3-е издание

LINUX

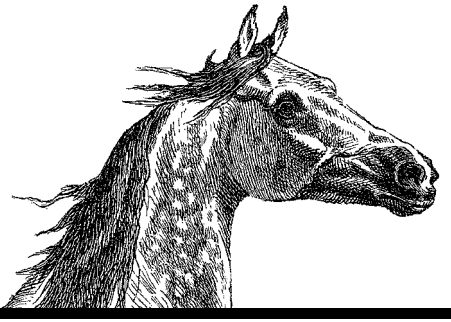
СПРАВОЧНИК

СИМВОЛ

O'REILLY®

Эллен Сивер, Стивен Спэйнауэр,
Стивен Фиггинс & Джессика П. Хекман

По договору между издательством «Символ-Плюс» и Интернет-магазином «Books.Ru – Книги России» единственный легальный способ получения данного файла с книгой ISBN 5-93286-012-X, название «Linux. Справочник, 3-е издание» – покупка в Интернет-магазине «Books.Ru – Книги России». Если Вы получили данный файл каким-либо другим образом, Вы нарушили международное законодательство и законодательство Российской Федерации об охране авторского права. Вам необходимо удалить данный файл, а также сообщить издательству «Символ-Плюс» (piracy@symbol.ru), где именно Вы получили данный файл.



LINUX IN A NUTSHELL

Third Edition

Ellen Siever, Stephen Spainhour,
Stephen Figgins & Jessica P. Hekman

O'REILLY®

Третье издание

Эллен Сивер, Стивен Спейнауэр,
Стивен Фиггинс и Джессика П. Хекман

СИМВОЛ

Санкт-Петербург
2001

Эллен Сивер, Стивен Спейнауэр,
Стивен Фиггинс и Джессика П. Хекман

Linux. Справочник

Перевод М. Зислиса

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*Т. Морозова*
Редакторы	*Н. Макарова, В. Овчинников*
Корректура	*С. Журавина*
Верстка	*Н. Макарова*

Сивер Э., Спейнауэр С., Фиггинс С., Хекман Д.
Linux. Справочник. – Пер. с англ. – СПб: Символ-Плюс, 2001. – 912 с., ил.
ISBN 5-93286-012-X

Это уже третье издание справочника, освещающего все основные команды Linux, доступные в любом дистрибутиве системы. Здесь вы найдете полноценный справочный материал по пользовательским командам, инструментам программирования, администрирования и работы с сетью. Книга необходима каждому пользователю Linux и включает все, что нужно для каждодневной работы с системой: команды с полным описанием параметров, синтаксис работы с интерпретаторами *bash*, *csh* и *tcsh*, поиск по шаблонам, команды редактирования Emacs и *vi*, команды *sed* и *gawk*, инструменты разработки программного обеспечения, команды системного администрирования.

В третье издание включен новый материал с описанием программ LILO и Loadlin, инструментов контроля версий RCS и CVS, методов настройки графических сред GNOME и KDE и оконного менеджера fvwm2. Подробно рассмотрены менеджер пакетов Red Hat – *rpm* и системы управления пакетами Debian – *dpkg*. Добавлено описание многих новых команд.

ISBN 5-93286-012-X
ISBN 0-59600-025-1 (англ)

© Издательство Символ-Плюс, 2001
Authorized translation of the English edition © 2000 O'Reilly & Associates Inc. This translation is published and sold by permission of O'Reilly & Associates Inc., the owner of all rights to publish and sell the same.

Все права на данное издание защищены Законодательством РФ, включая право на полное или частичное воспроизведение в любой форме. Все товарные знаки или зарегистрированные товарные знаки, упоминаемые в настоящем издании, являются собственностью соответствующих фирм.

Издательство «Символ-Плюс». 193148, Санкт-Петербург, ул. Пинегина, 4,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.

Подписано в печать 15.10.2001. Формат 70x100 1/16. Бумага офсетная.
Печать офсетная. Объем 57 печ. л. Тираж 3000 экз. Заказ N

Отпечатано с диапозитивов в ФГУП «Печатный Двор» им. А. М. Горького
Министерства РФ по делам печати, телерадиовещания
и средств массовых коммуникаций.
197110, Санкт-Петербург, Чкаловский пр., 15.

Оглавление

Предисловие 9

1. Введение 15

Великолепный Linux 16
Распространение и поддержка 17
Инструменты Linux 18
Что есть в этой книге 18
Исходные тексты и лицензирование 21
Руководство для начинающих 22

2. Обзор системного и сетевого администрирования 27

Общие команды 27
Обзор работы с сетями 33
Обзор TCP/IP 35
Брандмауэры и сокрытие адресов 41
Обзор NFS 43
Обзор NIS 45
Администрирование NIS 46
RPC и XDR 47

3. Команды Linux 48

Алфавитный перечень команд 49

4. Методы загрузки 459

Процесс загрузки 459
LILO: загрузчик Linux 461
Loadlin: загрузка из MS-DOS 473
Двойная загрузка Linux и Windows NT/2000 475
Параметры, передаваемые ядру при загрузке 477
initrd: использование RAM-диска 479

5. Менеджеры пакетов Red Hat и Debian 481

Менеджер пакетов Red Hat 482
Менеджер пакетов Debian 494

6. Обзор командных интерпретаторов Linux 514

Назначение командного интерпретатора 514
Разновидности интерпретаторов команд 515
Общие возможности 516
Различия 518

7. bash: командный интерпретатор Борна 519

Обзор возможностей 519
Запуск интерпретатора 521
Синтаксис 523
Переменные 534
Арифметические выражения 539
Журнал команд 541
Встроенные команды 544
Управление заданиями 573

8. Интерпретаторы csh и tcsh 575

Обзор свойств 576
Вызов интерпретатора 576
Синтаксис 577
Переменные 582
Выражения 595
Журнал команд 599
Работа с командной строкой 603
Управление заданиями 608
Встроенные команды csh и tcsh 609

9. Шаблоны и поиск 633

Имена файлов в сравнении с шаблонами 634
Метасимволы в программах Linux 634
Метасимволы 636
Примеры поиска 638

10. Редактор Emacs 641

Введение 641
Типичные проблемы 641
Конспект команд по группам 644
Конспект команд Emacs по клавишам 652
Конспект команд Emacs по именам 657

11. Редактор vi 664

Обзор операций vi 665
Параметры командной строки vi 667
Параметры командной строки ex 668
Команды навигации 670
Команды редактирования 672
Различные команды 677
Алфавитный перечень команд, действующих в командном режиме 678
Синтаксис команд ex 681
Алфавитный перечень команд ex 682
Настройка vi 695

12. Редактор sed 699

Общий обзор sed 699
Синтаксис командной строки 700
Синтаксис команд sed 701
Перечень команд sed по группам 702
Алфавитный перечень команд sed 704

13. Язык сценариев gawk 713

Обзор возможностей 714
Синтаксис командной строки 714
Шаблоны и процедуры 716
Системные переменные gawk 718
Операторы 719
Работа с переменными и массивами 720
Перечень команд awk по группам 721
Алфавитный перечень команд 721

14. CVS и RCS 729

Основные понятия 729
Инструмент CVS 732
Справочник администратора CVS 733
Справочник пользователя CVS 754
Инструмент RCS 782
Обзор команд RCS 783
Основные операции RCS 783
Общие стандарты RCS 784
Алфавитный перечень команд RCS 789

15. GNOME 803

Обзор рабочего стола 804
Панель 807
Главное меню 809
Центр управления GNOME 811

16. KDE 830

Обзор рабочего стола 831
Основная панель и панель задач 837
Центр управления KDE 839

17. Альтернативный оконный менеджер fvwm2 857

Запуск fvwm2 858
Файлы настройки 858
Модульный подход 859
Как вводить в действие новые настройки оконного менеджера 860
Краткий обзор среды fvwm 861
Настройка передачи фокуса по щелчку 863
Автоматический перевод окна с фокусом на передний план 864
Изменение размера рабочего стола 864
Работа с несколькими рабочими столами 865
Отображение окна на всех виртуальных страницах 867
Открытие окон на разных рабочих столах и виртуальных страницах 869
Настройка скорости перемещения указателя между страницами 870
Создание сочетаний клавиш 872
Настройка меню 877
FvwmWinList: переключение фокуса 879

Алфавитный указатель 882

Предисловие

Вы держите в руках книгу о Linux – свободно доступном клоне операционной системы Unix для персональных компьютеров. Linux был разработан Линусом Торвальдсом (Linus Torvalds), который создал ядро системы в его первоначальном виде и сегодня продолжает руководить его совершенствованием. Операционная система развивается совместными усилиями многочисленных программистов и хакеров со всего мира, общающихся через Интернет. Помимо кода ядра в Linux входят инструменты и программы, созданные в рамках Фонда свободно распространяемого программного обеспечения (Free Software Foundation, FSF) или разработанные для Berkley Unix (BSD), а также полная версия системы X Window (XFree86) от X Consortium и многочисленные программы, созданные непосредственно для системы Linux.

Linux непрерывно совершенствуется – это либо дополнения к ядру, такие как поддержка планирования выполнения процессов в реальном времени или поддержка RAID-массивов, либо создание библиотек и приложений, кардинально улучшающих жизнь пользователя. Наиболее известными примерами подобных усовершенствований являются графические среды GNOME и KDE, кратко описанные в этой книге.

Данное издание представляет собой краткий справочник по основным командам и функциям операционной системы Linux. Как и прочие справочники серии «In a Nutshell» издательства O'Reilly, эта книга рассчитана на пользователей, которые знают, что им необходимо сделать, и имеют представление о том, каким образом это делать, однако не могут сразу вспомнить конкретную команду или параметр. Надеемся, это руководство станет незаменимым настольным справочником для любого пользователя Linux.

Прочие источники информации

В этой книге вы не найдете сведений о том, как устанавливать Linux и поддерживать работоспособность системы. Информацию об этом можно найти в книгах Билла Маккарти (Bill McCarty), вышедших в издательстве O'Reilly, – «Изучаем Red Hat Linux» («Learning Red Hat Linux») и «Изучаем Debian GNU/Linux» («Learning Debian GNU/Linux»), каждая из которых является справочником по установке и конфигурированию Linux, а также содержит дистрибутив системы на прилагаемом компакт-диске. Кроме того, рекомендуем вам книгу Матта Уэлша (Matt Welsh), Маттиаса Калле Далхаймера (Matthias Kalle Dalheimer) и Лара Кауфмана (Lar Kaufman) «Запускаем Linux»[1] («Running Linux»), являющуюся подробным руководством по всем основным дистрибутивам. Информацию по работе с сетями можно найти в «Руководстве сетевого администратора Linux» («Linux Networking Administration Guide») Олафа Кирха (Olaf Kirch) и Терри Доусона (Terry Dawson). В дополнение к изданным O'Reilly книгам о Linux пользователя Linux могут заинтересовать и многочисленные книги по Unix, системе X Window, Perl и Java.

Сетевые ресурсы

Массу информации по Linux можно найти в Интернете. Одним из лучших источников информации является сайт Проекта документирования Linux (Linux Documentation Project), расположенный по адресу *http://www.linux-doc.org*. На сайте представлены многочисленные краткие руководства, называемые HOWTO, а также некоторые из подробных руководств. Информация по инструментам GNU, описанным в этой книге, доступна по адресу *http://www.gnu.org* (или на любом из десятков сайтов-зеркал по всему миру). Фонд свободно распространяемого программного обеспечения (FSF), координирующий проект GNU, публикует свою документацию, выпуская книги, посвященные различным инструментам.

«Linux Journal» и «Linux Magazine»

«Linux Journal» и «Linux Magazine» – это ежемесячные журналы, адресованные многочисленным пользователям Linux. Они составляются и публикуются целым рядом активистов движения Linux. В них вы сможете найти материалы на самые разные темы – от статей, отвечающих на вопросы новичков, до сведений о программировании ядра. «Linux Journal» – более старый журнал, публикуемый S.S.C. Incorporated по адресу *http://www.ssc.com*. «Linux Magazine» можно найти на *http://www.linuxmagazine.com*.

---

[1] Уэлш М., Далхаймер М., Кауфман Л. «Запускаем Linux», издательство «Символ-Плюс», 2000 г.

Конференции Usenet, посвященные Linux

При наличии доступа к группам новостей Usenet можно воспользоваться следующими конференциями:

comp.os.linux.announce

Модерируемая конференция, в которую попадает информация о новом программном обеспечении, новых дистрибутивах, сообщения о найденных ошибках в программах и информация о событиях в сообществе Linux. Всем пользователям Linux стоит посещать эту конференцию. Собственные сообщения можно отправлять по адресу *linux-announce@news.ornl.gov*.

comp.os.linux.help

Общие вопросы и ответы по установке и использованию системы Linux.

comp.os.linux.admin

Обсуждение тем, связанных с системным администрированием Linux.

comp.os.linux.networking

Обсуждение работы с сетями в Linux.

comp.os.linux.development

Обсуждение тем, связанных с развитием ядра Linux и системы в целом.

comp.os.linux.misc

Всеобъемлющая конференция, предназначенная для обсуждений, которые не попадают в предыдущие категории.

Существует также несколько конференций, посвященных Linux, в которых основным языком является не английский, а, например, французский (*fr.comp.os.linux*) и немецкий (*de.comp.os.linux*)[1].

Онлайновая поддержка Linux

Существует множество способов получения помощи в онлайновом режиме, поскольку добровольцы со всего мира предлагают свой опыт и услуги пользователям, у которых возникают вопросы и проблемы.

Сеть IRC-чатов OpenProjects полностью отведена под открытые проекты, причем как Open Source (Программное обеспечение с открытым кодом), так и Open Hardware (Открытое аппаратное обеспечение). Некоторые из каналов Сети предназначены для обеспечения онлайновой поддержки Linux. IRC (Internet Relay Chat) представляет собой сетевую службу, позволяющую участвовать в интерактивном общении с другими пользователями Интернета. Сети IRC обеспечивают поддержку многочисленных каналов, на каждом из ко-

[1] Существует также русскоязычная группа новостей *fido7.ru.linux*. – *Примеч. науч. ред.*

торых общается друг с другом произвольное количество людей. Все, что вы введете, находясь на канале, будет прочитано всеми присутствующими.

Существует целый ряд активных каналов в сети IRC OpenProjects, где 24 часа в сутки и семь дней в неделю можно застать пользователей, готовых и способных разрешить любые ваши проблемы с системой Linux или просто поболтать. Вы можете использовать службу IRC, установив на своей машине IRC-клиент, наподобие *irc-II*, подключившись к серверу *irc.openprojects.org:6667* и присоединившись к каналу `#linpeople`[1].

Группы пользователей Linux

Большое количество групп пользователей Linux по всему миру предлагают прямую поддержку пользователям. Многие такие группы участвуют в мероприятиях, посвященных установке системы, в семинарах и форумах, в вечерах демонстрации и других неофициальных встречах. Группы пользователей Linux – это замечательный способ познакомиться с другими пользователями системы, находящимися поблизости. Существует несколько списков пользовательских групп, публикуемых в Сети. Вот некоторые из наиболее популярных:

Groups of Linux Users Everywhere

http://www.ssc.com/glue/groups

LUG list project

http://www.nllgg.nl/lugww

LUG registry

http://www.linux.org/users

Обозначения, используемые в книге

В этом справочнике приняты следующие типографские соглашения:

Жирный шрифт

Используется при написании имен команд, программ и параметров. Все термины, обозначенные жирным шрифтом, вводятся буквально.

Курсив

Применяется для отображения аргументов и переменных, которые пользователь должен заменить конкретными значениями. Курсив используется для указания имен файлов и каталогов, а также для выделения комментариев в примерах.

`Моноширинный шрифт`

Используется для отображения содержимого файлов или вывода команд.

---

[1] Или `#rusunix` и `#unix.ru` на *irc.msu.ru*. – *Примеч. науч. ред.*

`Моноширинный жирный`

Используется в примерах и таблицах для отображения команд или иного текста, который должен вводиться пользователем буквально.

Моноширинный курсив

Используется в примерах и таблицах для отображения текста, который пользователь должен заменить конкретными значениями.

Символы % и $

Используются в некоторых примерах для обозначения приглашения интерпретатора *tcsh* (%) и интерпретатора Борна или **bash** ($).

Квадратные скобки [ ]

В квадратные скобки заключаются необязательные параметры команд в описаниях синтаксиса (сами по себе квадратные скобки никогда не вводятся). Обратите внимание, что многие команды имеют необязательный аргумент [*files*]. Если имя файла опущено, чтение происходит со стандартного ввода (т. е. с клавиатуры). Чтение заканчивается при получении символа конца файла.

EOF

Означает символ конца файла (обычно <Ctrl>+<D>).

Символ «|»

Используется при описании синтаксиса для разделения элементов, из которых единовременно может быть выбран только один.

Изображение совы означает примечание, которое относится к близлежащему тексту. Например:

Текст, расположенный рядом с изображением совы, является примечанием, подобно данному предложению.

Еще раз о синтаксисе. Во многих случаях пробел между параметром и его аргументом может быть опущен. В прочих ситуациях необходимо скрупулезно следовать описанию. Например, параметр **–w***n* (пробел отсутствует) может интерпретироваться иначе, чем **–w** *n*. Обращайте внимание на наличие пробелов в описании синтаксиса параметра.

Как с нами связаться

Мы приложили массу усилий, чтобы проверить правильность и точность информации в этой книге, но вы можете обнаружить, что формат команд изменился (и даже найти наши ошибки!). Пожалуйста, сообщайте нам обо всех найденных ошибках и присылайте ваши предложения, касающиеся следующих изданий, по адресу:

O'Reilly & Associates, Inc.
101 Morris Street
Sebastopol, CA 95472
800-998-9938 (в США или Канаде)
707-829-0515 (международный/местный телефон)
707-829-0104 (факс)

Вы также можете послать нам сообщение по электронной почте. Чтобы подписаться на список рассылки или заказать каталог, пишите по адресу:

info@oreilly.com

Чтобы задать технический вопрос или прокомментировать содержание книги, воспользуйтесь адресом:

bookquestions@oreilly.com

У данной книги существует веб-сайт, где представлены примеры, ошибки и планы будущих изданий. Эту страницу можно найти на:

http://www.oreilly.com/catalog/linuxnut3

Дополнительные сведения об этой и других книгах можно найти на сервере O'Reilly:

http://www.oreilly.com

Благодарности

Это издание книги «Linux. Справочник» является результатом совместной работы многих людей. Спасибо Энди Ораму (Andy Oram) за его редакторское мастерство, Вэл Кверкиа (Val Quercia) за ее умение управлять проектом и им обоим за вклад в эту работу – редактирование и проверку существующих глав, обновление и добавление материала по мере необходимости.

Спасибо Матту Уэлшу, известному по книгам «Запускаем Linux» («Running Linux») и «Installation and Getting Started Guide», за техническую рецензию; Майклу Джонсону (Michael K. Johnson) из Red Hat Software, Роберту Чесселу (Robert J. Chassell), Филу Хьюзу (Phil Hughes) и Лори Линн Такер (Laurie Lynne Tucker) из «Linux Journal»; Арнольду Роббинсу (Arnold Robbins), Юлианне Мидгли (Julian T.J. Midgley), Дугу Морину (Doug Moreen), Рону Пассерини (Ron Passerini) и Марку Стоуну (Mark Stone).

Глава 1

Введение

Всего за несколько лет из развлечения для студентов и программистов Linux превратился в нежданного конкурента на рынке серверных систем, занявшего достойное место в областях образования и корпоративных сетей. Свободно распространяемый клон операционной системы Unix, Linux проникает повсюду. Его используют как платформу для веб-серверов, файловых серверов и рабочих станций наряду с традиционными Unix-системами и Windows NT. Кроме той роли, которую Linux выполняет в крупных сетях (а в сетях он очень неплохо уживается с другими операционными системами), клон весьма популярен среди тех пользователей Windows, которые просто хотят испробовать нечто более быстрое, мощное и гибкое в управлении.

Проникновением на рынок коммерческих операционных систем список заслуг Linux не исчерпывается. Успех Linux вдохновил бесчисленные проекты разработки бесплатного и открытого программного обеспечения (*http://opensource.org*), включая такие, как Samba и GNOME, а также ошеломляющее количество новаторских проектов, о которых можно узнать на многочисленных сайтах, например на SourceForge (*http://sourceforge.net*). И в качестве платформы для разработчиков, и в качестве модели разработки Linux послужил сильнейшим катализатором для проекта GNU Фонда свободно распространяемого программного обеспечения, который, в свою очередь, стал источником программ, обеспечивающих дальнейшее развитие системы Linux. Если говорить кратко, то Linux играет главную роль в самом замечательном и продуктивном за всю историю компьютеров движении свободно распространяемого ПО.

Если у вас еще нет копии Linux или уже есть, но вы не слишком в себе уверены, чтобы начать пользоваться системой, загляните в предисловие к этой книге.

Великолепный Linux

Прежде всего, Linux – это свободно распространяемое программное обеспечение: любой может получить исходные тексты системы из Интернета или купить недорогую копию на компакт-диске. Однако Linux приобретает известность не только благодаря своей бесплатности, но и благодаря тому, что это еще и необыкновенно качественная система. Ваше аппаратное обеспечение будет использоваться более эффективно с Linux (особенно на платформе Intel, для которой он изначально и разрабатывался), вы будете гораздо реже переживать из-за критических сбоев в системе, а уровень безопасности, соблюдаемый в Linux, превосходит созданное во многих коммерческих системах.

Будучи бесплатным, Linux дал новую жизнь творческому вдохновению и открытости сообщества Unix-систем. Непревзойденная гибкость и открытость Unix (те вещи, которые обычно пугают и повергают в уныние новичков, и одновременно те вещи, без которых со временем бывшие новички уже не могут обойтись) располагают к постоянному развитию, появлению новых инструментов, подобных языку Perl, к экспериментам в информатике, которые иногда приводят к созданию повсеместно используемых коммерческих компьютерных систем.

Многие бережно хранят в памяти тот день, когда компания AT&T сделала исходные тексты системы Unix бесплатными для университетов, и университет Беркли начал распространять свою версию во всех доступных формах. Этим ветеранам-хакерам Linux может вернуть вдохновение и радость совместной работы, особенно если учесть, насколько развита сегодня сеть Интернет. А для тех, кто еще слишком молод, чтобы помнить о первой волне открытых систем (например, для тысяч студентов, которых сегодня привлекает Linux), или чей предыдущий опыт ограничивался, к сожалению, лишь коммерческими операционными системами, настало время открыть для себя великолепный мир свободно распространяемых исходных текстов и бесконечно гибких средств работы.

Ядро Linux было разработано Линусом Торвальдсом, студентом Хельсинского университета, и затем развивалось при сотрудничестве многочисленных добровольцев со всего мира. Под «ядром» понимается сама основа операционной системы, а не приложения, исполняемые в системе (такие, например, как компиляторы или командные интерпретаторы). Сегодня название «Linux» применяется для обозначения как самого ядра, так и приложений и системного окружения в целом.

Большую часть систем Linux с технической точки зрения нельзя назвать «версией Unix», поскольку они не прошли обязательных тестов и не были соответствующим образом сертифицированы.[1] Однако по крайней мере один из дистрибутивов Linux официально соответствует стандарту POSIX.1. Linux предоставляет все основные интерфейсы программирования стандартных

[1] Прежде чем получить статус Unix-системы, система должна быть сертифицирована концерном X/Open.

Unix-систем, и, как станет ясно из этой книги, многие из распространенных инструментов Unix были воссозданы в Linux. Это мощная, надежная, готовая к использованию система предназначена для тех, кому по душе Unix.

Экономическую основу популярности Linux составляет огромный спектр аппаратного обеспечения IBM-совместимых персональных компьютеров, поддерживаемого системой. Люди, привыкшие к работе с MS-DOS и Microsoft Windows, бывают поражены, обнаружив, насколько быстрее работают под управлением Linux устройства их компьютеров. Linux эффективно использует существующие ресурсы.

На заре существования Linux люди использовали его по финансовым или политическим соображениям, но вскоре обнаружили неожиданное преимущество: эта система работает лучше, чем многие коммерческие системы. Например, при использовании в качестве файлового сервера или сервера печати Samba Linux надежно, без сбоев в работе обслуживает большое количество PC-клиентов. При использовании с веб-сервером Apache Linux обеспечивает наибольшее (по сравнению с конкурентами) количество возможностей, необходимых веб-администраторам.

Распространение и поддержка

И хотя несложно загрузить одну или две новые программы из Интернета и вполне возможно загрузить что-то настолько же большое, как ядро Linux, попытка получить по телефонным линиям работающую систему целиком кажется просто абсурдной. В результате за несколько лет появилось множество коммерческих и бесплатных поставок системы, которые называются *дистрибутивами*. Первые дистрибутивы занимали приблизительно по 50 дискет, одна из которых, как правило, была испорченной, и ее приходилось заменять. Лишь с широким распространением приводов CD-ROM дистрибутивы Linux обрели настоящую жизнь.

Получив дистрибутив Linux, средний пользователь обычно задумывается о дальнейшей его поддержке. Помимо того что электронные конференции Usenet позволяют очень быстро найти ответ на интересующий вопрос (этим способом пользуются многие отважные пользователи системы), существует возможность купить услуги по поддержке у производителей основных дистрибутивов или у независимых специалистов. Можно сказать с определенностью, что поддержка Linux развита, по крайней мере, не хуже, чем у коммерческого ПО.

В настоящее время Linux в основном распространен на платформе Intel, но также доступен на коммерческой основе для целого ряда аппаратных платформ, таких как PowerPC, Alpha (создана Digital Equipment Corporation, в настоящее время Compaq), и на процессорах SPARC, MIPS и 64-разрядных процессорах Intel Itanium.

Инструменты Linux

Инструментарий Linux не идентичен предлагаемому стандартной системой Unix. Он лучше! Лучше по той причине, что большая часть программ предоставлена проектом GNU, развивающимся под управлением Фонда свободно распространяемого программного обеспечения. GNU – это рекурсивная аббревиатура «GNU's not Unix» (GNU – это не Unix).

Умудренные годами применения стандартных инструментов Unix, разработчики проекта GNU использовали последние достижения в информатике и создали новые версии программ, которые обладают большими возможностями, работают быстрее и эффективнее, а также свободны от ошибок и несовместимостей своих предшественников.

Проект GNU обеспечил систему Linux инструментарием для создания программ и стандартными утилитами, такими как *grep*, а большая часть программ системного и сетевого администрирования пришла в Linux из дистрибутива Berkeley Software (BSD). Кроме того, независимыми разработчиками были созданы программы непосредственно для Linux, например, для работы с файловыми системами, поддерживаемыми только Linux. В этой книге описаны все стандартные команды Unix, доступные в большинстве дистрибутивов Linux.

Третий тип программного обеспечения, обычно встречающийся на платформе Linux, – это система X Window, которую проект XFree86 сделал доступной на процессорах Intel. В этой книге мы не можем охватить широкий спектр X-инструментов, поэтому лишь кратко опишем некоторые из полезных настроек, которые вы можете применить к своему рабочему столу в KDE, GNOME или **fvwm**.

Что есть в этой книге

Книга основывается на классическом справочном издании O'Reilly & Associates «Unix in a Nutshell», но существенно дополнена информацией, специфичной для Linux. Третье издание включает главы, посвященные инструментам администрирования программных пакетов (которые упрощают процесс установки, обновления или удаления соответствующих файлов программ), интерфейсам графической среды KDE и GNOME, оконному менеджеру **fvwm**, а также информацию о новых командах и более подробное рассмотрение некоторых тем, в частности системы контроля версий CVS и интерпретатора команд **bash**.

«Linux. Справочник» не является учебником по Linux. В конце концов, это всего лишь краткий справочник, но данная книга будет весьма полезна как новичкам, так и профессиональным пользователям. Когда вы имеете представление о том, какая команда требуется, но точно не знаете, как она действует, или какая комбинация параметров даст нужный вам результат, воспользуйтесь этой книгой. Книга также раскроет вам глаза на возможности, о существовании которых вы раньше могли даже не подозревать.

Как и многие компьютерные системы эпохи создания Unix (начала 70-х), Linux управляется в основном командами. В большинстве Linux-систем существует некоторое количество графических приложений (в том числе коммерческих), но ни одно из них не является для Linux ведущим. Вот почему эта книга, следуя традициям справочника «Unix in a Nutshell», строится вокруг командных интерпретаторов и исполняемых в них команд.

Конечно же, у Linux есть графическая оконная система, богатая возможностями и очень гибкая, как и подобает гибкой и перспективной операционной системе. Но даже работая с ней, вы будете проводить большую часть времени в имитации терминала VT100 (программа **xterm**), осуществляя ручной ввод команд. Вы обнаружите, что при работе с графической оболочкой постоянно обращаетесь к командам, описанным в этой книге.

Поэтому первое, что вам необходимо сделать после преодоления барьера установки Linux, – это привыкнуть к основным инструментам, работа с которыми происходит в интерпретаторе команд. Если вы абсолютно ничего не знаете о Unix-системах, мы советуем прочесть руководство начального уровня. Вам могут помочь вводные главы таких книг O'Reilly, как «Изучаем Red Hat Linux» («Learning Red Hat Linux»), «Изучаем Debian GNU/Linux» («Learning Debian GNU/Linux») и «Запускаем Linux» («Running Linux»). В настоящей книге в главе 2 «Обзор системного и сетевого администрирования» дается контекст для понимания различных типов команд, включая команды программирования, системного администрирования и сетевого администрирования; сами команды описаны в главе 3 «Команды Linux». Очевидно, что третья глава является основой этой книги и занимает примерно треть ее объема.

Небольшие главы, непосредственно следующие за главой 3, помогут в настройке вашей системы. Поскольку большинство пользователей не хотят расставаться с другими операционными системами (например Microsoft Windows, OS/2 или какой-либо вариант Unix), Linux зачастую является не единственной операционной системой на машине. Пользователь может загружать ту или иную систему, исходя из текущих задач. В главе 4 «Методы загрузки» описаны наиболее популярные способы настройки загрузки на системах Intel, включая загрузчики LILO (Linux Loader) и Loadlin. В главе 5 «Средства управления программными пакетами в Red Hat и Debian» описан менеджер пакетов Red Hat (**rpm**), который поддерживается дистрибутивами Red Hat и SuSE, и менеджер пакетов Debian (**dpkg**). Менеджеры пакетов абсолютно необходимы при установке и обновлении программного обеспечения; они контролируют наличие необходимых для корректного функционирования файлов нужных версий.

Все команды исполняются командным интерпретатором. Командный интерпретатор – это просто программа, принимающая команды пользователя и исполняющая их. Синтаксис команд в различных интерпретаторах может отличаться. В системе Linux наиболее популярны два командных интерпретатора – **bash** и **tcsh**, которые немного отличаются друг от друга. (Одна из прелестей Linux и других Unix-систем заключается в возможности выби-

рать командный интерпретатор на свой вкус, руководствуясь теми или иными достоинствами и недостатками каждого из них.) Командным интерпретаторам посвящены несколько глав книги. Вероятно, к этим главам вы обратитесь тогда, когда у вас будет уже некоторый опыт работы с системой, поскольку в них описаны в основном мощные, нетривиальные возможности, необходимость в которых испытывают умелые пользователи.

Вам предстоит изучить мощные инструменты, позволяющие делать реальную работу: в частности, редактор и инструментарий для создания сценариев. Два основных редактора, используемых в Linux, – это **vi** и Emacs. За главами о редакторах следуют две главы о классических инструментах Unix, предназначенных для построчной обработки файлов: **sed** и **gawk** (GNU-версия традиционного **awk**). Издательством O'Reilly выпущены книги, посвященные каждой из этих тем. Они могут оказаться для вас полезными, поскольку перечисленные инструменты не могут похвастаться интуитивной легкостью изучения. Emacs, тем не менее, обладает превосходным встроенным руководством, доступ к которому можно получить, запустив редактор и нажав <Ctrl>+<H>, а затем <t> (от слова «tutorial» – руководство).

Системы контроля разработки – CVS (Concurrent Versions System) и RCS (Revision Control System) – позволяют при работе с файлами сохранять старые версии и одновременно обеспечивать удобную работу с несколькими вариантами новых версий. Изначально RCS и CVS использовались программистами с целью обеспечения поддержки сложных разработок и рационального развития приложений, но затем доказали свою полезность в любом проекте, где требуется контроль изменений файлов, особенно если в проекте задействовано большое количество людей. CVS является надстройкой для RCS и обеспечивает упрощенное одновременное редактирование файлов многими людьми. Команды CVS и RCS представлены в главе 14 «CVS и RCS».

Каждый дистрибутив Linux немного отличается от других, но, как вы увидите, команды, приведенные в этой книге, работают без изменений в любом дистрибутиве и используются наиболее часто. В книге описываются основные команды, инструменты разработки, системное администрирование и сетевое администрирование. Однако некоторые вопросы оказались настолько объемными, что мы вынуждены были их не затрагивать. Через этот фильтр не прошли также многие приложения, использующие систему X Window, TEX (инструмент для оформления текста, широко используемый в научных кругах и пользователями Linux вообще), многие полезные языки программирования, такие как Perl, Tcl/Tk и Python, с помощью которых пользователь может значительно расширить функциональность своей системы. Книга раздулась бы до непомерной толщины, если бы мы попытались рассмотреть и эти темы.

Нашей же целью при создании этой книги было удобство, а удобство подразумевает небольшие размеры. Конечно, здесь нет всей информации, которая содержится на страницах руководства, но вы найдете ответы на 95% вопросов, возникающих в процессе работы.

Исходные тексты и лицензирование

Получая Linux, вы получаете и исходные тексты системы. То же относится ко всем утилитам в Linux (за исключением случаев, когда поставщик предлагает коммерческие приложения или библиотеки в дополнение к дистрибутиву). Вы можете никогда не заглядывать в исходные тексты программ, но именно они являются ключом к развитию системы. Лицензия распространения Linux обязывает поставщика включать исходные тексты в дистрибутив, что позволяет знающим людям исправлять ошибки в работе программ, вносить предложения по общим принципам работы системы и таким образом улучшать жизнь пользователей системы. Речь идет о широко известной лицензии General Public License, так же известной как *GPL* или *copyleft* (противоположность *copyright*), которая разработана и активно продвигается Фондом свободно распространяемого программного обеспечения (FSF).

Фонд, основателем которого является Ричард Столлман (Richard Stallman), представляет собой явление, в возможность существования которого было бы трудно поверить, не будь оно реальностью. (Та же ситуация характерна для Linux: кто мог предположить десять лет назад, что возможно создать операционную систему путем сотрудничества через Интернет да еще сделать ее свободно распространяемой?) Именно Фондом создан один из самых популярных редакторов для Unix – GNU Emacs, а также **gcc** и **g++** (компиляторы языков C и C++), которые некоторое время устанавливали стандарты оптимизации и скорости работы кода. Одним из крупнейших проектов GNU является графическая среда GNOME, включающая несколько полезных библиотек общего назначения, оконные менеджеры и приложения. Разработчики GNOME в перспективе планируют создать среду, в которой будут интегрированы не только приложения пользовательской системы, но также и сервисы, охватывающие организацию в целом.

Работая на идею свободного ПО, FSF делает доступными исходные тексты и документацию к программам в сети Интернет, и каждый, кто жаждет вносить изменения в программы, волен это делать. Одним из проектов FSF является Linux-дистрибутив Debian.

В целях сохранения открытости FSF требует, чтобы измененные исходные тексты также распространялись вместе с программными продуктами по той же лицензии GPL. Это поощряет частных лиц и компании вносить усовершенствования и делиться ими с компьютерным миром. Единственное, что невозможно сделать в этих условиях, – это внести небольшие усовершенствования и пытаться распространять продукт на коммерческой основе, т. е. без исходных текстов. Это будет идти вразрез с интересами FSF и пользователей. Лицензию GPL можно найти в любом программном пакете, распространяемом под этой лицензией, а также по адресу *http://www.gnu.org/copyleft/gpl.html*.

Как уже говорилось, многие инструменты Linux происходят из BSD, а не GNU. BSD также является свободным программным обеспечением. Лицензия на распространение BSD достаточно сильно отличается от GPL, но вам

как пользователю это неважно. Отличия лицензии BSD позволяют разработчикам включать программное обеспечение в коммерческие продукты – практика, строго запрещенная лицензией GNU.

Руководство для начинающих

Если вы только начинаете работать с Linux, изобилие команд может испугать. Тематические выборки команд, представленные в таблицах, помогут вам сориентироваться.

Сети

ftp File Transfer Protocol (протокол передачи файлов).

login Регистрация.

rlogin Регистрация в удаленной системе.

rsh Выполнить интерпретатор или команду на удаленной системе.

talk Диалоговый обмен сообщениями с пользователями других терминалов.

telnet Соединение с удаленной системой.

tftp Trivial File Transfer Protocol (упрощенный протокол передачи файлов).

uudecode Декодирование файла, подготовленного к почтовой пересылке посредством **uuencode**.

uuencode Кодирование файла, содержащего двоичные данные, для пересылки по почте.

vacation Автоматический ответ на почтовые сообщения.

Поиск различий

cmp Посимвольное сравнение двух файлов.

comm Сравнение элементов в двух отсортированных файлах.

diff Построчное сравнение двух файлов.

diff3 Сравнение трех файлов.

Работа с файлами

cat Конкатенация файлов или отображение их содержимого.

chfn Изменение информации о пользователе, используемой командой finger, в электронной почте и т. д.

cksum Вычисление контрольной суммы.

chmod Изменение прав доступа для файлов.

chsh Изменение интерпретатора команд, используемого по умолчанию.

cp Копирование файлов.

csplit Разбиение файла на кусочки в соответствии с заданным критерием.
dd Прямое копирование файлов.
file Определение типа файлов.
head Отображение нескольких первых строк файла.
less Постраничный просмотр файлов.
ln Создание ссылок.
ls Отображение списка файлов или каталогов.
merge Объединение изменений из нескольких файлов.
mkdir Создание каталога.
more Постраничный просмотр файлов.
mv Перемещение или изменение имен файлов или каталогов.
newgrp Смена текущей группы.
pwd Вывести имя рабочего каталога.
rcp Копирование файлов на удаленную систему.
rm Удаление файлов.
rmdir Удаление каталогов.
split Разбиение файлов на секции одинакового размера.
tail Просмотр нескольких последних строк файла.
wc Подсчет количества строк, слов и символов в файле.

Печать

lpq Показать состояние заданий печати.
lpr Отправить на печать.
lprm Удалить задание печати.
lpstat Определить состояние принтера.
pr Форматирование и разбиение на страницы для печати.

Программирование

ar Создание и обновление файлов библиотек.
as Создание объектного файла.
bison Создание таблиц синтаксического разбора.
cpp Препроцессор исходных текстов на C.
flex Лексический анализатор.
g++ Компилятор GNU C++.
gcc Компилятор GNU C.
ld Редактор связей.
m4 Макропроцессор.
make Сборка программ.

ranlib Воссоздание архивной таблицы имен.

rpcgen Транслятор из RPC в код на C.

yacc Создание таблиц синтаксического разбора.

Сопровождение программ

cvs Управление различными версиями исходных файлов.

etags Создание списка имен для использования с редактором Emacs.

gctags Создание списка имен для использования с редактором **vi**.

gdb Отладчик GNU.

gprof Отображение профильных данных объектного файла.

imake Создание файлов сборки для использования с **make**.

make Сопровождение, обновление и пересборка необходимых программ и файлов.

nm Отображение таблицы имен объектного файла.

patch Наложение патчей на исходные тексты.

rcs Управление различными версиями исходных файлов.

size Отображение размера объектного файла в байтах.

strace Отслеживание системных вызовов и сигналов.

strip Удаление таблицы имен из объектного файла.

Поиск

apropos Поиск страниц руководства по теме.

egrep Расширенная версия утилиты **grep**.

fgrep Поиск слов в файлах.

find Поиск файлов.

grep Поиск по шаблону.

strings Поиск текста в двоичных файлах.

whereis Поиск файлов для команды.

Программирование в интерпретаторе

echo Вывод аргументов команды.

expr Выполнение арифметических действий и сравнений.

printf Вывод отформатированных аргументов команды.

sleep Сделать паузу во время работы.

test Проверка условия.

Хранение

bzip2 Сжатие файлов в целях экономии дискового пространства.
cpio Создание и распаковка файловых архивов.
gunzip Распаковка сжатых (*.gz* и *.Z*) файлов (предпочтительная).
gzip Сжатие файлов в целях экономии дискового пространства.
shar Создание архива командного интерпретатора.
tar Копирование или извлечение файлов с носителя архива.
zcat Отображение содержимого сжатых файлов.

Состояние системы

at Выполнение команд в заданное время.
atq Отображение команд, выполнение которых запланировано с помощью **at**.
atrm Удаление команд из очереди событий **at**.
chgrp Изменение группы файла.
chown Изменение владельца файла.
crontab Автоматизация выполнения команд.
date Отображение или установка даты.
df Отображение свободного дискового пространства.
du Использование дискового пространства.
env Отображение переменных среды.
finger Отображение информации о пользователях.
kill Принудительное завершение работающей программы.
printenv Отображение переменных среды.
ps Информация по процессам в системе.
stty Установка или отображение свойств терминала.
who Информация о пользователях, работающих в системе.

Работа с текстом

col Обработка управляющих символов.
cut Выбор колонок текста для отображения.
ex Строчный редактор, на котором основан **vi**.
expand Преобразование табуляций в пробелы.
fmt Форматирование с равной длиной строк.
fold Разрыв строк.
gawk Потоковая обработка строк или записей.
ghostscript Отображение файлов в формате PostScript или PDF.
groff Форматирование ввода для **troff**.

ispell Диалоговое средство проверки орфографии.
join Объединение колонок текста в базу данных.
paste Объединение колонок текста или изменение порядка следования.
rev Вывод строк в обратном порядке.
sed Потоковый текстовый редактор.
sort Сортировка или слияние файлов.
tac Вывод строк в обратном порядке.
tr Трансляция (переопределение) символов.
uniq Поиск повторяющихся или уникальных строк в файле.
vi Текстовый редактор.
xargs Обработка множественных аргументов управляемыми порциями.

Разное

banner Создание слов плакатного размера.
bc Калькулятор произвольной точности.
cal Отображение календаря.
clear Очистка экрана.
man Получение информации по команде.
nice Уменьшить приоритет задачи.
nohup Продолжение работы задачи после выхода пользователя из системы.
passwd Установка регистрационного пароля.
script Создание записи сеанса работы.
su Получение полномочий суперпользователя.
tee Одновременный вывод на экран и сохранение в файле.
which Вывод полного пути к исполняемому файлу команды.

Глава 2

Обзор системного и сетевого администрирования

Общие команды

Далее следуют таблицы часто используемых команд системного администрирования.

Часы и время

hwclock Управление аппаратными часами.

netdate Установить время по времени *узла* (*host*).

rdate Управление сервером времени.

zdump Вывести перечень часовых поясов.

zic Создать файлы информации о преобразовании времени.

Демоны

apmd Демон расширенного управления питанием.

bootpd Демон протокола загрузки через Интернет.

fingerd Демон finger.

ftpd Демон протокола передачи файлов (FTP).

gated Демон управления таблицами межсетевой маршрутизации.

identd Демон идентификации пользователя, использующего TCP/IP.

imapd Демон почтовой службы протокола IMAP.

inetd Демон интернет-служб.

kerneld Демон, обеспечивающий автоматическую загрузку модулей ядра.

klogd Управление демоном **syslogd**.

lpd Демон системы печати.

mountd Демон, обрабатывающий запросы на монтирование файловой системы NFS.

named Сервер доменных имен Интернета.

nfsd Демон NFS.

pop2d POP-сервер.

pop3d POP-сервер.

powerd Управление ИБП.

pppd Обеспечение сетевых соединений по протоколу PPP (точка-точка – point-to-point protocol).

rdistd Сервер удаленного распределения файлов.

rexecd Сервер удаленного выполнения.

rlogind Сервер **rlogin**.

routed Демон маршрутизации.

rshd Сервер удаленного использования интерпретатора команд.

rwhod Удаленный сервер запросов **who**.

syslogd Демон, регистрирующий события, происходящие в системе.

talkd Демон, позволяющий общаться при помощи **talk**.

tcpd Сетевой демон TCP.

tftpd Демон тривиального протокола передачи файлов.

update Демон сброса буферов.

ypbind Демон связывания NIS.

yppasswdd Сервер изменения паролей NIS.

ypserv NIS-сервер.

Аппаратная часть

agetty Запуск пользовательской сессии на терминале.

arp Управление кэшем ARP.

cardctl Управление PCMCIA-устройствами.

cardmgr Демон управления PCMCIA-устройствами.

cfdisk Работа с разделами диска (графический интерфейс).

fdisk Работа с разделами диска.

getty Запуск пользовательской сессии на терминале.

kbdrate Установка скорости и задержки перед автоповтором нажатой клавиши.

ramsize Вывести информацию о RAM-диске.

setserial Установка параметров последовательного порта.

slattach Использование последовательных соединений в качестве сетевых.

Информация об узле

arch Вывод информации об архитектуре машины.
dnsdomainname Отобразить DNS-имя домена.
domainname Отобразить NIS-имя домена.
free Вывести информацию по использованию памяти.
host Вывести информацию об узле и часовом поясе.
hostname Установка и отображение имени узла.
nslookup Опрос серверов доменных имен Интернета.
uname Вывод информации об узле.

Установка

cpio Копирование архивов файлов.
install Копирование файлов для последующего доступа к ним пользователей.
rdist Передача файлов удаленной системе.
tar Копирование файлов на архивный носитель или восстановление файлов с носителя.

Почта

fetchmail Получение почты с удаленного сервера.
formail Преобразование ввода в формат почтового сообщения.
mailq Вывод краткой информации по очереди почтовых сообщений.
makemap Обновление карты баз данных **sendmail**.
rmail Работа с почтой **uucp**.
sendmail Отправка и получение почты.

Управление файловыми системами

В системах Unix *файловая система* – это некое устройство (такое как жесткий диск, дискета или компакт-диск), отформатированное для хранения файлов. Файловые системы могут располагаться на дисках, дискетах, компакт-дисках и на любом другом носителе, к которому возможен произвольный доступ.

Конкретный формат и способ хранения файлов не столь важны; система обеспечивает общий интерфейс для всех *типов файловых систем*, с которыми умеет работать. В Linux существует собственная файловая система – Second Extended Filesystem, или *ext2fs*, которую вы, вероятно, и используете для хранения файлов. Вторая расширенная файловая система (Second Extended Filesystem), созданная специально для Linux, поддерживает имена файлов длиной до 256 символов, размер отдельного файла – до четырех терабайт и другие полезные функции. (Система называется «второй», поскольку

разработана на основе расширенной файловой системы.) Среди прочих часто встречающихся типов файловых систем можно назвать файловую систему MS-DOS, позволяющую из Linux обращаться к файлам, расположенным на DOS-разделах и дискетах, а также файловую систему ISO 9660, используемую на компакт-дисках.

debugfs Отладка файловой системы **ext2fs**.

dosfsck Проверка и устранение ошибок файловых систем DOS и VFAT.

dumpe2fs Вывод информации о суперблоке и группе блоков.

e2fsck Проверка и устранение ошибок **ext2fs**.

fdformat Форматирование гибкого диска.

fsck Проверка и устранение ошибок файловой системы.

fsck.minix Проверка и устранение ошибок файловой системы MINIX.

fuser Перечисление процессов, использующих файловую систему.

mke2fs Создание новой файловой системы типа **ext2fs**.

mkfs Создание новой файловой системы.

mkfs.ext2 Синоним **mke2fs**.

mkfs.minix Создание новой файловой системы MINIX.

mklost+found Создание каталога *lost+found*.

mkraid Настройка RAID-устройства.

mkswap Инициализация раздела подкачки.

mount Монтирование файловой системы.

raidstart Включение RAID-устройства.

raidstop Выключение RAID-устройства.

rdev Описание или изменение значений корневой файловой системы.

rootflags Перечисление или установка параметров корневой файловой системы, используемых при монтировании.

showmount Перечислить экспортированные каталоги.

swapdev Отобразить или установить параметры устройства подкачки.

swapoff Завершить использование устройства подкачки.

swapon Начать использование устройства подкачки.

sync Записать данные из буферов файловой системы на диск.

tune2fs Настройка **ext2fs**.

umount Отмонтирование файловой системы.

Управление ядром

depmod Создать список зависимостей модулей.

insmod Загрузить новый модуль ядра.

lsmod Перечислить загруженные модули.

modprobe Загрузить новый модуль и все модули, необходимые для его работы.

rmmod Удалить модуль.

Работа с сетями

dip Установить dial-up IP-соединения.

gdc Управление демоном маршрутизации **gated**.

ifconfig Настройка сетевых интерфейсов.

ipchains Администрирование брандмауэра (ядра ветви 2.2).

iptables Администрирование брандмауэра (ядра ветви 2.4).

named Перевод доменных имен в IP-адреса (и обратно).

netstat Вывести состояние сетевых интерфейсов.

portmap Связывание демонов с портами.

rarp Управление таблицами RARP.

route Управление таблицами маршрутизации.

routed Динамическое обновление таблиц маршрутизации.

rpcinfo Отображение информации RPC.

ruptime Информация о времени непрерывной работы удаленной системы.

rwho Сведения о пользователях, работающих с удаленной системой.

systat Показать состояние удаленных систем.

traceroute Отслеживание сетевого маршрута к удаленному узлу.

Администрирование NIS

domainname Задать или отобразить имя текущего NIS-домена.

makedbm Пересборка баз данных NIS.

ypbind Соединение с NIS-сервером.

ypcat Вывести значения из базы данных NIS.

ypchfn Изменить информацию о пользователе (для finger, электронной почты и т. д.) в базе данных NIS.

ypchsh Изменение интерпретатора команд по умолчанию (для пользователя) в базе данных NIS.

ypinit Создание новых баз данных NIS.

ypmatch Вывести значения по указанным ключам NIS.

yppasswd Изменить пароль пользователя в базе данных NIS.

yppasswdd Обновление базы данных NIS в связи с использованием **yppasswd**.

yppoll Определение версии карты NIS на NIS-сервере.

yppush Распространение карты NIS.

ypserv Демон NIS-сервера.

ypset Указание **ypbind** на определенный сервер.
ypwhich Отобразить имя NIS-сервера или владельца карты.
ypxfr Передача базы данных NIS с сервера на локальный узел.

Печать

lpc Управление печатающим устройством.
tunelp Настройка параметров печати.

Безопасность и целостность системы

badblocks Поиск дефектных секторов на диске.
chroot Смена корневого каталога.

Запуск и останов системы

bootpd Демон протокола загрузки по Интернету.
bootpgw Шлюз протокола загрузки по Интернету.
bootptest Проверка работы **bootpd**.
halt Останов или выключение системы.
init Изменение режима работы системы.
reboot Останов и перезагрузка системы.
runlevel Отображение режима работы (runlevel) системы.
shutdown Выключение системы.
telinit Изменение текущего режима работы.
uptime Отображение времени непрерывной работы локальных машин.

Активность системы и управление процессами

Некоторые дополнительные команды, описанные в главе 3 «Команды Linux», зачастую оказываются весьма полезными для управления процессами. В частности, это команды **kill**, **killall**, **killall5**, **pidof**, **ps** и **who**.

fuser Перечисление процессов, использующих определенный файл или файловую систему.
psupdate Обновление */boot/psupdate*.
renice Изменение приоритета выполняемых процессов.
top Вывод сведений об использовании процессами системных ресурсов.

Работа с пользователями

chpasswd Изменение нескольких паролей.
groupadd Создание новой группы.

groupdel Удаление группы.
groupmod Изменение групп.
grpck Проверка целостности системных файлов групп.
grpconv Преобразование файла групп в теневой файл групп.
lastlog Генерация отчета о предыдущих сеансах работы пользователей.
newusers Пакетное добавление новых пользователей.
pwck Проверка целостности системных файлов паролей.
pwconv Преобразование файла паролей в теневой файл паролей.
rusers Информация по удаленным машинам (аналогична информации, выдаваемой программой **who**).
rwall Отправка сообщения удаленным пользователям.
useradd Добавить нового пользователя.
userdel Удаление пользователя и его домашнего каталога.
usermod Изменение информации о пользователе.
w Перечислить пользователей, работающих с системой в текущий момент.
wall Посылка сообщения всем пользователям.
whoami Информация о пользователе текущего сеанса работы.

Разное

cron Выполнение команд в заданное время.
dmesg Просмотр сообщений, выданных системой при загрузке.
ldconfig Обновление информации о связях библиотек и их расположении.
logger Отправка сообщений регистратору системной активности.
logrotate Сжатие и вращение системных журналов.
rstat Отобразить информацию о состоянии *удаленной* системы.
run-parts Выполнить все файлы сценариев из определенного каталога.

Обзор работы с сетями

Сети объединяют компьютеры таким образом, чтобы различные системы могли обмениваться информацией. Системы Unix традиционно предоставляют пользователям и администраторам набор простых, но очень удобных сетевых сервисов, которые позволяют проверить состояние системы по сети, просмотреть файлы, находящиеся на другом узле, общаться посредством электронной почты и т. д.

Чтобы обеспечить работу большинства программ в сети, на каждой системе должен непрерывно работать обслуживающий фоновый процесс, реагирующий на запросы клиентов. Такой процесс называется *демоном*. Вот примеры демонов, обеспечивающих основные возможности вашей Linux-системы:

named (переводит численные IP-адреса в адреса буквенно-цифровые, которые так нравятся людям), **lpd** (отправляет документы на печать, возможно, и по сети), **ftpd** (позволяет соединяться с другими узлами по протоколу **ftp**).

Большая часть сетевых команд Unix основывается на протоколах Интернета (Internet Protocols, IP). Протоколы являются стандартизированными способами обмена информацией по сети и разбиты на иерархические слои. Действия протоколов простираются от адресации и маршрутизации пакетов (на относительно низком уровне) до поиска пользователей и выполнения пользовательских команд (на более высоких уровнях).

Основные пользовательские команды, поддерживаемые большинством систем через интернет-протоколы, обычно называются командами TCP/IP (имя, унаследованное от двух наиболее распространенных сетевых протоколов). Все эти команды можно использовать для связи с любыми Unix-системами, а не только с Linux. Многие из команд пригодны также и для работы с не-Unix-системами, поскольку протоколы TCP/IP поддерживаются множеством систем.

Кроме того, в этом разделе описываются системы NFS и NIS, позволяющие производить прозрачный обмен файлами и информацией в сетях, а также **sendmail**.

Команды TCP/IP-администрирования

ftpd Сервер передачи файлов.

gated Управление межсетевыми таблицами маршрутизации.

host Вывод информации об узле и часовом поясе.

ifconfig Настройка параметров сетевых интерфейсов.

named Перевод доменных имен в IP-адреса (и обратно).

netstat Вывод состояния сетевых соединений.

nslookup Опрос серверов доменных имен.

ping Проверить, работает ли удаленный узел и отвечает ли на запросы.

pppd Создание последовательного PPP-соединения.

rdate Уведомление сервера времени об изменении времени.

route Управление таблицами маршрутизации.

routed Динамическое обновление таблиц маршрутизации.

slattach Использование последовательных соединений в качестве сетевых интерфейсов.

telnetd Сервер Telnet-сеансов с удаленных узлов.

tfptd Сервер для организации ограниченной передачи файлов.

Команды администрирования NFS и NIS

domainname Задать или отобразить имя текущего NIS-домена.

makedbm Пересборка баз данных NIS.

portmap Связывание портов DARPA с программами RPC.
rpcinfo Отчет RPC.
ypbind Соединение с NIS-сервером.
ypcat Вывод значений из базы данных NIS.
ypinit Создание новых баз данных NIS.
ypmatch Вывод значений по указанным ключам NIS.
yppasswd Изменение пароля пользователя в базе данных NIS.
yppasswdd Обновление базы данных NIS в связи с использованием **yppasswd**.
yppoll Определение версии карты NIS на NIS-сервере.
yppush Распространение карты NIS.
ypserv Демон NIS-сервера.
ypset Указание **ypbind** на определенный сервер.
ypwhich Отобразить имя NIS-сервера или владельца карты.
ypxfr Передача базы данных NIS с сервера на локальный узел.

Обзор TCP/IP

TCP/IP – это набор протоколов связи, определяющих способы общения компьютеров различных типов. Название происходит от имен двух наиболее распространенных протоколов: протокола управления передачей (TCP, Transmission Control Protocol) и межсетевого протокола (Internet Protocol). Межсетевой протокол обеспечивает движение данных между узлами: он разбивает информацию на пакеты, которые затем отправляются по сети целевому узлу. Использование протокола управления передачей (TCP) гарантирует, что пакеты, достигнув целевого узла, будут собраны в правильном порядке, и что любые отсутствующие пакеты будут посылаться повторно до корректного их получения. Кроме того, в набор TCP/IP входят:

Address Resolution Protocol, ARP (Протокол преобразования адресов)

Обеспечивает перевод между адресами Интернет и локальными аппаратными адресами (адаптеров Ethernet и прочих).

Internet Control Message Protocol, ICMP (Протокол управляющих сообщений)

Обеспечивает управление сообщениями об ошибках и контроль соединений.

Point-to-Point Protocol, PPP (Протокол соединения «точка-точка»)

Обеспечивает работу протокола TCP/IP (и других) через последовательные синхронные и асинхронные соединения.

Reverse Address Resolution Protocol, RARP (Протокол обратного преобразования адресов)

Обеспечивает перевод локальных аппаратных адресов в адреса Интернет (обратно ARP).

Serial Line Internet Protocol, SLIP (Межсетевой протокол последовательных соединений)

Обеспечивает работу IP через последовательные соединения.

Simple Mail Transport Protocol, SMTP (Простой протокол передачи почты)

Протокол, используемый **sendmail** для отправки почты по протоколам TCP/IP.

Simple Network Management Protocol, SNMP (Простой протокол управления сетью)

Обеспечивает функции управления распределенными сетями по протоколам TCP/IP.

User Datagramm Protocol, UDP (Протокол пользовательских дейтаграмм)

Обеспечивает передачу данных без надежности доставки, присущей TCP.

Фундаментальное описание TCP/IP можно найти в трехтомном труде Дугласа Комера (Douglas R. Comer) «TCP/IP и Сети» («Internetworking with TCP/IP»), издательство Prentice-Hall. Команды этой и следующей глав более подробно описаны в следующих двух книгах издательства O'Reilly & Associates: Крэг Хант (Craig Hunt) «Администрирование TCP/IP сетей» («TCP/IP Network Administration», Second Edition) и Олаф Кирх (Olaf Kirch), Терри Доусон (Terry Dawson) «Руководство сетевого администратора Linux» («Linux Networking Administration Guide»).

В архитектуре протоколов TCP/IP данные при посылке в сеть передаются вниз по уровням (к уровню доступа к сети), а при поступлении из сети – вверх к уровню приложения (рис. 2.1).

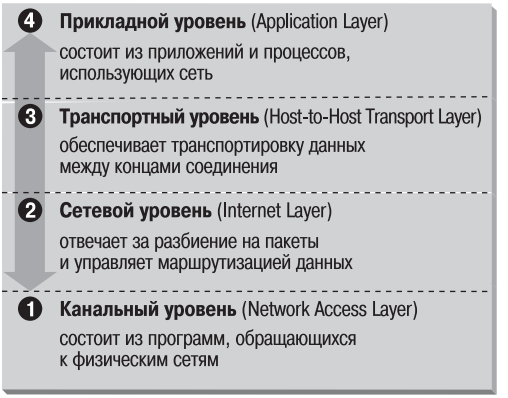


Рис. 2.1. Уровни архитектуры протокола TCP/IP

IP-адресация

IP-адрес (интернет-адрес) является 32-битным двоичным числом, которое отличает вашу машину от всех остальных в сети. У каждой машины должен быть уникальный IP-адрес, состоящий из двух частей – сетевой и машинной. Количество бит адреса, используемых для определения сети и узла, варьируется в зависимости от класса адреса. Существуют адреса трех основных классов: A, B и C (рис. 2.2). Крайние слева биты определяют, к какому классу относится адрес.

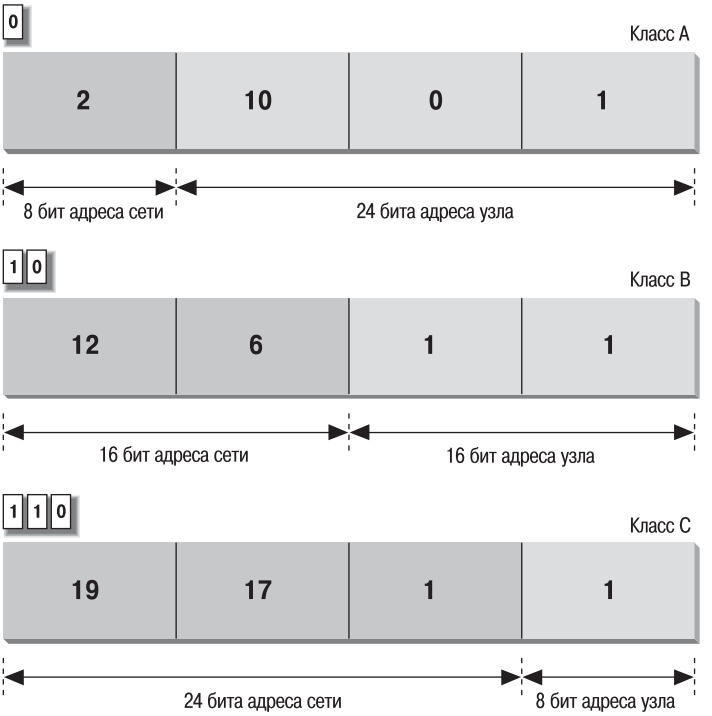


Рис. 2.2. Структура IP-адреса

Более современный стандарт под названием Бесклассовая междоменная маршрутизация (Classless Inter-Domain Routing, CIDR) развивает идею системы классов, состоящую в использовании первых битов (а именно, первый байт адреса) для идентификации адресата пакетов. В стандарте CIDR новый домен может иметь любое количество (не обязательно кратное 8) закрепленных, крайних слева бит.

Еще один новый стандарт, IPv6, изменяет метод адресации и увеличивает количество полей IP-адреса, однако должно пройти какое-то время, чтобы стандарт получил широкое распространение.

При желании подключиться к сети Интернет следует связаться с Информационным центром Сети (Network Information Center) с целью выделения вам адреса. Если вы не подключены к какой-либо внешней сети, можно выбрать любой сетевой адрес, соответствующий стандарту задания адресов. Следует использовать специальные зарезервированные адреса, описанные в RFC 1597. В этом документе перечислены сетевые IP-адреса частных сетей, которые не нужно регистрировать в организации IANA (Internet Assigned Numbers Authority). IP-адрес отличается от Ethernet-адреса, который назначается производителем сетевого адаптера.

Шлюзы и маршрутизация

Шлюзы являются узлами, отвечающими за обмен информацией о маршрутах и передачу данных из одной сети в другую. Каждая подсеть, имеющая собственное независимое управление, называется автономной системой (АС). Автономные системы соединяются с другими посредством внешних шлюзов. АС также может состоять из более мелких частей, соединяемых посредством внутренних шлюзов.

Протоколы шлюзов

Существуют следующие протоколы шлюзов:

EGP (Exterior Gateway Protocol) – протокол внешнего шлюза

BGP (Border Gateway Protocol) – протокол пограничного шлюза

Протоколы обмена данными для внешних шлюзов.

RIP (Routing Information Protocol) – протокол данных маршрутизации

Протокол внутреннего шлюза; особенно распространен в локальных сетях.

Hello protocol

OSPF (Open Shortest Path First) – протокол выбора кратчайшего пути

Протоколы внутренних шлюзов.

Демоны маршрутизации

Демоны маршрутизации – **gated** и **routed** – могут быть запущены на произвольном узле в целях установки шлюза. Их использование взаимоисключающе. **gated** является демоном шлюзовой маршрутизации и позволяет узлу функционировать в качестве как внешнего, так и внутреннего шлюза. Он упрощает настройку маршрутизации, объединяя протоколы RIP, Hello, BGP, EGP и OSPF в один пакет.

routed – демон сетевой маршрутизации, использующий протокол RIP. Этот демон управляет таблицами маршрутизации и позволяет узлу работать только в качестве внутреннего шлюза. Более подробно **gated** и **routed** описаны в главе 3.

Таблицы маршрутизации

Таблицы маршрутизации хранят информацию, необходимую для отправки пакетов адресатам. Эта информация включает данные о сети адресата, задействованном шлюзе, состоянии маршрутов и количестве отправленных пакетов. Таблицы маршрутизации можно просмотреть при помощи команды **netstat**.

Служба имен

Каждый узел в сети имеет имя, по которому можно найти информацию об узле. Имя узла можно присвоить любому устройству, имеющему IP-адрес. Служба имен преобразует легкие для запоминания имена в IP-адреса (числа, с которыми работают машины).

DNS и BIND

Система доменных имен (Domain Name System, DNS) – это распределенная база данных, в которой хранится информация об узлах сети. Структура базы данных схожа со структурой файловой системы Unix, т. е. напоминает дерево, растущее ветвями вниз. Ветви дерева называются *доменами* (или *поддоменами*) и соответствуют IP-адресам. Самой распространенной программной реализацией DNS является BIND (Berkeley Internet Name Domain).

DNS работает по модели клиент-сервер. *Resolver* – это программа-клиент, запрашивающая информацию об узле. Сервер имен (*name server*) представляет собой процесс, реагирующий на запросы. Сервером BIND является демон **named**. Вы можете в диалоговом режиме запрашивать у серверов имен информацию об узлах с помощью команды **nslookup**. Более подробно **named** и **nslookup** описаны в главе 3.

В качестве сервера доменных имен ваша машина будет ответственна за хранение (и предоставление по запросу) данных по именам машин в вашем домене. Именно сюда будут передавать запросы по этим именам все остальные серверы имен в сети.

Доменные имена

Полное имя домена представляет собой последовательность имен, начиная с имени текущего домена и заканчивая корнем, при этом имена разделяются точками. Например, *oreilly.com* содержит имя домена *oreilly* (от O'Reilly & Associates), который расположен в домене *com* (коммерческий). Одна из машин в этом домене имеет имя *www.oreilly.com*. Перечислим домены верхнего уровня:

com – коммерческие организации

edu – образовательные организации

gov – правительственные организации

mil – военные службы

net – коммерческие организации, деятельность которых связана с Интернетом (как правило, поставщики интернет-услуг)

org – различные организации

Каждая страна также имеет домен верхнего уровня.

Настройка TCP/IP

ifconfig

Сетевой интерфейс представляет собой способ, которым сетевое программное обеспечение использует аппаратное обеспечение, а именно: драйвер, IP-адрес и т. п. Для настройки сетевого интерфейса воспользуйтесь командой **ifconfig**. С ее помощью во время загрузки можно присвоить адрес сетевому интерфейсу, а также установить маску сети, широковещательный адрес и IP-адрес. Кроме того, можно установить параметры сетевого интерфейса, включая применение ARP, использование привязанного к драйверу отладочного кода, работу в монопакетном режиме, а также адрес сетевого интерфейса на другом конце соединения точка-точка. Более подробно применение **ifconfig** описано в главе 3.

Связь через последовательное соединение

Существует два протокола для связи по последовательным линиям: межсетевой протокол для последовательного канала (SLIP) и протокол точка-точка (PPP). Эти протоколы позволяют передавать информацию через последовательный порт вместо сетевого адаптера и по последовательному кабелю вместо кабеля Ethernet.

В Linux драйвер протокола SLIP входит в состав ядра. Для перевода последовательной линии в SLIP-режим используют команду **slattach** (подробнее о **slattach** смотрите в главе 3). Следует помнить, что после перевода линии в SLIP-режим по-прежнему необходимо выполнить команду **ifconfig** для настройки сетевого интерфейса. Например, если ваша машина с именем **tanuki** соединяется с **ruby**:

```
# ifconfig s10 tanuki pointopoint ruby
# route add ruby
# route add default gw ruby
```

Теперь интерфейс настроен на связь с **ruby** по протоколу точка-точка, добавлен маршрут к **ruby** и этот маршрут сделан маршрутом по умолчанию, определяя тем самым узел **ruby** в качестве шлюза.

Создатели протокола PPP намеревались избежать некоторых недостатков SLIP. Посредством PPP возможна передача пакетов по не-интернет протоколам, в PPP реализована авторизация клиента и обнаружение/исправление ошибок, PPP динамически конфигурирует задействованные сетевые интерфейсы. В Linux PPP существует и в качестве драйвера ядра, и в качестве демона **pppd**. Подробно **pppd** описан в главе 3.

Диагностирование TCP/IP

Для диагностирования протоколов TCP/IP можно использовать следующие команды. Подробнее эти команды рассмотрены в главе 3.

ifconfig

Предоставляет информацию об основных настройках сетевого интерфейса.

netstat

Показать состояние сетевых соединений.

ping

Проверить, доступен ли удаленный узел.

nslookup

Опрос службы имен DNS.

traceroute

Отследить маршрут пакетов, отправленных определенному адресату.

Брандмауэры и сокрытие адресов

Компьютер-брандмауэр – это система безопасности, расположенная между внутренней и внешней (т. е. Интернетом) сетями. Ее настройка осуществляется при помощи набора правил, определяющих, что можно пропустить, а что необходимо отсечь. Несмотря на то что брандмауэр обычно используется как средство защиты сети от умышленной или случайной атаки извне, его можно применять и для фильтрации пакетов, покидающих внутреннюю сеть. Являясь единственной точкой входа во внутреннюю сеть, брандмауэр значительно упрощает обеспечение безопасности и наблюдение за сетью.

Брандмауэр можно настроить таким образом, чтобы для внешнего мира он представлял собой единственный IP-адрес, хотя для внутреннего пользования может использоваться произвольное количество IP-адресов. Эта возможность известна как *сокрытие адресов* (*masquerading*). Сокрытие адресов дает несколько преимуществ, включая и то, что оно дополнительно защищает сеть, скрывая самый факт ее существования. Кроме того, это избавляет от издержек на получение многочисленных IP-адресов.

Все, сказанное про **iptables,** применимо к ядрам Linux ветви 2.4. Во время написания этой книги как **iptables**, так и ветвь ядра 2.4 все еще были в процессе разработки. Конечная реализация может немного отличаться от описанной здесь. Более подробно вопрос рассмотрен в книге издательства O'Reilly «Руководство сетевого администратора Linux» («Linux Network Administrator's Guide») Олафа Кирха (Olaf Kirch) и Терри Доусона (Terry Dawson), а также в документе «Linux IPTABLES-HOWTO». Этот HOWTO-документ, как и многие другие, можно получить с любого веб-сайта Проекта документирования Linux (см. предисловие).

Как брандмауэры, так и сокрытие адресов реализуются в ядре Linux ветви 2.2 при помощи утилиты **ipchains**, а в ядре ветви 2.4 – при помощи **iptables**. В ядре ветви 2.0 использовалась команда **ipfwadm**, которая описана в разделе, посвященном командам для старых систем, но здесь она опущена. Две более новые команды очень похожи, но отличаются организацией используемых правил. Возможности создания брандмауэров, встроенные в ядра ветви 2.4, задумывались как расширяемые. Если какая-то возможность отсутствует в текущей реализации, ее можно добавить. В документе «Linux netfilter Hacking HOWTO» подробно описано, как это сделать.

В большинстве дистрибутивов поддержка брандмауэров уже встроена в ядро, но если в вашем дистрибутиве это не так, вам придется пересобрать ядро, чтобы добавить поддержку брандмауэров. Для этого необходимо выполнить **make config** для ядра 2.2 и выбрать следующие параметры:

- Network firewalls
- TCP/IP networking
- IP: firewalling

Для того чтобы включить поддержку брандмауэром прозрачного прокси-сервера, следует выбрать следующий параметр:

- IP: transparent proxy support

Для поддержки сокрытия адресов нужно выбрать также параметры:

- IP: masquerading
- IP: ICMP masquerading

Для ядер ветви 2.4 необходимо выбрать параметры:

- Network packet filtering (замена **ipchains**)
- IP tables support (требуется для filtering/masq/NAT)
- Packet filtering

Существует несколько расширенных модулей целей и наборов правил, которые также могут пригодиться. Поведение этих расширенных модулей описано вместе с командой **iptables**. При наличии работающего брандмауэра на ядре ветви 2.2 или 2.0 можно добавить поддержку этих более старых команд и использовать их с новым ядром вместо более современного стиля фильтрации пакетов, присущего **iptables**.

Брандмауэр предоставляет набор встроенных правил, или цепей (chains), на соответствие которым проверяется каждый сетевой пакет. В ядрах ветви 2.4 эти цепи, помимо прочего, сводятся в таблицы, в которых функции фильтрации отделены от функций сокрытия адресов и искажения пакетов. В любом ядре, если пакет соответствует правилу, увеличивается соответствующий счетчик и выполняется действие, связанное с правилом. Пакет может быть принят, отвергнут, подвергнут сокрытию адреса или даже передан в другую цепь для обработки. Более подробные принципы работы цепей в **iptables** и **ipchains** можно найти в описании этих команд.

В дополнение к существующим цепям можно определить и свои собственные. Можно создать специальную цепь для PPP-интерфейсов или для пакетов из определенного узла. Для активации пользовательской цепи необходимо сделать ее действием, привязанным к правилу.

Это можно сделать посредством цепи без применения каких-либо правил, связанных с действиями. Если пакет не подпадает ни под одно из правил цепи, определенной пользователем, управление возвращается цепи, из которой произошел вызов, и проверяется следующее правило в этой цепи. Если ни одно из правил встроенной цепи не применяется к пакету, используется поведение по умолчанию для этой цепи. Поведение по умолчанию может заключаться в специальных действиях, совершаемых над пакетом. Допустимые действия для каждой команды подробно описаны в разделе команд.

В ядрах ветви 2.2 для определения правил используется **ipchains**. Когда правила определены, можно использовать команду **ipchains-save** для создания файла, содержащего определения правил, и **ipchains-restore** – для восстановления определений после перезагрузки системы. Эквивалентная команда для ядер ветви 2.4 – **iptables**. Во время написания этой книги команды **iptables-save** и **iptables-restore** еще находились в стадии разработки, но должны работать подобно их **ipchains**-аналогам.

Более подробно о принятии решений и соображениях, влияющих на составление правил, можно прочесть в литературе по брандмауэрам, например в книге издательства O'Reilly «Создание защиты в Интернете»[1] («Building Internet Firewalls»), написанной Элизабет Цвики (Elizabeth D. Zwicky), Саймоном Купером (Simon Cooper) и Брентом Чепменом (D. Brent Chapman). Подробности о командах **ipchains** и **iptables** можно найти во втором издании «Руководства сетевого администратора Linux» («Linux Network Administrator's Guide») Олафа Кирха (Olaf Kirch) и Терри Доусона (Terry Dawson) или в одном из тематических документов HOW-TO, например в «Linux IPCHAINS HOW-TO» или «Linux IPTABLES HOW-TO».

Обзор NFS

Сетевая файловая система (Network File System, NFS) – это распределенная файловая система, предоставляющая пользователям возможность монтировать удаленные файловые системы так, как если бы они были локальными. NFS работает по модели клиент-сервер: сервер экспортирует разделяемые каталоги, а клиенты монтируют их, чтобы получить доступ к файлам в каталогах. NFS делает ненужным хранение копий одних и тех же файлов на разных машинах, позволяя всем клиентам разделять единственную копию файла на сервере. NFS является протоколом прикладного уровня на основе RPC. Более подробную информацию об архитектуре сетевых протоколов смотрите ранее в этой главе в разделе «Обзор TCP/IP».

---

[1] «Создание защиты в Интернете», издательство «Символ-Плюс», IV кв. 2001 г.

Администрирование NFS

Установка клиентов и серверов NFS включает запуск демонов NFS, экспортирующих файловые системы на серверах NFS и монтирующих эти системы для клиентов. Файлом настроек NFS является */etc/exports*; он контролирует, какие файлы и каталоги экспортируются и какие виды доступа к ним разрешены. Имена и адреса клиентов, пользующихся службой NFS, содержатся в файле */etc/hosts*.

Демоны

Демоны сервера NFS, называемые *nfsd-демоны*, работают на сервере и принимают RPC-запросы от клиентов. Серверы NFS также используют демон **mountd** для обслуживания запросов на монтирование. На стороне клиента кэширование и буферизация реализованы посредством **biod** – демона блочного ввода/вывода. Демон **portmap** ставит в соответствие номера RPC-программ и номера TCP/IP-портов.

Экспортирование файловых систем

Прежде чем устанавливать NFS-сервер, проверьте наличие доступа к вашему узлу у узлов-клиентов. Затем отредактируйте файл */etc/exports*, включив в него точку монтирования экспортируемой файловой системы. При использовании **mountd** файлы будут экспортироваться в соответствии с правами доступа, описанными в */etc/exports*.

Монтирование файловых систем

После запуска NFS клиент может монтировать удаленную файловую систему либо используя команду **mount**, либо определив удаленные файловые системы, монтируемые по умолчанию, в файле */etc/fstab*. На запрос **mount** реагирует демон сервера **mountd**, проверяя права доступа клиента и возвращая указатель на файловую систему. Будучи смонтированным, каталог остается таковым до тех пор, пока его не размонтируют с помощью **umount** либо локальная система не будет перезагружена.

Как правило, только привилегированный пользователь может монтировать файловые системы с помощью NFS. Однако можно разрешить и другим пользователям монтировать и размонтировать отдельные файловые системы при помощи команд **mount** и **umount**, установив для этих систем параметр **user** в файле */etc/fstab*. Это может снизить объем передаваемой информации за счет того, что файловые системы монтируются только при необходимости. Чтобы обеспечить возможность монтирования файловых систем пользователем, создайте для каждой из них соответствующую запись в */etc/fstab*.

Обзор NIS

Network Information System (NIS) – это служба, известная ранее как Желтые страницы Sun (Sun Yellow Pages, YP) и используемая для синхронизации настроек большого количества машин в сети. Для этого один узел назначается *хозяином*, владеющим всеми системными файлами и базами данных и распространяющим эту информацию всем узлам сети. Информация хранится в специальных базах данных, называемых *картами* (*maps*). NIS работает на основе протокола RPC. В настоящее время существует два бесплатных NIS-сервера для Linux – **yps** и **ypserv**.

Серверы

В NIS существует два типа серверов – хозяин (master) и подчиненный (slave). Сервер-хозяин отвечает за работу с картами и распределение их по подчиненным серверам. После распределения файлы становятся локально-доступными по соответствующим запросам.

Домены

Домен NIS – это группа узлов, которые пользуются одним и тем же набором карт. Карты хранятся в подкаталоге */var/yp*, имеющем такое же имя, что и домен. Машины одного домена разделяют пароль, информацию об узлах и групповые права владения файлами. Доменные имена NIS устанавливаются с помощью команды **domainname**.

Карты NIS

В NIS информация хранится в базах данных, называемых картами. Каждая карта состоит из пары **dbm**-файлов базы данных, первый из которых содержит иерархию ключей (бит-карту индексов), а второй – конкретные значения полей данных. Нетекстовая (не-ASCII) структура **dbm**-файлов делает необходимым использование специальных инструментов NIS, например **yppush** для переноса карт между машинами.

Файл */var/yp/YP_MAP_X_LATE* содержит полный список активных карт NIS, а также псевдонимов для NIS-карт. Здесь должны быть перечислены все карты, которые обслуживаются NIS.

Инструменты для работы с картами

Для работы с картами NIS используются следующие инструменты:

makedbm

Создание **dbm**-файлов. Изменениям подвергаются только карта *ypservers* и любые дополнительные (не по умолчанию) карты.

ypinit

Сборка и установка баз данных NIS. Управление картами при инициализации NIS. Не рекомендуется использование при уже работающей системе NIS.

yppush

Распределение обновленных карт с сервера-хозяина.

Администрирование NIS

Для использования NIS необходимо установить NIS-клиенты и NIS-серверы. Описание, приводимое ниже, относится к установке NIS с использованием сервера **ypserv**, который не поддерживает конфигурацию хозяин/подчиненный. Все команды NIS находятся в прямой зависимости от RPC-программы **portmap**, которая должна быть установлена и запущена при установке и настройке NIS.

Установка NIS-сервера

Установка NIS-сервера сводится к следующим шагам:

1. Установка доменного имени для NIS с помощью команды **domainname**.
2. Редактирование файла *ypMakefile*, определяющего, какие базы данных собирать и какие источники при этом использовать.
3. Копирование файла *ypMakefile* в */var/yp/Makefile*.
4. Выполнение **make** в каталоге */var/yp*, т. е. сборка баз данных и инициализация сервера.
5. Запуск **ypserv** – демона сервера NIS.

Установка NIS-клиента

Установка клиента NIS сводится к установке доменного имени NIS с помощью **domainname** (имя должно совпадать с доменным именем для NIS-сервера) с последующим запуском **ypbind**.

Учетные записи пользователей NIS

В сетях NIS существует два вида учетных записей пользователей: распределяемые и локальные. Распределяемые учетные записи должны обслуживаться на узле-хозяине; они содержат информацию, одинаковую для всех узлов домена NIS. Изменения в распределяемых учетных записях клиенты получают в виде карт NIS. Локальные учетные записи изменяются непосредственно на локальных машинах, т. е. представляют собой информацию, уникальную для отдельного узла. Карты NIS не влияют на локальные настройки, как и изменения локальных учетных записей не ведут к измене-

нию карт. При установке NIS существующие учетные записи автоматически становятся локальными.

RPC и XDR

Удаленный вызов процедур (Remote Procedure Call, RPC) является протоколом работы, используемым как NFS, так и NIS. Он позволяет узлу делать запросы, которые с виду являются локальными, но выполняются на удаленном узле в сети. RPC реализован как библиотека функций плюс сетевой стандарт порядка следования байтов и структур данных, называемый XDR (eXternal Data Representation, представление внешних данных).

Глава 3

Команды Linux

В данной главе представлены пользовательские команды, а также команды программирования и системного администрирования. Имя каждой из команд вынесено в отдельную колонку. В первой строке рассматривается применение (синтаксис) команды, затем следует ее краткое описание и перечисление всех возможных параметров. Для многих команд описание завершается примерами использования. Если требуется лишь вспомнить, как на практике работать с командой либо узнать о конкретных применениях, можно сразу перейти к примерам.

Типографские соглашения в описаниях синтаксиса команд перечислены в предисловии к книге. Для более эффективного поиска команд можно воспользоваться указателем в конце книги.

Мы старались как можно более аккуратно составлять перечни параметров команд. Большая часть информации и параметров команд должна быть верна; однако существует много дистрибутивов Linux и версий включаемых в них инструментов. Время от времени новые параметры добавляются, а старые изымаются из обращения. Поэтому возможны различия между параметрами, описанными здесь, и доступными в конкретной системе. В случае подобных несоответствий загляните в страницы руководства. Для большинства команд доступен параметр **––help**, позволяющий получить краткую справку. (Если такого параметра нет, обычно выдается сообщение об ошибке – «invalid option» – и справка по использованию команды.)

Традиционно однобуквенным параметрам команд предшествует один дефис (**–d**). Более новые правила позволяют указывать длинные имена параметров, предваряемые парой дефисов (**––debug**). Очень часто возможности команды могут быть задействованы с использованием и того и другого стиля.

Алфавитный перечень команд

agetty [*options*] *port baudrate* [*term*]

agetty

Команда системного администрирования. Linux-версия команды **getty**. Устанавливает тип терминала, режим, скорость и характеристики линии. Запускается демоном **init**. Это второй из процессов в последовательности **init-getty-login-shell**, которая в конечном итоге соединяет пользователя с системой. **agetty** считывает имя пользователя и выполняет команду **login** с этим именем в качестве аргумента. В процессе считывания имени **agetty** пытается адаптировать систему к скорости и типу используемого устройства.

Необходимо указать порт, поиск которого **agetty** производит в каталоге */dev*. Если указан символ «–», то **agetty** производит чтение со стандартного ввода. Необходимо также указывать скорость линии *baudrate*, которая может быть задана списком скоростей, перебираемых **agetty**. Также возможно указание типа терминала *term*, которое для **agetty** имеет больший приоритет, чем значение переменной окружения TERM.

Параметры

–h Установить аппаратный, а не программный контроль передачи.

–i Подавить вывод содержимого файла */etc/issue* перед выдачей приглашения входа в систему.

–l *program*

Использовать указанную программу вместо */bin/login*.

–m Автоматический подбор подходящей скорости линии.

–t *timeout*

Предписание **agetty** завершать работу, если произошло удачное соединение **(open)**, но ответ на приглашение входа (login prompt) не получен в течение *timeout* секунд.

–L Не отслеживать наличие несущего сигнала в линии. Это подразумевает только локальное обслуживание. Параметр используется для подключения терминалов.

apmd

apmd [*options*]

Команда системного администрирования. **apmd** обрабатывает события, о которых сообщает драйвер расширенного управления питанием BIOS (Advanced Power Management BIOS driver). Драйвер передает сведения об уровне зарядки батарей и запросы на переход в режим пониженного энергопотребления. **apmd** заносит всю получаемую информацию в системный журнал посредством демона **syslogd** и надлежащим образом обрабатывает запросы на переход в режим пониженного энергопотребления или приостановки. Поведение **apmd** можно подстроить, указав команду **apmd_proxy**, которая выполняется при наступлении определенного события.

Параметры

–c *n*, **––check** *n*

Установить интервал ожидания события в секундах перед повторной проверкой уровня питания. По умолчанию происходит бесконечное ожидание. Установка этого параметра обеспечивает более частую проверку уровня зарядки батарей.

–P *command*, **––apmd_proxy** *command*

Выполнить указанную команду при получении событий от драйвера APM. Обычно команда является сценарием интерпретатора. Команда *command* выполняется с аргументами, определяющими, какое именно событие произошло. Аргументы перечислены ниже.

–p *n*, **––percentage** *n*

Заносить в журнал информацию об изменении уровня питания на *n* процентов. По умолчанию $n = 5$. При $n > 100$ подобные изменения не фиксируются.

–V, **––version**

Вывести номер версии и завершить работу.

–v, **––verbose**

Заносить в журнал информацию обо всех событиях.

–W, **––wall**

Уведомить пользователей об опасной разрядке батарей.

–w *n*, **––warn** *n*

Заносить в журнал предупреждение, если заряд батарей опускается ниже *n* процентов. По умолчанию $n = 10$. При отрицательных значениях предупреждения в журнал не заносятся.

–q, ––quiet

Отключить предупреждения о низком уровне зарядки батарей.

–?, ––help

Вывести справку по **apmd** и завершить работу.

Передаваемые аргументы

Команда обработки события **(apmd_proxy)** выполняется со следующими аргументами:

start

Вызывается при запуске демона.

stop

Вызывается при остановке демона.

suspend [system | user]

Запускается, когда получен запрос на приостановку работы системы. Второй параметр позволяет определить происхождение запроса.

standby [system | user]

Запускается, когда получен запрос на перевод в режим пониженного энергопотребления. Второй параметр позволяет определить происхождение запроса.

resume [suspend | standby | critical]

Запускается, когда система восстанавливается для продолжения работы. Второй параметр определяет режим работы, в котором находилась система до этого события. Приостановка работы по **critical** означает аварийное выключение системы. Система после такого выключения может быть нестабильной; **resume** выполняется для продолжения работы в этом случае.

change power

Выполняется при изменении источника питания.

change battery

Запускается, когда драйвер APM BIOS уведомляет о низком уровне зарядки батарей.

change capability

Запускается, когда драйвер APM BIOS уведомляет об изменениях в аппаратной части компьютера, которые могут повлиять на производительность.

apropos	**apropos** *string* ... Поиск вхождений перечисленных *строк* в записи базы данных программы **whatis** и выдача результатов на стандартный вывод. Действует подобно **whatis**, за тем исключением, что производит поиск строк, а не отдельных слов. Команда эквивалентна **man –k**.
ar	**ar** [**–V**] *key* [*args*] [*posname*] *archive* [*files*] Сохранение группы файлов (*files*) в файловом архиве (*archive*). Наиболее часто применяется для создания или обновления библиотечных файлов, используемых редактором связей (**ld**). Допускается указание только одного ключевого символа, но каждый из символов можно сочетать с дополнительными аргументами *args* (без пробелов между символами и аргументами). *posname* – это имя файла в архиве (*archive*). При перемещении или замене файлов (*files*) их можно указывать как перед файлом *posname*, так и после него. Параметр **–V** отображает номер версии **ar** в стандартный поток ошибок. **Ключ** **d** Удалить файлы (*files*) из архива (*archive*). **m** Переместить файлы в конец архива. **p** Распечатать файлы (*files*) из архива (*archive*). **q** Добавить файлы к архиву. **r** Заменить файлы в архиве. **t** Отобразить содержимое архива или список указанных файлов. **x** Извлечение содержимого архива или только указанных файлов. **Аргументы** **a** Используется с ключами **r** или **m** для помещения новых файлов в архив после файла *posname*. **b** То же, что **a**, но файлы помещаются перед файлом *posname*. **c** Создание архива (*archive*) без отображения соответствующей информации. **f** Укоротить длинные имена файлов. **i** Идентично **b**.

ar

l Аргумент существует для поддержания обратной совместимости; его применение в Linux бессмысленно.

o Сохранять исходную информацию о времени создания файлов.

s Принудительная перегенерация таблиц архивных имен (рекомендуется после выполнения команды **strip**).

S Запретить перегенерацию таблицы имен.

u Выполняется с ключом **r** для обновления только тех файлов, которые были изменены после создания архива.

v Подробная диагностика; отображение описаний всех действий **ar**.

Пример

Заменить **mylib.a** объектными файлами из текущего каталога:

```
ar r mylib.a `ls *.o`
```

arch

arch Выдать тип архитектуры компьютера на стандартный вывод. Команда эквивалентна **uname –m**.

arp

arp [*options*]

Команда TCP/IP. Очистить, дополнить или отобразить содержимое ARP-кэша ядра (*/proc/net/arp*).

Параметры

–v Диагностика выполняемых действий.

–t *type*

Поиск записей для *type* в ARP-кэше. *type* имеет значение либо **ether** (Ethernet), либо **ax25** (пакетное радио AX.25); по умолчанию принимается значение **ether**.

–a [*hosts*]

Показать записи для узлов *hosts* либо все записи, если узел не указан.

–d *host*

Удалить запись для узла *host*.

arp	**–s** *host hardware-address* Добавить запись вида *host hardware-address*, где адрес **ether**-класса представлен шестью шестнадцатеричными значениями, разделяемыми двоеточиями. **–f** *file* Добавить в кэш записи из указанного файла.
as	**as** [*options*] *files* Создание объектного файла для каждого из перечисленных файлов (*files*) с исходными текстами на ассемблере. Объектные файлы получают те же имена, что и исходные, но суффикс **.s** заменяется на **.o**. **as** может иметь не описанные ниже системно-зависимые параметры. **Параметры** **––** [| *files*] Читать исходные файлы со стандартного ввода или из файлов (*files*) при использовании конвейера. **–a[dhlns][=***file***]** Одиночный параметр **–a** вызывает отображение исходного кода, ассемблерного листинга и таблицы имен. Прочие параметры задают дополнения к этому отображению или его сокращение: **–ad** Опускать директивы отладки. **–ah** Включить исходный текст на языке высокого уровня, если таковой имеется в наличии. **–al** Включить ассемблерный листинг. **–an** Подавить обработку форм. **–as** Включить таблицу имен. **=***file* Ассемблерный текст в файле *file*. **–defsym** *symbol=value* Определить значение имени *symbol*. Значение *value* должно быть целым. **–f** Пропустить действия препроцессора.

as

−−gstabs

Генерировать отладочную информацию.

−o *objfile*

Поместить результат в объектный файл *objfile* (по умолчанию этот файл имеет имя *file.o*).

−v Отобразить информацию о номере версии ассемблера.

−I *path*

Включить *path* в пути поиска директив **.include**.

−K Выдавать предупреждение перед изменением таблиц приращений.

−L Не удалять локальные имена, начинающиеся с **L**.

−R Сочетать данные и текст в текстовом разделе.

−W «Тихий» режим.

at

at [*options*] *time*

Выполнение команд в заданное время (*time*) и в определенный день (*date*). Команды поступают со стандартного ввода или из файла. (См. также **batch**.) Последовательность команд завершается символом *EOF*. Время (*time*) может задаваться как числами (часы, минуты и модификаторы), так и ключевыми словами. Может содержать необязательную дату (*date*) в виде месяца, числа и дня недели или специального ключевого слова (*today* или *tomorrow*). Также может быть задано приращение. Ниже даны более подробные описания параметров.

Команда **at** всегда доступна привилегированному пользователю. Прочие пользователи должны быть перечислены в файле */etc/at.allow*, если таковой существует; в противном случае они не должны быть упомянуты в файле */etc/at.deny*. Если не существует ни один из файлов, командой может пользоваться только привилегированный пользователь.

Параметры

−c *job* [*job*...]

Выдать перечисленные задания на стандартный вывод. Этому параметру не требуется задание времени.

−d *job* [*job*...]

Удалить указанные задания. Действует эквивалентно **atrm**.

at

–f *file*

Читать задания из *file*, а не со стандартного ввода.

–l Получить список всех заданий, имеющихся в расписании пользователя, применяющего команду. Эквивалентно **atq**.

–m Отправить пользователю почтовое уведомление, когда выполнение задания завершится, независимо от результатов завершения.

–q *letter*

Поместить задание в очередь, имеющую буквенный идентификатор (*letter*). *letter* – это буква из диапазона [a–z] или [A–Z]. Очередь заданий по умолчанию имеет идентификатор **a** (для очереди пакетов умолчанием является **b**). Приоритет выполнения очередей заданий понижается с увеличением старшинства букв.

–V Выдать номер версии программы.

Время

hh:mm [*modifiers*]

Час может быть представлен одной или двумя цифрами (по умолчанию принимается 24-часовой стандарт отображения времени); необязательные минуты также могут представляться одной или двумя цифрами; двоеточие может быть опущено, если время представлено как *h*, *hh* (только часы) или *hhmm* (часы и минуты, по две обязательных цифры на каждое поле). Таким образом, правильными являются следующие варианты: 5, 5:30, 0530, 19:45. При добавлении модификатора **am** или **pm** время представляется в 12-часовом формате. При добавлении ключевого слова **zulu** время соответствует среднему времени по Гринвичу (Greenwich Mean Time).

midnight | **noon** | **teatime** | **now**

Перечисленные ключевые слова могут заменять цифровое представление времени. **teatime** приравнивается к 4 часам дня; после **now** должно быть обязательно указано приращение (*increment*).

Дата

month num[, *year*]

Месяц (*month*) представляет собой название одного из двенадцати месяцев года, полное или сокращенное до первых трех букв; *num* – это календарное число месяца;

at

год (*year*) задается четырьмя цифрами. Если месяц, на который запланировано выполнение задания, уже прошел, **at** переносит выполнение на соответствующий месяц следующего года.

day

Название одного из семи дней недели, полное или сокращенное до первых трех букв.

today | **tomorrow**

Ключевые слова, указывающие на текущий и следующий дни. Если дата (*date*) не задана, **at** планирует задание как **today**, когда указанное время наступает позже текущего времени, иначе задание переносится на следующий день (**tomorrow**).

Приращение

Можно задать численное приращение, если необходимо, чтобы задание было выполнено через определенное время после текущего времени или текущей даты. Число должно предшествовать ключевым словам **minute**, **hour**, **day**, **week**, **month** или **year** (и их множественным формам). Ключевое слово **next** является синонимом приращения **+ 1**.

Примеры

Заметьте, что первые две команды эквивалентны:

```
at 1945 pm December 9
at 7:45pm Dec 9
at 3 am Saturday
at now + 5 hours
at noon next day
```

atq

atq [*options*]

Выдача списка не отработавших задач пользователя. Если команда выполняется привилегированным пользователем, выдается список задач всех пользователей системы. Эквивалентно **at –l**.

Параметры

–q *queue*

Получить список из определенной очереди заданий, очереди с другими идентификаторами игнорировать.

–v Выдать список заданий, уже отработавших, но еще не удаленных из очереди.

–V Выдать номер версии программы.

atrm

atrm [*options*] [*job*] [*job*...]

Удаление задания из очереди. Эквивалентно **at –d**.

Параметры

–q *queue*

Удалить задание из указанной очереди.

–V Выдать номер версии программы и завершить работу.

badblocks

badblocks [*options*] *device block-count*

Команда системного администрирования. Поиск сбойных блоков на устройстве *device*. Необходимо указать количество блоков на устройстве (*block-count*).

Параметры

–b *blocksize*

Блоки имеют размер *blocksize* байт.

–o *file*

Перенаправить вывод в указанный файл.

–v Диагностика работы.

–w Проверка каждого блока посредством записи и последующего чтения данных.

banner

banner [*options*] [*characters*]

Выдача символов (*characters*) плакатного размера на стандартный вывод. Если параметр *characters* отсутствует, **banner** запрашивает символы для печати и выполняет чтение со стандартного ввода. По умолчанию результаты отправляются на стандартный вывод, но предназначены для печати на принтере.

Параметр

–w *width*

Установить ширину печатаемых символов в *width* знакомест. Следует помнить, что строчные символы будут *у*же. Если не задан параметр **–w**, ширина символа принимается равной 132. Если параметр **–w** задан, но не задана ширина, то она принимается равной 80.

Пример

```
/usr/games/banner -w50 Happy Birthday! |lpr
```

basename *name* [*suffix*]
basename *option*

basename

Удаление имен каталогов из пути файла. Если задан и совпадает с окончанием имени параметр *suffix*, он также удаляется. Результат выводится на стандартный вывод.

Параметры

––help

Отображение справки по использованию команды и завершение работы.

––version

Отображение номера версии программы **basename** и завершение работы.

Примеры

```
% basename /usr/lib/libm.a
libm.a

% basename /usr/lib/libm.a  .a
libm
```

---

bash [*options*] [*file* [*arguments*]]
sh [*options*] [*file* [*arguments*]]

bash

Стандартный командный интерпретатор Linux, который принимает все прочие команды. Подробно **bash** описан в главе 7 «bash: интерпретатор команд Борна».

---

batch [*options*] [*time*]

batch

Выполнение команд, поступающих со стандартного ввода. Если не указано время, команды выполняются, когда позволит загрузка системы (т. е. при уровне ниже 0,8). По существу, программа почти аналогична **at**, за тем исключением, что не требует обязательного указания времени. Подробнее см. команду **at**.

Параметры

–f *file*

Читать команды из файла, а не со стандартного ввода.

–m Отправить пользователю почтовое уведомление по завершении выполнения задания независимо от результатов завершения.

batch	**–q** *letter* Поместить задание в очередь, имеющую буквенный идентификатор (*letter*). *letter* – это буква из диапазона [a–z] или [A–Z]. Очередь заданий по умолчанию имеет идентификатор **a** (для очереди пакетов умолчанием является **b**). Приоритет выполнения очередей заданий понижается с увеличением старшинства букв. **–V** Выдача номера версии и завершение работы. **–v** Отображение времени, когда будет выполняться задание.
bc	**bc** [*options*] [*files*] **bc** является языком (и компилятором), синтаксис которого похож на синтаксис C. **bc** предназначен для выполнения арифметических вычислений с любой точностью. Элементами языка являются идентификаторы, ключевые слова и символы, которые кратко описываются ниже. В конце описания приведены примеры. Выполнение вычислений с произвольной точностью или перевод чисел в различные системы счисления происходит в диалоговом режиме. Данные могут читаться из файла (*file*) или поступать со стандартного ввода. Для выхода из **bc** наберите **quit** или **EOF**. **Параметры** **–l, ––mathlib** Сделать доступными функции из математической библиотеки. **–s, ––standard** Отключить все расширения и работать в стандарте POSIX. **–w, ––warn** Выдавать предупреждение при использовании расширений POSIX **bc**-стандарта. **–q, ––quiet** Не выводить приветствие при начале работы. **–v, ––version** Вывести номер версии.

Идентификаторы

Идентификаторы представляют собой последовательности из одного или более символов. Идентификатор должен начинаться со строчной буквы, но может содержать также цифры и символы подчеркивания. Прописные буквы запрещены. Идентификаторы используются в качестве имен переменных, массивов и функций. Обычно в переменных хранятся числа произвольной точности. В одной и той же программе у переменной, массива и функции может быть одинаковый идентификатор. Так, следующие идентификаторы не противоречат друг другу:

x Переменная *x*.

x[*i*] Элемент массива *x* с номером *i*. *i* лежит в интервале от 0 до 2047 и может являться выражением.

x(*y*,*z*)

Вызов функции *x* с параметрами *y* и *z*.

Ключевые слова ввода–вывода

Ключевые слова **ibase**, **obase**, **scale** и **last** имеют значения. Текущие значения отображаются при простом вводе этих ключевых слов. Текущие значения могут быть изменены присваиванием. Буквы A–F трактуются как цифры с диапазоном значений 10–15.

ibase = *n*

Вводимые (например, вручную) числа считаются представленными в системе счисления с основанием *n* (по умолчанию *n* = 10).

obase = *n*

Отображаемые числа представлены в системе счисления с основанием *n* (по умолчанию *n* = 10). Примечание: для того чтобы вернуть параметрам **ibase** и **obase** значения по умолчанию, используйте цифру A.

scale = *n*

Отображать результаты вычислений, ограничивая длину дробной части до *n* десятичных цифр (значение по умолчанию равно 0, т. е. результат округляется до целого числа). Ключевое слово **scale** обычно используется только для вычислений в десятичной системе.

last

Выдать значение последнего напечатанного числа.

bc

Ключевые слова программных операторов

Операторы отделяются друг от друга точкой с запятой или новой строкой. Фигурные скобки необходимы для объединения нескольких операторов.

if (*rel-expr*) {*statements*} [**else** {*statements*}]

Выполнить один или несколько операторов (*statements*), если выражение *rel-expr* истинно. В противном случае не производить никаких действий либо, если задано расширение **else**, выполнять альтернативные операторы. Например:

```
if (x==y) {i = i + 1} else {i = i - 1}
```

while (*rel-expr*) {*statements*}

Повторять один или более операторов (*statements*), пока условие *rel-expr* истинно. Например:

```
while (i>0) {p = p*n; q = a / b; i = i -1}
```

for (*expr1*; *rel-expr*; *expr2*) {*statements*}

Оператор схож с оператором **while**. Например, чтобы напечатать десять чисел, кратных пяти, начиная с 5, можно ввести:

```
for (i = 1; i <= 10; i++)  i*5
```

В GNU-версии **bc** оператор **for** может иметь меньше трех аргументов. Если опущен первый или третий аргумент, это означает, что данное выражение не вычисляется. Отсутствующий второй аргумент всегда принимает значение 1.

break

Прерывает выполнение оператора **while** или **for**.

print *list*

Расширение GNU-версии. Оператор обеспечивает альтернативное средство вывода. Список (*list*) представляет собой последовательность строк и выражений, разделенных запятыми. **print** отображает элементы списка в порядке следования и не выводит символ новой строки по завершении печати. Каждое выражение вычисляется, результат выводится на печать, а затем специальная переменная *last* принимает значение этого выражения. Строки просто выводятся на печать (и могут содержать специальные символы, т. е. начинающиеся с символа «\»). Специальные символы:

a звуковой сигнал

b забой (backspace)

f новая страница (form feed)

n новая строка (newline)

r возврат каретки (carriage return)

q двойные кавычки (double quote)

t табуляция (tab)

\ обратный слэш (backslash)

continue

GNU-расширение. Немедленный переход к следующей итерации в операторе **for**.

halt

GNU-расширение. Немедленное прекращение работы **bc**.

limits

GNU-расширение. Выдать информацию об ограничении рабочих диапазонов (для данной версии **bc**).

Ключевые слова функций

define *f*(*args*) {

Начало определения функции *f*, принимающей аргументы *args*. Аргументы разделяются запятыми. На следующих строках записываются программные операторы. Определение завершается символом «}».

auto *x*, *y*

Определить переменные *x* и *y* как локальные для функции и не имеющие смысла за ее пределами, инициализировать их значением 0. Ключевое слово должно предшествовать всем прочим.

return(*expr*)

Вернуть значение выражения *expr* в программу. Функция **return** возвращает 0, если выражение отсутствует. Используется при определении функций.

sqrt(*expr*)

Вычислить квадратный корень выражения *expr*.

length(*expr*)

Определить количество значащих цифр в выражении *expr*.

bc

scale(*expr*)

То же, что и **length**, но в расчет принимаются только цифры после десятичной запятой.

read()

GNU-расширение. Прочитать число со стандартного ввода. Возвращаемое значение равно введенному числу, преобразованному в соответствии со значением **ibase**.

Функции математической библиотеки

Перечисленные ниже функции доступны лишь при запуске **bc** с параметром **–l**. Подключение математической библиотеки устанавливает значение **scale** в 20.

s(*angle*)

Вычислить синус угла (*angle*); константа или выражение в радианах.

c(*angle*)

Вычислить косинус угла (*angle*); константа или выражение в радианах.

a(*n*)

Вычислить арктангенс *n* и вернуть угол в радианах.

e(*expr*)

Возвести **e** в степень выражения (*expr*).

l(*expr*)

Вычислить натуральный логарифм выражения (*expr*).

j(*n*, *x*)

Вычислить функцию Бесселя от целого порядка *n*.

Операторы

Состоят из символов операторов и операндов. Операторы делятся на арифметические, одноместные, операторы присваивания и операторы отношений.

арифметические

+ – * / % ^

одноместные

– ++ ––

присваивания

=+ =– =* =/ =% =^ =

отношений

< <= > >= == !=

Прочие символы

/* */

Границы комментария.

( ) Круглые скобки служат для контроля вычисления выражений, в частности, для изменения приоритета операций. Кроме того, оператор присваивания может заключаться в круглые скобки, что вызывает печать результата.

{} Фигурные скобки используются для группирования операторов.

[] Квадратные скобки содержат индекс при обращении к массиву.

"*text*"

Двойные кавычки используются для печати текста (*text*).

Примеры

Обратите внимание, что при вводе числа (в цифрах или выражении) оно вычисляется и печатается, однако при присваивании никакого вывода не происходит.

`ibase = 8`	*Ввод в восьмеричной системе счисления*
`20`	*«Вычислить» это восьмеричное число*
`16`	*На терминале отображается десятичный эквивалент*
`obase = 2`	*Отображать числа в двоичной системе, а не в десятичной*
`20`	*Восьмеричный ввод*
`10000`	*На терминале отображается двоичный эквивалент*
`ibase = A`	*Вернуться к вводу в десятичной системе*
`scale = 3`	*Усечение результата до трех цифр после запятой*
`8/7`	*Вычислить результат деления*
`1.001001000`	*Однако! А кто будет за результирующей системой счисления следить?*
`obase = 10`	*Сейчас ввод происходит в десятичной системе счисления, так что можно обойтись без A.*
`8/7`	
`1.142`	*На терминале отображается результат, усеченный до 3 знаков после запятой.*

bc

Ниже показано использование функций:

`define p(r, n) {`	*Функция p принимает два аргумента*
`auto v`	*Переменная v является локальной для функции*
`v = r^n`	*r возводится в степень n*
`return (v)}`	*Возврат значения*
`scale = 5`	
`x=p(2.5, 2)`	*x = 2,5^2*
`x`	*Вывести значение x*
`6.25`	
`length(x)`	*Число цифр*
`3`	
`scale(x)`	*Число десятичных цифр после запятой*
`2`	

biff

biff [*arguments*]

Уведомление пользователя о пришедшей почте и имени отправителя. **biff** работает асинхронно. Уведомление действует, только если в системе запущен сервер **comsat(8)**. Команда **biff y** включает режим уведомления, а **biff n** отключает его. Команда без аргументов информирует пользователя о своем текущем статусе.

bison

bison [*options*] *file*

Построение таблиц для последующего грамматического разбора на основе файла (*file*), содержащего описание контекстно-независимой грамматики. Результат записывается в файл *file.c*. Этот инструмент был назван по аналогии с **yacc** и очень во многом совместим с ним. Имена всех исходных файлов должны содержать суффикс *.y*, а имена получаемых файлов – исходный префикс имени. Все длинные имена параметров (предваряемые обычно символами «––») могут предваряться символом «+».

Параметры

–b *prefix*, **––file-prefix=***prefix*

Для всех получаемых файлов использовать префикс *prefix*.

–d, **––defines**

Создать файл *file.h*, содержащий директивы **#define**, связывающие коды лексем **bison** с именами лексем, определяемыми пользователем.

bison

–r, ––raw

В файле *file.h* использовать номера лексем, а не **yacc**-совместимые преобразования.

–k, ––token-table

Включить в файл *file.c* имена лексем и значения YYNTOKENS, YYNNTS, YYNRULES и YYNSTATES.

–l, ––no-lines

Исключить директивы **#line** из кода в *file.c* (обычно применяется после завершения отладки).

–n, ––no-parser

Подавить выдачу результирующего кода грамматического разбора, оставив только объявления. Собрать все результаты преобразований в теле оператора switch и сохранить оператор в файле *file.act*.

–o *file*, **––output-file=***file*

Выдача результата в указанный файл.

–p *prefix*, **––name-prefix=***prefix*

Заменить префикс (*prefix*) на **yy** во всех внешних именах.

–t, ––debug

Скомпилировать код отладки времени выполнения.

–v, ––verbose

Режим детальной диагностики. Сообщения записываются в файл *file.output*.

–V, ––version

Отобразить номер версии **bison**.

–y, ––yacc, ––fixed-output-files

Следовать принятым в **yacc** соглашениям по именованию получаемых файлов.

bootpd [*options*] [*configfile* [*dumpfile*]]

bootpd

Команда TCP/IP. Сервер протокола загрузки через Интернет (Internet Boot Protocol). **bootpd** обычно выполняется */etc/inetd* включением следующей строки в файл */etc/inetd.conf*:

```
bootps dgram udp wait root /etc/bootpd bootpd
```

bootpd

Благодаря этому **bootpd** запускается, только когда приходит соответствующий запрос на загрузку. Сервер также может быть запущен в автономном режиме из командной строки. При старте **bootpd** читает файл настройки */etc/bootptab* (или файл *configfile*, имя которого указывается в качестве аргумента при вызове), а затем переходит к ожиданию пакетов BOOTREQUEST.

В файле */etc/services* **bootpd** читает номер порта, который необходимо обслуживать. Извлекаются две записи: **bootps** – порт-приемник сервера **bootp** и **bootpc** – порт, предназначенный для ответа на запросы клиентов.

Если **bootpd** собран с параметром –DDEBUG, получение сигнала SIGUSR1 приводит к записи постоянно находящейся в памяти базы данных в файл */etc/bootpd.dump* или в указанный в командной строке *dumpfile*.

Параметры

–c *directory*

Рабочий каталог для **bootpd**.

–d *level*

Уровень отладки. Если значение *level* опущено, происходит приращение уровня отладки на 1.

–t *timeout*

Определить интервал ожидания в минутах. При нулевом значении ожидание длится бесконечно.

Файл настройки

В каждой записи файла настройки **bootpd** содержатся двухсимвольные, чувствительные к регистру символов теги параметров узла. Определения тегов разделяются двоеточиями. Общий формат записи следующий:

```
hostname:tg=value:tg=value:tg=value
```

где *hostname* – имя узла-клиента bootp, а *tg* – тег, получающий значение *value*. Список существующих тегов приведен ниже.

Теги

bf Файл загрузки.

bs Размер файла загрузки в блоках по 512 октетов.

cs Список адресов сервера файлов cookie.

ds Список адресов сервера доменных имен (DNS).

bootpd

gw Список адресов шлюзов.

ha Аппаратный адрес узла.

hd Домашний каталог файла загрузки.

hn Имя узла для посылки.

ht Тип аппаратного обеспечения узла (см. RFC Assigned Numbers – назначение номеров).

im Внедрить список адресов сервера.

ip IP-адрес узла.

lg Список адресов log-сервера.

lp Список адресов **lpr**-сервера.

ns Список адресов сервера имен IEN-116.

rl Список адресов сервера протокола обнаружения ресурсов.

sm Маска подсети узла.

tc Продолжение таблицы.

to Разница во времени с UTC (в секундах).

ts Список адресов сервера времени.

vm Волшебный селектор cookie производителей.

Существует также общий тег **T***n*, в котором *n* является номером производителя (по записи производителя из RFC 1048). Общие данные могут представляться либо потоком шестнадцатеричных чисел, либо строками ASCII-символов, заключенными в кавычки.

bootpgw

bootpgw [*options*] *server*

Шлюз протокола загрузки через Интернет (Internet Boot Protocol Gateway). Поддерживает шлюз, направляющий серверу *server* запросы от **bootpd**. Обрабатывает пакеты BOOTREPLY, а также пакеты BOOTREQUEST. **bootpgw** обычно запускается */etc/inetd* при включении в */etc/inetd.conf* следующей строки:

```
bootps dgram udp wait root /etc/bootpgw bootpgw
```

При этом **bootpgw** запускается только при поступлении запроса на загрузку. **bootpgw** опознает те же параметры, что и **bootpd**, кроме **–c**.

bootptest	**bootptest** [*options*] *server* [*template*] Команда TCP/IP. Проверка работы демона **bootpd** сервера *server* путем посылки запросов каждую секунду в течение 10 секунд либо до отклика сервера. Если указан файл *template*, производится чтение настроек из этого файла. **Параметры** **–f** *file* Прочитать имя файла загрузки из указанного файла. **–h** Опознавать клиента по аппаратному, а не по IP-адресу. **–m** *magic-number* Сделать число *magic-number* первым полем записи информации о производителе.
bzip2	**bzip2** [*options*] *filenames* **bunzip2** [*options*] *filenames* **bzcat** [*option*] *filenames* **bzip2recover** *filenames* Утилита сжатия и распаковки файлов, аналогичная **gzip**, но в ней для получения лучшего сжатия реализованы более совершенные алгоритмы и методы кодирования. **bzip2** заменяет каждый из перечисленных файлов сжатой версией, добавляя расширение *.bz2*. **bunzip2** производит распаковку файлов, сжатых **bzip2** (все прочие файлы игнорируются). **bzcat** распаковывает файлы на стандартный вывод, а **bzip2recover** используется для попытки восстановления данных из поврежденных упакованных файлов. **Параметры** **––** Все последующие параметры считаются именами файлов. **–***dig* Размер блока при сжатии – *dig* × 100 Кбайт, где *dig* – цифра в интервале от 1 до 9. **–c, ––stdout** Сжатие или распаковка с записью полученных данных на стандартный вывод. **–d, ––decompress** Принудительная распаковка.

bzip2

–f, ––force

Принудительная перезапись конечных файлов. По умолчанию перезапись не происходит. Параметр также является указанием разрывать жесткие ссылки на файлы.

–k, ––keep

Не удалять исходные файлы.

–L, ––license, –V, ––version

Отобразить информацию о лицензировании и версии, завершить работу.

–q, ––quiet

Отображать минимум сообщений.

–s, ––small

Жертвовать скоростью, используя при работе меньший объем памяти.

–t, ––test

Проверить целостность файлов, но не выполнять сжатие.

–v, ––verbose

Подробные сообщения. Отображать степень сжатия для каждого обработанного файла. Дополнительные параметры **–v** увеличивают объем отображаемой информации.

–z, ––compress

Принудительное сжатие, даже если программа запущена под именем **bunzip2** или **bzcat**.

––repetetive-fast, ––repetetive-best

Параметр может быть полезен в версиях до 0.9.5 (в которых использовался улучшенный алгоритм сортировки) для получения возможности управлять работой алгоритма.

---

c++ [*options*] *files*

c++

См. **g++**.

---

cal [–*jy*] [[*month*] *year*]

cal

Печать календаря на 12 месяцев (начиная с января) текущего года или календаря на заданный месяц текущего года. Месяц (*month*) может принимать значения от 1 до 12, а год (*year*) – от 1 до 9999. Команда без аргументов выдает календарь на текущий месяц.

cal

Параметры

–j Даты Юлианского календаря (дни нумеруются числами от 1 до 365, начиная с первого января).

–m Отображать понедельник в качестве первого дня недели.

–y Календарь на весь год.

Примеры

```
cal 12 1995
cal 1994 > year_file
```

cardctl

cardctl [*options*] *command*

Команда системного администрирования. Позволяет управлять гнездами PCMCIA и выбирать текущую рабочую схему. Текущая схема, наряду с адресом любой установленной карты, передается сценариям настройки (которые по умолчанию расположены в каталоге */etc/pcmcia*). Команда **scheme** позволяет изменить или отобразить текущую схему. Прочие команды работают с указанным номером гнезда PCMCIA-карты либо, если номер не задан, со всеми гнездами.

Команды

config [*socket*]

Отобразить текущие настройки гнезда.

eject [*socket*]

Подготовить систему к извлечению карты (или нескольких карт).

ident [*socket*]

Отобразить идентификационную информацию для карты.

insert [*socket*]

Уведомить систему об установке карты.

reset [*socket*]

Передать карте сигнал сброса.

resume [*socket*]

Восстановить питание гнезда и переконфигурировать для работы.

scheme [*name*]

Отобразить текущую схему или изменить на указанную (*name*).

status [*socket*]

Показать текущее состояние гнезда.

suspend [*socket*]

Остановить работу устройства и прекратить подачу питания на гнездо.

Параметры

–c *directory*

Искать данные для настройки карты в указанном каталоге вместо */etc/pcmcia*.

–f *file*

Использовать для отслеживания текущей схемы указанный файл, а не */var/run/pcmcia-scheme*.

–s *file*

Искать текущую информацию о гнезде в указанном файле, а не в */var/run/stab*.

cardctl

---

cardmgr [*options*]

cardmgr

Команда системного администрирования, демон карт PCMCIA. **cardmgr** отслеживает гнезда PCMCIA на предмет появления и удаления устройств. Обнаружив карту, демон пытается получить ее идентификатор и настроить ее по базе настроек (которая обычно хранится в */etc/pcmcia/config*). По умолчанию **cardmgr** при обнаружении карты создает запись в системном журнале и выдает звуковой сигнал. Два высоких сигнала означают, что устройство успешно обнаружено и настроено. Один высокий сигнал, за которым следует один низкий, – устройство обнаружено, но настройка прошла неудачно. Один низкий сигнал означает, что демон не смог определить вставленную карту. Информация о сконфигурированных в данный момент времени картах хранится в файле */var/run/stab*.

Параметры

–c *directory*

Искать базу данных настройки карт в указанном каталоге *directory* вместо */etc/pcmcia*.

–d Для загрузки драйвера PCMCIA-устройства использовать **modprobe** вместо **insmod**.

cardmgr	**–f** Во время настройки присутствующих карт выполняться в качестве приоритетной задачи, затем перейти в режим демона. **–m***directory* Искать модули драйверов PCMCIA-устройств в указанном каталоге, а не в */lib/modules/*`**uname –r**`. **–o** Произвести настройку имеющихся в наличии карт, затем завершить работу. **–p***file* Записывать идентификатор процесса **cardmgr** в указанный файл, а не в */var/run/cardmgr.pid*. **–q** Не выдавать звуковые сигналы. **–s***file* Записывать текущую информацию о гнездах в указанный файл, а не */var/run/stab*. **–v** Подробная диагностика работы. **–V** Вывести номер версии программы и завершить работу.
cat	**cat** [*options*] [*files*] Чтение одного или более файлов (*files*) и печать их содержимого на стандартный вывод. Если файлы не заданы или один из файлов задан как «–», происходит чтение со стандартного ввода. Ввод заканчивается символом *EOF*. Можно объединить несколько файлов в один новый при помощи оператора перенаправления > или добавить файлы в конец уже существующего при помощи >>. **Параметры** **–A, ––show-all** То же, что и **–vET**. **–b, ––number-nonblank** Пронумеровать все непустые строки перед выдачей на стандартный вывод. Первая строка имеет номер 1. **–e** То же, что и **–vE**. **–E, ––show-ends** Добавлять символ $ в конец каждой строки. **–n, ––number** Пронумеровать все строки. Первая строка имеет номер 1.

cat

–s, ––squeeze-blank

Сокращать идущие подряд пустые строки до одной пустой строки.

–t То же, что **–vT**.

–T, ––show-tabs

Заменять символы табуляции (TAB) на ^I.

–u Параметр игнорируется, оставлен для совместимости с Unix-версией.

–v, ––show-nonprinting

Отображать управляющие и неотображаемые символы, за исключением символов переноса строки (LINEFEED) и табуляции (TAB).

Примеры

`cat ch1`	*Вывод содержимого файла*
`cat ch1 ch2 ch3 > all`	*Копирование файлов в один*
`cat notes5 >> notes`	*Добавление к файлу*
`cat > temp1`	*Создать файл с терминала, закончить по получении EOF*
`cat > temp2 << STOP`	*Создать файл с терминала, закончить по получении* `STOP`

cc

cc [*options*] *files*

См. **gcc**.

cfdisk

cfdisk [*options*] [*device*]

Команда системного администрирования. Разбиение жесткого диска на разделы. Устройство *device* может принимать значения */dev/hda* (по умолчанию), */dev/hdb*, */dev/sda*, */dev/sdb*, */dev/sdc*, */dev/sdd*. См. также **fdisk**.

Параметры

–a Отмечать текущий раздел положением курсора, а не инвертированием.

–c *cylinders*

Указать количество цилиндров.

cfdisk	**–h** *heads* Указать количество головок. **–s** *sectors* Указать количество секторов в одной дорожке. **–z** Не производить чтение таблицы разделов; производить разбивку «с нуля». **–P** *format* Отображать таблицу разделов в указанном формате: **r** (просто данные), **s** (по секторам) и **t** (обычный формат). **Команды** Курсорные клавиши <**Вверх**> и <**Вниз**>. Навигация по разделам. **b** Переключение флага загрузки (bootable) раздела. **d** Удаление раздела (освобождается место, которое может быть использовано под другие разделы). **g** Изменение геометрии диска. Выдается запрос на изменение количества цилиндров, головок или секторов (**c**, **h** и **s**, соответственно). **h** Справка. **m** Попытка обеспечить максимальное использование дискового пространства раздела. **n** Создать новый раздел. Запрашивается дополнительная информация. **p** Отобразить таблицу разделов. **q** Завершение работы без сохранения изменений. **t** Изменение типа файловой системы на указанный. **u** Изменение единиц измерения раздела (мегабайты, сектора, цилиндры и обратно по кругу). **W** Запись изменений. Обратите внимание, что команда должна быть набрана именно в верхнем регистре.
chattr	**chattr** [*options*] *mode files* Изменение атрибутов файла. Это специфическая команда файловой системы Linux (Second Extended Filesystem). Работает аналогично символьному варианту **chmod** с использованием +, – и =. Режим (*mode*) представляется в виде *операция атрибут*. См. также **lsattr**.

chattr

Параметры

–R Изменять рекурсивно атрибуты каталогов и их содержимое.

–V Отображать состояние атрибутов после их изменения.

–v *version*

Установить версию файла в *version*.

Операции

+ Включить атрибут.

– Выключить атрибут.

= Присвоить атрибуты (выключив те, что не заданы).

Атрибуты

A Не обновлять атрибут времени доступа при изменении файла.

a Разрешить только добавление к содержимому файла. Атрибут может устанавливаться или сниматься только привилегированным пользователем.

c Файл сжат.

d Запретить вывод содержимого программой **dump**.

i Неизменяемый. Атрибут может устанавливаться или сниматься только привилегированным пользователем.

s Безопасное удаление; содержимое при удалении обнуляется.

u Неудаляемый.

S Синхронное обновление.

Пример

`chattr +a myfile` *Команда, выполненная суперпользователем*

chfn

chfn [*options*] [*username*]

Изменение информации, хранимой в файле */etc/passwd* и отображаемой по запросу finger. Если опущены параметры (*options*), **chfn** переходит в диалоговый режим и запрашивает изменения. Чтобы оставить поле пустым, введите ключевое слово **none**. Информацию о другом пользователе может изменять только привилегированный пользователь. Обычные пользователи должны ввести свой пароль перед внесением изменений.

chfn

Параметры

–f, ––full-name

Задать новое полное имя пользователя.

–h, ––home-phone

Задать новый номер телефона.

–o, ––office

Задать новый номер кабинета.

–p, ––office-phone

Задать новый номер рабочего телефона.

–u, ––help

Отобразить справку по использованию команды и завершить работу.

–v, ––version

Выдать номер версии команды и завершить работу.

Пример

```
chfn -f "Ellen Siever" ellen
```

chgrp

chgrp [*options*] *newgroup files*
chgrp [*options*]

Изменить группу владельца файлов *files*. Новая группа-владелец (*newgroup*) задается либо идентификатором группы, либо именем группы, определенным в файле */etc/groups*. Только владелец файлов или привилегированный пользователь могут менять группу-владельца.

Параметры

–c, ––changes

Отобразить информацию об изменяемых файлах.

–f, ––silent, ––quiet

Подавлять сообщения об ошибках, если для каких-либо файлов смена группы не может быть выполнена.

––help

Отобразить справку по использованию команды и завершить работу.

–R, ––recursive

Совершить рекурсивный обход каталогов, внося изменения в групповые права владения.

chgrp

−−reference=*filename*

Изменить группу на ассоциированную с файлом *filename*. Параметр *newgroup* при этом не указывается.

−v, −−verbose

Подробно описать произошедшие изменения.

−−version

Вывести номер версии команды и завершить работу.

chmod

chmod [*options*] *mode files*
chmod [*options*] **−−reference=***filename files*

Изменение прав доступа (*mode*) указанных файлов (*files*). Права доступа для файла может менять только его владелец или привилегированный пользователь. Права доступа (*mode*) могут быть заданы в виде числа или выражения вида *кто операция режим* (*who opcode permission*), где часть *кто* может быть опущена и имеет по умолчанию значение **a**. Единовременно к изменению режима (доступа) можно применять только одну операцию. Допускается задание нескольких параметров *mode*, разделенных запятыми.

Параметры

−c, −−changes

Отобразить информацию о файлах, для которых изменяются права доступа.

−f, −−silent, −−quiet

Не уведомлять пользователя, если **chmod** не может выполнить смену прав доступа для определенного файла.

−−help

Отобразить справку по использованию команды и завершить работу.

−R, −−recursive

Совершить рекурсивный обход каталогов, меняя права доступа.

−−reference=*filename*

Изменить права доступа на права для файла *filename*.

−v, −−verbose

Вывести информацию обо всех рассмотренных файлах, независимо от того, произошли ли изменения прав доступа.

chmod

--version

Выдать номер версии команды и завершить работу.

Кто

u Пользователь.

g Группа.

o Прочие.

a Все (по умолчанию).

Операции

+ Включить режим доступа.

– Отменить режим доступа.

= Присвоение прав (и отмена незаданных режимов).

Права

r Чтение.

w Запись.

x Исполнение.

s Установка идентификатора пользователя (группы).

t Бит «липкости». Сохранять текстовый режим (файла) или предотвращать удаление файлов не владельцем (для каталогов).

u Текущие права пользователя.

g Текущие права группы.

o Текущие права прочих.

Как вариант можно задавать права доступа с помощью трехзначного восьмеричного числа. Первая цифра определяет права для владельца файла, вторая – права для группы, третья – права для всех остальных. Права вычисляются в результате сложения следующих восьмеричных значений:

4 Чтение.

2 Запись.

1 Исполнение.

Примечание: еще одна, четвертая цифра, может предшествовать последовательности. Эта цифра определяет следующие режимы:

4 Создать процесс с правами пользователя, владеющего файлом программы, а не с правами пользователя, запускающего программу.

chmod

2 Создать процесс с правами группы, владеющей файлом программы, а не с правами пользователя, запускающего программу.

1 Установить бит «липкости».

Примеры

Добавить право на исполнение для владельца файла (execute-by-user permission):

```
chmod u+x file
```

Обе следующие команды дают права на чтение, запись и выполнение владельцу (7), права на чтение и выполнение группе (5) и право только на выполнение всем остальным (1) файла (*file*):

```
chmod 751 file
chmod u=rwx, g=rx, o=x file
```

Любая из следующих команд дает всем доступ только для чтения к файлу (*file*):

```
chmod =r file
chmod 444 file
chmod a-wx, a+r file
```

Установить бит смены идентификатора пользователя, присвоить право на чтение, запись и выполнение владельцу, право на чтение и выполнение – группе и всем прочим:

```
chmod 4755 file
```

chown

chown [*options*] *newowner files*
chown [*options*] **––reference=***filename files*

Изменение владельца перечисленных файлов (*files*) на нового владельца (*newowner*). Новый владелец задается числовым идентификатором пользователя или регистрационным именем, которое записано в файле */etc/passwd*. **chown** также допускает форму записи *владелец:группа* или *владелец.группа*. Это позволяет менять одновременно и группу-владельца. Если новый владелец не указан, он не изменяется. Если указан только владелец, но после двоеточия или точки отсутствует новая группа, будет установлена группа нового владельца. Владельца файлов имеет право менять только их текущий владелец или привилегированный пользователь.

chown	**Параметры** **–c, ––changes** Отобразить информацию о файлах, для которых изменяется владелец. **––dereference** Следовать по символическим ссылкам. **–f, ––silent, ––quiet** Подавлять сообщения об ошибках в случае, если владелец какого-либо файла не может быть изменен. **–h, ––no-dereference** Изменить владельца для символической ссылки (если такая возможность существует в системе), но не файла, на который она ссылается. **–v, ––verbose** Вывести информацию обо всех файлах, независимо от того, произошла ли смена их владельца. **–R, ––recursive** Совершить рекурсивный обход каталогов, меняя владельцев для всех файлов. **––reference=***filename* Изменить владельца на владельца файла *filename* вместо явного задания нового владельца. **––help** Отобразить справку по использованию команды и завершить работу. **––version** Выдать номер версии **chown** и завершить работу.
chpasswd	**chpasswd** [*option*] Команда системного администрирования. Изменение паролей пользователей в пакетном режиме. **chpasswd** допускает ввод в форме *username:password* (каждая пара в отдельной строке). Если не указан параметр **–e**, пароль (*password*) будет зашифрован перед сохранением. **Параметр** **–e** Считать поступающие пароли уже зашифрованными.

chroot *newroot* [*command*]

chroot

Команда системного администрирования. Изменить корневой каталог на время выполнения команды *command* либо, если команда не указана, на время выполнения новой копии командного интерпретатора. Команда или интерпретатор выполняется относительно нового корневого каталога. Значение первого символа / в именах заменяется на *newroot* для команды и всех порождаемых процессов. Начальным рабочим каталогом также является *newroot*. Команда доступна только для привилегированного пользователя.

chsh [*options*][*username*]

chsh

Изменение командного интерпретатора, запускаемого при входе в систему, в диалоговом режиме или в командной строке. Если интерпретатор отсутствует в списке файла */etc/shells*, выдается предупреждение. Необходимо указывать полный путь к интерпретатору. Выполнение **chsh** требует ввода пароля пользователя. Только привилегированный пользователь может изменять интерпретатор для других пользователей.

Параметры

–l, ––list-shells

Отобразить существующие имена интерпретаторов, перечисленные в */etc/shells*, и завершить работу.

–s *shell*, **––shell** *shell*

Определить новый интерпретатор, запускаемый при входе в систему.

–u, ––help

Отобразить справку по использованию команды и завершить работу.

–v, ––version

Выдать номер версии **chsh** и завершить работу.

Пример

```
chsh -s /bin/tcsh
```

cksum	**cksum** [*files*] Вычисление контрольной суммы (CRC) для перечисленных файлов (*files*) в целях проверки целостности информации после передачи файла. Если имена файлов опущены или задан параметр «–», происходит чтение со стандартного ввода. Программа отображает результат вычисления, размер каждого файла в байтах и имя файла (в том случае, если чтение производится не со стандартного ввода).
clear	**clear** Очистка терминала.
cmp	**cmp** [*options*] *file1 file2* [*skip1* [*skip2*]] Сравнение файлов *file1* и *file2*. Если имя первого файла – «–» или не указано, производится чтение со стандартного ввода. См. также команды **comm** и **diff**. Файлы могут быть произвольного типа. Параметры *skip1* и *skip2* являются необязательными значениями смещений от начала каждого файла при сравнении. **Параметры** **–c, ––print-chars** Вывести на экран различающиеся байты в виде символов. **–i** *num*, **––ignore-initial=***num* Пропустить первые *num* байт. **–l, ––verbose** Отображать смещения и коды различающихся байтов. **–s, ––quiet, ––silent** Подавлять вывод; ничего не печатать, но возвращать коды завершения: **0** Файлы идентичны. **1** Файлы различаются. **2** Доступ к файлам запрещен. **Пример** Выдать соответствующее сообщение, если файлы идентичны (код возврата 0): `cmp -s old new && echo 'no changes'`

col [*options*] **col**

Фильтр для обработки обратных переносов строк и escape-последовательностей позволяет отображать результаты работы **tbl** или **nroff** в виде, пригодном для чтения с терминала.

Параметры

–b Пропускать символы забоя (backspace); полезно при печати страниц руководства пользователя (manpages).

–f Обрабатывать частичные обратные переносы строки, но не полные обратные переносы. (Обычно частичный перенос строки при вводе отображается на следующей полной строке.)

–l *n*

Буферизация по меньшей мере *n* строк в памяти. Значение по умолчанию составляет 128.

–x Как правило, **col** экономит время, необходимое для печати результата, преобразованием идущих подряд пробелов в символы табуляции. Режим **–x** подавляет такое преобразование.

Примеры

Обработать файл *myfile* утилитами **tbl** и **nroff** и постранично выдать результат на экран после предварительной фильтрации с параметрами **col** и **more**:

```
tbl myfile | nroff | col | more
```

Сохранить man-страницу (manpage) команды **ls** в файле *out.print*, удалив символы забоя (backspace), которые иначе отображались бы как ^H.

```
man ls | col -b > out.print
```

colcrt [*options*] [*files*] **colcrt**

Фильтр для обработки обратных переносов строк и escape-последовательностей, позволяющий отображать результаты работы **tbl** или **nroff** в виде, пригодном для чтения с терминала. Нестрочные символы (например, индексы и степени) и символы подчеркивания (изменяемые на дефисы) выносятся на отдельные строки.

Параметры

– Не подчеркивать.

–2 Удваивать межстрочное расстояние печатью всех полустрок.

colrm	**colrm** [*start* [*stop*]] Удаление заданных колонок текста из файла, причем ширина колонки принимается равной одному символу. Чтение происходит со стандартного ввода, результат выдается на стандартный вывод. Нумерация колонок начинается с 1; удаление начинается с колонки *start* (включительно) и заканчивается колонкой *stop* (включительно). Табуляция увеличивает счетчик колонок до следующего кратного *start* или *stop*; забой (backspace) уменьшает его на 1. **Пример** `colrm 3 5 < test1 > test2`
column	**column** [*options*] [*files*] Форматировать ввод из одного или нескольких файлов (*files*) колонками, заполняя их построчно. Если имя файла опущено, происходит чтение со стандартного ввода. **Параметры** **–c** *num* Определить число (*num*) колонок при форматировании. **–s** *char* Разграничить колонки таблицы символом *char*. Имеет смысл применять только совместно с параметром **–t**. **–t** Форматировать ввод таблицей. Разграничивать элементы пробелами, если не задан альтернативный разграничитель в параметре **–s**. **–x** Поменять порядок заполнения, производить заполнение по колонкам.
comm	**comm** [*options*] *file1 file2* Построчное сравнение отсортированных файлов *file1* и *file2*. Результатом работы программы является таблица из трех колонок: в первой колонке содержатся уникальные строки первого файла, во второй – уникальные строки второго файла, в третью колонку помещаются строки, существующие в обоих файлах. Утилита **comm** схожа с **diff**, т. к. обе они предназначены для сравнения файлов. Но **comm** можно использовать и как **uniq**. Дело в том, что **comm** находит повторяющиеся или уникальные строки в *двух* отсортированных файлах, тогда как **uniq** производит поиск уникальных или повторяющихся строк внутри *одного* отсортированного файла.

comm

Параметры

– Чтение строк со стандартного ввода.

–*num*

Подавление вывода колонки *num*. Можно задавать несколько номеров колонок, не разделяя их пробелами.

––help

Отобразить справку и завершить работу.

––version

Отобразить информацию о версии программы и завершить работу.

Пример

Сравнить два списка лучшей десятки фильмов и отобразить строки, которые встречаются в обоих списках:

```
comm -12 siskel_top10 ebert_top10
```

compress [*options*] *files*

compress

Сжатие одного или нескольких файлов. К именам сжатых файлов добавляется суффикс *.Z*. Если имена файлов не указаны, происходит сжатие информации, поступающей со стандартного ввода. Каждый из перечисленных файлов сжимается отдельно от других. Файлы, являющиеся символическими ссылками, игнорируются. См. также **gzip**.

Параметры

–b *maxbits*

Ограничить максимальное количество бит.

–c Записывать результаты на стандартный вывод, а не в файл с суффиксом *.Z*.

–d Производить распаковку, а не сжатие. Действие идентично вызову **uncompress**.

–f Принудительно создать конечный файл, даже если он уже существует.

–r Выполнять рекурсивную обработку и сжатие содержимого каталогов.

–v Вывести статистику сжатия.

–V Отобразить номер версии и данные о том, как была скомпилирована программа, а затем завершить работу.

cp

cp [*options*] *file1 file2*
cp [*options*] *files directory*

Копирование файла *file1* в *file2* либо копирование одного или нескольких файлов *files* без изменения имен в каталог *directory*. Если конечный файл существует, он перезаписывается; если существует конечный каталог, файл копируется в него (каталог *не* перезаписывается).

Параметры

–a, ––archive

По возможности сохранять атрибуты копируемых файлов. То же, что и **–dpR**.

–b, ––backup

Создавать резервные копии перезаписываемых файлов.

–d, ––no-dereference

Не разыменовывать символические ссылки, сохранять отношения жестких ссылок между исходным файлом и копией.

–f, ––force

Удаление существующего файла перед копированием в него.

–i, ––interactive

Запрашивать подтверждение на перезапись существующих файлов.

–l, ––link

Вместо копий создавать жесткие ссылки для всех файлов (но не каталогов).

–p, ––preserve

Сохранять всю информацию о владельцах, правах доступа и времени доступа к файлам.

–P, ––parents

Сохранять структуру промежуточных каталогов в описании пути. Последним аргументом команды должно быть имя существующего каталога. Например, команда

```
cp --parents jphekman/book/ch1 newdir
```

скопирует файл *jphekman/book/ch1* в */newdir/jphekman/book/ch1*, создав при необходимости промежуточные каталоги.

cp

–r, –R, ––recursive

Рекурсивное копирование каталогов.

–S *backup-suffix*, **––suffix=***backup-suffix*

Установить символ суффикса, добавляемого при создании резервных копий к именам файлов. Суффикс также можно задать с помощью переменной окружения SIMPLE_BACKUP_SUFFIX. Суффикс по умолчанию – «~» (тильда). Для того чтобы отделить суффикс от имени, необходимо явно задать точку (т. е. *.bak*, а не *bak*).

–s, ––symbolic-link

Вместо копий создавать символические ссылки для всех файлов. Для копируемых файлов должен быть задан полный путь.

–u, ––update

Не копировать файл, если конечный файл уже существует и имеет такое же (или более позднее) время изменения.

–v, ––verbose

Выводить имя каждого файла перед копированием.

–V *type*, **––version-control=***type*

Определить тип создаваемых резервных копий. Можно также воспользоваться переменной окружения VERSION_CONTROL. По умолчанию имеет значение **existing**. Возможные аргументы:

t, numbered

Всегда создавать пронумерованные резервные копии.

nil, existing

Создавать нумерованные резервные копии только для файлов, у которых они уже есть, в противном случае создавать обычные резервные копии.

never, simple

Всегда создавать простые резервные копии.

–x, ––one-file-system

Игнорировать подкаталоги, расположенные в других файловых системах.

cpio *flags* [*options*]

cpio

Копирование файловых архивов с ленты (или диска) на ленту (или диск) либо в другое место на локальной машине. Каждый из трех флагов (*flags*) **–i**, **–p** и **–o** имеет свои параметры.

cpio

Флаги

–i, ––extract [*options*] [*patterns*]

Извлечь из архива файлы, имена которых совпадают с заданными шаблонами имен (*patterns*). Каждый шаблон может содержать специальные символы командного интерпретатора Борна. (Шаблон необходимо заключать в кавычки или экранировать, чтобы интерпретатор не пытался обработать его вместо **cpio**.) Если не были заданы шаблоны, извлекаются все файлы. В процессе извлечения существующие файлы не перезаписываются более старыми версиями из архива (обратное достигается заданием параметра **–u**).

–o, ––create [*options*]

Поместить в архив файлы, список имен которых поступил со стандартного ввода.

–p, ––pass-through [*options*] *directory*

Копировать файлы в другой каталог (*directory*) на той же системе. Конечные пути файлов строятся от заданного каталога.

Сравнение допустимых параметров

Параметры, доступные для использования с каждым из флагов **–i**, **–o** и **–p**, перечислены ниже. Символ «–» опущен для упрощения.

```
i:   bcdf mnrtsuv B SVCEHMR IF
o: 0a c         vABL VC HM O F
p: 0a  d lm    uv  L V    R
```

Параметры

–0, ––null

Список файлов завершается нулевым символом, а не символом новой строки, что позволяет производить операции над файлами, в именах которых содержится символ новой строки.

–a, ––reset-access-time

Сбросить (вернуть старое значение) время последнего доступа к файлу после прочтения файла.

–A, ––append

Добавить файлы к существующему архиву (который в этом случае должен являться дисковым файлом). Файл архива задается параметром **–O** или **–F**.

–b, ––swap

Обменивать значения соседних байтов и машинных полуслов в целях межплатформенного преобразования 32-битных целых чисел.

–B Блочный ввод-вывод, размер блока устанавливается равным 5120 байт на одну запись (по умолчанию 512).

––blocksize=*size*

Установить размер блока (blocksize) ввода/вывода равным *size* × 512 байт.

–c Чтение или запись информации из заголовков в виде ASCII-символов. Позволяет преодолеть несовместимость при переносе на машину другой архитектуры.

–C *n*, **––io-size=***n*

Аналогично **–B**, но размер блока может быть равен любому положительному целому числу *n*.

–d, ––make-directories

По необходимости создавать каталоги.

–E *file*, **––pattern-file=***file*

Извлечь из архива файлы с именами из шаблонов в *file*.

–f, ––nonmatching

Изменение правила отбора файлов: обрабатываются только файлы, *не* совпадающие с перечисленными шаблонами *patterns*.

–F *file*, **––file=***file*

Использовать в качестве архива файл *file*, вместо стандартных потоков ввода **(stdin)** или вывода **(stdout)**. Файл может находиться на другой машине, тогда его имя представляется в форме *user@hostname:file* (часть *user@* не обязательна).

––force-local

Считать, что файл (заданный параметром **–F**, **–I** или **–O**) находится на локальной машине, даже если в имени файла присутствует символ «:».

–H *type*, **––format=***type*

Использовать формат *type*. По умолчанию для извлекаемых файлов принимается формат **bin**. Допустимые форматы:

bin Исполняемый файл.

odc Старый (POSIX.1) переносимый формат.

cpio

newc

Новый (SVR4) переносимый формат.

crc

Новый (SVR4) переносимый формат, учитывающий контрольную сумму.

tar

Tar-формат.

ustar

Tar стандарта POSIX.1 (опознаются также архивы, созданные GNU-версией tar).

hpbin

Исполняемый файл системы HP-UX (параметр устарел, не используется).

hpodc

Переносимый формат системы HP-UX.

–I *file*

Считать файл (*file*) обрабатываемым архивом. Файл может быть расположен на удаленной машине (см. параметр **–F**).

–k Параметр игнорируется, оставлен для обратной совместимости.

–l, ––link

Создавать ссылки на файлы вместо копирования.

–L, ––dereference

Обрабатывать файлы по ссылкам.

–m, ––preserve-modification-time

Сохранять дату последнего изменения файла.

–M *msg*, **––message=***msg*

Отображать сообщение *msg* о смене носителя в качестве приглашения перед сменой. Использовать в сообщении переменную **%d**, хранящую численный идентификатор следующего носителя. Режим **–M** применяется только совместно с **–I** или **–O**.

–n, ––numeric-uid-gid

При отображении содержимого архива печатать идентификатор пользователя и группы в численном виде.

––no-absolute-filenames

Создавать все копируемые в архив файлы относительно текущего каталога.

––no-preserve-owner

Делать пользователя, совершающего операцию, владельцем обрабатываемых файлов. Имеет смысл только для привилегированного пользователя.

–O *file*

Перенаправить вывод в файл (*file*). Файл может находиться на другой машине (см. **–F**).

––only-verify-crc

Для архивов CRC-формата вычислять CRC каждого файла, но не выполнять добавление в архив.

––quiet

Не отображать число скопированных блоков.

–r Переименование файлов в диалоговом режиме.

–R [*user*][*:group*], **––owner** [*user*][*:group*]

Переопределить пользователя и группу, владеющих файлами. Имеет смысл только для привилегированного пользователя.

–s, ––swap-bytes

Обменивать значения соседних байт.

–S, ––swap-half-words

Обменивать значения соседних машинных полуслов.

––sparse

При извлечении файлов из архива или копировании записывать файлы с длинными последовательностями нулей как разреженные.

–t, ––list

Вывести содержание ввода (без создания каких-либо файлов). При использовании совместно с **–v** результат напоминает работу команды **ls –l**.

–u, ––unconditional

Безусловное копирование; старые файлы записываются поверх более новых, если таковые существуют.

–v, ––verbose

Вывести список имен обрабатываемых файлов.

–V, ––dot

Отображать точку (символ) для каждого прочитанного или записанного файла (это позволяет наблюдать за работой **cpio**, не загромождая экран информацией).

cpio

−−version

Отобразить информацию о версии программы и завершить работу.

Примеры

При помощи команды **find** создать список файлов, имена которых заканчиваются на *.old*. Передать список **cpio:**

```
find . -name “*.old” - print | cpio -ocBv > /dev/rst8
```

Извлечь из архива с ленты все файлы, имена которых содержат **save** (подкаталоги создаются по необходимости):

```
cpio -icdv “*save*” < /dev/rst8
```

Переместить дерево каталогов:

```
find . -depth -print | cpio -padm /mydir
```

cpp

cpp [*options*] [*ifile* [*ofile*]]

GNU-версия препроцессора для языка C. Вызывается при первом проходе компиляции любой C-программы с помощью **gcc**. Результат работы является вводом для следующего прохода компилятора C, и обычно **cpp** вызывает **gcc,** закончив обработку. *ifile* и *ofile* являются, соответственно, именами исходного и получаемого файла для препроцессора; по умолчанию этими файлами являются стандартный ввод и стандартный вывод.

Параметры

−$ Символ $ в идентификаторах запрещен.

−dM

Подавление нормального вывода. Отображаются последовательности макроопределений **#define**, использованных в исходном файле.

−dD

Аналогично параметру **−dM**, но исключаются заранее заданные макроопределения и включаются результаты работы препроцессора.

−idirafter *dir*

Искать заголовочные файлы в каталоге *dir*, если файл не найден ни в одном из каталогов с включаемыми файлами.

–imacros *file*

Обрабатывать макроподстановки в файле *file* до обработки основных исходных файлов.

–include *file*

Обрабатывать файл *file* прежде основного файла.

–iprefix *prefix*

При добавлении каталогов посредством **–iwithprefix** предварять имя каталога префиксом (*prefix*).

–iwithprefix *dir*

Добавить *dir* к списку каталогов, в которых производится дополнительный поиск, если заголовочный файл не был найден в обычных путях поиска. Если задан параметр **–iprefix**, добавить указанный префикс к *dir*.

–lang-c, **–lang-c++**, **–lang-objc**, **–lang-objc++**

Исходный текст написан на языке C, C++, Objective C или Objective C++, соответственно.

–lint

Отображать все команды lint в качестве комментариев следующего вида: **#pragma lint** *command*.

–nostdinc

Производить поиск заголовочных файлов только в указанных, но не в стандартных каталогах поиска.

–nostdinc++

Запретить поиск в каталогах, содержащих специфичные для C++ заголовочные файлы.

–pedantic

Предупреждения о возможных ошибках.

–pedantic-errors

Считать ошибкой сомнительные места, в которых команда **–pedantic** выдавала бы только предупреждение.

–traditional

Препроцессор традиционного C, а не ANSI C .

–undef

Запретить определение нестандартных макросов.

–A *name*[=*def*]

Контроль имени *name* по значению *def*, как при определении в директиве **#assert**.

cpp

–C Передавать компилятору комментарии (кроме тех, что в строках директив **cpp**). По умолчанию **cpp** удаляет комментарии в стиле C.

–D*name*[=*def*]

Определить имя *name* как имеющее значение *def* (по аналогии с директивой **#define**). Если значение (=*def*) опущено, *name* получает значение 1. Приоритет **–D** меньше, чем у параметра **–U**.

–H Отобразить пути к включаемым файлам (по одному имени в строке) в стандартный поток ошибок.

–I*dir*

Искать в каталоге *dir* включаемые директивой **#include** файлы, имена которых не начинаются с символа «/», прежде чем искать в стандартных каталогах. Поиск включаемых файлов, имена которых заключены в двойные кавычки и не начинаются с символа «/», выполняется сначала в текущем каталоге, затем в каталогах, заданных параметром **–I**, и в последнюю очередь в стандартных каталогах.

–M [–MG]

Подавление нормального вывода. Отобразить правило **make**, описывающее зависимости для главного исходного файла. Если указан параметр **–MG**, считать, что недостающие файлы заголовков являются генерируемыми и искать их в каталоге с исходными файлами.

–MD *file*

Аналогично **–M**, но с выводом в файл и компиляцией исходного текста.

–MM

Аналогично **–M**. Описываются только файлы, прямо включенные директивой **#include** "*file*".

–MMD *file*

Аналогично **–MD**, но описываются только заголовочные файлы пользователя.

–P Препроцессор не генерирует управляющую информацию по строкам, используемую на втором проходе компилятора C.

–U*name*

Удалить определение имени *name*, которое может являться зарезервированным и определенным предварительно препроцессором либо именем, определенным параметром **–D**. Имена, предопределяемые **cpp** на системах Intel, – это **unix** и **i386**.

–Wcomment, –Wcomments

Предупреждать о вложенных комментариях.

–Wtraditional

Предупреждать о конструкциях, которые в ANSI C интерпретируются иначе, чем в традиционном C.

Специальные имена

Препроцессор **cpp** распознает различные специальные имена. Вот некоторые из них:

__DATE__

Текущая дата (например Oct 10 1997).

__FILE__

Имя текущего файла (в формате строки C).

__LINE__

Текущая строка исходного файла (десятичное целое).

__TIME__

Текущее время (например 12:00:00).

Эти специальные имена могут использоваться в любом месте, в том числе и в макроопределениях, как и любые другие определяемые имена. Интерпретация номера строки и имени файла в **cpp** может изменяться с помощью директивы **#line**.

Директивы

Все директивы препроцессора **cpp** начинаются с символа # в первой колонке строки. Допускается любое количество пробелов и табуляций между символом # и непосредственно директивой. Существуют следующие директивы:

#assert *name* (*string*)

Определить вопрос с именем *name* и ответом *string*. Предположения (assertions) могут вычисляться в директивах **#if**. Для архитектурно-зависимых изменений в исходном тексте могут использоваться предположения для **#system**, **#cpu** и **#machine**.

cpp

#unassert *name*

Удалить предположение для вопроса *name*.

#define *name token-string*

Определяет макрос с именем *name* и значением *token-string*. Все последующие вхождения имени *name* заменяются на *token-string*.

#define *name(arg, . . . , arg) token-string*

Осуществляет подстановку макросов с аргументами. Параметр *name* будет заменен в исходном файле на *token-string*. Аргументы из каждого вхождения *name* в исходном файле будут подставлены в *token-string*.

#undef *name*

Удалить определение макроса с именем *name*. Имя должно быть единственным параметром директивы **#undef**.

#ident *string*

Поместить строку *string* в раздел комментариев в объектном файле.

#include "*filename*", **#include** <*filename*>

Включение содержимого файла *filename* в данной точке исходного текста. Использование дополнительных имен после закрывающего символа " или > запрещено.

#line *integer-constant* "*filename*"

cpp генерирует управляющую информацию для следующего прохода компилятора C. Компилятору передается порядковый номер следующей строки, которую следует обработать (*integer-constant*) в файле *filename*. По умолчанию *filename* является именем текущего файла. Указание дополнительных имен после имени файла внутри директивы недопустимо.

#endif

Директива, закрывающая секцию текста, начатую условной директивой (**#if**, **#ifdef** или **#ifndef**). Использование имен в строке этой директивы препроцессора запрещено.

#ifdef *name*

Строки, следующие за этой директивой вплоть до закрывающей директивы **#endif** или следующего **#else**, или **#elif**, входят в результат работы препроцессора, если имя *name* определено. Указание дополнительных имен в строке директивы запрещено.

#ifndef *name* **cpp**

Строки, следующие за этой директивой вплоть до закрывающей директивы **#endif** или следующего **#else** или **#elif**, входят в результат работы препроцессора, если имя *name* не определено. Нельзя указывать дополнительные имена в строке директивы.

#if *constant-expression*

Строки, следующие за этой директивой вплоть до соответствующей директивы **#endif** или следующего **#else** или **#elif**, входят в результат работы препроцессора, если значение выражения *constant-expression* не равно нулю.

#elif *constant-expression*

Допускается произвольное количество директив **#elif** между директивами **#if**, **#ifdef** или **#ifndef** и директивами **#else** или **#endif**. Строки, следующие за **#elif** вплоть до следующего **#else**, **#elif** или **#endif**, входят в результат работы препроцессора, если значения выражений предшествующей условной директивы и всех промежуточных директив **#elif** равны нулю, а значение выражения *constant-expression* нулю не равно. В этом случае все последующие директивы **#elif** и **#else** игнорируются.

#else

Строки, следующие за этой директивой вплоть до соответствующего **#endif**, входят в результат работы препроцессора, если значения выражений предшествующей условной директивы и всех промежуточных директив **#elif** равны нулю. Использование каких-либо имен в строке директивы **#else** запрещено.

#error

Выдавать сообщения о неисправимых ошибках.

#warning

Выдавать предупреждения, но продолжать работу.

---

cron **cron**

Команда системного администрирования. Обычно запускается из стартового файла системы. Позволяет выполнять команды по расписанию, заданному в пользовательских файлах из каталога */etc/cron/tabs*. Имя каждого файла совпадает с именем владельца. Работать с этими файлами позволяет команда **crontab**.

crontab

crontab [*options*] [*file*]

Просмотреть, создать или удалить текущий файл конфигурации *crontab*. Привилегированный пользователь может запустить **crontab** для любого пользователя, указав параметр **–u** *user*. Файл конфигурации *crontab* представляет собой список команд (по одной на каждую строку), автоматически выполняющихся в определенное время. Время выполнения команд задается цифрами непосредственно перед каждой командой. Цифры, обозначающие время запуска команды, разделены на пять полей:

Минута	0–59
Час	0–23
День месяца	1–31
Месяц	1–12 Jan, Feb, Mar, ...
День недели	0–6, где 0 означает воскресенье Sun, Mon, Tue, ...

Несколько значений могут перечисляться через запятую, интервал обозначается с помощью дефиса, все возможные значения – символом звездочки (*). Предположим, файл *crontab* содержит следующие строки:

```
59 3 * * 5    find / -print | backup_program
0 0 1,15 * *  echo “Timesheets due” | mail user
```

Первая команда выполняет резервное копирование файлов системы каждую пятницу в 3:59 утра, а вторая отправляет уведомление почтой 1-го и 15-го числа каждого месяца.

Привилегированный пользователь может выполнять команду **crontab** всегда. Прочие пользователи должны быть упомянуты в файле */etc/cron.allow*, если таковой существует, в противном случае – не должны быть упомянуты в файле */etc/cron.deny*. Если не существует ни один из файлов, команду может запускать только привилегированный пользователь.

Параметры

Параметры **–e**, **–l** и **–r** недопустимы, если задан хотя бы один файл в командной строке.

–e Редактировать существующий файл *crontab* текущего пользователя (или создать его, если файл не существует).

–l Выдать файл *crontab* текущего пользователя на стандартный вывод.

crontab

−r Удалить файл *crontab* текущего пользователя.

−u *user*

Указать пользователя, для которого будет проводиться работа с файлом *crontab*.

csh [*options*] [*file* [*arguments*]]

csh

Командный интерпретатор C (C-shell), который принимает и обрабатывает все прочие команды. Более подробно интерпретатор описан в главе 8 «Интерпретаторы csh и tcsh».

csplit [*options*] *file arguments*

csplit

Разбить файл на несколько частей и поместить каждую из них в отдельный файл с именем от xx00 до xx*n* (n < 100). Разбивка файла выполняется в соответствии с шаблоном, заданным в аргументах. См. также **split**.

Параметры

− Считывать файл со стандартного ввода.

−b *suffix*, **−−suffix-format=***suffix*

Добавить к именам получаемых файлов суффикс (*suffix*). В этом случае параметр **−n** игнорируется. Суффикс определяет, как двоичное целое преобразуется в символьную, читаемую форму. Существуют следующие варианты задания суффиксов:

%**d** Знаковое десятичное целое.

%**i** То же, что и %**d**.

%**u** Беззнаковое десятичное целое.

%**o** Восьмеричное целое.

%**x** Шестнадцатеричное целое.

%**X** То же, что и %**x**.

−f *prefix*, **−−prefix=***prefix*

Именовать файлы с *prefix***00** до *prefixn* (по умолчанию с **xx00** до **xx***n*).

−k, **−−keep-files**

Сохранять только что созданные файлы, даже если произошла ошибка (обычно ошибка приводит к удалению файлов). Этот оператор полезен, если необходимо задать произвольно большой аргумент повторений {*n*}, а в резуль-

csplit

тате ошибки «out of range» (выход за пределы диапазона) могут быть уничтожены все созданные в процессе работы файлы.

–n *num*, **––digits=***num*

Заканчивать имена создаваемых файлов числом в *num* знаков длиной. По умолчанию *num* = 2.

–s, **–q**, **––silent**, **––quiet**

Подавлять статистику по символам.

–z, **––elide-empty-files**

Не создавать пустые файлы. Однако остальные файлы нумеровать так, как если бы пустые были созданы.

Аргументы

Аргументом является любое из следующих выражений или их комбинация. Аргументы, содержащие пустые или специальные символы, должны заключаться в одинарные кавычки.

/*expr*/ [*offset*]

Создать файл и включить в него все строки, начиная с текущей и заканчивая строкой, содержащей регулярное выражение *expr*. Смещение (*offset*) представляется в виде +*n* или –*n*, где *n* – число строк после или перед выражением *expr*.

%*expr*%[*offset*]

Аналогично /*expr*/, но не создается файл для строк, предшествующих строке, содержащей выражение *expr*.

num

Создать файл и включить в него строки от текущей до строки с номером *num*. При наличии счетчика повторений ({*n*}) следующие *num* строк файла помещаются в отдельный файл.

{*n*}

Повторить аргумент *n* раз. Может следовать за любым из вышеперечисленных аргументов. Файлы будут разделены по выражению *expr*, являющемуся границей секций, или на секции по *num* строк. Если на месте *n* стоит символ *, аргумент применяется к данным, пока не будет достигнут конец обрабатываемого файла.

Примеры

Создать до 20 файлов-глав из файла *novel*.

```
csplit -k -f chap. novel '/CHAPTER/' '{20}'
```

csplit

Создать до 100 файлов с адресами (xx00 – xx99), по четыре строки в каждом файле, из базы данных *address_list*.

```
csplit -k address_list 4 {99}
```

ctags

ctags [*options*] *files*

Создать список имен функций и макросов, определенных в указанных исходных текстах программ C, C++, FORTRAN, Java, Perl, **yacc** или других. Результат работы (по умолчанию файл *tags*) содержит строки следующего вида:

```
name  file  context
```

где *name* – имя функции или макроса, *file* – файл с исходным текстом, в котором определено это имя, а *context* – шаблон поиска, которому соответствует строка кода, содержащая имя *name*. После создания списка имен (тегов) можно вызвать **vi** для любого файла и ввести:

```
:set tags=tagsfile
:tag name
```

Редактор **vi** переключается на исходный файл, ассоциированный с именем *name* в файле тегов *tagsfile* (который можно задать параметром **–t**).

etags создает эквивалентный файл для использования с редактором Emacs.

Параметры

–a, ––append

Добавлять теги в уже существующий список.

–d, ––defines

Включать теги для определяющих директив препроцессора C.

–i *file*, **––include**=*file*

Добавить в файл *tags* упоминание об имени файла *file*, который необходимо прочитать в дополнение к данному.

–l *language*, **––language**=*language*

Считать все файлы, следующие за данным параметром, файлами на языке *language*. Параметр **–h** позволяет получить перечень языков и расширений имен файлов по умолчанию для этих языков.

–o *file*, **––output**=*file*

Запись результата в файл *file*.

ctags	**–r***regexp*, **––regex=***regexp* Включать теги для строк в тексте, соответствующих регулярному выражению *regexp*. Применять ко всем файлам, следующим за параметром. **–R, ––no-regex** Не включать теги для строк в тексте, соответствующих регулярному выражению. Применять ко всем файлам, следующим за параметром. **–t, ––typedefs** Включать строки для определений типов с помощью **typedefs**. **–u, ––update** Обновить файл тегов, чтобы учесть новое местоположение функций (например, если функции перемещены в другой исходный файл). Старые теги удаляются, новые добавляются в конец файла. **–v, ––vgrind** Выдать на стандартный вывод распечатку (указатель) для каждой функции, исходный файл и номер страницы (одна страница содержит 64 строки). **–w, ––no-warn** Не выводить предупреждения. **–x, ––cxref** Выводить распечатку каждой функции, номер строки, исходный файл и контекст. **–B, ––backward-search** Поиск тегов в файлах от конца к началу. **–C, ––c++** Считать, что файлы *.c* и *.h* содержат текст на языке C++, а не C. **–H, –h, ––help** Вывести справку по использованию и завершить работу. **–S, ––ignore-indentation** Обычно **ctags** использует отступы в тексте для разбора файла тегов; данный параметр делает работу менее зависимой от отступов. **–T, ––typedefs-and-c++** Включать строки тегов для определений typedef, struct, enum, union и функций-членов C++.

−V, −−version **ctags**

Вывести информацию о версии программы и завершить работу.

cut *options* [*files*] **cut**

Удаление определенных колонок или полей записей из перечисленных файлов. В последующем описании параметров список (*list*) представляет собой последовательность из целых чисел. Элементы списка разделяются запятыми, интервалы задаются с помощью дефиса (например, **1–10,15,20** или **50−**). См. также **paste** и **join**.

Параметры

−b *list*, **−−bytes** *list*

Указать список *list* позиций; печататься будут только байты, находящиеся в этих позициях строк.

−c *list*, **−−characters** *list*

Удалять колонки в позициях из списка.

−d *c*, **−−delimiter** *c*

Применяется с параметром **−f** для задания символа *c* в качестве разделителя полей (по умолчанию используется символ табуляции). Специальные символы (например, пробел) должны заключаться в кавычки.

−f *list*, **−−fields** *list*

Вырезать поля, определенные в списке *list*.

−n Не разделять символы, занимающие более одного байта.

−s, −−only-delimited

Применяется совместно с **−f** для подавления обработки строк без разделителей.

−−output-delimiter=*string*

Использовать указанную строку в качестве разделителя при выводе. По умолчанию при выводе разделитель такой же, как при вводе.

−−help

Вывести справку по работе команды и завершить работу.

−−version

Вывести номер версии **cut** и завершить работу.

cut

Примеры

Извлечь регистрационные и реальные имена пользователей из файла */etc/passwd*:

```
cut -d: -f1,5 /etc/passwd
```

Выяснить, кто из пользователей находится в системе (отображать только регистрационные имена):

```
who | cut -d" " -f1
```

Вырезать символы в четвертой колонке файла (*file*) и вставить их в первую колонку того же файла:

```
cut -c4 file | paste - file
```

date

date [*options*] [+*format*] [*date*]

Отобразить текущее время и дату. Может быть задан формат отображения (*format*). Формат может состоять из текстовых строк, интерпретируемых буквально (пробелы должны экранироваться кавычками), и описателей полей формата, вид которых представлен ниже. Элементы расположены в логическом порядке. Привилегированный пользователь может менять системное время и дату.

Параметры

+*format*

Отобразить текущее время в нестандартном виде, например:

```
% date +"%A %j %n%k %p"
Tuesday 248
15 PM
```

По умолчанию установлен формат **%a %b %e %T %Z %Y**, т. е. Tue Sep 5 14:59:37 EDT 2000.

–d *date*, **––date** *date*

Отобразить дату *date* (заключенную в кавычки), представленную в формате *d* **days** или *m* **months** *d* **days** для получения даты, большей текущей. Ключевое слово **ago** позволяет вывести дату более раннюю, чем текущая. Можно использовать также форматирование (см. далее раздел «Формат»).

–f *datefile*, **––file**=*datefile*

Аналогично **–d** для каждой строки файла *datefile*.

–I [*timespec*], **––iso-8601**[=*timespec*]

Отображать дату в формате стандарта ISO-8601. Если указан параметр *timespec*, он может принимать одно из следующих значений: **date** (только дата), **hours**, **minutes**, **seconds**. Этот параметр определяет точность отображения времени.

–r *file*, **––reference**=*file*

Отобразить время последнего изменения файла.

–R, **––rfc-822**

Отобразить дату в формате RFC 822.

––help

Вывести справку и завершить работу.

––version

Вывести информацию о версии и завершить работу.

–s *date*, **––set** *date*

Изменить дату.

–u, **––universal**

Установить время по Гринвичу (Greenwich Mean Time), а не местное.

Формат

% Символ %.

– Не производить заполнение полей (по умолчанию производится заполнение нулями).

_ Заполнение полей пробелами (по умолчанию – нулями).

%**a** Сокращенное название дня недели.

%**b** Сокращенное название месяца.

%**c** Специфичный для страны формат даты и времени.

%**d** День месяца (01–31).

%**h** Идентично %**b**.

%**j** День года по Юлианскому календарю (001–366).

%**k** Час в 24-часовом формате времени без предваряющих нулей (0–23).

%**l** Час в 12-часовом формате времени без предваряющих нулей (1–12).

%**m** Месяц года (01–12).

%**n** Новая строка.

date

%**p** Отображение половины суток: a.m. – до полудня или p. m. – после полудня (по умолчанию AM или PM).

%**r** Время в 12-часовом формате %**I**:%**M**:%**S**%**p**.

%**s** Количество секунд, прошедших с начала «Эпохи», т. е. с момента времени 00:00:00 UTC 01.01.1970 (это нестандартное расширение).

%**t** Вставить табуляцию.

%**w** День недели (где Sunday = 0).

%**x** Специфичный для страны формат даты.

%**y** Последние две цифры года (00–99).

%**z** Часовой пояс, код по RFC 822.

%**A** Полное имя дня недели.

%**B** Полное имя месяца.

%**D** Дата в формате %**m**/%**d**/%**y**.

%**H** Час в 24-часовом формате (00–23).

%**I** Час в 12-часовом формате (01–12).

%**M** Минуты (00–59)

%**S** Секунды (00–59).

%**T** Время в формате %**H**:%**M**:%**S**.

%**U** Порядковый номер недели в году (00–53), неделя начинается с воскресенья.

%**V** Порядковый номер недели в году (01–52), неделя начинается с понедельника.

%**W** Порядковый номер недели в году (00–53), неделя начинается с понедельника.

%**X** Специфичный для страны формат времени.

%**Y** Год в четыре цифры (например, 1996).

%**Z** Название часового пояса.

Строки задания времени и даты

Строки, задающие время и дату, могут быть численными и смешанными. Численные строки состоят из времени, дня и года в формате *MMDDhhmm[[CC]YY][.ss]*. Смешанные строки могут включать названия месяцев, часовых поясов, идентификатор половины суток.

time

Время представляется в формате *hhmm*: две цифры на час в 24-часовом формате и две – на минуты.

date

day

День представляется четырьмя цифрами в формате *MMDD* – две на месяц и две на день месяца. Значение по умолчанию – текущий день текущего месяца.

year

Год представляется полностью (четыре цифры) или последними двумя цифрами. Значение по умолчанию – текущий год.

Примеры

Установить следующую дату: 1 июля **(0701)**, 4 часа утра **(0400)**, 1995 год **(95)**:

```
date 0701040095
```

Команда

```
date +"Hello%t Date is %D %n%t Time is %T"
```

выдает дату в следующем виде:

```
Hello     Date is 05/09/93
     Time is 17:53:39
```

dd

dd [*option*=*value*]

Создание копии обрабатываемого файла **(if)** с учетом заданных параметров, запись результата в конечный файл (или на стандартный вывод, если параметр **of** не задан). Может быть перечислено любое количество параметров, но **of** и **if** традиционно идут первыми, поскольку чаще всего используются. В **dd** имеется возможность задания произвольного размера блока информации, что делает программу полезной для переноса на устройство прямого доступа.

Параметры

bs=*n*

Установить размер блока для чтения и записи в *n* байт; этот параметр имеет больший приоритет и отменяет действие параметров **ibs** и **obs**.

cbs=*n*

Установить размер буфера преобразования (размер логической записи) в *n* байт. Используется только для флагов преобразования **ascii**, **ebcdic**, **ibm**, **block** или **unblock**.

dd

conv=*flags*

Производить преобразование исходного файла в соответствии с перечисленными (через запятую) флагами. Первые пять флагов *flags* взаимно исключают друг друга.

ascii

Преобразование из системы EBCDIC в ASCII.

ebcdic

Преобразование из системы ASCII в EBCDIC.

ibm

Преобразование из системы ASCII в EBCDIC с учетом стандартов IBM.

block

Преобразование записей переменной длины (т. е. тех, которые завершаются символом новой строки) в записи фиксированной длины.

unblock

Обратно предыдущему параметру.

lcase

Преобразование прописных букв в строчные.

ucase

Преобразование строчных букв в прописные.

noerror

Продолжать обработку при возникновении ошибки чтения.

notrunc

Не усекать конечный файл.

swab

Обменивать значения пар байтов.

sync

При чтении дополнять нулями блоки до размера, заданного параметром **ibs**.

count=*n*

Копировать первые *n* прочитанных блоков.

ibs=*n*

Установить размер блока для чтения в *n* байт (512 по умолчанию).

if=*file*

Читать данные из файла *file* (по умолчанию выполняется чтение со стандартного ввода).

dd

obs=*n*

Установить размер блока для записи в *n* байт (512 по умолчанию).

of=*file*

Записывать результат работы в файл *file* (по умолчанию запись выполняется в стандартный вывод).

seek=*n*

Пропустить *n* блоков (размер блока соответствует размеру блока для записи, **obs**) от начала получаемого файла.

skip=*n*

Пропустить *n* блоков (размер блока соответствует размеру блока для чтения, **ibs**) от начала входного файла.

––help

Отобразить справку по использованию **dd** и завершить работу.

––version

Вывести номер версии **dd** и завершить работу.

Значения размера (*n*) можно умножать на 1024, 512 или 2, добавляя к числу, соответственно, буквы **k**, **b** или **w**. Букву **x** можно использовать как оператор умножения двух произвольных чисел.

Примеры

Преобразовать все символы файла в строчные:

```
dd if=caps_file of=small_file conv=lcase
```

Обработать записи переменной длины (полученные командой *data_retrieval_cmd*) и записать их в файл **out**, зафиксировав размер записи:

```
data_retrieval_cmd | dd of=out conv=sync,block
```

debugfs

debugfs [[*option*] *device*]

Команда системного администрирования. Отладка файловой системы **ext2**. Параметр *device* является специальным файлом, соответствующим устройству, содержащему файловую систему **ext2** (например, */dev/hda3*).

Параметр

–w Открытие файловой системы для чтения и записи.

debugfs

Команды

cat *file*

Отобразить содержимое inode-блока на стандартный вывод.

cd *directory*

Сменить текущий рабочий каталог на указанный.

chroot *directory*

Сменить корневой каталог на указанный inode-блок.

close

Закрыть открытую в настоящий момент файловую систему.

clri *file*

Очистить содержимое inode-блока, соответствующего файлу *file*.

dump *file out_file*

Записать содержимое inode-блока в файл *out_file*.

expand_dir *directory*

Раскрыть каталог.

find_free_block [*goal*]

Найти первый свободный блок, начиная с *goal* (если задан), и занять его.

find_free_inode [*dir* [*mode*]]

Найти свободный inode-блок и занять его.

freeb *block*

Пометить блок *block* как свободный.

freei *file*

Освободить inode-блок, соответствующий указанному файлу *file*.

help

Вывести список команд **debugfs**.

icheck *block*

Выполнить перевод блок → inode-блок.

initialize *device blocksize*

Создать файловую систему **ext2** на указанном устройстве.

kill_file *file*

Удалить файл и освободить занимаемые им блоки.

ln *source_file dest_file*

Создать ссылку.

ls [*pathname*]

Эмуляция команды **ls**.

modify_inode *file*

Изменить содержимое inode-блока, соответствующего файлу.

mkdir *directory*

Создать каталог.

mknod *file* [**p** | [[**c** | **b**] *major minor*]]

Создать специальный файл устройства.

ncheck *inode*

Выполнить перевод inode-блок → имя.

open [**−w**] *device*

Открыть указанную файловую систему.

pwd

Вывести текущий рабочий каталог.

quit

Завершить работу с **debugfs**.

rm *file*

Удалить файл.

rmdir *directory*

Удалить каталог.

setb *block*

Пометить указанный блок как занятый.

seti *file*

Пометить как используемый inode-блок, соответствующий файлу.

show_super_stats

Отобразить содержимое суперблока.

stat *file*

Отобразить содержимое inode-блока, соответствующего указанному файлу.

testb *block*

Проверить, занят ли указанный блок.

debugfs	**testi** *block* Проверить, занят ли inode-блок, соответствующий указанному файлу. **unlink** *file* Удалить ссылку на файл. **write** *source_file file* Создать в файловой системе файл *file* и скопировать в него содержимое файла *source_file*.
depmod	**depmod** [*options*] *modules* Команда системного администрирования. Создать файл зависимостей для указанных в командной строке модулей (*modules*). Файл зависимостей может использоваться программой **modprobe** для автоматической загрузки необходимых модулей. Обычной практикой применения **depmod** является включение вызова **/sbin/depmod –a** в один из файлов каталога */etc/rc.d*, чтобы корректные зависимости модулей были доступны сразу же после загрузки системы. **Параметры** **–a** Создать зависимости для всех модулей, перечисленных в файле */etc/conf.modules*. **–d** Режим отладки. Отображать все выполняемые команды. **–e** Перечислять не найденные в модулях имена (unresolved symbols). **–v** Перечислять обработанные модули. **Файлы** */etc/conf.modules* Информация о модулях и их зависимостях, о каталогах, в которых расположены различные типы модулей. */sbin/insmod, /sbin/rmmod* Программы, с которыми работает **depmod**.

df [*options*] [*name*]

df

Отображение количества свободного дискового пространства на всех подмонтированных файловых системах либо для системы с определенным именем (*name*). **df** не отображает информацию по неподмонтированным файловым системам. Дисковое пространство отображается по умолчанию в блоках по 1 Кбайт либо по 512 байт (если установлена переменная окружения POSIXLY_CORRECT). Имя (*name*) может быть именем устройства (например, */dev/hd**), путем точки монтирования (например, */usr*) или простым путем к каталогу (в этом случае **df** выдает сведения в целом по файловой системе, в которой расположен каталог).

Параметры

–a, ––all

Отображать информацию даже для файловых систем с нулевым количеством блоков.

––block-size=*n*

Показать размеры в блоках по *n* байт.

––h, ––human-readable

Вывести размеры в виде, удобном для чтения (т. е. 1,9 Гбайт вместо 1967156).

––H, ––si

Аналогично **–h**, но в качестве основания для размеров используется число 1000, а не 1024.

–i, ––inodes

Отображать информацию по свободным и использованным единицам **inode**, а также их процентное соотношение.

–k, ––kilobytes

Вывести размер в килобайтах, а не блоках по 512 байт.

–l, ––local

Показать информацию только для локальных файловых систем.

–m, ––megabytes

Вывести размер в мегабайтах.

––no-sync

Не выполнять **sync** (сброс буферизованных данных) перед вычислением и отображением информации. Режим по умолчанию.

df	**–P, ––portability** Установить формат выдачи стандарта POSIX (информация по каждой файловой системе занимает не более одной строки). **––sync** Выполнять **sync** (сброс буферизованных данных) перед вычислением и отображением информации. **–t** *type*, **––type**=*type* Дать информацию только для файловых систем заданного типа (*type*). **–T, ––print-type** Вывести тип каждой файловой системы наряду с информацией о размерах. **–x** *type*, **––exclude-type**=*type* Дать информацию для всех файловых систем, кроме имеющих тип *type*. **––help** Вывести справку по использованию и завершить работу. **––version** Отобразить номер версии программы и завершить работу.
diff	**diff** [*options*] [*diroptions*] *file1 file2* Сравнение двух текстовых файлов. **diff** отображает информацию о несовпадающих строках в файлах *file1* и *file2*. Результат состоит из соответствующих строк пары файлов, причем строки файла *file1* маркируются символом <, а строки файла *file2* – символом >. Строки предваряются командой редактора **ed** (**a**, **c** или **d**), которая необходима для преобразования файла *file1* в *file2*. Если задан параметр «–» вместо одного из имен файлов, то происходит чтение со стандартного ввода. Если одно из имен является именем каталога, **diff** обращается к тому файлу в заданном каталоге, который соответствует второму параметру (например, **diff my_dir junk** эквивалентно **diff my_dir/junk junk**). Если оба аргумента являются именами каталогов, **diff** обрабатывает попарно файлы, имеющие совпадающие имена (например, *olddir/program* и *newdir/program*). Кроме того, **diff** перечисляет уникальные имена для каждого каталога, а также подкаталоги, общие для обоих каталогов. См. также команду **cmp**.

Параметры

–a, ––text

Считать все файлы текстовыми. Параметр полезен при сравнении бинарных файлов.

–b, ––ignore-space-change

Игнорировать повторяющиеся пробелы и пробелы в конце строки; считать повторяющиеся пробелы за один символ.

–B, ––ignore-blank-lines

Игнорировать пустые строки в файлах.

–c Контекстное сравнение: выводить по три строки текста до и после измененной строки.

–C *n*, **––context[=***n***]**

Контекстное сравнение: выводить по *n* строк текста до и после измененной строки.

–d, ––minimal

Для того чтобы ускорить сравнение, игнорировать последовательности строк, содержащих многочисленные отличия и, таким образом, вывести наиболее короткий результирующий список.

–D*symbol*, **––ifdef=***symbol*

При обработке файлов с исходными текстами на языке C создавать выходной файл, включающий содержимое обоих исходных файлов, плюс директивы **#ifdef** и **#ifndef**, которые присутствуют в обоих файлах.

–e, ––ed

Создать сценарий команд **(a, c** и **d)**, преобразующих *file1* в *file2* при помощи редактора **ed**.

–F *regexp*, **––show-function-line[**=*regexp***]**

Для контекстного и объединенного вариантов **diff** отображать последние строки, содержащие соответствие регулярному выражению *regexp*, перед каждым блоком изменившихся строк.

–H Ускорить обработку больших файлов, включив в рассмотрение строки с отдельными небольшими отличиями; при этом длинные участки с изменениями могут не отображаться.

diff	**--help** Вывести краткую справку по использованию. **--horizon-lines=***n* При сравнении стремиться к уменьшению объема результата, оставляя при этом по *n* строк до и после строк с изменениями. **-i, --ignore-case** При сравнении не учитывать регистр символов. Строчный символ и его прописной вариант считаются одинаковыми. **-I** *regexp*, **--ignore-matching-lines=***regexp* Пропускать строки, которые соответствуют регулярному выражения *regexp*. **-l, --paginate** Выводить результат постранично, передавая его **pr**. **-L** *label*, **--label** *label*, **--label=***label* В случае контекстного или объединенного **diff** отображать метку *label* вместо имени сравниваемого файла. Первый такой параметр относится к первому имени файла, второй, соответственно, – ко второму. **--left-column** Для режима вывода в две колонки (**-y**) отображать только левую колонку для совпадающих строк. **-n, --rcs** Отображать результаты в формате **diff** RCS. **-N, --new-file** Реагировать на несуществующие файлы как на пустые. **-p, --show-c-function** При работе с файлами, содержащими исходные тексты на C и C-подобных языках вроде Java, отображать функцию, содержащую каждый из блоков измененных строк. Подразумевается включение параметра **-c**, но также может использоваться с объединенным **diff**. **-P, --unidirectional-new-file** Если при сравнении двух каталогов парный файл отсутствует в первом каталоге, считать этот файл существующим и пустым. **-q, --brief** Сообщать лишь о том, что в файлах найдены отличия.

–r, ––recursive

Рекурсивно сравнивать подкаталоги.

–s, ––report-identical-files

Сообщать о неотличимых файлах.

–S *filename*, **––starting-file=***filename*

При сравнении каталогов начинать с файла *filename*, пропустив все файлы до него, исходя из стандартного порядка перечисления.

––suppress-common-lines

Для режима вывода в две колонки (**–y**) не отображать совпадающие строки.

–t, ––expand-tabs

При отображении результатов преобразовывать символы табуляций в пробелы.

–T, ––initial-tab

Дополнять результат символами табуляции для получения форматированного отображения.

–u Объединенный вариант **diff**: отображать последовательно старую и новую версии строки в одном блоке, окружая каждый блок отличающихся строк тремя строками контекста.

–U *n*, **––unified[=***n***]**

Объединенный вариант **diff**: отображать последовательно старую и новую версии строки плюс по *n* строк контекста до и после измененной строки. Размер контекста по умолчанию составляет 3 строки.

–v, ––version

Вывести информацию о версии **diff**.

–w, ––ignore-all-space

При сравнении игнорировать все пробелы.

–W *n*, **––width=***n*

Для режима вывода в две колонки (**–y**) задать максимальную ширину колонки в *n* символов. Значение по умолчанию равно 130.

–x *regexp*, **––exclude=***regexp*

Не сравнивать файлы, имена которых соответствуют регулярному выражению *regexp*.

diff	**−X** *filename*, **−−exclude-from=***filename* Не сравнивать файлы, имена которых совпадают с шаблонами, перечисленными в файле *filename*. **−y, −−side-by-side** Выводить результат в две колонки. **−***n* Для контекстного и объединенного вариантов установить количество строк контекста в *n*. Идентично указанию количества строк в параметрах **−C** и **−U**.
diff3	**diff3** [*options*] *file1 file2 file3* Сравнение трех файлов и вывод результатов. Только один из файлов может быть обозначен символом «−» (что является указанием читать файл со стандартного ввода). Результат отображается в следующей форме: **====** Все три файла различны. **====1** *file1* отличен, *file2* и *file3* идентичны. **====2** *file2* отличен, *file1* и *file3* идентичны. **====3** *file3* отличен, *file1* и *file2* идентичны. **diff3** также предназначается для объединения разных изменений одного и того же файла (например, если два пользователя внесли изменения в один и тот же файл, создав каждый свою версию). **diff3** способен находить различия между исходным файлом и каждым из измененных, а результат работы можно использовать для добавления изменений к исходному файлу. Изменения называются *необъединенными* (*unmerged changes*), когда оба измененных файла отличаются друг от друга в одном и том же месте и по крайней мере один из них отличается в том же месте от исходного файла. Совпадающие отличия измененных файлов от исходного называются *объединенными* (*merged changes*). Ситуация, когда все три файла отличаются в одном и том же месте, носит название *наложения отличий* (*overlapping change*). Параметры описывают схему работы, при которой исходный файл (*file2*) сравнивается с одним из измененных файлов (*file3*), а результаты этого сравнения накладываются на *file1*.

Параметры

–3, ––easy-only

Создать сценарий **ed**, добавляющий в *file1* необъединенные, не налагающиеся отличия между файлами *file2* и *file3*.

–a, ––text

Все файлы считаются текстовыми.

–A, ––show-all

Создать сценарий **ed**, добавляющий в файл *file1* все отличия, отображая в квадратных скобках все противоречия.

–e, ––ed

Создать сценарий **ed**, добавляющий к файлу *file1* все необъединенные отличия между *file2* и *file3*.

–E, ––show-overlap

Создать сценарий **ed**, добавляющий к файлу *file1* необъединенные отличия, отображая в квадратных скобках все противоречия.

–x, ––overlap-only

Создать сценарий **ed**, добавляющий к файлу *file1* все налагающиеся отличия между тремя файлами.

–X Аналогично **–x**, но противоречия наложения отличий отображаются в квадратных скобках.

–m, ––merge

Создать файл, объединяющий изменения (не сценарий **ed**).

–L *label*, **––label**=*label*

Заменить имя выходного файла на *label* при выводе.

–i Добавить команду сохранения **(w)** и завершения работы **(q)** к сценарию **ed**.

–T, ––initial-tab

Начинать строки символом табуляции (вместо пары пробелов) с целью получения форматированного отображения.

–v, ––version

Вывести информацию о версии программы и завершить работу.

dip

dip [*options*] [*chat scriptfile*]

Команда системного администрирования. Настройка и установка соединения с Интернетом. **dip** применяется для установки как исходящих, так и входящих соединений. Команды можно вводить в диалоговом режиме либо помещать в файл сценария для последующего выполнения при установке соединения. При обработке входящих соединений **dip** часто используется в качестве командного интерпретатора и может запускаться с помощью команд **diplogin** или **diplogini**.

Параметры

–a В режиме приема соединений запрашивать имя пользователя и пароль. Идентично команде **diplogini**.

–i Запустить интерпретатор для входящих соединений. Идентично команде **diplogin**.

–k Удалить самый последний процесс **dip** либо процесс, работающий на устройстве, которое задано параметром **–l**.

–l *device*

Используется совместно с параметром **–k** для указания устройства терминала.

–m *mtu*

Максимальный размер передаваемого блока (Maximum Transfer Unit). По умолчанию равен 296.

–p *protocol*

Используемый протокол: SLIP, CSLIP, PPP или TERM.

–t Командный режим. Обычно применяется для проверки.

–v Режим диагностики.

Команды

Большая часть этих команд могут выполняться как в диалоговом режиме, так и в файле сценария.

beep *times*

Выдать звуковой сигнал на терминал указанное количество раз.

bootp

Получить локальный и удаленный IP-адреса, используя протокол BOOTP.

break

Послать сигнал BREAK.

chatkey *keyword code*

Привязать ключевое слово (ответ модема) к числовому коду.

config [interface | routing] [pre | up | down | post] *arguments*

Изменение настроек интерфейса **(interface)** или таблицы маршрутизации **(routing)** для состояний: до установки соединения **(pre)**, при установленном соединении **(up)**, при разрыве соединения **(down)** и после разрыва соединения **(post)**. Синтаксис аргументов такой же, как для команд **ifconfig** и **route**.

databits 7 | 8

Установить количество бит данных.

dec *$variable*[*value*]

Уменьшить значение переменной *variable* на *value*. По умолчанию уменьшается на 1.

default

Установить маршрут по умолчанию в IP-адрес узла, с которым установлено соединение.

dial *phonenumber* [*timeout*]

Набрать телефонный номер *phonenumber*. Завершить попытку соединения, если удаленный модем не ответит в течение *timeout* секунд. Установить значение **$errlvl** в соответствии с ответом модема.

echo on | off

Включить или выключить отображение команд модема.

exit [*n*]

Завершить работу сценария с необязательным кодом завершения *n*.

flush

Очистить буфер ввода.

get *$variable* **[ask | remote [***timeout***]]** *value*

Установить значение переменной *variable* равным *value*. Если указан параметр **ask**, то значение вводится пользователем. Если указан параметр **remote**, значение должно быть получено от удаленной системы. Завершение ожидания происходит через *timeout* секунд.

goto *label*

Переход к секции с меткой *label*.

dip

help

Перечислить доступные команды.

if *expr* **goto** *label*

Переход к секции с меткой *label*, если выражение *expr* истинно. В выражении происходит сравнение переменных и констант при помощи одного из операторов: =, !=, <, >, <= или >=.

inc *$variable* [*value*]

Увеличить значение переменной *variable* на *value*. По умолчанию – на 1.

init *string*

Задание строки, используемой для инициализации модема. По умолчанию это строка ATE0 Q0 V1 X1.

mode *protocol*

Задание протокола соединения. Допустимые значения: SLIP, CSLIP, PPP и TERM. По умолчанию – SLIP.

netmask *mask*

Задание маски подсети.

parity E | O | N

Установка контроля четности линии: по четности (even), по нечетности (odd) или отсутствие контроля (none).

password

Запросить пароль у пользователя.

proxyarp

Добавить запись для ARP-прокси в локальную таблицу ARP.

print *$variable*

Отобразить значение переменной *variable*.

psend *command*

Выполнить команду в интерпретаторе и направить вывод команды в последовательное устройство. Команды выполняются с реальным идентификатором пользователя.

port *device*

Указать последовательное устройство, к которому подключен модем.

quit

Завершить работу с ненулевым кодом завершения. Происходит разрыв соединения.

reset

Сброс модема.

securid

Запрашивать переменную часть пароля ACE System SecureID и посылать ее вместе с хранимым постоянным префиксом удаленной системе.

securidf *prefix*

Сохранить фиксированную часть пароля ACE System SecureID.

send *string*

Послать строку *string* в последовательное устройство.

shell *command*

Выполнить команду в интерпретаторе, используя реальный идентификатор пользователя.

skey [*timeout*]

Ожидать запрос S/Key, затем запросить у пользователя секретный ключ. Сгенерировать и послать ответ. Завершить ожидание, если запрос не получен в течение *timeout* секунд. Программа **dip** должна быть собрана с поддержкой S/Key.

sleep *time*

Выждать *time* секунд.

speed *bits-per-second*

Установить скорость порта. По умолчанию – 38400.

stopbits 1 | 2

Установить количество стоповых бит.

term

Перейти в терминальный режим. Передавать ввод с клавиатуры напрямую в последовательное устройство.

timeout *time*

Установить интервал бездействия на линии, по истечении которого происходит разрыв связи.

wait *text* [*timeout*]

В течение *timeout* секунд ожидать прихода текста *text* от удаленной системы. Если время не указано, ожидание длится бесконечно.

dirname	**dirname** *pathname* Отобразить путь *pathname*, исключив последний уровень. Используется для удаления имени файла из путевого имени. Если символы «/» в имени отсутствуют, команда **dirname** отображает точку «.» (текущий каталог). См. также **basename**.
dmesg	**dmesg** [*options*] Команда системного администрирования. Отобразить контрольные сообщения системы из кольцевого буфера ядра. В этом буфере сохраняются все сообщения с момента последней загрузки системы (ядра) либо, в случае заполнения буфера, несколько последних сообщений. **Параметры** **–c** Очистить буфер после вывода сообщений. **–n** *level* Установить уровень отображаемых системных сообщений.
dnsdomainname	**dnsdomainname** Команда TCP/IP. Вывод DNS (доменного имени системы). См. также **hostname**.
domainname	**domainname** [*name*] Команда NFS/NIS. Установка или отображение имени текущего NIS-домена. Команда без аргумента отображает существующее имя. Устанавливать имя домена, задавая аргумент, может только привилегированный пользователь; обычно это происходит в стартовом сценарии.
dosfsck	**dosfsck** [*options*] *device* **fsck.ext2** [*options*] *device* Команда системного администрирования. Аналогична **fsck**, но предназначена специально для файловых систем MS-DOS. При проверке файловых систем MS-DOS эта программа вызывается **fsck**. Обычно **dosfsck** хранит все изменения в памяти и записывает их, когда проверка заканчивается.

dosfsck

Параметры

–a Автоматически исправлять ошибки в файловой системе, не запрашивая подтверждения пользователя.

–A Работа с версией MS-DOS для систем Atari.

–d *file*

Удалить указанный файл из таблицы размещения файлов. Принудительно произвести проверку, даже если файловая система помечена ядром как корректная. В ином случае, если для файловой системы установлен флаг целостности, **dosfsck** просто завершает работу.

–l *file*

Прочитать список сбойных блоков из указанного файла и проверить все остальные.

–n Не производить никаких изменений файловой системы. На все запросы следует давать отрицательный ответ.

–p Автоматически обрабатывать все сбойные блоки.

–t Отображать статистику по времени выполнения.

–v Диагностика работы.

–y На все запросы следует давать положительный ответ.

–B *size*

Считать, что суперблок расположен по адресу *size*; если это не так, завершить работу.

–F Сбрасывать буферы перед проверкой.

–L *file*

Прочитать список сбойных блоков из указанного файла *file* вместо проверки файловой системы на сбойные блоки.

du

du [*options*] [*directories*]

Отображение информации по заполнению дискового пространства (т. е. количество блоков по 1 Кбайт), занимаемого заданным каталогом (и содержащимися в нем подкаталогами).

Параметры

–a, ––all

Вывести размеры всех файлов, а не только подкаталогов в целом.

du	**–b, ––bytes** Вывести размеры в байтах. **–c, ––total** Помимо обычной обработки аргументов подсчитать суммарный размер всех заданных каталогов. **–D, ––dereference-args** Обрабатывать символические ссылки, если они указаны в качестве аргументов команды. **–h, ––human-readable** Отображать размеры в удобном для чтения виде. **–H, ––si** Аналогично **–h**, но за основание для размеров принимается число 1000, а не 1024. **–k, ––kilobytes** Вывести размеры в килобайтах (режим по умолчанию). **–l, ––count-links** Учитывать размер всех файлов, даже тех, которые уже были учтены по жесткой ссылке. **–L, ––dereference** Следовать по символическим ссылкам. **––exclude=***pattern* Не учитывать файлы, имена которых соответствуют шаблону *pattern*. **––max-depth=***num* Отображать размер каталогов, находящихся не ниже уровня *num* от начальной точки, которая считается нулевым уровнем. **–m, ––megabytes** Выводить размеры в мегабайтах. **–s, ––summarize** Отображать только окончательный результат для каждого из перечисленных каталогов. **–S, ––separate-dirs** Не включать размеры подкаталогов при подсчете суммарного размера каталогов.

du

–x, ––one-file-system

При подсчете не выходить за пределы текущей файловой системы.

–X, ––exclude-from=*file*

Не учитывать размеры файлов, имена которых соответствуют какому-либо шаблону в *file*.

––help

Вывести справку по использованию и завершить работу.

––version

Отобразить информацию о версии программы и завершить работу.

dumpe2fs *device*

dumpe2fs

Команда системного администрирования. Вывод информации о суперблоке устройства *device* и группе блоков.

dumpkeys [*options*]

dumpkeys

Вывод информации о текущей таблице преобразования драйвера клавиатуры. Более подробную информацию можно почерпнуть на страницах руководства по *keytables*.

Параметры

–1, ––separate-lines

Отображать каждую пару модификатор/код в отдельной строке, предваряя немодифицированные коды клавиш строкой **plain**.

–c*charset*, **––charset=***charset*

Указать кодировку, по которой будут определяться значения кодов символов. Кодировка по умолчанию – **iso-8859-1**. Полный перечень допустимых кодировок доступен по ключу **––help**.

––compose-only

Выводить только клавиатурные комбинации для compose-клавиш.[1] Требуется поддержка compose-клавиш ядром.

[1] Клавиши, одновременно нажимаемые для вывода на экран какого-либо символа и позволяющие набирать те символы, которых нет на клавиатуре (например, буквы французского или немецкого алфавита и т. д.). – *Примеч. науч. ред.*

dumpkeys

–f, ––full-table

Отобразить каноническую таблицу: на каждую клавишу отводится по строке, где в колонках расположены значения с различными модификаторами.

––funcs-only

Вывести только строковые определения с функциональными клавишами; не выводить клавиатурные комбинации или обычные строковые определения.

–h, ––help

Вывести справку и номер версии.

–i, ––short-info

Краткий формат выдачи информации. Включает информацию о корректных ключевых словах для кодов клавиш в клавиатурных таблицах, о количестве действий, которые возможно связать с комбинацией, а также список диапазонов кодов действий (значений, расположенных справа от определения клавиши) и количество поддерживаемых ядром функциональных клавиш.

––keys-only

Вывести только клавиатурные комбинации; не отображать строковые определения.

–l, ––long-info

Та же информация, что и в **––short-info**, плюс список поддерживаемых символов действий и их численных значений.

–n, ––numeric

Отображать значения кодов действий в шестнадцатеричной записи; не пытаться преобразовывать их в символьную запись.

–S *num*, **––shape=***num*

Использовать число *num* для определения способа отображения. Возможные значения *num*:

0 По умолчанию.

1 То же, что и **––full-table**.

2 То же, что и **––separate-lines**.

3 По строке на каждый код клавиши до первого пропуска, затем по строке на пару модификатор/код.

e2fsck [*options*] *device*
fsck.ext2 [*options*] *device*

e2fsck

Команда системного администрирования, аналогичная **fsck**, но предназначенная специально для расширенной файловой системы Linux. **e2fsck** вызывается **fsck** при проверке *ext2fs*.

Параметры

–b *superblock*

Использовать указанный суперблок вместо суперблока по умолчанию.

–d Режим отладки.

–f Принудительная проверка файловой системы, даже если она уже отмечена ядром как корректная. Обычно же **e2fsck** завершает работу без проверки, если файловая система в порядке.

–l *file*

Прочитать список сбойных блоков из файла *file* и проверить все остальные.

–n Не вносить изменения в файловую систему. На запросы отвечать отрицательно.

–p «Чистка». Отмечать сбойные блоки в пакетном режиме.

–t Отображать статистику по времени работы.

–v Режим диагностики.

–y Отвечать положительно на все запросы.

–B *size*

Искать суперблок по адресу *size*; завершить работу, если суперблок не найден.

–F Произвести сброс буферов перед проверкой.

–L *file*

Взять список сбойных блоков из указанного файла вместо проверки устройства на их наличие.

echo [**–n**] [*string*]

echo

Команда **/bin/echo**. Существует также команда **echo**, встроенная в интерпретаторы C shell и **bash**. Следующие последовательности символов имеют специальное значение:

\a Звуковой сигнал.

echo

\b Забой.

\c Подавление заключительного символа новой строки.

\f Новая страница (перевод формата).

\n Новая строка.

\r Возврат каретки.

\t Горизонтальная табуляция.

\v Вертикальная табуляция.

**** Обратный слэш.

nnn

Символ с восьмеричным ASCII-кодом *nnn*.

Параметры

–e Распознавать escape-последовательности (см. выше) внутри строк. В некоторых версиях этот параметр не требуется.

–E Отключить распознавание escape-последовательностей.

–n Подавить вывод заключительного символа новой строки.

––help

Отобразить информацию по использованию и завершить работу.

––version

Вывести информацию о версии программы и завершить работу.

Примеры

```
/bin/echo “testing printer” | lp
/bin/echo “TITLE\nTITLE” > file; cat doc1 doc2 >> file
/bin/echo “Warning: ringing bell \a”
```

egrep

egrep [*options*] [*regexp*] [*files*]

Поиск в файлах (*files*) строк, соответствующих регулярному выражению *regexp*. В **egrep** не поддерживаются следующие регулярные выражения: \(, \), \n, \<, \>, \{ и \}. Прочие регулярные выражения имеют силу, как и расширения +, ?, | и ( ). Помните, что эти символы необходимо заключать в кавычки. Подробно регулярные выражения описаны в главе 9 «Шаблоны и поиск». Коды возврата **egrep**: 0 – если результат поиска не равен нулю, 1 – в обратном случае, 2 – произошла ошибка.

egrep

Перечень параметров командной строки см. в описании **grep**. См. также **fgrep**. **egrep**, как правило, работает быстрее этих двух программ.

Примеры

Искать имена *Victor* и *Victoria* в файле:

```
egrep 'Victor(ia)*' file
egrep '(Victor|Victoria)' file
```

Найти в файлах и отобразить такие строки, как *old.doc1* или *new.doc2*, включая номера строк:

```
egrep -n '(old|new)\.doc?' files
```

emacs

emacs [*options*] [*files*]

Текстовый редактор и многоцелевая рабочая среда. Более подробно Emacs описан в главе 10 «Редактор Emacs».

env

env [*options*] [*variable=value* . . .] [*command*]

Отобразить параметры текущего окружения либо, если задана переменная окружения (*variable*), присвоить ей новое значение (*value*) и отобразить измененные параметры окружения. Если задана команда (*command*), выполнить ее в измененном окружении.

Параметры

–, –i, ––ignore-environment

Полностью игнорировать существующее окружение.

–u *name*, **––unset** *name*

Сбросить заданную переменную окружения.

––help

Вывести справку по использованию и завершить работу.

––version

Отобразить информацию о версии программы и завершить работу.

etags

etags [*options*] *files*

Создать список имен функций и макросов, определенных в указанном исходном тексте программы на языке C, Pascal, FORTRAN, **yacc** или **flex**. Результат работы (по умолчанию файл *tags*) содержит строки следующего вида:

etags

```
name   file   context
```

где *name* – имя функции или макроса, *file* – файл с исходным текстом, в котором определено это имя, а *context* – шаблон поиска, которому соответствует строка кода, содержащая имя *name*. После создания списка имен (тегов) можно вызвать Emacs для любого файла и ввести:

```
ESC-x visit-tags-table
```

Emacs предложит задать имя файла; по умолчанию – TAGS. Чтобы переключиться на редактирование исходного файла, связанного с именем *name* в файле *tagsfile*, введите:

```
ESC-x find-tag
```

а затем имя тега, которое необходимо найти. **ctags** создает эквивалентный файл для работы с редактором **vi.**

Параметры

–a, ––append

Добавлять теги в уже существующий список.

–d, ––defines

Включать строку для каждой из определяющих директив препроцессора.

–i *file*, **––include=***file*

Добавить к файлу тегов примечание о том, что в дополнение к исходному файлу необходимо прочитать файл *file*.

–l *language*, **––language=***language*

Файлы, перечисленные после этого параметра, содержат исходные тексты на языке *language*. При помощи параметра **–h** можно получить перечень узнаваемых языков и стандартных расширений файлов с текстами на этих языках.

–o *file*, **––output=***file*

Запись результата в файл *file*.

–r *regexp*, **––regex=***regexp*

Включать тег для каждой из строк, содержащей соответствие регулярному выражению *regexp,* из всех файлов, следующих после параметра.

–C, ––c++

Считать, что файлы *.c* и *.h* содержат текст на языке C++, а не C.

etags

–D, ––no-defines

Не включать строки для определяющих директив препроцессора.

–H, –h, ––help

Вывести справку по использованию **etags**.

–R, ––noregex

Не включать теги для строк, соответствующих регулярным выражениям, из всех файлов, следующих за этим параметром.

–S, ––ignore-indentation

Обычно **etags** использует отступы в тексте для разбора файлов тегов; данный параметр делает работу независящей от отступов.

–V, ––version

Вывести информацию о номере версии **etags**.

ex

ex [*options*] *file*

Диалоговый командный редактор. Более подробно описан в главе 11 «Редактор vi».

expand

expand [*options*] *files*

Преобразование символов табуляции в перечисленных файлах или, если именем файла является «–», то в потоке данных, поступающих со стандартного ввода. Символы табуляции заменяются соответствующим количеством пробелов. Результат выдается на стандартный вывод.

Параметры

–*tabs*, **–t, ––tabs** *tabs*

tabs – это список целых чисел, разделенных запятыми, которые определяют размеры табуляций в зависимости от их порядковых номеров в строках. Если список состоит из одного числа, каждый символ табуляции заменяется таким же количеством пробелов. По умолчанию размер табуляции составляет 8 пробелов. При использовании **–t** и **––tabs** значения в списке могут быть разделены пробелами, а не запятыми.

expand	**–i, ––initial** Подвергать преобразованию только символы табуляции, расположенные в начале строк. **––help** Вывести справку по использованию и завершить работу. **––version** Отобразить информацию о версии программы и завершить работу.
expr	**expr** *arg1 operator arg2* [*operator arg3*...] Вычисление выражения, заданного аргументами, и вывод результата. Аргументы и операторы должны разделяться пробелами. В большинстве случаев аргументами являются целые числа, представленные цифрами или переменными командного интерпретатора. Существует три вида операторов: арифметические, операторы отношения и логические. В выражениях могут использоваться ключевые слова. Коды возврата **expr: 0** – выражение не равно нулю или результат не пустой, **1** – выражение равно нулю или результат пустой, **2** – выражение недопустимо. **Арифметические операторы** Применяются для вычисления математических выражений с выводом результата: + Сложить аргументы *arg2* и *arg1*. – Вычесть *arg2* из *arg1*. * Перемножить аргументы. / Поделить *arg1* на *arg2*. % Взять остаток от деления *arg1* на *arg2*. Операции сложения и вычитания выполняются в последнюю очередь, если они не сгруппированы внутри скобок. Символ «*» и круглые скобки имеют специальное значение для интерпретатора, поэтому их необходимо экранировать (символом «\» или одинарными кавычками). **Операторы отношений** Используются для сравнения двух аргументов. Аргументы могут быть словами, тогда сравнение определяется текущими параметрами локализации (locale). Если сравнение истинно, **expr** возвращает 1, в противном случае – 0. Символы «>» и «<» должны быть экранированы символом «\».

=, ==

Проверка на равенство аргументов.

!= Проверка на неравенство аргументов.

> Проверить, что *arg1* больше *arg2*.

>= Проверить, что *arg1* больше или равен *arg2*.

< Проверить, что *arg1* меньше, чем *arg2*.

<= Проверить, что *arg1* меньше или равен *arg2*.

Логические операторы

Предназначены для сравнения двух аргументов. В зависимости от значений аргументов результат может равняться *arg1* (или его части), arg2 или нулю. Символы «|» и «&» должны быть экранированы при помощи символа «\».

| Логическое ИЛИ; если аргумент *arg1* имеет ненулевое (и непустое) значение, результат равен arg1. В противном случае результат равен arg2.

& Логическое И; если оба аргумента имеют ненулевое (и непустое) значение, результат равен arg1; в противном случае результат равен 0.

: Аналог **grep**. Аргумент *arg2* является шаблоном для поиска в *arg1*. *arg2* должен быть регулярным выражением. Если шаблон *arg2* заключен в пары символов \(\), то результатом является часть аргумента *arg1*, соответствующая шаблону, в противном случае результатом является количество символов, совпадающих с шаблоном. По умолчанию происходит выравнивание шаблона по началу аргумента *arg1* (регулярное выражение неявно начинается с символа «^»); но можно производить поиск и в произвольном месте строки, тогда шаблон должен начинаться с символов «.*».

Ключевые слова

index *string character-list*

Возвращает номер первой позиции строки *string*, символ которой совпадает с любым символом из списка возможных (*character-list*), причем предпочтение отдается символу, встретившемуся в списке раньше прочих. Если ни один символ в строке не найден, возвращается 0.

length *string*

Вернуть длину строки *string*.

expr

match *string regex*

Идентично *string* **:** *regex*.

quote *token*

Считать *token* строкой, даже если его можно интерпретировать как ключевое слово или оператор.

substr *string start length*

Вернуть часть строки (*string*), начиная с символа *start*. *length* определяет максимальное количество возвращаемых символов. При задании отрицательных или нечисловых параметров *start* и *length* возвращается пустой результат.

Примеры

Приоритет операции деления выше; результат вычисления равен 10:

```
expr 5 + 10 / 2
```

Сначала происходит сложение; результат равен 7 (усечение 7,5):

```
expr \( 5 + 10 \)  /  2
```

Прибавить 1 к переменной *i*. В сценариях командных интерпретаторов именно так увеличиваются значения переменных:

```
i = `expr $i + 1`
```

Вывести 1 (истина), если значение переменной **a** равняется строке «hello»:

```
expr $a = hello
```

Вывести 1 (истина), если сумма **b** и **5** больше или равна 10:

```
expr $b + 5 \>= 10
```

Найти 5-ю, 6-ю и 7-ю буквы слова *character*:

```
expr substr character 5 3
```

В следующих примерах переменная **p** является строкой и имеет значение «version.100». Данная команда выводит количество символов в **p**:

```
expr $p : '.*'
```

Результат равен 11

Найти все символы и напечатать их:

expr

```
expr $p : '\(.*\)'        Результат: «version.100»
```

Вывести число строчных букв в начале строки **p**:

```
expr $p : '[a-z]*'        Результат: 7
```

Найти строчные буквы в начале строки **p**:

```
expr $p : '\([a-z]*\)'    Результат: «version»
```

Усечь **$x**, если **$x** содержит 5 или более символов; в противном случае просто напечатать **$x** (логическое ИЛИ возвращает второй аргумент в том случае, если первый равен нулю или пуст, т. е. когда сравнение с шаблоном возвращает отрицательный результат.)

```
expr $x : '\(.....\)' \| $x
```

Строка из сценария командного интерпретатора. Усечь имена файлов до первых пяти символов.

```
mv $x `expr $x : '\(.....\)' \| $x`
```

(Чтобы избежать перезаписи уже существующих файлов, используйте **mv –i**.)

false

false

«Пустая» команда. Возвращает код завершения с ошибкой (не 0). Обычно применяется в сценариях интерпретатора **bash**. См. также **true**.

fdformat [*options*] *device*

fdformat

Низкоуровневое форматирование дискет. Устройством (*device*) для форматирования обычно является **/dev/fd0** или **/dev/fd1**.

Параметр

–n Не производить проверку после завершения форматирования.

fdisk [*options*] [*device*]

fdisk

Команда системного администрирования. Работа с разделами диска через меню. **fdisk** отображает информацию о существующих разделах, позволяет создавать и удалять разделы, а также менять активный раздел. Можно иметь различные

fdisk

операционные системы на всех четырех разделах, но активным разделом может быть только один из них. Также можно разбить физический раздел на несколько логических. Минимальный рекомендуемый размер раздела Linux – 40 Мбайт. Обычно устройство (*device*) – это */dev/hda*, */dev/hdb*, */dev/sda*, */dev/sdb*, */dev/hdc*, */dev/hdd* и т. д. См. также **cfdisk**.

Параметры

–l Перечислить таблицы разделов и завершить работу.

–s*partition*

Отобразить размер раздела *partition*, если это не раздел DOS.

Команды

a Переключение флага bootable (загружаемый) текущего раздела.

d Удалить текущий раздел.

l Перечислить существующие типы разделов.

m Главное меню.

n Создать новый раздел; дополнительная информация вводится пользователем.

p Вывести список всех разделов и отобразить информацию о каждом из них.

q Завершить работу без сохранения изменений.

t Сменить тип текущего раздела.

u Изменить единицы отображения/ввода размеров (цилиндры или сектора).

v Проверить таблицу разделов на наличие ошибок, отобразить суммарное количество свободного (не принадлежащего разделам) дискового пространства.

w Сохранить изменения и завершить работу.

fetchmail

fetchmail [*options*] [*servers*...]

Команда системного администрирования. Получение почты с почтовых серверов и пересылка в локальную почтовую систему доставки. **fetchmail** работает с почтовыми серверами по протоколам POP2, POP3, IMAP2bis и IMAP4. Сообщения доставляются посредством SMTP через порт 25 локального узла и посредством агента доставки почты системы (например, *sendmail*), а затем могут быть прочитаны с помощью

пользовательского почтового клиента. Настройки **fetchmail** хранятся в файле ~/*.fetchmailrc*. Параметры и серверы также могут указываться в командной строке (в этом случае они имеют больший приоритет, чем настройки *.fetchmailrc*). **fetchmail** совместима с программой **popclient**, так что пользователи могут работать с обеими программами, не меняя настроек.

Параметры

–a, ––all

Получить все сообщения с сервера, включая те, что уже были прочитаны, но оставлены на сервере. По умолчанию происходит получение только новых сообщений.

–A *type*, **––auth** *type*

Указать тип авторизации. *type* может иметь значение **password**, **kerberos_v5** или **kerberos**. Тип авторизации обычно определяется **fetchmail** автоматически, так что этот параметр не особо полезен.

–B *n*, **––fetchlimit** *n*

Ограничить максимальное количество сообщений, которое может быть получено с сервера за один запрос.

–b *n*, **––batchlimit** *n*

Ограничить максимальное количество сообщений, которое может быть передано приемнику на SMTP-порту. При достижении предела соединение разрывается и немедленно устанавливается повторно. Значение по умолчанию равно 0 (ограничений нет).

–c, ––check

Проверка наличия почтовых сообщений на одном сервере без получения или удаления сообщений. Работает с протоколом IMAP, а с другими протоколами работает не очень хорошо (либо совсем не работает).

–D [*domain*], **––smtpaddress** [*domain*]

Указать домен, помещаемый в строке RCPT TO при пересылке по протоколу SMTP. По умолчанию это имя локального домена (localhost).

–E *header*, **––envelope** *header*

Изменить заголовок, предположительно содержащий адрес на «конверте» (обычно «X-Envelope-to:»), на *header*.

fetchmail

–e *n*, **––expunge** *n*

Послать IMAP-серверу команду EXPUNGE, стирающую отмеченные для удаления сообщения, после *n* удалений. Значение 0 означает стирание только в конце сеанса. Как правило, **expunge** выполняется при каждом удалении.

–F, **––flush**

Для протоколов POP3 и IMAP: удалить уже полученные сообщения перед получением новых.

–f *file*, **––fetchmailrc** *file*

Задать альтернативное имя файла настройки **fetchmail**.

–I *specification*, **––interface** *specification*

Проверять, что почтовый сервер запущен и работает на указанном IP-адресе (или в диапазоне IP-адресов) перед его опросом. Спецификация (*specification*) указывается в виде *interface/ipaddress/mask*. Первая часть означает тип ожидаемого TCP-соединения (*sl0*, *ppp0* и т. д.), вторая является IP-адресом, а третья – битовой маской IP, предположительно 255.255.255.255.

–K, **––nokeep**

Удалять все полученные сообщения с почтового сервера.

–k, **––keep**

Сохранять копии всех полученных сообщений на сервере.

–l *size*, **––limit** *size*

Ограничить максимальный размер для получаемых сообщений. Сообщения, превышающие ограничение, будут оставлены на сервере и отмечены как непрочтенные.

–M *interface*, **––monitor** *interface*

В режиме демона проверять указанный TCP/IP-интерфейс на наличие какой-либо внешней активности и пропускать опрос, если таковая отсутствует. Полезно для PPP-соединений, которые производят автоматическое рассоединение при отсутствии активности.

–m *command*, **––mda** *command*

Передать почту напрямую агенту доставки почтовых сообщений, вместо того чтобы посылать их через 25 порт. Команда *command* является путем с параметрами вызываемого агента, например, **/usr/lib/sendmail -oem**. Параметр %**T** команды заменяется локальным адресом доставки, а %**F** – адресом отправителя сообщения.

–n, ––norewrite

Не расширять локальные почтовые идентификаторы до полных адресов. Этот параметр отключает обычную адресацию и должен использоваться только для отладки.

–P *n*, **––port** *n*

Указать порт для соединения с почтовым сервером. Обычно бывает достаточно значений по умолчанию для портов поддерживаемых протоколов.

–p *proto*, **––protocol** *proto*

Указать протокол, используемый для опроса сервера. Возможные значения *proto*:

POP2 Post Office Protocol 2.

POP3 Post Office Protocol 3.

APOP POP3 с авторизацией по MD5.

RPOP POP3 с авторизацией по RPOP.

KPOP POP3 с авторизацией Kerberos v4 через порт 1109.

IMAP

IMAP2bis, IMAP4 или IMAP4rev1. **fetchmail** автоматически определяет их характеристики.

IMAP-K4

IMAP4 или IMAP4rev1 с авторизацией по Kerberos v4.

IMAP-GSS

IMAP4 или IMAP4rev1 с авторизацией по GSSAPI.

ETRN ESMTP.

–Q *string*, **––qvirtual** *string*

Удалить префикс *string*, являющийся локальным идентификатором узла пользователя, из адреса в заголовке сообщения (например, «Delivered-To:»).

–r *folder*, **––folder** *folder*

Получить указанную почтовую папку *folder* с почтового сервера.

–s, ––silent

Не выводить сообщения о состоянии в процессе получения почты.

–U, ––uidl

Для протокола POP3 отслеживать возраст сохраненных на сервере сообщений с помощью списка уникальных идентификаторов.

fetchmail

–u *name*, **––username** *name*

Указать имя пользователя для авторизации на почтовом сервере.

–V, **––version**

Вывести информацию о номере версии **fetchmail** и набор параметров для каждого из почтовых серверов. Получение сообщений не производится.

–v, **––verbose**

Выводить все сообщения о состоянии в процессе получения почты. Имеет бо́льший приоритет, чем **–q**.

–Z *nnn*, **––antispam** *nnn*

Указать номер SMTP-ошибки, *nnn*, который будет использован для предупреждения о спам-блоке от клиента. Если *nnn* = –1, параметр не имеет влияния.

fgrep

fgrep [*options*] *pattern* [*files*]

Поиск в перечисленных файлах *files* строк, соответствующих заданному шаблону *pattern*. Коды завершения: 0 – если найдена хотя бы одна строка, удовлетворяющая критерию поиска, 1 – если не найдено таких строк, 2 – если произошла ошибка.

Полный перечень параметров приведен в описании команды **grep**. См. также **egrep**.

Примеры

Вывести строки файла *file*, не содержащие пробелы:

```
fgrep -v ' ' file
```

Вывести строки файла *file*, содержащие слова из **spell_list**:

```
fgrep -f  spell_list file
```

file

file [*options*] *files*

Классификация файлов по типу данных, содержащихся в них. Программа **file** обращается к «волшебному файлу» (как правило, */usr/share/magic*), чтобы опознать некоторые типы файлов.

Параметры

–b Режим сокращенного отображения – не печатать имена файлов в начале строк вывода.

file

−c Проверить правильность формата волшебного файла (одновременное использование аргументов *files* несовместимо с параметром **−c**). Обычно применяется совместно с параметром **−m**.

−f *file*

Обработать файлы, имена которых обнаружены в файле (*file*).

−L Следовать по символическим ссылкам. По умолчанию этого не происходит.

−m *file*

Использовать указанный файл в качестве «волшебного» вместо */usr/share/magic*.

−n Сбросить стандартный вывод после определения типа файла.

−s Помимо обычных обрабатывать также файлы блочных и символьных устройств.

−v Вывести версию **file**.

−z Пытаться обработать сжатые файлы.

file распознает многочисленные типы файлов. Результатом является список, в каждой строке которого содержится имя файла и краткая классификация, например:

```
ascii text
c program text
c-shell commands
data
empty
iAPX 386 executable
directory
[nt]roff, tbl, or eqn input text
shell commands
symbolic link to ../usr/etc/arp
```

Пример

Перечислить все файлы, которые могут быть похожи на исходные данные для troff/nroff:

```
file * |  grep roff
```

find

find [*pathnames*] [*conditions*]

Мощнейший инструмент для отбора определенных групп файлов (см. многочисленные примеры в конце описания). **find** выполняет обход дерева каталогов, начиная с *pathname* и

find

отбирает файлы, удовлетворяющие перечисленным критериям (*conditions*). По умолчанию дерево каталогов начинается с текущего каталога. Наиболее полезные критерии и действия: **print** (автоматически включается по умолчанию, если не заданы никакие другие критерии), **–name** и **–type** (часто используются), **–exec** и **–size** (для опытных пользователей), **–mtime** и **–user** (для администраторов).

Критерии отбора могут быть: сгруппированы путем заключения их в экранированные скобки **\(\)**, инвертированы с помощью символа «**!**» (**\!** для интерпретатора C shell), заданы списком альтернатив (разделение с помощью **–o**) или повторены (наложением дополнительных ограничений на поиск; повторение обычно используется только с критериями **–name**, **–type**, **–perm**). Редактированием называется изменение содержимого файла, смена прав доступа или владельца и т. д. Следовательно, критерий **–ctime** охватывает больше параметров, чем **–atime** или **–mtime**.

Критерии и действия

–atime +*n* | –*n* | *n*

Найти файлы, обращение к которым в последний раз производилось более чем *n* (+*n*) дней назад либо менее чем *n* (–*n*) дней назад, либо ровно *n* дней назад. Следует помнить, что работа **find** меняет время последнего обращения для каталогов, указанных в командной строке.

–ctime +*n* | –*n* | *n*

Найти файлы, которые были изменены более чем *n* (+*n*) дней назад либо менее чем *n* (–*n*) дней назад, либо ровно *n* дней назад. Изменением называется действие, приводящее к модификации записи для файла, например, изменение с помощью **chmod**.

–depth

Спускаться по дереву каталогов и обрабатывать сначала файлы (и только *потом* содержащие их каталоги). Режим полезен, если файлы находятся в каталоге, запись в который запрещена (например, может пригодиться при совместном использовании **find** и **cpio**).

–exec *command* { } \;

Выполнить команду *command* для всех файлов, отобранных **find** (при условии успешного завершения **find**, т. е. возврата 0). При выполнении *command* фигурные скобки заменяются именем текущего файла. Параметр завершается экранированным символом точка с запятой «\;».

–follow

Следовать по символическим ссылкам, отмечая каталоги как обработанные (не применять совместно с **–type l**).

–group *gname*

Найти файлы, принадлежащие группе *gname*. *gname* может быть как именем, так и идентификатором группы.

–inum *n*

Найти файлы, для которых число единиц **inode** равно *n*.

–links *n*

Найти файлы, на которые существует *n* ссылок.

–mount, **–xdev**

Искать только файлы, находящиеся внутри файловой системы, в которой начат поиск.

–mtime +*n* | –*n* | *n*

Найти файлы, которые в последний раз были модифицированы более чем *n* (+*n*) дней назад либо менее чем *n* (–*n*) дней назад, либо ровно *n* дней назад. Под модификацией понимается изменение содержимого файла.

–name *pattern*

Найти файлы, имена которых соответствуют шаблону (*pattern*). При использовании специальных символов необходимо экранировать их кавычками или символом «\».

–newer *file*

Найти файлы, содержимое которых изменялось позже, чем содержимое указанного файла (*file*); критерий схож с **–mtime**. **–follow** влияет на этот критерий, только если указан в командной строке раньше него.

–ok *command* { } \;

Идентично **–exec**, но перед выполнением команды запрашивается подтверждение пользователя.

–perm *nnn*

Найти файлы, восьмеричное значение прав доступа которых (например, **rwx**) в точности совпадает с *nnn* (например, **–rw–rw–r––** соответствует значению 664). Укажите знак «минус», чтобы произвести поиск по шаблону для любых незаданных прав доступа (например, **–perm –600** отметит все файлы с правами доступа **–rw–******, где * может принимать любое значение).

find

–print

Отображать абсолютные пути для всех найденных файлов и каталогов. Код возврата – true.

–regex *pattern*

Сходно с **–path**, но с использованием регулярных выражений в стиле **grep** вместо шаблонов командного интерпретатора, таких как в **–name** и **–path**.

–size *n*[**c**]

Найти файлы, состоящие из *n* блоков либо, если задан модификатор **c**, то из *n* символов.

–type *c*

Найти файлы типа *c*. Возможные значения *c*: **b** (специальный файл блочного устройства), **c** (специальный файл символьного устройства), **d** (каталог), **p** (конвейер или именованный канал), **l** (символическая ссылка), **s** (сокет) или **f** (обычный файл).

–user *user*

Найти файлы, принадлежащие пользователю *user* (имя или идентификатор).

–daystart

Отсчитывать время от начала текущего дня.

–maxdepth *num*

Не спускаться по дереву каталогов более чем на *num* уровней.

–mindepth *num*

Производить обработку только на уровнях, расположенных ниже уровня *num*.

–noleaf

Обычно **find** полагает, что для каждого каталога существуют две жесткие ссылки, которые следует игнорировать (это ссылка на имя каталога и на «.»), т. е. реальных каталогов на два меньше, чем жестких ссылок. Критерий **–noleaf** заставляет **find** обрабатывать все жесткие ссылки без исключения, что бывает полезно на файловых системах не-Unix систем (жесткая ссылка может на самом деле быть каталогом, который необходимо обработать). В системах Unix этот параметр только замедляет работу программы.

–amin +*n* | –*n* | *n*

Найти файлы, обращение к которым в последний раз выполнялось более чем *n* (+*n*) минут назад либо менее чем *n* (–*n*) минут назад, либо ровно *n* минут назад.

–anewer *file*

Найти файлы, обращение к которым производилось после последнего изменения содержимого файла *file*. **–follow** влияет на этот критерий, только если указан в командной строке раньше него.

–cmin +*n* | –*n* | *n*

Найти файлы, которые были изменены более чем *n* (+*n*) минут назад либо менее чем *n* (–*n*) минут назад, либо ровно *n* минут назад.

–cnewer *file*

Найти файлы, которые изменялись позднее последнего изменения содержимого файла *file*. **–follow** влияет на этот критерий, только если указан раньше него в командной строке.

–empty

Продолжать работу, если файл пуст. Критерий применим к обычным файлам и каталогам.

–false

Вернуть значение false для каждого найденного файла.

–fstype *type*

Отмечать только файлы, расположенные в файловой системе типа *type*. Могут быть заданы следующие типы: **minix**, **ext**, **ext2**, **xia**, **msdos**, **umsdos**, **vfat**, **proc**, **nfs**, **iso9660**, **hpfs**, **sysv**, **smb** и **ncpfs**.

–gid *num*

Найти файлы, принадлежащие группе с идентификатором *num*.

–ilname *pattern*

Нечувствительный к регистрам символов вариант **–lname**.

–iname *pattern*

Нечувствительный к регистрам символов вариант **–name**.

–ipath *pattern*

Нечувствительный к регистрам символов вариант **–path**.

find

–iregex *pattern*

Нечувствительный к регистрам символов вариант **–regex**.

–lname *pattern*

Искать символические ссылки на файлы, соответствующие шаблону *pattern*. Шаблон может включать специальные символы командного интерпретатора и не трактует «/» (слэш) и «.» (точку) специальным образом.

–mmin +*n* | –*n* | *n*

Найти файлы, содержимое которых редактировалось более чем *n* (+*n*) минут назад либо менее чем *n* (–*n*) минут назад, либо ровно *n* минут назад.

–nouser

Искать файлы с незарегистрированным в системе идентификатором пользователя.

–nogroup

Искать файлы с незарегистрированным в системе идентификатором группы.

–path *pattern*

Найти файлы, имена которых соответствуют шаблону *pattern*. Выполняется обработка имен относительно точки начала поиска, т. е. «/» (слэш) и «.» (точка) не трактуются специальным образом.

Примеры

Перечислить все файлы (и подкаталоги) из домашнего каталога:

```
find $HOME -print
```

Перечислить все файлы с именем *chapter1* из каталога */work*:

```
find /work -name chapter1 -print
```

Перечислить все файлы, начинающиеся с *memo*, владельцем которых является пользователь *ann*:

```
find /work -name 'memo*' -user ann -print
```

Искать (начиная с корневого каталога файловой системы) каталоги, содержащие страницы руководств (manpages):

```
find / -type d -name ‘man*’ -print
```

Найти в текущем каталоге имена файлов, не начинающиеся с прописной буквы, и послать их на принтер:

find

```
find . \! -name '[A-Z]*' -exec lpr {} \;
```

Найти и сжать файлы, имена которых не заканчиваются на *.gz*:

```
gzip `find . \! -name '*.gz' -print`
```

Удалить все пустые файлы в системе (с запросом подтверждений):

```
find / -size 0 -ok rm {} \;
```

Найти файлы, содержимое которых изменялось в последние два дня (первые претенденты на резервное копирование):

```
find / -mtime -2 -print
```

Рекурсивный поиск по шаблону (**grep**) в дереве каталогов:

```
find /book -print | xargs grep '[Nn]utshell'
```

Если в текущем каталоге есть файлы *kt1* и *kt2*, их имена можно вывести с помощью команды:

```
$ find . -name 'kt[0-9]'
./kt1
./kt2
```

Поскольку команда выводит имена, предваряя их символами «./», необходимо задать эти символы с критерием **–path**:

```
$ find . -path './kt[0-9]'
./kt1
./kt2
```

Критерий **–regex**, как и **–path**, полагает путевое имя полным, но интерпретирует аргументы как регулярные выражения, а не шаблоны интерпретатора (хотя в данном случае результат не изменится):

```
$ find . -regex './kt[0-9]'
./kt1
./kt2
```

finger [*options*] *users*

finger

Отобразить информацию о перечисленных пользователях, включая и ту, что содержится в файлах *.plan* и *.project* в домашнем каталоге каждого пользователя. Пользователи (*users*) могут задаваться точным регистрационным именем либо именем или фамилией (выполняется поиск всех пользователей с таким именем или фамилией). В сетевом окружении можно использовать имена вида *user@host* и *@host*.

finger	**Параметры** **–l** Полный формат выдачи данных (режим по умолчанию): все, перечисленное в параметре **–s**, а также домашний каталог, домашний телефон, командный интерпретатор по умолчанию, статус почтового ящика, файлы *.plan*, *.project* и *.forward*. **–m** Подавлять поиск по реальным именам пользователей. **–p** Не отображать содержимое файлов *.plan* и *.project*. **–s** Краткий формат выдачи: регистрационное имя, реальное имя, имя терминала, разрешение на получение сообщений, время бездействия, расположение кабинета и номер рабочего телефона.
fingerd	**in.fingerd** [*option*] Команда TCP/IP. Удаленный сервер информации о пользователях. **fingerd** представляет собой сетевой интерфейс к программе **finger**. Сервер ожидает TCP-соединение через порт **finger** и в ответ на каждое соединение производит чтение единственной строки, которую передает программе **finger**, а вывод программы перенаправляет клиенту, пославшему запрос. **fingerd** запускается **inetd** и должен иметь соответствующую запись в файле настройки */etc/inetd.conf*. **Параметр** **–w** Включать в ответ дополнительную информацию, такую как суммарное время работы и имя операционной системы.
flex	**flex** [*options*] [*file*] **flex** (Fast Lexical Analyzer Generator) является более эффективным вариантом **lex**. **flex** генерирует программу лексического анализа (с именем *lex.yy.c*), основываясь на регулярных выражениях и операторах C, содержащихся в одном или нескольких исходных файлах. См. также описания **bison**, **yacc** и книгу Джона Левина (John Levine), Тони Мэйсона (Tony Mason) и Дага Брауна (Doug Brown) «lex & yacc», изданную O'Reilly. **Параметры** **–b** Создавать резервную копию данных в файле *lex.backup*. **–d** Режим отладки.

–f Ускоренная компиляция (для небольших программ).

–h Краткая справка по использованию **flex**.

–i Нечувствительное к регистрам символов сканирование.

–l Максимальная совместимость с **lex**.

–o *file*

Записать результат в файл *file*, вместо *lex.yy.c*.

–p Вывести отчет о производительности.

–s Завершение работы при встрече сканером данных, не соответствующих ни одному из правил.

–t Отображение на стандартный вывод (по умолчанию вывод производится в файл *lex.yy.c*).

–v Вывести краткую статистику.

–w Не выводить предупреждения.

–B Генерировать пакетный (а не диалоговый) сканер.

–F Использовать таблицы, ускоряющие работу сканера.

–I Генерировать диалоговый сканер (режим по умолчанию).

–L Не обрабатывать директивы **#line** в *lex.yy.c*..

–P *prefix*

Изменить префикс по умолчанию (**yy**) на *prefix* для всех глобальных переменных и имен функций.

–V Отобразить номер версии **flex**.

–7 Генерировать 7-битный сканер.

–8 Генерировать 8-битный сканер (режим по умолчанию).

–+ Генерировать сканер класса C++.

–C Сжимать таблицы сканера, но не использовать классы эквивалентности.

–Ca

Выравнивать таблицы для более быстрого доступа и вычислений. Таблицы увеличиваются в размерах, но позволяют достичь большей производительности.

–Ce

Создавать классы эквивалентности. Это позволяет уменьшить размеры таблиц, но немного снижает производительность (режим по умолчанию).

–Cf

Генерировать полные таблицы сканера (несжатые).

flex	**–CF** Генерировать более эффективные таблицы сканера аналогично **–F**. **–Cm** Создавать классы метаэквивалентности (режим по умолчанию). **–Cr** Не использовать стандартную библиотеку ввода/вывода; вместо этого работать с системными вызовами **read()**.
fmt	**fmt** [*options*] [*files*] Придание тексту форматирования с определенной длиной строк (происходит построчное заполнение, символы новой строки удаляются). Сращение файлов, заданных в командной строке, или текста, поступающего со стандартного ввода. По умолчанию сохраняются пустые строки, пробелы и отступы. Команда **fmt**, по возможности, разбивает строки по концам предложений и избегает разрывов после первого слова в предложении или перед последним. **Параметры** **–c, ––crown-margin** Не изменять отступы первых двух строк, отступы всех последующих строк выравнивать по отступу второй строки. **–p** *prefix*, **––prefix=***prefix* Форматировать только строки, начинающиеся с заданного префикса (*prefix*). **–s, ––splt-only** Запретить слияние строк. **–t, ––tagged-paragraph** Режим разметки абзацев. Работает аналогично режиму **–c**, если отступы первой и второй строк различны. Если отступы одинаковы, первая строка считается отдельным абзацем. **–u, ––uniform-spacing** Вставлять один пробел между словами и по два между предложениями.

−w *width*, **−−width=***width*

Задать длину строки (*width*) для форматирования. По умолчанию равна 75.

−−help

Отобразить справку по использованию и завершить работу.

−−version

Вывести информацию о версии программы и завершить работу.

fmt

---

fold [*options*] [*files*]

fold

Разрыв строк в перечисленных файлах выполняется таким образом, чтобы длина получаемых строк не превышала заданной (80 символов по умолчанию). **fold** разрывает строки точно по заданной длине, даже если разрыв приходится на середину слова. Если вместо имени файла задан параметр «−», то выполняется чтение со стандартного ввода.

Параметры

−b, −−bytes

Производить разбивку по байтам, а не по колонкам текста (т. е. считать каждый из символов табуляции, забоя и возврата каретки одной колонкой).

−s, −−spaces

По возможности производить разрыв на символах пробелов.

−w, −−width *width*

Установить максимальную ширину строки в *width*. По умолчанию равна 80.

---

formail [*options*]

formail

Преобразование данных, поступающих со стандартного ввода, в формат сообщения файла почтового ящика. Если не найден адрес отправителя, **formail** подставляет фиктивный адрес *foo@bar*. По умолчанию дубликаты строки **From** экранируются символом цитирования «>».

Параметры

+skip

Не производить разбивку заданного количества (*skip*) первых сообщений.

formail

–*total*

Прекратить работу после разбивки заданного количества (*total*) сообщений.

–a *headerfield*

Добавить к заголовку письма поле *headerfield*; добавление не происходит, если поле уже присутствует в заголовке. Если задано пустое поле **Message-ID** или **Resent-Message-ID**, сообщению присваивается уникальный идентификатор.

–b Не экранировать дубликаты поля **From**.

–c Если суммарная длина полей заголовка превышает длину строки, произвести объединение полей в одну строку.

–d Не предполагать, что на ввод поступают данные строго в формате почтового сообщения.

–e Сообщения могут следовать одно за другим, без разделяющих отступов.

–f Не изменять строки, представленные не в формате строк почтового сообщения. По умолчанию **formail** добавляет поле **From** к таким строкам.

–i *headerfield*

Добавить поле *headerfield* к заголовку, даже если это поле уже присутствует. Прежние поля с именем *headerfield* переименовываются в **Old-***headerfield*, если они не пустые.

–k Используется только совместно с параметром **–r**. Сохранять тело сообщения и поля, заданные параметром **–r**.

–m *minfields*

Опознавать начало сообщения по минимально допустимому количеству полей заголовка (*minfields*). По умолчанию требуется хотя бы 2 поля.

–n Разрешить выполнение нескольких параллельных процессов **formail**.

–p *prefix*

Экранировать строки префиксом *prefix* вместо символа «>».

–q Не выводить сообщения об ошибках записи, дубликатах сообщений и неверных полях **Content-Length**. Режим задан по умолчанию и отключается параметром **–q–**.

–r Отбрасывать все существующие поля, оставляя только поле **X-Loop**, взамен добавлять заголовок автоматически

создаваемого ответа на сообщение. С помощью параметра **–i** можно принудительно сохранять необходимые поля.

–s Этот параметр должен быть последним в списке; все, что следует за **–s**, считается аргументами параметра. Разделить ввод на несколько сообщений и передать их указанной почтовой программе либо (по умолчанию) сплошным блоком данных передать на стандартный вывод.

–t Предполагать, что обратный адрес отправителя существует (по умолчанию **formail** отдает предпочтение автоматически созданным адресам).

–u *headerfield*

Удалить все, кроме первого вхождения поля *headerfield*.

–x *headerfield*

Выделить полю *headerfield* отдельную строку.

–z По необходимости добавлять пробелы между названиями полей и содержимым. Удалять пустые поля.

–A *headerfield*

Добавлять поле *headerfield*, даже если оно уже существует.

–B Считать, что на вход поступает сообщение в BABYL-формате **rmail**.

–D *maxlen idcache*

Сохранять идентификаторы сообщений в буфере *idcache*, размер которого не может превысить *maxlen*. В режиме разделения сообщений игнорировать дубликаты. В противном случае возвращать **true** при нахождении дубликата. Если задан параметр **–r**, учитывать почтовый адрес отправителя вместо идентификатора сообщения.

–I *headerfield*

Добавлять поле *headerfield*, даже если поле с таким именем существует. Удалять существующие поля.

–R *oldfield newfield*

Переименовать все поля с именем *oldfield* в *newfield*.

–U *headerfield*

Удалить все, кроме последнего вхождения поля с именем *headerfield*.

–Y Форматировать в классическом стиле системы Беркли, т. е. игнорировать поля **Content-Length**.

formail	**–X** *headerfield* Выделить полю *headerfield* и значению поля отдельную строку.
free	**free** [*options*] Отображение статистики по использованию памяти системой: объем свободной, используемой, физической памяти, размер файла подкачки, разделяемая память, память, используемая ядром под буферизацию данных. **Параметры** **–b** Рассчитывать объемы памяти в байтах. **–k** Режим по умолчанию. Рассчитывать объемы памяти в килобайтах. **–m** Рассчитывать объемы памяти в мегабайтах. **–o** Не отображать строки с информацией по использованию буферной памяти (строки «–/+ buffers»). **–s** *time* Обновлять информацию по использованию памяти каждые *time* секунд. **–t** Отображать суммарные показатели в одну строку в последней строке вывода. **–V** Выдать информацию о версии **free**.
fsck	**fsck** [*options*] [*filesystem*] ... Команда системного администрирования. Вызов соответствующей программы проверки и устранения ошибок файловой системы. Если файловая система в порядке, отображается количество файлов, количество занятых и свободных блоков. Если же файловая система содержит ошибки, то **fsck** выдает запрос на подтверждение перед каждой попыткой внесения исправлений. Коды завершения **fsck** могут интерпретироваться, исходя из того, что они являются суммами следующих величин: **1** Были найдены и устранены ошибки. **2** Предлагается перезагрузить систему. **4** Ошибки были найдены, но не были устранены. **8** Ошибка в работе **fsck**. **16** Неверный вызов **fsck**. **128** Ошибка разделяемой библиотеки.

fsck

Параметры

–– Передать все последующие параметры вызываемой программе проверки. Также передаются все параметры, не распознанные **fsck**.

–r Диалоговый режим; запрашивать подтверждение пользователя перед внесением исправлений.

–s Последовательный режим.

–t *fstype*

Указать тип файловой системы. Не проверять файловые системы других типов.

–A Проверка всех файловых систем, перечисленных в файле */etc/fstab*.

–N Подавить нормальное выполнение; отображаются только описания действий, которые будут выполнены.

–R Имеет смысл только совместно с **–A**: проверка всех файловых систем, перечисленных в */etc/fstab*, кроме корневой.

–T Подавить вывод названия.

–V Режим диагностики.

fsck.minix [*options*] *device*

fsck.minix

Команда системного администрирования. Аналог **fsck**, предназначенный специально для работы с файловой системой MINIX.

Параметры

–a Режим автоматического исправления ошибок без запросов подтверждений.

–f Принудительная проверка, даже если файловая система помечена ядром как корректная. **fsck.minix** обычно завершает работу, если файловая система не требует проверки.

–l Перечислить файловые системы.

–m Включить предупреждения «mode not cleared» в стиле MINIX.

–r Диалоговый режим; запрашивать подтверждение перед внесением каких-либо исправлений.

–s Отображение информации о суперблоках.

–v Режим диагностики работы.

ftp

ftp [*options*] [*hostname*]

Программа, организующая передачу файлов между удаленными машинами в сети. *hostname* – имя удаленной сетевой машины. Работа с **ftp** происходит в диалоговом режиме. Команды перечислены ниже, после параметров. Некоторые команды являются переключателями режимов, т. е. включают определенный режим работы, если он выключен, и отключают, если включен.

Параметры

–d Включить режим отладки.

–g Отключить расширение имен.

–i Отключить диалоговый режим работы команд.

–n Запретить автоматический вход в систему сразу после соединения.

–v Все сообщения, посланные удаленной машиной, отображаются полностью.

Команды

![*command* [*args*]]

Запустить интерпретатор команд на локальной машине. Если заданы аргументы, то первый из них считается исполняемой командой, а все остальные передаются этой команде в качестве параметров.

$ *macro-name* [*args*]

Выполнить макрокоманду *macro-name*, которая была определена с помощью команды **macdef**. Параметры передаются в макрокоманду буквально, расширение по маске не происходит.

account [*passwd*]

Задать дополнительный пароль, который будет запрашиваться после успешного входа в удаленную систему для авторизации доступа к ресурсам. Если команда выполняется без аргументов, пользователю предлагается ввести пароль (символы при наборе не отображаются).

append *local-file* [*remote-file*]

Добавить локальный файл в конец файла на удаленном узле. Если не задано имя конечного файла (*remote-file*), используется имя локального файла, измененное в соответствии с установками **ntrans** и **nmap**. При передаче файла действуют текущие установки типа, формата, режима и структуры.

ascii

Установить для передачи файлов режим ASCII (по умолчанию).

bell

Выдавать звуковой сигнал по завершении каждой команды передачи файлов.

binary

Установить двоичный режим передачи файлов.

bye Разорвать соединение, завершить работу и выйти из **ftp**.

case

Переключить режим интерпретации регистров символов в именах файлов на удаленной машине. По умолчанию режим выключен. Если режим активен, при передаче файлов с удаленной машины по команде **mget** все прописные буквы в именах файлов заменяются на строчные для локальных копий.

cd *remote-directory*

Сменить текущий каталог на удаленной машине на *remote-directory*.

cdup

Сменить текущий каталог на удаленной машине, переместившись в родительский каталог.

chmod [*mode*] [*remote-file*]

Изменить права доступа для файла (*remote-file*) на удаленной машине. Если параметры опущены, они запрашиваются в диалоговом режиме.

close

Завершить сеанс работы с FTP и вернуться в интерпретатор команд.

cr Переключение режима удаления символов возврата каретки при передаче файлов ASCII.

delete *remote-file*

Удалить файл *remote-file* на удаленной машине.

debug [*debug-value*]

Переключение режима отладки. Если задана переменная *debug-value*, она определяет уровень отладочной диагностики.

ftp

dir [*remote-directory*] [*local-file*]

Отобразить содержимое каталога (*remote-directory*) с удаленной машины и при необходимости поместить результат в локальный файл (*local-file*). Если каталог не задан, команда работает с текущим каталогом на удаленной машине. Если опущено имя файла (или задан параметр «–»), происходит отображение информации на терминале.

disconnect

Синоним команды **close**.

form *format*

Установить формат передаваемых данных (*format*). По умолчанию это файл (*file*).

get *remote-file* [*local-file*]

Получить файл (*remote-file*) с удаленной машины и сохранить его на локальной машине. Если не задано новое имя для сохраняемого файла (*local-file*), имя файла не меняется, подвергаясь лишь изменениям, предусмотренным командами **case**, **ntrans** и **nmap**. Если вместо локального файла задан параметр «–», то файл выводится на терминал.

glob

Переключение режима расширения имен для команд **mdelete**, **mget** и **mput**. Если расширение отключено, имена файлов в аргументах команд воспринимаются буквально и не расширяются.

hash

Переключение режима отображения символа # после каждого успешно переданного блока данных.

help [*command*]

Вывести справку по использованию команды (*command*). **help** без параметров выводит список команд **ftp**.

idle [*seconds*]

Получить (или установить) значение (*seconds*) таймера бездействия на удаленной машине. Команда без параметров отображает текущее значение таймера.

image

Синоним **binary**.

lcd [*directory*]

Сменить текущий каталог на локальной машине. Если путь не указан, выполняется переход в домашний каталог пользователя.

ls [*remote-directory*] [*local-file*]

Отобразить содержимое каталогов на удаленной машине в формате, заданном удаленным ftp-узлом. Если не задан каталог (*remote-directory*), команда отображает содержание текущего рабочего каталога.

macdef *macro-name*

Определить макрокоманду. Последующие вводимые строки сохраняются в макрокоманде *macro-name*; пустая строка завершает ввод. Если макрокоманда содержит **$i**, то выполняется последовательный перебор аргументов из списка. Символ $ необходимо экранировать символом «\».

mdelete *remote-files*

Удалить указанные файлы (*remote-files*) на удаленной машине.

mdir *remote-files local-files*

Аналог команды **dir** с возможностью задания списка файлов на удаленной машине.

mget *remote-files*

Произвести расширение масок имен из *remote-files* на удаленной машине, а затем передать каждый из найденных файлов на локальную машину посредством команды **get**.

mkdir *directory-name*

Создать каталог (*directory-name*) на удаленной машине.

mls *remote-files local-file*

Аналог **nlist** с возможностью задания списка файлов на удаленной машине. Имя локального файла должно быть задано.

mode [*mode-name*]

Установить режим передачи файлов. Режим передачи по умолчанию – **stream** (поток).

modtime [*file-name*]

Отобразить время последнего редактирования файла на удаленной машине.

ftp

mput [*local-files*]

Произвести расширение имен файлов в параметре *local-files* и передать найденные файлы на удаленную машину посредством команды **put**.

newer *remote-file* [*local-file*]

Получить файл с удаленной машины, если локальный файл старше.

nlist [*remote-directory*] [*local-file*]

Отобразить содержимое каталога на удаленной машине в файл *local-file* (или на экран, если имя файла не задано). Если опущено имя удаленного каталога, отображается содержание текущего каталога.

nmap [*inpattern outpattern*]

Включить или выключить механизм разбора имен файлов. Преобразование происходит в соответствии с шаблонами *inpattern* (для имен получаемых файлов) и *outpattern* (для передаваемых файлов). Комбинации от **$1** до **$9** считаются переменными. Например, шаблон для получаемых файлов **$1.$2** при получаемом файле **readme.txt** установит значение **$1** в **readme**, а значение **$2** в **txt**. Шаблон для передаваемых файлов **$1.data** даст результат **readme.data**. **$0** соответствует полному имени файла. Шаблон [*string1*, *string2*] заменяется строкой *string1*, если она не пуста, иначе происходит замена на *string2*.

ntrans [*incharts* [*outchars*]]

Включить или выключить механизм преобразования имен файлов. В имени файла символы, присутствующие в списке символов *inchars*, заменяются соответствующими символами из списка *outchars*. Если аргументы опущены, механизм преобразования имен выключается. Если аргументы заданы:

- Символы в именах файлов на удаленной машине преобразуются в процессе выполнения команд **mput** и **put**, когда не указывается имя (или имена) конечного файла.
- Символы в именах файлов на локальной машине преобразуются в процессе выполнения команд **mget** и **get**, когда не указывается имя (или имена) локального файла.

open *host* [*port*]

Установить соединение с указанным узлом (*host*) FTP. Необязательный параметр *port* указывает **ftp**, что следует устанавливать соединение с сервером через указанный порт.

prompt

Переключение диалогового режима работы.

proxy *ftp-command*

Выполнить команду FTP через второе контрольное соединение. Например, одновременно послать команду двум различным удаленным узлам.

put *local-file* [*remote-file*]

Передать файл (*local-file*) на удаленную машину. Если имя конечного файла (*remote-file*) не задано, то в качестве такого имени используется исходное имя файла, обработанное в соответствии с установками **ntrans** и **nmap**. При передаче файла действуют текущие настройки типа, файла, структуры и режима передачи.

pwd

Отобразить имя текущего рабочего каталога на удаленной машине.

quit

Синоним **bye**.

quote *arg1 arg2...*

Послать перечисленные команды удаленному FTP-серверу.

recv *remote-file* [*local-file*]

Синоним **get**.

reget *remote-file* [*local-file*]

Получить файл (аналогично **get**). При указании имени локального файла (*local-file*) производится «докачка» (команду удобно использовать при обрыве связи или временном рассоединении).

remotehelp [*command-name*]

Запросить справку по существующим командам с удаленного FTP-сервера либо справку по указанной команде *command-name*.

remotestatus [*filename*]

Отобразить состояние удаленного узла или файла на удаленном узле, если указано имя файла (*filename*).

rename [*from*] [*to*]

Переименовать файл *from* на удаленной машине в файл *to*.

ftp	**reset** Повторно синхронизировать обмен с сервером. **restart** *marker* Продолжить передачу файла с определенного адреса (в байтах). **rmdir** [*directory-name*] Удалить указанный каталог на удаленной машине. **runique** Переключение режима сохранения файлов в локальной системе под уникальными именами. Если режим включен, файлы по необходимости переименовываются добавлением к имени символов **.1**, **.2** и т. д. По умолчанию режим выключен. **send** *local-file* [*remote-file*] Синоним **put**. **sendport** Переключение режима использования команд PORT. **site** [*command*] Получить/установить информацию, специфическую для удаленной машины. **size** *filename* Вернуть размер файла *filename* на удаленной машине. **status** Отобразить текущий статус **ftp**. **struct** [*struct-name*] Установить структуру передачи данных в *struct-name*. По умолчанию используется структура **stream** (поток). **sunique** Переключение режима сохранения файлов под уникальными именами в удаленной системе. **system** Отобразить тип операционной системы, используемой на удаленной машине. **tenex** Установить тип передаваемых файлов для обмена с машинами архитектуры TENEX.

ftp

trace

Переключение режима отслеживания маршрута пакетов.

type [*type-name*]

Установить тип передачи данных (*type-name*). Если параметр опущен, отображается текущий тип передачи. По умолчанию установлен режим ASCII.

umask [*mask*]

Установить указанную маску прав доступа для создаваемых файлов на удаленной машине. Если параметр опущен, отображается текущее значение маски.

user *username* [*password*] [*account*]

Передать серверу данные, идентифицирующие пользователя. Если пароль опущен, но необходим для доступа к серверу, сервер запросит пароль и дополнительный параметр account (при необходимости).

verbose

Переключение режима отображения максимального количества информации.

? [*command*]

Идентично **help**.

ftpd

in.ftpd [*options*]

Команда TCP/IP. Сервер протокола передачи файлов по Интернету (Internet File Transfer Protocol). Сервер использует протокол TCP и производит прием запросов через порт, указанный в спецификации сервисов **ftp**. **ftpd** запускается демоном **inetd**, в файле настройки которого, */etc/inetd.conf*, должна содержаться соответствующая запись.

Параметры

–d Запись отладочной информации в syslog.

–l Запись каждого FTP-сеанса в syslog.

–T*maxtimeout*

Установить максимальный интервал ожидания в секундах. По умолчанию интервал равен 15 минутам.

–t*timeout*

Установить интервал ожидания в секундах.

fuser

fuser [*options*] [*files* | *filesystems*]

Идентификация процессов, использующих перечисленные файлы или файловые системы. Отображаемая информация интерпретируется следующим образом: **c** – текущий каталог, **e** – исполняемый файл, **f** – открытый файл, **m** – разделяемая библиотека, **r** – корневой каталог. Выполнять команду **fuser** разрешено пользователям, у которых есть права на чтение файлов */dev/kmem* и */dev/mem*, но принудительно завершать процессы, принадлежащие другим пользователям, может только привилегированный пользователь. **fuser** не работает с файлами на удаленных файловых системах (NFS).

Если указано более одной группы файлов, параметры могут быть заданы отдельно для каждой группы. Стоящий отдельно символ «**–**» (дефис) отменяет действие текущих параметров, а для следующей группы файлов вступают в силу новые параметры.

Параметры

– Вернуться к параметрам по умолчанию.

–*signal*

Послать процессу сигнал *signal* вместо SIGKILL.

–a Отобразить информацию для всех перечисленных файлов, а не только для тех, с которым работает по меньшей мере один процесс.

–i Запрашивать подтверждение пользователя для принудительного завершения процесса. Параметр игнорируется, если не указан также ключ **–k**.

–k Послать сигнал SIGKILL всем процессам.

–l Отобразить список имен существующих сигналов.

–m Считать, что файлы (*files*) существуют на подмонтированной файловой системе; включать все файлы, принадлежащие этой файловой системе.

–s Подавление вывода при поиске.

–u Отображать имена владельцев процессов в скобках после идентификатора процесса.

–v Подробная информация.

–V Отобразить информацию о версии **fuser**.

g++ [*options*] *files* — **g++**

Выполнить **gcc** с параметрами, необходимыми для распознавания и работы с C++. **g++** опознает все расширения файлов, известные **gcc**, в дополнение к исходным файлам C++ (файлы *.C*, *.cc* или *.cxx*) и обработанным препроцессором C++ (файлы *.ii*). См. также **gcc**.

gated [*options*] — **gated**

Команда TCP/IP. Демон шлюзовой маршрутизации. **gated** работает с многочисленными протоколами маршрутизации и заменяет **routed** или любой другой демон, использующий протоколы Hello, EGP или BGP. В настоящий момент **gated** понимает протоколы RIP, BGP, EGP, Hello и OSPF, а также может быть настроен для работы с любой комбинацией этих пяти протоколов.

Параметры

–c Проверить файл настройки на наличие синтаксических ошибок, а затем завершить работу, сохранив результаты в файле */usr/tmp/gated_dump*.

–f *config_file*

Использовать альтернативный файл настройки. По умолчанию это */etc/gated.conf*.

–n Не изменять таблицу маршрутизации ядра.

–t [*trace_options*]

Выполнить **gated** с заданными режимами трассировки. Если флаги не указаны, использовать **general**. Флаги трассировки:

adv Управление блоками правил.

all Включает флаги **normal**, **policy**, **route**, **state**, **task** и **timer**.

general

Включает **normal** и **route**.

iflist

Список интерфейсов ядра.

normal

Обычные протоколы.

parse

Лексический анализ и разбор.

gated	**policy** Случаи, в которых правила применяются к импортируемым и экспортируемым маршрутам. **route** Любые изменения таблицы маршрутизации. **state** Состояние машинных переходов. **symbols** Имена, читаемые из ядра. Обратите внимание, что чтение происходит до разбора файла настройки, поэтому этот параметр должен быть указан в командной строке. **task** Системные задачи и интерфейсы. **timer** Использование таймера. **–C** Проверка файла настройки на наличие ошибок и завершение работы. Если ошибки были найдены, код возврата равен 1, иначе – 0. **–N** Работать в режиме приоритетной задачи, а не демона.
gawk	**gawk** [*options*] '*script*' [*var=value*...] [*files*] **gawk** [*options*] **–f** *scriptfile* [*var=value*...] [*files*] GNU-версия **awk** – программы, которая выполняет поиск по шаблону, обработку записей и предоставляет прочие формы работы с текстами. Более подробно этот инструмент описан в главе 13 «Язык сценариев gawk».
gcc	**gcc** [*options*] *files* Компиляция одного или более исходных файлов на языке C (*file.c*), ассемблера (*file.s*) или обработанных препроцессором исходных текстов C (*file.i*). Если суффикс файла не опознан, считать файл объектным или библиотекой. **gcc** автоматически запускает редактор связей **ld** (если не задан какой-либо из параметров **–c**, **–S** или **–E**). В некоторых случаях **gcc** создает объектный файл, имя которого образовано именем исходного файла и суффиксом *.o*. По умолчанию результат компиляции получает имя *a.out*. Системно-зависимые параметры **gcc** здесь не рассмотрены.

Примечание: **gcc** является GNU-вариантом **cc**; на большинстве Linux-систем по команде **cc** будет выполнен **gcc**. Команда **g++** выполняет **gcc** с набором параметров, необходимым для распознавания C++.

Параметры

–a Создавать отладочную информацию для основных блоков.

–ansi

Полное соответствие стандарту ANSI.

–b *machine*

Скомпилировать для использования на архитектуре *machine*.

–c Создавать связываемый объектный файл для каждого исходного файла, но не призводить связывание.

–dD

Отображать директивы **#define**.

–dM

Подавление нормального вывода. Отобразить последовательности **#define**, которые еще действуют в конце обработки препроцессором.

–dN

Отображать только имена макроподстановок **#define**, но не списки аргументов и не значения.

–fno-asm

Не считать **asm**, **inline** или **typeof** ключевыми словами. Подразумевается параметром **–ansi**.

–fno-builtin

Игнорировать имена встроенных функций, если они не начинаются с двух символов подчеркивания.

–fno-gnu-keywords

Не считать ключевыми словами **classof**, **headof**, **signature**, **sigof** и **typeof**.

–fno-ident

Игнорировать команды директивы **#ident**.

gcc	**–fsigned-bitfields** **–funsigned-bitfields** **–fno-signed-bitfields** **–fno-unsigned-bitfields** Установить знак битовых полей, если он не задан явно. **–fsigned-char** Считать тип **char** знаковым по умолчанию. **–fsyntax-only** Производить только проверку синтаксиса, но не собственно компиляцию. **–funsigned-char** Считать тип **char** беззнаковым по умолчанию. **–g** Включить отладочную информацию, используемую **gdb**. **–g***level* Степень подробности отладочной информации. Параметр *level* может принимать значения 1, 2 или 3. Значение 1 обеспечивает наименьшее количество отладочной информации. По умолчанию принимается уровень 2. **–idirafter** *dir* Включить *dir* в список каталогов поиска включаемых файлов, проверяемых в том случае, когда включаемый файл не найден в стандартных путях поиска. **–include** *file* Обрабатывать файл *file* прежде остальных. **–imacros** *file* Обрабатывать содержащий макросы файл *file* прежде остальных. **–iprefix** *prefix* При добавлении каталогов с помощью **–iwithprefix** предварять имена префиксом *prefix*. **–isystem** *dir* Добавить *dir* к каталогам, в которых производится поиск, если системный файл не найден в основных путях поиска. **–iwithprefix** *dir* Добавить *dir* к каталогам, в которых производится поиск, если заголовочный файл не найден в основных путях поиска. Если установлен **–iprefix**, добавить указанный префикс к имени *dir*.

–l*lib*

Связывание с библиотекой *lib*.

–nostartfiles

Указание редактору связей игнорировать стандартные стартовые файлы системных библиотек.

–nostdinc

Искать файлы заголовков только в указанных, но не в стандартных каталогах.

–nostdinc++

Запретить поиск в каталогах, содержащих заголовочные файлы C++.

–nostdlib

Запретить связывание со стандартными библиотеками.

–o *file*

Указать имя получаемого файла. По умолчанию – *a.out*.

–p Генерирование отладочной информации для использования с **prof**.

–pedantic

Выдавать предупреждения.

–pedantic-errors

Выдавать сообщение об ошибке в тех случаях, когда было бы сгенерировано предупреждение при использовании параметра **–pedantic**.

–pg

Генерирование отладочной информации для использования с **gprof**.

–pipe

Передавать информацию между проходами компиляции через конвейеры, а не через временные файлы.

–s Удалить из исполняемого файла все таблицы имен и информацию о смещениях.

–save-temps

Не удалять временные файлы после завершения компиляции.

–static

Запретить связывание с разделяемыми библиотеками.

gcc	**–traditional** Поведение, максимально близкое к поведению традиционного компилятора C. **–traditional-cpp** Поведение препроцессора, соответствующее традиционному языку C. **–trigraphs** Включить поддержку триграфов. **–u** *symbol* Предписание редактору связей искать имя *symbol* в библиотеках и связывать с библиотекой программу, если результат поиска положительный. **–undef** Определить только константы, требуемые стандартом языка, а не системно-зависимые константы вроде **unix**. **–v** Режим подробной диагностики. Команды отображаются по мере выполнения. Также отображается номер версии **gcc** и номер версии препроцессора. **–w** Подавить вывод предупреждений. **–x** *language* Считать, что исходные файлы содержат текст на языке *language*. Аргумент может быть равен **c**, **objective-c**, **c-header**, **c++**, **cpp-output**, **assembler** или **assembler-with-cpp**. Значение **none** является предписанием для автоматического распознавания. **–A***question*(*answer*) Препроцессор, встретив условную директиву *question*, например **#if** *question*, добавляет в ответ указание *answer*. Отключение стандартных указаний производится параметром **–A–**. **–B***path* Указать каталог *path*, в котором расположены файлы компилятора. **–C** Указание препроцессору сохранять комментарии. Имеет смысл использовать только с параметром **–E**. **–D***name*[=*def*] Определить имя *name* как имеющее значение *def* (по аналогии с директивой **#define**). Если значение (=*def*) опущено, *name* получает значение 1. **–D** имеет меньший приоритет , чем **–U**.

–E Обрабатывать исходные тексты препроцессором, но не производить компиляцию. Отображать результаты на стандартный вывод.

–I*dir*

Включить *dir* в список каталогов, в которых выполняется поиск включаемых файлов. Если *dir* имеет значение «–», искать в тех каталогах, которые заданы параметром **–I** *раньше* параметра **–I–**, причем только для директив вида **#include** "*file*", но не **#include** <*file*>.

–L*dir*

В дополнение к стандартным каталогам производить поиск в каталоге *dir*.

–M Вместо компиляции создать, основываясь на директивах **#include**, правило, описывающее зависимости исходного файла и пригодное для включения в файл сборки (makefile). Подразумевается параметр **–E**.

–MD

Действует аналогично **–M**, но информация о зависимостях записывается в файлы с расширением *.d* и выполняется обычная компиляция.

–MG

Применяется совместно с **–M** или **–MM**. Не выводить сообщение об ошибке, если не найден включаемый файл. Используется, если включаемый файл автоматически генерируется при сборке.

–MMD

Аналогично **–MD**, но записывается информация только о включаемых файлах пользователя, а не о системных включаемых файлах.

–MM

Действует аналогично **–M**, но правило ограничивается нестандартными включаемыми файлами, т. е. определяемые директивой **#include** "***file***", но не **#include** <*file*>.

–H Отображать пути включаемых файлов (по одному в строке) в стандартный поток ошибок.

–O[*level***]**

Оптимизация. Параметр *level* может принимать значения 1, 2, 3 или 0. По умолчанию значение *level* равно 1. 0 – оптимизация отключена; 3 – максимальная оптимизация.

gcc

–P Обработка препроцессором без генерирования информации управления строками для следующего прохода компилятора C. Имеет смысл использовать только совместно с **–E**.

–S Транслировать исходный текст в ассемблерный, но не вызывать ассемблер.

–U*name*

Удалить существующее определение имени *name*, которое может являться зарезервированным и предварительно определенным препроцессором либо именем, определенным параметром **–D**. **unix** и **i386** – это имена, предопределяемые **cpp**.

–V *version*

Попытаться выполнить **gcc** указанной версии.

–W

Более подробные, чем обычно, предупреждения.

–Wl,*option*

Запустить редактор связей с параметром *option*, который может быть списком параметров, разделяемых запятыми.

–Wa,*option*

Вызвать ассемблер с параметрами *option* (список параметров разделяется запятыми).

–Waggregate-return

Выдавать предупреждения об объявленных или вызываемых функциях, возвращающих экземпляры структур или объединений.

–Wall

Включить параметры **–W**, **–Wchar-subscripts**, **–Wcomment**, **–Wformat**, **–Wimplicit**, **–Wparentheses**, **–Wreturn-type**, **–Wswitch**, **–Wtemplate-debugging**, **–Wtrigraphs**, **–Wuninitialized** и **–Wunused**.

–Wcast-align

Выдавать предупреждение о приведении указателя к типу с выравниванием по увеличенной границе.

–Wcast-qual

Выдавать предупреждение о приведении указателя к типу с потерей квалификатора исходного типа.

–Wchar-subscripts

Выдавать предупреждение, если индекс массива имеет тип **char**.

–Wcomment

Выдавать предупреждение, когда встречается вложенный комментарий.

–Wconversion

Выдавать предупреждения о некоторых вариантах приведения типов данных.

–Werror

Завершать работу при первой встреченной ошибке.

–Wformat

Выдавать предупреждение о неверном задании формата в функциях **printf** и **scanf**.

–Wimplicit

Выдавать предупреждение о неявно определяемой функции или параметре.

–Winline

Выдавать предупреждение о неверных inline-функциях.

–Wmissing-declarations

Выдавать предупреждение о наличии определения глобальной функции, для которой отсутствует объявление.

–Wmissing-prototypes

Предупреждать об отсутствии прототипов для определений глобальных функций.

–Wnested-externs

Выдавать предупреждение, если ключевое слово **extern** встречается внутри функции.

–Wno-import

Не выдавать предупреждение об использовании **#import**.

–Wp,*options*

Передача параметров (*options*) препроцессору. Множественные параметры разделяются запятыми. Параметр не относится к параметрам предупреждений.

–Wparentheses

Более подробные предупреждения об опущенных скобках.

gcc	**–Wpointer-arith** Выдавать предупреждение о попытке определить размер функции или типа void. **–Wredundant-decls** Выдавать предупреждение, если какой-либо объект объявляется дважды в пределах одного блока видимости. **–Wreturn-type** Выдавать предупреждения о функциях, объявленных без возвращаемого типа или с неверным возвращаемым типом. **–Wshadow** Выдавать предупреждение о перекрытии имен локальных переменных. **–Wstrict-prototypes** Типы аргументов функций должны быть указаны и в объявлении, и в определении функции. **–Wswitch** Выдавать предупреждение, если в операторе switch обрабатываются не все возможные значения перечислимого типа. **–Wtemplate-debugging** Выдавать предупреждение о невозможности отладки для шаблонов C++. **–Wtraditional** Выдавать предупреждение о наличии кода, работающего различным образом в традиционном C и в ANSI C. **–Wtrigraphs** Выдавать предупреждение о наличии триграфов. **–Wuninitialized** Выдавать предупреждение о наличии неинициализированных автоматических переменных. **–Wunused** Выдавать предупреждение о наличии неиспользуемых переменных и функций. **–Xlinker** *option* Передать параметр (*option*) редактору связей. Параметр с аргументом должен передаваться парой параметров **–X**, где первый определяет параметр, а второй определяет аргумент.

Директивы pragma

gcc

#pragma interface [*header-file*]

Применяется в файлах заголовков для включения в объектные файлы ссылок на определения, а не самих определений. Директива является специфичной для C++.

#pragma implementation [*header-file*]

Размещается в основном файле программы для принудительного генерирования полной информации о файле *header-file* (если имя опущено, используется заголовочный файл с таким же именем, как у файла, содержащего директиву). Информация становится доступной глобально. Обычно указанный файл содержит директиву **#pragma interface**.

---

gdb [*options*] [*program* [*core*|*pid*]]

gdb

Отладчик GDB (GNU DeBugger) обеспечивает пошаговое выполнение программ на C, C++ или Modula-2 с целью локализации причины сбоя. Имя программы для отладки, как правило, указывается в командной строке. Можно также указать образ памяти (*core*) завершившейся задачи или идентификатор работающего процесса (*pid*).

Параметры

–s *file*, **–symbols=***file*

Прочитать таблицу имен из файла *file*. При указании совместно с параметром **–e** файл также считается исполняемым.

–e *file*, **–exec=***file*

Использовать указанный файл в качестве исполняемого в паре с исходными текстами программы. В сочетании с параметром **–s** может применяться для чтения имен из файла.

–c *file*, **–core=***file*

Прочитать информацию из файла *file*, считая его образом оперативной памяти.

–x *file*, **–command=***file*

Выполнить команды **gdb** из файла *file*.

–d *directory*, **–directory=***directory*

Включить каталог *directory* в пути поиска файлов с исходными текстами программы.

gdb

–n, –nx

Игнорировать файл *.gdbinit*.

–q, –quiet

Подавить отображение вводной информации и сведений о правообладании.

–batch

Завершить работу после выполнения всех команд из файла *.gdbinit* и файлов, указанных в параметре **–x**. Не отображать никаких сообщений при старте.

–cd=*directory*

Сделать *directory* рабочим каталогом **gdb**.

–f, –fullname

Отображать полное имя файла и номер строки для каждого уровня стека.

–b *bps*

Установить скорость работы последовательного устройства, используемого GDB, равной значению *bps*.

–tty=*device*

Сделать устройство *device* стандартным вводом и выводом.

Общие команды

Вот некоторые из основных команд **gdb**; общее число команд слишком велико, чтобы приводить их здесь.

bt Отобразить текущее положение в программе и маршрут, приведший выполнение в эту точку. (Действие идентично работе команды **where**.)

break

Установить точку останова программы.

cd Сменить текущий рабочий каталог.

clear

Удалить точку останова, в которой только что произошло прерывание выполнения.

commands

Перечислить команды, выполняемые по прерыванию в точке останова.

c Продолжить выполнение с точки останова.

delete

Удалить точку останова или точку наблюдения; также применяется в сочетании с другими командами.

display

Отобразить значение переменных или выражений при останове программы.

down

Передвинуться вниз по стеку, чтобы сделать текущей другую функцию.

frame

Выбрать процедурную секцию для следующей команды **continue**.

info

Отображение информации о **gdb**. Например, **info breakpoints** перечисляет установленные точки останова и наблюдения.

jump

Продолжить выполнение программы с другой строки в исходном файле.

kill Завершить процесс, порожденный **gdb**.

list Отобразить содержимое исходного файла, соответствующего исполняемой в настоящий момент программе.

next

Выполнить следующую строку исходного текста, вызвав все необходимые функции.

print

Отобразить значение переменной или выражения.

pwd

Отобразить имя текущего рабочего каталога.

ptype

Отобразить содержимое типа данных, например, структуры или класса C++.

quit

Завершение работы **gdb**.

reverse-search

Обратный поиск по регулярному выражению в исходном тексте программы.

run Выполнить программу.

gdb	**search** Поиск по регулярному выражению в исходном тексте программы. **set variable** Присвоить значение переменной. **signal** Послать сигнал работающему процессу. **step** Выполнить следующую строку исходного текста программы, по необходимости переходя внутрь тела функции. **undisplay** Отменить действие команды **display**; не отображать выражения. **until** Завершить текущий цикл. **up** Передвинуться вверх по стеку, сделав текущей другую функцию. **watch** Установить точку наблюдения (т. е. точку останова для данных) в программе. **whatis** Отобразить тип переменной или функции.
gdc	**gdc** [*options*] *command* Команда TCP/IP. Используется для администрирования **gated**. Различные команды останавливают и запускают демон, посылают ему сигналы, модифицируют файлы настройки, а также управляют состояниями и распечаткой содержимого образа памяти. **Параметры** **–c** *size* Указать максимальный размер сохраняемого образа памяти (core dump). **–f** *size* Указать максимальный размер сохраняемых данных (file dump).

–m *size*

Указать максимальный размер сегмента данных.

–n Запретить редактирование таблицы ретрансляции ядра.

–q Тихий режим. Предупреждения не выводятся, сообщения об ошибках направляются в **syslogd**, а не в стандартный поток ошибок.

–s *size*

Указать максимальный размер стека.

–t *seconds*

Выждать указанное количество секунд (по умолчанию равно 10). Время отводится на завершение операций **gated** при запуске и останове.

Команды

BACKOUT

Восстановить файл */etc/gated.conf* из файла */etc/gated.conf–*, независимо от того, существует ли последний.

backout

Восстановить файл */etc/gated.conf* из файла */etc/gated.conf–*, предполагая, что последний существует.

checkconf

Сообщить о любых синтаксичских ошибках в файле */etc/gated.conf*.

checknew

Сообщить о любых синтаксичских ошибках в файле */etc/gated.conf+*.

COREDUMP

Принудительно создать файл образа памяти и завершить работу.

createconf

Создать пустой файл */etc/gated.conf+*, если таковой еще не существует, и установить права доступа в 664 с владельцем **root** и группой **gdmaint**.

dump

Принудительный сброс данных в файл */usr/tmp/gated_dump* и продолжение нормальной работы.

interface

Перезагрузить настройки интерфейса.

gdc

KILL

Немедленное (и неэлегантное) принудительное завершение.

modeconf

Установить права доступа для всех файлов настройки в режим 664 с владельцем **root** и группой **gdmaint**.

newconf

Убедиться, что */etc/gated.conf+* существует, и перенести его содержимое в */etc/gated.conf*. Старый файл */etc/gated.conf* сохраняется под именем */etc/gated.conf–*.

reconfig

Перезагрузить файл настройки.

restart

Завершить и перезапустить **gated**.

rmcore

Удалить все найденные файлы образов.

rmdmp

Удалить все найденные файлы образов состояния.

rmparse

Удалить все файлы **gated**, в которых найдены ошибки разбора. Они генерируются командами **checkconf** и **checknew**.

running

Завершение с кодом 0, если **gated** запущен, и с ненулевым кодом – в противном случае.

start

Выполнить **gated**, если он еще не запущен. В последнем случае вернуть ошибку.

stop

Завершить выполнение **gated** как можно мягче.

term

Принудительное мягкое завершение.

toggletrace

Переключение трассировки.

gdc

Файлы

/etc/gcd.conf+

Испытательный файл настройки. Если конфигурация работает удовлетворительно, необходимо выполнить *gated newconf*, чтобы испытательный файл стал постоянным файлом настройки */etc/gated.conf*.

/etc/gated.conf–

Резервная копия старого файла настройки.

/etc/gated.conf––

Резервная копия резервной копии старого файла настройки.

/etc/gated.conf

Собственно файл настройки.

/etc/gated.pid

Идентификатор процесса **gated**.

/usr/tmp/gated_dump

Файл состояния.

/usr/tmp/gated_parse

Перечень ошибок разбора, найденных при чтении файла настройки.

getkeycodes

getkeycodes

Команда выдает таблицы преобразования скан-кодов в коды клавиш.

getty [*options*] *port* [*speed* [*term* [*lined*]]]

getty

Команда системного администрирования. Установка типа терминала, режимов, скорости и характеристик линии. На системах Linux иногда для этой цели используется **agetty**, что подразумевает отличный синтаксис. **getty** запускается программой **init**. Это второй из процессов в последовательности **init-getty-login-shell**, которая в конечном итоге соединяет пользователя с системой. **getty** считывает имя пользователя и выполняет команду **login** с именем пользователя в качестве аргумента. В процессе считывания имени **getty** пытается адаптировать систему к скорости и типу установленного устройства.

getty

Необходимо указать порт, который используется **getty** для прикрепления к устройству */dev/port*. После прикрепления **getty** просматривает файл значений, заданных по умолчанию (как правило, это */etc/default/getty*), считывая значения и параметры работы. Большинство параметров могут быть указаны также и в командной строке, но значения из этого файла имеют больший приоритет. Аргумент *speed* служит для указания на одну из записей в */etc/gettydefs*, которая содержит начальную скорость, настройки терминала и приглашения login, а также конечную скорость и параметры соединения. По умолчанию производится чтение первой такой записи из */etc/gettydefs*. *term* определяет тип терминала, а *lined* – необязательные характеристики линии.

Параметры

–c *file*

Проверка файла *gettydefs*. Именем такого файла является аргумент *file*. Значения из файла и сообщения об ошибках разбора выдаются на стандартный вывод.

–d *file*

Назначить указанный файл как файл значений по умолчанию.

–h Не выполнять принудительное рассоединение порта при инициализации.

–r *delay*

Ожидать появления единственного символа из порта, затем еще *delay* секунд перед продолжением работы.

–t *timeout*

Если имя пользователя не введено в течение *timeout* секунд, закрыть соединение.

–w *string*

Ожидать появления *string* символов из порта перед продолжением работы.

gprof

gprof [*options*] [*object_file*]

Отобразить профилировочную информацию для объектного файла. Таблица имен файла сравнивается с таблицей вызовов файла *gmon.out* (предварительно созданного компиляцией **gcc –pg**).

gprof

Параметры

–a Не отображать информацию по статическим функциям. Поскольку эти данные могут представлять интерес, добавлять их к информации о функциях, загруженных непосредственно перед статическими.

–b Не отображать сведения по каждому из полей в записи профилировки.

–c Попытаться определить статистику вызовов статических функций в программе по текстовому сегменту объектного файла. Отображать только статических предков и порожденных потомков с нулевым числом вызовов.

–e *routine*

Не отображать информацию по *routine* и порождаемым объектам.

–f *routine*

Отображать только информацию по *routine*, а также время, затраченное на выполнение всех вызовов.

–k *from to*

Удалить арки между вызовами *from* и *to*.

–s Обобщить профилировочную информацию в файле *gmon.sum*.

–v Отобразить номер версии и завершить работу.

–z Включать неиспользованные вызовы.

–E *routine*

Не отображать информацию о вызовах *routine* и порождаемых объектах или не включать затраченное на них время в суммарное время вызовов.

–F *routine*

Отображать информацию только о вызовах *routine*. Не включать время, потраченное на выполнение прочих функций.

grep [*options*] *pattern* [*files*]

grep

Поиск в перечисленных файлах строки текста, соответствующей регулярному выражению *pattern*. Регулярные выражения подробно рассмотрены в главе 9 «Шаблоны и поиск». Код завершения: 0 – если найдена хотя бы одна строка, удовлетворяющая критерию поиска, 1 – если такие строки не найдены, 2 – если произошла ошибка. См. также **egrep** и **fgrep**.

grep

Параметры

–a, ––text

Разрешить вывод строк с двоичными данными; считать их обычным текстом.

–b, ––byte-offset

Отображать для каждой печатаемой строки смещение (в байтах) от начала исходного файла.

–c, ––count

Отображать только число строк, соответствующих шаблону поиска. При использовании совместно с параметром **–v** или **––revert-match** отображается число строк, не соответствующих шаблону.

–d *action*, **––directories=***action*

Определить правило *action* для обработки каталогов. Возможные варианты:

read

Читать каталоги как обычные файлы (режим по умолчанию).

skip

Не обрабатывать каталоги.

recurse

Выполнять рекурсивную обработку каталогов. Идентично заданию **–r**.

–e *pattern*, **––regexp=***pattern*

Поиск по шаблону *pattern*. Применяются шаблоны, начинающиеся с символа «–».

–f *file*, **––file=***file*

Брать список шаблонов из файла (*file*). Каждый шаблон располагается в отдельной строке файла.

–h, ––no-filename

Отображать найденные строки, но не имена файлов (обратно действию параметра **–l**).

–i, ––ignore-case

Не различать регистр символов.

–l, ––files-with-matches

Отображать имена файлов, но не найденные строки. При поиске в одном файле работа прекращается по нахождению первой строки.

–n, ––line-number

Отображать строки и номера строк.

–q, ––quiet, ––silent

Подавлять нормальный вывод, поиск прекращается по нахождению первой строки.

–r, ––recursive

Рекурсивно обрабатывать файлы в каталогах. Идентично заданию **–d recurse**.

–s, ––no-messages

Подавлять сообщения о ненайденных или недоступных файлах.

–v, ––revert-match

Отображать все строки, которые не совпадают с шаблоном.

–w, ––word-regexp

Искать только целые слова, совпадающие с шаблоном. Слова состоят из букв, цифр и символов подчеркивания, остальные символы являются разделителями слов.

–x, ––line-regexp

Отображать строку только в том случае, если она целиком совпадает с шаблоном.

–A *num*, **––after-context=***num*

Вывести *num* строк текста после найденной строки.

–B *num*, **––before-context=***num*

Вывести *num* строк текста перед найденной строкой.

–C[*num***], ––context=[***num***], –***num*

Отображать *num* строк, предшествующих найденной, и *num* строк, следующих за ней. По умолчанию объем контекста равен 2 строкам.

–L, ––files-without-match

Перечислить имена файлов, поиск в которых не дал положительных результатов.

–V, ––version

Отобразить номер версии и завершить работу.

Примеры

Перечислить пользователей, применяющих интерпретатор **tcsh**:

```
grep -c /bin/tcsh /etc/passwd
```

grep

Перечислить включаемые файлы, содержащие хотя бы одну директиву **#include**:

```
grep -l '^#include' /usr/include/*
```

Перечислить файлы, которые не содержат регулярное выражение *pattern*:

```
grep -c pattern files | grep :0
```

groff

groff [*options*] [*files*]
troff [*options*] [*files*]

Интерфейс к системе форматирования документов **groff**, который обычно исполняет **troff** и производит необходимую последующую обработку для указанного устройства. Параметры без аргументов можно группировать после одного отдельно стоящего символа «–». Тот же символ вместо имени файла указывает на стандартный ввод.

Параметры

–a Создавать ASCII-приближение форматированного вывода.

–b Отображение трассы (backtrace) для каждого предупреждения или ошибки.

–C Режим совместимости.

–d*cs*, **–d***name*=*s*

Определить символ *c* или строку *name* со значением *s* (где *s* – строка).

–e Предварительная обработка **eqn**.

–E Не отображать сообщения об ошибках.

–f*fam*

Определить используемое по умолчанию семейство шрифтов.

–F*dir*

Искать *DESC* и шрифтовые файлы в указанном каталоге и лишь затем в каталоге по умолчанию */usr/lib/groff/font*.

–h Получение помощи по **groff**.

–i Читать стандартный ввод после того, как все файлы будут обработаны.

–l Вывести результаты работы на принтер (посредством команды печати, приведенной в файле описания устройства).

–L*arg*

Поместить аргумент *arg* в буферную зону (spooler). Каждый аргумент следует передавать с отдельным параметром **–L**.

–m*name*

Прочитать макрокоманды из файла *tmac.name*.

–M*dir*

Искать файлы макрокоманд сначала в указанном каталоге и лишь затем в каталоге по умолчанию */usr/lib/groff/tmac*.

–n*num*

Установить номер первой страницы в *num*.

–N Исключить символ новой строки из списка допустимых разделителей **eqn**. Действует эквивалентно **eqn –N**.

–o*list*

Выводить только страницы, перечисленные в списке *list*. Значения разделяются запятыми и могут задаваться интервалами.

–p Предварительная обработка **pic**.

–P*arg*

Передать аргумент *arg* пост-обработчику. Каждый аргумент следует передавать с отдельным параметром **–P**.

–r*cn*, **–***name*=*n*

Установить значение численного регистра *c* или имени *name*. Параметр *c* представляет собой единственный символ, *n* может быть произвольным численным выражением **troff**.

–R Предварительная обработка **refer**.

–s Предварительная обработка **soelim**.

–S Безопасный режим (т. е. параметр **–S** передается **pic**, а макрокоманда **–msafer** используется в **troff**).

–t Предварительная обработка **tbl**.

–T*dev*

Подготовить результат к выводу на устройство *dev*. По умолчанию это устройство **ps**.

groff

–v Выдавать версии программ, запускаемых **groff**.

–V Не исполнять конвейерную последовательность команд, а выдать ее на стандартный вывод.

–w*name*

Включить предупреждения вида *name*. Может быть задано произвольное число параметров **–w**. Виды предупреждений перечислены в руководстве по команде **troff**.

–W*name*

Отключить предупреждения вида *name*. Может быть задано произвольное число параметров **–W**. Виды предупреждений перечислены в руководстве по команде **troff**.

–z Подавить выдачу информации **troff** (кроме сообщений об ошибках).

–Z Не выполнять дополнительную обработку результатов, выдаваемых **troff**. По умолчанию соответствующая пост-обработка производится **groff** автоматически.

Устройства

ascii

Устройство, сходное с печатной машинкой.

dvi dvi-формат системы TEX.

latin1

Устройство, сходное с печатной машинкой. Таблица символов ISO Latin-1.

ps PostScript.

X75

Устройство X11 для просмотра с разрешением 75 dpi.

X100

Устройство X11 для просмотра с разрешением 100 dpi.

lj4 Устройство, совместимое с принтером HP LaserJet4 (и другими PCL5-совместимыми устройствами).

Переменные окружения

GROFF_COMMAND_PREFIX

Если переменная имеет значение **X**, **groff** использует **Xtroff** вместо **troff**.

groff

GROFF_FONT_PATH

Список каталогов (элементы разделяются двоеточием), в которых выполняется поиск каталогов с именем *devname*.

GROFF_TMAC_PATH

Список каталогов (элементы разделяются двоеточием), в которых происходит поиск файлов макрокоманд.

GROFF_TMPDIR

Если переменная установлена, то при работе в этом каталоге создаются временные файлы, в противном случае либо в каталоге, определяемом переменной TMPDIR (если она установлена), либо в каталоге */tmp*.

GROFF_TYPESETTER

Устройство по умолчанию.

PATH

Пути поиска команд, выполняемых **groff**.

---

groupadd

groupadd [*options*] *group*

Команда системного администрирования. Создать новую группу пользователей *group*.

Параметры

–g *gid*

Определить численный идентификатор группы. (По умолчанию это первое доступное число, большее 500.) Значение должно быть уникальным, за исключением случаев, когда указывается параметр **–o**.

–o Использовать неуникальный идентификатор, указанный в параметре **–g**.

---

groupdel

groupdel *group*

Команда системного администрирования. Удаление учетной записи для указанной группы. Может потребоваться найти и изменить права доступа для файлов, принадлежащих удаленной группе.

---

groupmod

groupmod [*options*] *group*

Команда системного администрирования. Изменение информации об указанной группе.

groupmod	**Параметры** **–g** *gid* Изменить численное значение идентификатора группы. Все файлы, имеющие прежний идентификатор группы, необходимо обработать вручную. Новый идентификатор группы должен быть уникальным, если только не указан параметр **–o**. **–n** *name* Изменить имя группы на *name*. **–o** Обход ограничений. Принять неуникальный идентификатор *gid* в качестве нового.
groups	**groups** [*options*] [*users*] Отображение групп, к которым принадлежат пользователи *users*. Команда без параметров выводит список групп, к которым принадлежит текущий пользователь. Группы пользователей перечислены в файлах */etc/passwd* и */etc/group*. **Параметры** **––help** Отобразить справку. **––version** Вывести информацию о версии программы.
grpck	**grpck** [*option*] [*files*] Команда системного администрирования. Удалить некорректные или дублирующиеся записи из файлов */etc/group* и */etc/gshadow*. Выдавать предупреждения, если найдены другие ошибки. Перед удалением записи **grpck** запрашивает подтверждение. При отрицательном ответе работа завершается. В пакетном режиме автоматически даются отрицательные ответы на все запросы. Также можно проверять альтернативные файлы (*files*) group и gshadow. При наличии других ошибок лучше использовать **groupmod**. **Параметр** **–n** Пакетный режим. **Коды завершения** **0** Успешное завершение.

grpck

1 Синтаксическая ошибка.

2 Одна или более некорректных записей для групп.

3 Невозможно открыть файлы групп.

4 Невозможно заблокировать файлы групп.

5 Запрещена запись в файлы групп.

grpconv

grpconv
grpunconv

Команда системного администрирования. Аналогична **pwconv**; создает скрытый файл группы, чтобы защитить зашифрованные пароли группы от взлома специальными программами. **grpconv** создает файл */etc/gshadow* на основе существующего */etc/groups* и заменяет шифрованные пароли символом **x**. Если в файл */etc/groups* добавляются новые записи, команда **grpconv** может быть выполнена снова, чтобы добавить новую информацию в */etc/gshadow*. При этом будут игнорироваться записи, содержащие **x** вместо пароля, а все остальные будут преобразованы. Команда **grpunconv** выполняет обратную операцию – создает */etc/groups* на основе */etc/gshadow*, удаляя при этом последний файл.

gs

gs [*options*] [*files*]

Интерпретатор языков PostScript (Adobe Systems) и PDF (Portable Document Format); применяется для обработки документов. Если вместо имени файла подставляется параметр «–», происходит чтение со стандартного ввода.

Параметры

––*filename arg1*...

Первый аргумент параметра считается именем файла, все последующие заносятся не в системный (systemdict), а в пользовательский словарь (userdict) массивом строк, прежде чем начнется обработка файла.

–g*number1***x***number2*

Задать ширину и высоту устройства; параметр предназначается для систем подобных X Window.

–q Тихий запуск.

gs

–*rnumber*, **–r***number1***x***number2*

Установить вертикальное и горизонтальное разрешения (для устройств, способных менять разрешение вывода, например, принтеров). Если задано только одно число, оно используется для инициализации и вертикального, и горизонтального разрешения.

–D*name*=*token*, **–d***name*=*token*

Определить в системном словаре systemdict имя и его значение (*token*). Значение должно быть только одно (это определяется оператором token) и не должно содержать пробелов.

–D*name*, **–d***name*

Определить в системном словаре (systemdict) имя с пустым значением.

–I*directories*

Добавить список указанных каталогов в начало списка путей поиска файлов библиотек.

–S*name*=*string*, **–s***name*=*string*

Определить в системном словаре имя со значением, соответствующим заданной строке.

Специальные имена

–dDISKFONTS

Изображения символа, который встречается в тексте впервые, подгружаются с диска.

–dNOBIND

Подавить действие оператора **bind**. Используется только при отладке.

–dNOCACHE

Отключить буферизацию символов. Используется только при отладке.

–dNODISPLAY

Не выполнять обычную инициализацию устройства, на которое выдается результат. Может быть полезно при отладке.

–dNOPAUSE

Не делать паузу и не выдавать запрос на продолжение в конце каждой страницы.

gs

–dNOPLATFONTS

Подавить использование шрифтов, установленных на текущей платформе (например, X Window).

–dSAFER

Подавить действие операторов **deletefile** и **renamefile** и запретить открытие файлов не в режиме read-only (только для чтения).

–dWRITESYSTEMDICT

Разрешить запись в системный словарь (systemdict).

–sDEVICE=*device*

Сделать указанное устройство основным устройством вывода.

–sOUTPUTFILE=*filename*

Сделать указанный файл или конвейер (pipe) основным устройством вывода.

---

gunzip [*options*] [*files*]

gunzip

Распаковка файлов, сжатых программой **gzip**. Перечень параметров приведен в описании **gzip**.

---

gzexe [*options*] [*files*]

gzexe

Сжатие исполняемых файлов. При запуске сжатых файлов происходит их автоматическое разворачивание в памяти. В результате снижается производительность, но выигрывается место для хранения файлов. Команда **gzexe** создает резервные копии файлов (добавляя к имени файла символ «~» (тильда)), которые следует удалить после проверки корректности работы сжатого файла.

Параметр

–d Развернуть сжатый файл.

---

gzip [*options*] [*files*]
gunzip [*options*] [*files*]
zcat [*options*] [*files*]

gzip

Сжатие перечисленных файлов (или данных со стандартного ввода) кодированием Лемпеля-Зива (Lempel-Ziv, LZ77). Сжатый файл переименовывается в *filename.gz*; сохраняются все права доступа, информация о владельцах, о времени

gzip

изменения файла и времени доступа к нему. Символические ссылки не обрабатываются. Распаковка сжатых файлов производится с помощью **gunzip**, который опознает большинство параметров **gzip**. Вызов **zcat** идентичен **gunzip –c**. Из параметров, описанных ниже, **zcat** опознает параметры **–fhLV**. Этими программами могут быть распакованы файлы, сжатые командой **compress**.

Параметры

–*n*, ––fast, ––best

Регулировка скорости сжатия по указанной цифре *n*. Режимы **–1** или **––fast** означают самый быстрый метод сжатия (коэффициент сжатия уменьшается), а **–9** или **––best** – самый медленный и самый эффективный метод. По умолчанию уровень сжатия равен **–6**.

–a, ––ascii

Текстовый ASCII-режим. Преобразование концов строк в соответствии с локальными настройками. Этот режим поддерживается лишь в некоторых не-Unix системах.

–c, ––stdout, ––to-stdout

Отображать результаты работы на стандартный вывод, не менять исходные файлы.

–d, ––decompress, ––uncompress

Идентично **gunzip**.

–f, ––force

Принудительное сжатие. Обычно **gzip** запрашивает подтверждение для продолжения операций, если файл имеет несколько ссылок либо для файла уже существует сжатая *.gz*-версия, либо если на вход поступают уже сжатые данные.

–h, ––help

Отобразить информацию по использованию **gzip** и завершить работу.

–l, ––list

Указывает, что аргументами являются сжатые файлы. Файлы могут быть сжаты любым из следующих методов: **gzip**, **deflate**, **compress**, **lzh** и **pack**. Для каждого имеющегося файла выдать полный и упакованный размеры (последний всегда равен –1, если сжатие производилось не **gzip**), степень сжатия и имя. При использовании совместно с параметром **–v** также отображается метод сжатия, контрольная сумма (32-битная CRC) распакованных дан-

gzip

ных и информация о времени для файла. Если также указан параметр **–N**, то отобразить имя несжатого файла и информацию о времени.

–L, ––licence

Отобразить лицензию распространения **gzip** и завершить работу.

–n, ––no-name

При сжатии не сохранять исходное имя файла и информацию о времени. При распаковке не восстанавливать исходное имя и информацию о времени, если они присутствуют. При распаковке это режим по умолчанию.

–N, ––name

Режим по умолчанию. Сохранять исходное имя файла и информацию о времени. При распаковке восстанавливать эти параметры.

–q, ––quiet

Не выводить предупреждения.

–r, ––recursive

Если в качестве аргумента задано имя каталога, рекурсивно обрабатывать все файлы в этом каталоге.

–S *suffix*, **––suffix** *suffix*

Добавлять суффикс (.*suffix*) к имени сжатого файла. Суффикс по умолчанию – **gz**. Если задан пустой суффикс в режиме распаковки, **gzip** будет пытаться распаковать все файлы.

–t, ––test

Проверить целостность сжатого файла.

–v, ––verbose

Отображать имя и процентное уменьшение размера для каждого файла.

–V, ––version

Вывести номер версии и параметры компиляции **gzip**, затем завершить работу.

---

halt [*options*]

halt

Команда системного администрирования. Вставить уведомление в файл */var/log/wtmp*; если система работает в режиме (runlevel) 0 или 6, остановить все процессы; в противном случае выполнить **shutdown –nf**.

halt

Параметры

–d Подавить запись в */var/log/wtmp*.

–f Принудительно выполнить **halt** вместо вызова **shutdown –nf** (т. е. выполнять **halt** даже тогда, когда система не работает в режиме 0 или 6).

–n Запретить обычный вызов **sync**.

–w Подавить нормальное выполнение; просто добавить запись в */var/log/wtmp*.

head

head [*options*] [*files*]

Отображение нескольких первых строк (по умолчанию – 10) указанных файлов (*files*). Если опущены имена файлов или задан параметр «–», выполняется чтение со стандартного ввода. Если файлов больше одного, отображаются первые строки каждого из файлов.

Параметры

–c *num*[**bkm**], **––bytes** *num*

Отобразить первые *num* байт файла или первые *num* блоков по 512 байт, 1 Кбайт или 1 Мбайт при модификаторах **b**, **k** и **m**, соответственно.

––help

Вывести справку по использованию и завершить работу.

–n *num*, **––lines** *num*, **–***num*

Отобразить *num* первых строк. По умолчанию – 10.

–q, **––quiet**, **––silent**

«Тихий» режим; не выводить заголовки (имена) указанных файлов.

–v, **––verbose**

Вывести заголовки (имена) файлов, даже если задано всего одно имя файла.

––version

Отобразить информацию о версии **head** и завершить работу.

Примеры

Отобразить первые 20 строк файла **phone_list**:

```
head -20 phone_list
```

head

Отобразить первую десятку телефонных номеров с кодом района 202:

```
grep '(202)' phone_list | head
```

host

host [*options*] *host* [*server*]
host [*options*] *zone* [*server*]

Команда системного администрирования. Вывод информации по указанному узлу или зоне в DNS. Узел может задаваться именем или IP-адресом; по умолчанию **host** преобразует IP-адреса в имена и добавляет локальный домен к узлам без завершающей точки. Серверы, опрашиваемые по умолчанию, определены в файле */etc/resolv.conf*. Более подробная информация об узлах и зонах содержится в первых двух главах книги Пола Албица (Paul Albitz) и Крикета Лью (Cricket Liu) «DNS и BIND», издательство O'Reilly & Associates.[1]

Параметры

–a Действует идентично **–t ANY**.

–c *class*

Поиск указанного класса записей ресурсов (IN, INTERNET, CS, CSNET, CH, CHAOS, HS, HESIOD, ANY или *). По умолчанию – IN.

–d Режим отладки. Более подробным вариантом является **–dd**.

–e Не отображать информацию о доменах, находящихся за пределами указанной зоны. Для запросов по именам узлов не выводить дополнительную информацию («additional information») и авторитативный сервер имен («authoritative nameserver»).

–f *file*

Записать копию вывода в указанный файл.

–i Для IP-адреса вернуть соответствующий ему адрес вида *in-addr.arpa*, его класс (всегда PTR) и имя узла.

–l *zone*

Перечислить все машины зоны *zone*.

---

[1] П. Албиц, К. Лью «DNS и BIND», издательство «Символ-Плюс», IV кв. 2001 г.

host

–m Отображать только записи MR, MG и MB; рекурсивно расширять записи MR (переименованный почтовый ящик) и MG (почтовая группа) в записи MB (почтовый ящик).

–o Запретить печать на стандартный вывод.

–p [*server*]

Используется совместно с **–l**. Опрашивать только первичный сервер имен (или *server*) зоны вместо отвечающих авторитативных серверов. Параметр полезен для проверки незарегистрированных зон.

–q Тихий режим. Предупреждения не выводятся, а только ошибки.

–r Не позволять опрашиваемому серверу передавать запрос другим серверам, а требовать только буферизованную информацию.

–t *type*

Поиск элементов *type* в записях ресурсов. *type* может принимать значения A, NS, PRT, ANY или * (все).

–u Использовать TCP, а не UDP.

–v Диагностика. Включать все поля записи ресурсов, даже «time-to-live» и «class», а также «дополнительную информацию» и «авторитативный сервер имен» (данные, полученные от удаленного сервера имен).

–vv

Очень подробная диагностика. Включать информацию о значениях по умолчанию для узла *host*.

–w Не прекращать попытки получить ответ от опрашиваемого сервера.

–x Разрешить указание нескольких узлов или зон. Если задан также и сервер, аргумент должен предваряться параметром **–X**.

–A Для имен узлов искать IP-адреса, а затем, наоборот, искать имя по IP-адресу и проверить, совпадают ли имена. Для IP-адресов выполнять поиск связанного имени и определять, считает ли узел этот адрес своим собственным. Для зон проверять IP-адреса всех узлов. Завершить работу без вывода, если несоответствий не обнаружено.

–C Аналогично **–l**, но также проверять, являются ли в действительности сервера имен зоны авторитативными. Запись SOA (Start of Authority) для зоны определяет авторитативные сервера имен (в полях NS). Происходит опрос

этих серверов; если у какого-либо из них нет записи в SOA, **host** сообщает о «неверном делегировании». Помимо этого выполняются и другие проверки.

–D Аналогично **–H**, но включаются имена узлов, имеющие более одного адреса.

–E Аналогично **–H**, но не считаются ошибочными внезональные имена. Внезональными называются имена из неопределенного поддомена.

–F *file*

Перенаправить стандартный вывод в файл и отображать дополнительные записи ресурсов только на стандартный вывод.

–G *zone*

Аналогично **–H**, но включаются имена шлюзов.

–H *zone*

Вывести число уникальных узлов в зоне *zone*. Псевдонимы не включаются. Также перечислить все найденные ошибки (внезональные и дублирующиеся узлы).

–I *chars*

Не выдавать предупреждения об именах доменов, содержащих недопустимые символы (например, «_») из списка *chars*.

–L *level*

Используется совместно с **–l**. Перечислить рекурсивно все делегированные зоны внутри указанной до уровня глубины *level*.

–P *servers*

Применяется совместно с **–l**. Элементы списка *servers* разделяются запятыми и являются предпочтительными вторичными серверами для копирования данных о зоне. Наивысший приоритет получают серверы, имена которых более всего соответствуют доменным компонентам в указанной части *servers*.

–R Обработка неполных имен узлов в стиле BIND – путем поиска каждого компонента узла в локальном домене.

–S Используется с **–l**. Вывести на стандартный вывод все узлы зоны. Не выводить узлы подзон. Включать класс и IP-адрес. Отображать предупреждения (неверные имена, «неверное делегирование», отсутствующие записи и т. д.) в стандартный поток ошибок.

host	**–T** Вывести значения time-to-live (сколько еще времени информация об узле будет находиться в кэше, пока сервер имен не обновит ее). **–X** *server* Указать сервер для опроса и разрешить указание множественных узлов или зон. **–Z** При выводе записей ресурсов включать последнюю точку в именах доменов, отображать значение time-to-live и имя класса узла.
hostid	**hostid** Отображение идентификационного номера текущего узла в шестнадцатеричном виде.
hostname	**hostname** [*option*] [*nameofhost*] Установка или вывод текущего имени узла. Привилегированный пользователь может изменить имя узла, задав соответствующий параметр (*nameofhost*). **Параметры** **–a, ––alias** Отобразить синоним имени узла (если такой существует). **–d, ––domain** Вывести имя домена DNS. **–f, ––fqdn, ––long** Вывести полное доменное имя. **–F** *file*, **––file** *file* Извлечь имя узла из файла *file*. **–h, ––help** Вывести информацию по использованию **hostname** и завершить работу. **–i, ––ip-address** Отобразить IP-адрес(а) узла. **–s, ––short** Удалить информацию о домене из отображаемого имени. **–v, ––verbose** Описательный режим.

hostname

–V, ––version

Вывести информацию о версии **hostname**, завершить работу.

–y, ––yp, ––nis

Отобразить имя NIS-домена. Привилегированный пользователь может установить новое имя при помощи аргумента *nameofhost*.

hwclock [*options*]

hwclock

Команда системного администрирования. Чтение или установка аппаратных часов. **hwclock** хранит в файле */etc/adjtime* информацию об изменениях, которая может быть использована для подводки часов, если они отстают или спешат. **hwclock** является заменой программы **clock**. Однобуквенные параметры включены для сохранения совместимости с прежней командой.

Параметры

Можно указывать лишь один из перечисленных ниже параметров:

–a Подводка аппаратных часов на основе информации из */etc/adjtime* и установка системных часов на новое время.

––adjust

Подводка аппаратных часов на основе информации из */etc/adjtime*.

––date *date*

Имеет смысл, только если параметр указан совместно с параметром **––set**. Аргумент *date* является строкой, пригодной к использованию с командой **date**.

––debug

Диагностика действий **hwclock**.

–r, ––show

Вывести текущее время по аппаратным часам.

–s, ––hctosys

Привести системное время в соответствие с временем аппаратных часов.

hwclock	**––set** Установить аппаратные часы на время, указанное в параметре **––date**. **––test** Ничего не менять реально. Параметр удобен для проверки правильности синтаксиса. **–u, ––utc** Аппаратное время хранится в формате UTC (Universal Coordinated Time). **––version** Отобразить номер версии и завершить работу. **–w, ––systohc** Привести аппаратные часы в соответствие с системным временем.
icmpinfo	**icmpinfo** [*options*] Команда TCP/IP. Перехват и интерпретация пакетов ICMP. Вывод адреса и имени отправителя, исходного и конечного портов, последовательности и размера пакета. По умолчанию отображается информация только о пакетах со странным поведением. **Параметры** **–k** Принудительно завершить процесс **syslogd**, порожденный параметром **–l**. **–l** Запись посредством **syslogd**. Параметр доступен только привилегированному пользователю. **–n** Использовать IP-адреса вместо имен узлов. **–p** Подавить расшифровку номера порта: не пытаться подобрать имя сервиса, обслуживающего порт. **–s** Включать IP-адрес интерфейса, через который был получен пакет, в том случае, если на машине существует несколько интерфейсов. **–v** Подробный режим. Включить информацию об обычных ICMP-пакетах. Дополнительные детали можно получить, указав параметр **–vv** или **–vvv**.

id [*options*] [*username*] **id**

Отображение информации о пользователе, выполняющем команду: идентификатор пользователя, идентификатор группы пользователя, эффективные идентификаторы пользователя или группы, если необходимо, и дополнительные идентификаторы группы.

Параметры

–g, ––group

Вывести только идентификатор группы.

–G, ––groups

Отображать только идентификаторы дополнительных групп.

–n, ––name

Отображать имя пользователя или группы, а не численные идентификаторы. Указывается совместно с параметрами **–u**, **–g** или **–G**.

–r, ––real

Отображать реальный, а не эффективный идентификатор пользователя или группы. Указывается совместно с параметрами **–u**, **–g** или **–G**.

–u, ––user

Отображать только идентификатор пользователя.

––help

Вывести справку и завершить работу.

––version

Вывести информацию о версии **id**.

---

in.identd [*options*] [*kernelfile* [*kmemfile*]] **identd**

Команда TCP/IP. Выяснение имени пользователя, процесс которого владеет определенным соединением TCP/IP. Можно указать ядро и его пространство в памяти.

Параметры

–a *ip_address*

Привязка к IP-адресу. Имеет смысл только с параметром **–b**. По умолчанию происходит привязка к адресу INADDR_ANY.

–b Автономное выполнение; не для использования с **inetd**.

identd	**–d** Разрешить отладочные запросы. **–g***gid* Выполняться, по возможности, в группе *gid*. Имеет смысл только с параметром **–b**. **–i** Выполнение в стиле демона – по одному процессу на каждый запрос. **–l** Ведение журнала через **syslogd**. **–m** Разрешить множественные запросы в течение одного сеанса. **–n** Возвращать идентификаторы пользователей вместо имен. **–N** Не возвращать ни имени, ни идентификатора пользователя, если в домашнем каталоге пользователя существует файл *.noident*. **–o** При запросе типа операционной системы всегда возвращать OTHER. **–p***port* Прием через указанный порт, а не через стандартный порт 113. **–t***seconds* Завершить работу, если новые запросы не поступали в течение *seconds* секунд. Обратите внимание, что при указанных параметрах **–i** или **–w** следующий запрос приведет к повторному запуску **identd**. По умолчанию устанавливается бесконечное время ожидания. **–u***uid* Выполняться, по возможности, с идентификатором владельца *uid*. Имеет смысл только при использовании совместно с **–b**. **–V** Вывести номер версии и завершить работу. **–w** Работать в стиле демона – один процесс на все запросы.
ifconfig	**ifconfig** [*interface*] **ifconfig** [*interface address_family parameters addresses*] Команда TCP/IP. Присваивание адреса сетевому интерфейсу либо настройка параметров сетевого интерфейса. **ifconfig**, как правило, применяется для определения сетевых адресов всех интерфейсов машины при загрузке системы. Позже его можно использовать для переопределения адреса интерфейса

и других параметров. Команда без аргументов отображает текущую конфигурацию сетевого интерфейса. При выполнении с единственным аргументом (*interface*) **ifconfig** отображает текущие настройки указанного интерфейса.

Аргументы

interface

Строка вида *name unit* (*имя номер*), например **en0**.

address_family

Поскольку интерфейс может принимать пакеты разных протоколов, каждый из которых требует разных схем задания имен, существует возможность задать адресное семейство (*address_family*) для изменения интерпретации остальных параметров. Существует семейство **inet** (по умолчанию; протоколы TCP/IP), **ax25** (AX.25 Packet Radio), **ddp** (Appletalk Phase 2) или **ipx** (Novell).

Parameters

Следующие параметры могут быть установлены с помощью **ifconfig**:

allmulti/–allmulti

Включить/отключить пересылку входящих блоков данных в сетевой слой ядра.

arp/–arp

Включить/отключить использование протокола разрешения адресов (ARP) при связывании адресов сетевого уровня и адресов ссылочного уровня.

broadcast

(только для **inet**.) Указать адрес для посылки широковещательных сообщений по сети. По умолчанию это адрес, все биты узла которого установлены, т. е. x.y.z.255 для сети класса C.

debug/–debug

Включить/отключить работу зависимого от драйвера отладочного кода.

dest_address

Указать адрес второй стороны соединения PPP (point-to-point).

down

Пометить интерфейс как «down» (не отвечающий).

hw *class address*

Установить аппаратный класс интерфейса и аппаратный адрес. Параметр *class* может принимать значения

ifconfig

ether (Ethernet), **ax25** (AX.25 Packet Radio) или **ARCnet**.

irq *addr*
Установить прерывание для устройства.

metric *n*
Установить метрику маршрутизации интерфейса в *n*. По умолчанию – 0.

mtu *num*
Установить размер максимального передаваемого блока данных (MTU, Maximum Transfer Unit).

multicast
Установить флаг многоадресной передачи.

netmask *mask*
(только для **inet**.) Указать, какую часть адреса резервировать для разделения сетей на подсети. Маска (*mask*) может задаваться в виде четырехбайтового шестнадцатеричного числа (начинается с 0x) в записи с точками (подобно IP-адресу) или псевдосетевым именем (иногда оно называется логическим) из сетевой таблицы */etc/networks*.

pointopoint/–pointopoint [*address*]
Включить/отключить работу интерфейса с протоколом PPP, чтобы соединение между машинами было выделенным.

up Пометить интерфейс как «up» (готовый к принятию и посылке данных).

trailers/–trailers
Запросить/отключить использование трейлеров уровня ссылок при формировании посылаемых пакетов данных.

address

Имя узла из базы данных */etc/hosts* или адрес Интернета, заданный в стандартной записи «через точку».

imake

imake *options*

Интерфейс препроцессора (**cpp**) и утилиты **make**. **imake** (от *include make*) решает проблему переносимости **make** путем хранения машинно-зависимой информации в основном наборе файлов настройки отдельно от описания объектов, которые необходимо будет собрать для получения готовой программы. Файл *Imakefile* содержит машинно-независимые

описания собираемых целей в виде макросов **cpp**. Команда **imake** вызывает **cpp**, для того чтобы создать машинно-зависимые файлы сборки (*Makefiles*), используемые впоследствии **make**.

Один из файлов настройки – это основной для **imake** файл шаблонов (по умолчанию – *Imake.tmpl*), который включает в себя (посредством директивы **#include**) прочие файлы настройки, содержащие машинные зависимости (такие как присваивания значений переменным, определения точек сборки, макросы **cpp**), а также содержит указания по порядку обработки файлов. Каждый из файлов влияет на интерпретацию последующих файлов и различных секций *Imake.tmpl*. В файлы настроек **imake** можно включать комментарии, причем первый символ комментария «#» должен предваряться пустым комментарием C:

```
/**/#
```

См. также описания **cpp** и **make**. Дополнительную информацию можно найти в книге из серии Nutshell Handbook: «Software Portability with imake» (Переносимость программ и imake) Пола Дюбуа (Paul DuBois).

Параметры

–D*define*

Установка каталого-зависимых переменных. Параметр передается **cpp** без изменений.

–e Выполнить сгенерированный файл *Makefile*. По умолчанию это действие оставлено пользователю.

–f *filename*

Задать имя файла *imake* в каждом из каталогов собираемого проекта. По умолчанию это *Imakefile*.

–I*directory*

Каталог, в котором следует производить поиск шаблонов и файлов настройки **imake**. Параметр передается **cpp** без изменений.

–s *filename*

Имя создаваемого файла сборки **make**. Если вместо имени задан дефис, результат работы записывается на стандартный вывод. По умолчанию создается (не выполняется) файл *Makefile*.

imake

–T*template*

Имя основного файла шаблонов, используемого **cpp**. Этот файл, как правило, находится в каталоге, который задан параметром **–I**. Имя файла по умолчанию – *Imake.tmpl*.

–v Вывести командную строку **cpp**, по которой будет создан *Makefile*.

Инструменты

Далее следует список инструментов, используемых вместе с **imake**:

makedepend [*options*] *files*

Создать зависимости для заголовочных файлов в файлах сборки *Makefiles*. **makedepend** выполняет последовательное чтение исходных файлов и разбор директив **#include**, **#define**, **#undef**, **#ifdef**, **#ifndef**, **#endif**, **#if** и **#else**, для того чтобы определить, какие директивы **#include** будут использованы при компиляции. **makedepend** определяет зависимости и записывает их в *Makefile*. Таким образом **make** узнает, какие из объектных файлов изменились и должны быть перекомпилированы. Параметры **makedepend**:

–– *options* **––**

Игнорировать все неопознанные параметры, следующие за двойным дефисом. Второй двойной дефис отменяет действие предыдущего. Опознанные параметры между дефисами обрабатываются как обычно.

–a Добавлять новые зависимости к уже существующим, а не заменять их.

–f *filename*

Записывать зависимости в файл *filename*, а не в *Makefile*.

–m Вывести предупреждение о множественном включении файла.

–s *string*

Использовать строку *string* вместо строки «**#DO NOT DELETE THIS LINE – make depend depends on it**» в качестве разделителя внутри файла.

–v Диагностика. Перечислить все файлы, включаемые в основной исходный файл.

−D *name=value*
−D *name*

Определить имя с заданным значением (первый вариант) или со значением 1 (второй вариант).

−I *dir*

Добавить каталог *dir* к каталогам, в которых производится поиск.

−Y *dir*

Искать включаемые файлы только в каталоге *dir*. Игнорировать стандартные каталоги включаемых файлов.

makedirhier *dir*...

Создать каталог *dir* и все недостающие родительские каталоги во время процедуры установки файла.

xmkmf [*options*] [*topdir*] [*curdir*]

Инициализировать *Makefile* содержимым *Imakefile*. *topdir* определяет корневой каталог дерева проекта, *curdir* (обычно отсутствует) задается как относительное (от вершины дерева сборки) имя пути текущего каталога. Параметр **−a** эквивалентен последовательности команд:

```
% xmkmf
% make Makefiles
% make includes
% make depend
```

Файлы настройки

Ниже приведены описания файлов настройки **imake**:

Imake.tmpl

Основной файл шаблонов **imake**. *Imake.tmpl* включает все прочие файлы настройки и *Imakefile* из текущего каталога.

Imake.params

Содержит определения, которые действуют в различных проектах и для различных платформ.

Imake.rules

Содержит макроопределения **cpp** для текущей платформы. Макроопределения передаются программе **imake**, которая затем выполняет **cpp** для обработки этих макросов. Символы новой строки обозначаются последовательностью @@\ (два «at» и обратный слэш).

imake

site.def

Содержит специфичные для проекта данные, такие как каталоги установки, набор компилируемых программ, использование специальных версий программ при сборке проекта. На разных машинах файл *site.def* различается.

Project.tmpl

Файл, содержащий переменные, специфичные для X.

Library.tmpl

Файл, содержащий правила работы с библиотеками.

Server.tmpl

Файл, содержащий правила, специфичные для сервера.

.cf Файлы *.cf* (*VendorFiles*, расположенные в *Imake.vb*) содержат настройки разработчика. В каждом таком файле находится информация о платформе разработки, в частности, о версии операционной системы и компилятора, а также альтернативные пути реализации отсутствующих возможностей. Определения в файлах *.cf* имеют более высокий приоритет, чем соответствующие значения по умолчанию, описанные в файле *Imake.params*.

Файл `Imakefile`

Файл *Imakefile*, существующий в каждом из каталогов проекта, определяет цели сборки, установки и правила, которым необходимо следовать. Команда **imake** читает *Imakefile* и создает из правил целевые записи для файла *Makefile*. Каждый *Imakefile* также может содержать определение переменных для **make** и список зависимостей целей. Зависимости выражаются в виде макросов **cpp**, определенных в файле *Imake.rules*. При изменении какого-либо файла *Imakefile* необходимо произвести пересборку файла *Makefile* и зависимостей заголовочных файлов. Более подробно о работе с **imake** можно прочесть в книге Пола Дюбуа (Paul DuBois) из серии Nutshell Handbook: «Software Portability with imake» (Переносимость программ и imake).

imapd

imapd

Команда TCP/IP. Демон сервера диалогового протокола доступа к почте IMAP (Interactive Mail Access Protocol). **imapd** запускается командой **inetd** и принимает запросы IMAP-клиентов через порт 143. IMAP позволяет почтовым программам получать доступ к удаленным почтовым ящикам, как если бы они были локальными. Возможности IMAP шире,

чем у протокола POP, поскольку он позволяет клиенту получать информацию о сообщениях в почтовом ящике на сервере, а не только все содержимое почтового ящика. IMAP может использоваться для чтения почты как в online-, так и в offline-режиме. Популярный почтовый клиент Pine поддерживает работу с IMAP.

imapd

inetd [*option*] [*configuration_file*]

inetd

Команда TCP/IP. Демон сервисов Интернета. **inetd** работает в качестве приемника запросов на множественных портах. Когда запрос получен, **inetd** запускает соответствующий сервер. При запуске **inetd** считывает настройки либо из файла *configuration_file*, либо из стандартного файла настройки */etc/inetd.conf*. Затем **inetd** вызывает **getservbyname**, создает сокет для каждого из серверов и связывает сокеты с соответствующими портами. **inetd** выполняет команду **listen** для всех сокетов соединений, затем переходит в ожидание, используя **select** для соединений или пересылки пакетов.

При получении запроса на соединение ожидающим сокетом **inetd** выполняет **accept**, создавая новый сокет. Затем происходит ветвление (fork), дублирование и запуск соответствующего сервера с необходимыми аргументами. Вызываемый сервер выполняет операции ввода/вывода для потоков **stdin**, **stdout** и **stderr** нового сокета, соединяя клиентский процесс с сервером.

Если для пакетного сокета есть данные, **inetd** выполняет ветвление (fork), дублирование и запуск нового сервера с передачей ему соответствующих аргументов сервера. Если пакетный сокет помечен как **wait**, запущенный сервер должен обработать сообщение до того, как **inetd** посчитает сокет доступным для новых соединений. Если сокет помечен как **nowait**, то **inetd** продолжает обрабатывать поступающие на этот порт запросы.

Команда **inetd** может запускать следующие серверы: **bootpd**, **bootpgw**, **fingerd**, **ftpd**, **imapd**, **popd**, **rexecd**, **rlogind**, **rshd**, **talkd**, **telnetd** и **tftpd**. Не пытайтесь запускать с помощью **inetd** серверы **named**, **routed**, **rwhod**, **sendmail**, **listen** или какие-либо серверы NFS.

При получении сигнала рассоединения (SIGHUP) **inetd** выполняет повторное чтение файла настройки. Сервисы могут добавляться, удаляться или изменяться перед повторным чтением настроек.

inetd

Параметр

–d Включить отладку уровня сокета и выдавать отладочную информацию на стандартный вывод.

Файлы

/etc/inetd.conf

Стандартный файл конфигурации.

/var/run/inetd.pid

Идентификатор процесса **inetd**.

info

info [*options*] [*topics*]

GNU-средство просмотра гипертекста: отображает оперативную документацию, предварительно созданную из исходных файлов Texinfo. Информационные файлы (info-файлы) имеют иерархическую структуру, содержат меню и подтемы. Команда без параметров отображает info-файл самого высокого уровня (обычно это */usr/local/info/dir*). Если заданы темы (*topics*), происходит поиск тем в системе меню (рекурсивным спуском по иерархии). Отображение информации можно контролировать параметрами **–f** и **–n**.

Параметры

–d *directories*, **––directory** *directories*

Искать info-файлы в каталогах из списка (элементы разделяются двоеточием). Если этот параметр не указан, для поиска используется переменная окружения INFOPATH или каталог по умолчанию (обычно */usr/local/info*).

––dribble *file*

Сохранять ввод с клавиатуры в файле *file*, который позже можно использовать при работе с параметром **––restore** для быстрого поиска определенного места в info-файле.

–f *file*, **––file** *file*

Отобразить содержимое указанного info-файла.

–n *node*, **––node** *node*

Отобразить указанный пункт (*node*) info-файла.

–o *file*, **––output** *file*

Отобразить вывод в файл (*file*), а не на экран.

––help

Краткая информация по использованию **info**.

info

−−restore *file*

Отобразив содержимое info-файла, восстановить последовательность ввода с клавиатуры из файла (*file*).

−−subnodes

Отображать подтемы.

−−version

Отобразить версию **info**.

−−vi-keys

Использовать клавиатурные сочетания, аналогичные принятым в редакторе **vi**.

init

init [option] [*runlevel*]

Команда системного администрирования.

Параметр

−t *seconds*

При изменении режима работы (runlevel) посылать сигнал SIGKILL через *seconds* секунд после SIGTERM. По умолчанию – 20 секунд.

Файлы

init является самым первым процессом, выполняемым на Unix-машине при загрузке. Он проверяет целостность всех файловых систем и создает другие процессы, используя ветвление **fork** и **exec**, как указано в файле */etc/inittab*. Допустимость выполнения тех или иных процессов в системе регулируется режимами работы (*runlevel*). Все завершения процессов регистрируются в файлах */var/run/utmp* и */var/log/wtmp*. При изменении режима работы **init** посылает сигнал SIGTERM, а через двадцать секунд – SIGKILL всем процессам, которые не могут выполняться в новом режиме работы.

Режимы работы

Текущий режим работы может быть изменен при помощи программы **telinit**, которая часто представляет собой просто ссылку на **init**. Режим работы по умолчанию зависит от дистрибутива, но следующие определения являются стандартом:

0 Останов системы.

1, **s**, **S** Однопользовательский режим.

init	**6** Перезагрузка системы. **q**, **Q** Повторное чтение */etc/inittab*. Сведения о режимах работы системы можно уточнить в файле */etc/inittab*.
insmod	**insmod** [*options*] *file* [*symbol=value* . . .] Команда системного администрирования. Загрузить модуль ядра *file*, изменив значения имен на указанные в командной строке. Если файл модуля называется *file.o* или *file.mod*, необходимо указать имя *file*. **Параметры** **–f** Принудительная загрузка модуля, даже при наличии определенных затруднений. **–m** Вывод карты загрузки. **–o** *name* Дать модулю имя *name*, не учитывая имя объектного файла. **–x** Отменить экспорт: не добавлять никакие внешние имена из модуля к таблице имен ядра.
install	**install** [*options*] [*file*] *directories* Команда системного администрирования. Используется преимущественно в файлах сборки для обновления файлов. **install** копирует файлы в каталоги, указанные пользователем. Файлы не перезаписываются. Программа работает аналогично **cp**, но пытается устанавливать права доступа, владельца и группу для копируемых файлов. **Параметры** **–d**, **––directory** Создавать недостающие каталоги. **–g** *group*, **––group** *group* Установить идентификатор группы для нового файла в *group* (доступно только привилегированному пользователю). **–m** *mode*, **––mode** *mode* Установить права доступа для нового файла в *mode* (восьмеричная или символьная запись). По умолчанию устанавливаются права **0755**.

install

–o [*owner*], **––owner** [*owner*]

Установить владельца нового файла. Если владелец не задан, им становится пользователь **root** (доступно только привилегированному пользователю).

–s, **––strip**

Удалить таблицы имен.

ipchains

ipchains *command* [*options*]

Команда системного администрирования. Изменение правил для IP-брандмауэра в ядре Linux ветви 2.2. Если ядро Linux 2.2 собрано с поддержкой брандмауэринга, то заголовки всех сетевых пакетов будут проверяться на соответствие установленным правилам с целью определения дальнейших действий, выполняемых над пакетом. Правила брандмауэра состоят из критерия отбора и цели, а также действия, совершаемого в случае соответствия пакета критерию. Правила объединяются в цепи (chains) и могут применяться для создания брандмауэра или просто наложения запрета на определенные виды сетевых соединений.

Правила брандмауэра, организованные в виде цепей, представляют собой упорядоченные списки правил, которые просматриваются ядром в поисках соответствия критерию. Существует три предопределенных цепи: **input**, **output** и **forward**. Входящие пакеты проверяются на наборе правил цепи **input**. Выходящие пакеты проверяются на наборе правил цепи **output**. Если входящий пакет адресован какой-либо другой системе, он проверяется на наборе правил цепи **forward**. У каждой из этих цепей есть цель по умолчанию для тех случаев, когда ни одно правило не может быть применено. Цепи, определяемые пользователем, могут выступать в качестве целей для пакетов, но они не имеют правил, применяемых по умолчанию. Если ни одно из правил цепи, определенной пользователем, не может быть применено, пакет возвращается той цепи, из которой произошел вызов, и проверяется на соответствие следующему правилу этой цепи.

Команда **ipchains** изменяет правила только в процессе работы ядра системы. Когда система выключается, все изменения теряются. Можно воспользоваться командой **ipchains-save** для создания сценария, который позже, будучи выполненным командой **ipchains-restore**, восстановит настройки брандмауэра. Такие сценарии часто называются загрузочными, и во многие дистрибутивы входят сценарии инициализации **ipchains**, использующие результаты работы **ipchains-save**.

ipchains

Команды

Команда **ipchains** всегда выполняется с указанием одной из следующих команд:

–A *chain rules*, **––append** *chain rules*

Добавить новые правила (*rules*) к цепи (*chain*).

–I *chain number rules*, **––insert** *chain number rules*

Вставить правила (*rules*) в цепь *chain* в указанной позиции *number*.

–D *chain rules*, **––delete** *chain rules*

Удалить правила из цепи. Правила могут определяться своими порядковыми номерами в цепи, а также общими описаниями правил.

–R *chain number rule*, **––replace** *chain number rule*

Заменить правило в цепи. Заменяемое правило определяется порядковым номером *number*.

–C *chain rule*, **––check** *chain rule*

Создать сетевой пакет, соответствующий критериям указанного правила и проверить, как этот пакет будет обработан цепью. Правило должно описывать источник, адресат, протокол и интерфейс конструируемого пакета.

–L [*chain*], **––list** *$PARAMETER*

Перечислить правила цепи. Если цепь не указана, перечислить все правила всех цепей.

–ML, **––masquerading ––list**

Перечислить маскируемые соединения.

–MS *tcp tcpfin udp*, **––masquerading ––set** *tcp tcpfin udp*

Установить значение интервала ожидания (в секундах) для соединений, маскируемых сокрытием адресов. Команда **–MS** всегда должна иметь три аргумента, определяющие значения интервалов для сеансов TCP, для сеансов TCP, в которых были получены пакеты FIN и пакеты UDP.

–F *chain*, **––flush** *chain*

Удалить все правила из указанной цепи.

–Z [*chain*], **––zero** [*chain*]

Обнулить счетчики байт и пакетов для указанной цепи. Если цепь не указана, произойдет обнуление для всех цепей, а при использовании совместно с командой **–L** перед обнулением перечисляются текущие значения всех счетчиков.

–N *chain*, **––new-chain** *chain*

Создать новую цепь. Указанное имя цепи должно быть уникальным.

–X [*chain*], **––delete-chain** *chain*

Удалить указанную цепь. Могут быть удалены только цепи, определенные пользователем, при этом не должно существовать ссылок на удаляемую цепь. Если цепь не указана, удаляются все определенные пользователем цепи.

–P *chain target*, **––policy** *chain target*

Установить правило для предопределенной цепи. Цель не может являться цепью.

–h [icmp]

Вывести краткую справку по использованию. Если задан аргумент **icmp**, перечислить доступные ICMP-типы.

Цели

Цель может являться именем цепи либо одним из следующих специальных значений:

ACCEPT

Пропустить пакет.

DENY

Удалить пакет.

MASQ

Скрыть адреса для данного пакета. Источником пакета становится текущая система. Ответные пакеты автоматически подвергаются обратному преобразованию. Это правило доступно только для цепи **forward** или пользовательских цепей, в которых осуществляется ретрансляция пакетов. Для использования этого правила необходима поддержка сокрытия адресов ядром (IP masquerading).

REDIRECT [*port*]

Перенаправить входящий пакет в указанный локальный порт, на котором работает «прозрачный» прокси-сервер. Если номер порта равен 0 или порт не определен, целевой порт пакета выступает в качестве порта перенаправления. Правило **REDIRECT** может применяться либо для цепи **input**, либо для определяемых пользователем цепей, которые обрабатывают входящие пакеты. Ядро должно быть собрано с поддержкой «прозрачного» прокси-сервера.

ipchains

REJECT

Удалить пакет и отправить ICMP-сообщение отправителю пакета с уведомлением об удалении.

RETURN

Вернуть управление в цепь, из которой произошел вызов данной цепи, и проверить следующее правило. Если **RETURN** является целью предопределенной цепи, то используется правило по умолчанию для этой цепи.

Параметры определения правил

Следующие параметры служат для создания правил, которые применяются с вышеперечисленными командами. Правила обычно состоят из критериев отбора и целей, к которым происходит переход (**–j**) при совпадении. Со многими критериями отбора может использоваться отрицание, определяемое восклицательным знаком (**!**). Такие правила будут выбирать все пакеты, кроме тех, что удовлетворяют указанному критерию.

–p [!] *name*, **––protocol [!]***$PARAMETER*

Отбор пакетов протокола *name*. Значение *name* может быть строковым именем или соответствующим числом (из файла */etc/protocols*). Наиболее часто встречаются значения **tcp**, **udp**, **icmp**, а также ключевое слово **all**. Число 0 идентично **all**, это значение является стандартным, если иное не определяется явно данным параметром.

–s [!] *address*[/*mask*] **[!]** [*port*], **––source [!]** *address*[/*mask*] **[!]** [*port*]

Отбор пакетов, приходящих с указанного адреса и порта. Адрес может указываться в виде имени узла, имени сети или в виде IP-адреса. Необязательный параметр *mask* определяет сетевую маску и может быть представлен в классическом формате (например, /255.255.255.0) или в современной, более краткой форме (например, /24). Необязательный параметр *port* определяет порт TCP, UDP или ICMP. Порт может быть указан только в том случае, если был использован параметр **–p** с указанием протокола **tcp**, **udp** или **icmp**. Для задания включающего диапазона портов или значений ICMP, которые будут использованы при отборе пакетов, применяется двоеточие (например, 20:25 для портов с 20 по 25). Если отсутствует первое значение *port*, принимается значение по умолчанию, а именно 0. Если отсутствует второе значение, оно принимается равным 65535.

–d [!] *address*[/*mask*] **[!]** [*port*], **––destination [!]** *address*[/*mask*] **[!]** [*port*]

Отбор пакетов с определенным адресом назначения (*address*). Синтаксис параметра идентичен синтаксису параметра **–s**.

–j *target*, **––jump** *target*

Переход к специальной цели или к определенной пользователем цепи. Если этот параметр для правила не указан, соответствие пакета правилу лишь увеличивает счетчики для правила, а пакет передается для проверки на соответствие следующему правилу.

–i [!] *name*, **––interface** *name*

Отбор пакетов интерфейса *name*[+]. *name* – это сетевой интерфейс системы (например, **eth0** или **ppp0**). Символ «+» может использоваться для создания маски. Так, значению **ppp+** соответствуют все имена интерфейсов, начинающиеся с **ppp**.

[!] –f, [!]––fragment *$PARAMETER*

Правило отбирает все, кроме первого, фрагменты разбитого на части пакета.

––source-port [!] *port*

Отбор пакетов с указанным значением исходного порта. Синтаксис задания портов см. в описании параметра **–s**.

––destination-port [!] *port*

Отбор пакетов с указанным значением целевого порта. Синтаксис задания портов см. в описании параметра **–s**.

––icmp-type [!] *type*

Отбор пакетов, которые имеют указанный (по имени или номеру) тип ICMP.

Параметры

–b, ––bidirectional

Добавить правило в цепи **input** и **output**, чтобы на соответствие этому правилу проверялись как входящие, так и исходящие пакеты.

–v, ––verbose

Режим подробной диагностики.

ipchains	**–n, ––numeric** Отображать все IP-адреса и номера портов в численном виде. По умолчанию, где возможно, отображаются имена. **–l, ––log** При соответствии пакета правилу создавать соответствующую запись в системном журнале. **–t** *andmask xormask*, **––TOS** *andmask xormask* Изменить поле Type of Service (Тип службы) в заголовке пакета. Поле пакета TOS – первое, над которым выполняется операция логического умножения AND на восьмибитную маску *andmask*, затем результат используется в качестве первого операнда для XOR *xormask*, где *xormask* – также восьмибитная маска. Правила, которые могут повлиять на наименее значимый бит (LSB) поля TOS, не применяются. **–x, ––exact** Производить подстановку чисел при перечислении (**–L**). Отображать точные значения счетчиков байт и пакетов вместо округленных. **[!] –y, ––syn** Отбор только входящих запросов на TCP-соединение, т. е. запросов с установленным битом SYN и сброшенными битами ACK и FIN. Таким образом, входящие TCP-соединения блокируются, но не затрагиваются исходящие соединения. **––line-numbers** Используется совместно с командой **–L**. Добавлять номер строки в начало каждого правила в перечислении, определяя, таким образом, позицию правила в цепи. **––no-warnings** Отключить выдачу предупреждений.
ipchains-restore	**ipchains-restore** [*options*] Команда системного администрирования. Восстановить набор правил брандмауэра. **ipchains-restore** использует для восстановления правил брандмауэра в каждой цепи команды, сгенерированные **ipchains-save**. Часто применяется в сценариях инициализации для восстановления набора правил при загрузке системы.

ipchains-restore

Параметры

–f Принудительное обновление существующих цепей.

–v Отображать правила по мере добавления.

–p Если целью правила является несуществующая цепь, создать ее.

ipchains-save

ipchains-save [*chain*] [*option*]

Команда системного администрирования. Вывести правила IP-брандмауэра, хранящиеся в памяти ядра, на стандартный вывод. Если цепь не указана, отображаются правила из всех цепей. Вывод обычно перенаправляется в файл, который затем может быть использован совместно с **ipchains-restore** для восстановления значений.

Параметр

–v Отображать правила дополнительно в стандартный поток ошибок (stderr), облегчая их просмотр при перенаправлении вывода.

ipfwadm

ipfwadm *category command parameters* [*options*]
ipfwadm –*M* [–*l* | –*s*] [*options*]

Управление брандмауэром и работа с правилами, ведение учетных записей и сокрытие IP-адресов для ядер Linux ветви 2.0. В ветви 2.2 эта команда заменена на **ipchains**, а **ipchains** – на **iptables** в ветви 2.4.

Существуют четыре категории правил: учет IP-пакетов, входной брандмауэр, выходной брандмауэр и ретранслирующий брандмауэр. Правила каждой категории представляют собой отдельный список. Более подробное описание работы списков правил можно найти на страницах руководства (manpages) для **ipfw**(4).

Каждая команда **ipfwadm** задает только одну категорию и одно правило. Для создания надежного брандмауэра необходимо выполнить множество команд **ipfwadm**; комбинация определенных ими правил гарантирует, что брандмауэр работает точно так, как задумывалось. Вторая форма команды предназначена для сокрытия адресов. Команды **–l** и **–s**, описанные ниже, являются единственными, которые могут использоваться с категорией сокрытия **–M**.

ipfwadm

Категории

Для определения категории правил, к которой относится команда, необходимо указать один из следующих флагов:

–A [*direction*]

Правила учета IP-пакетов. Может быть задано направление:

in Только входящие пакеты.

out Только исходящие пакеты.

both

Все пакеты; режим по умолчанию.

–F Правила ретранслирующего брандмауэра.

–I Правила входного брандмауэра.

–M Управление сокрытием адресов. Может применяться только с командами **–l** и **–s**.

–O Правила выходного брандмауэра.

Команды

За категорией следует команда, указывающая на предпринимаемое действие. Если не указано другое, то в командную строку может включаться только одно действие. Для команд, которые учитывают варианты поведения, доступны следующие варианты:

accept

Разрешить получение, отправку или ретрансляцию пакетов, соответствующих правилу.

deny

Запретить получение, отправку или ретрансляцию пакетов, соответствующих правилу.

reject

Запретить получение, отправку или ретрансляцию пакетов, соответствующих правилу, а также вернуть ICMP-сообщение об ошибке узлу-отправителю.

Существуют следующие команды:

–a [*policy*]

Добавить одно или более правил в конец списка правил для категории. Для правил учета пакетов поведение (*policy*) не указывается. Правила для брандмауэров требуют задания варианта поведения. Когда происходит разрешение исходного и/или конечного имени в более чем одном

адресе, добавляется правило для каждой возможной комбинации адресов.

–c Проверить, будет ли данный IP-пакет принят (accepted), отвергнут (denied) или отклонен (rejected) указанным типом брандмауэра. Параметр допустим только в категориях **–O**, **–I** и **–F**. Должен быть указан параметр **–V**. (См. ниже раздел «Настройки».)

–d [*policy*]

Удалить одну или более записей из списка правил для данной категории. Для правил учета поведение (*policy*) не указывается. Аргументы этой команды должны в точности совпадать с аргументами команды добавления или вставки, иначе соответствие не будет найдено, а правило не будет удалено. Удаляется только первое соответствующее правило из списка.

–f Удалить (сбросить) все правила данной категории.

–h Отобразить справку с кратким описанием синтаксиса команд. Используется без задания категории:

```
% ipfwadm -h
```

–i [*policy*]

Вставить новое правило в начало списка правил данной категории. Для правил учета поведение (*policy*) не указывается. Для правил брандмауэров указание варианта поведения обязательно. Когда происходит разрешение исходного и/или конечного имени в более чем одном адресе, добавляется правило для каждой возможной комбинации адресов.

–l Перечислить все правила данной категории. Эта команда может сочетаться с параметром **–z** для сброса счетчиков байт и пакетов после отображения их текущих значений. Если не задан также и параметр **–x**, значения счетчиков отображаются в формате *number***K** или *number***M** с округлением до ближайшего целого. См. также описание параметра **–e** в разделе «Параметры».

–p *policy*

Изменить заданный по умолчанию вариант поведения для выбранного типа брандмауэра на *policy*. Поведение по умолчанию вступает в силу, если пакет не соответствует ни одному из правил. Применение команды корректно только с категориями **–I**, **–O** или **–F**.

ipfwadm

–s *tcp tcpfin udp*

Установить интервалы ожидания для сокрытия адресов. Применение корректно только с категорией **–M**. Все три аргумента являются обязательными и представляют собой значения интервалов ожидания в секундах для TCP-сеансов, TCP-сеансов после получения пакета FIN и UDP-сеансов. Если указан 0, то текущее значение аргумента сохраняется.

–z Обнулить счетчики байт и пакетов для всех правил в данной категории. Команда может использоваться совместно с командой **–l**.

Настройки

С командами **–a**, **–i**, **–d** и **–c** могут быть указаны следующие параметры. Обратное отмечается специально. Несколько параметров можно задать в одной командной строке **ipfwadm**.

–D *address*[/*mask*] [*port*...]

Спецификация пункта назначения (необязательная). Синтаксис, значения по умолчанию и прочие требования приведены в описании команды **–S**. Типы ICMP не могут указываться с **–D**.

–P *protocol*

Протокол правила или пакета; корректные значения: **tcp**, **udp**, **icmp** или **all**. По умолчанию – **all**, т. е. все протоколы. **–P** не может использоваться совместно с командой **–c**.

–S address[/*mask*] [*port*...]

Исходный IP-адрес, указанный в виде имени узла, имени сети или IP-адреса. Исходный адрес и маска по умолчанию имеют вид 0.0.0.0/0. Если указан параметр **–S**, также должен быть указан и параметр **–P**. Необязательная маска задается как маска сети либо как число единиц в левой части двоичного представления маски (например, маска 24 эквивалентна 255.255.255.0). По умолчанию значение маски равно 32. Может быть определено одно или несколько необязательных значений *port*, указывающих, к каким типам ICPM-портов должно применяться правило. По умолчанию *port* имеет значение **all**. Порты могут быть указаны в записях файла */etc/services*. Синтаксис для указания диапазона портов следующий:

```
lowport:highport
```

Например:

```
-S 172.29.16.1/24 ftp:ftp-data
```

–V *address*

Адрес сетевого интерфейса, через который получен пакет (для категории **–I**) либо через который посылается пакет (для категории **–O**). *address* может быть представлен в виде имени узла или IP-адреса и по умолчанию равен 0.0.0.0, что соответствует адресу любого интерфейса. Параметр **–V** обязателен к использованию с командой **–c**.

```
-V 172.29.16.1
```

–W *name*

Идентично **–V**, но вместо адреса указывается имя устройства:

```
-W ppp0
```

Параметры

–b Двунаправленный режим. Правило действует на входящие и исходящие IP-пакеты. Параметр можно применять только с командами **–a**, **–i** и **–d**.

–e Расширенный вывод. Используется совместно с командой **–l** для отображения адреса интерфейса и любых настроек правил. При перечислении правил для брандмауэров отображаются счетчики байт и пакетов, а также маски типов сервисов TOS (Type of Service). В случае применения с категорией **–M** отображается информация, связанная с числами дельта-последовательностей.

–k Работать с TCP-пакетами подтверждений, т. е. только с пакетами, содержащими установленный флаг ACK. При работе со всеми прочими протоколами этот параметр игнорируется, его использование корректно только с командами **–a**, **–i** и **–d**.

–m Пропустить для ретрансляции пакеты со скрытыми адресами таким образом, будто их отправителем является локальный узел. Происходит распознавание реверсных пакетов и автоматическое раскрытие адресов в обход ретрансляционного брандмауэра. Этот параметр действителен только для правил ретрансляционных брандмауэров с поведением **accept**. Ядро должно быть собрано с ключом CONFIG_IP_MASQUERADE.

–n Численный вывод. Отображать IP-адреса и номера портов в численном формате.

–o Вести записи в системном журнале о пакетах, которые соответствуют данному правилу. Параметр можно применять только с командами **–a**, **–i** и **–d**. Ядро должно быть собрано с ключом CONFIG_IP_FIREWALL_VERBOSE.

ipfwadm	**–r** [*port*] Перенаправить пакеты в локальный сокет, даже если адресатом является удаленный узел. Если указан порт 0 (значение по умолчанию), используется порт адресата пакета. Этот параметр действителен только для правил входных брандмауэров с поведением **accept**. Ядро должно быть собрано с ключом CONFIG_IP_TRANSPARENT_PROXY. **–t** *andmask xormask* Определить маски, применяемые для изменения поля TOS в заголовке пакета IP. Когда пакет принимается (со скрытыми адресами или без) правилом брандмауэра, происходит поразрядная операция (TOS AND *andmask*) XOR *xormask*. Маски задаются 8-битными шестнадцатеричными числами. Применение параметра корректно только с командами **–a**, **–i** и **–d** и не имеет смысла с правилами учета пакетов или с правилами брандмауэров, для которых выбрано поведение **deny** или **reject**. **–v** Подробный вывод. Отображается детальная информация о правиле или пакете – добавляемом, удаляемом или проверяемом. Параметр можно применять только с командами **–a**, **–i**, **–d** и **–c**. **–x** Расширение чисел. Отображается точное значение счетчиков пакетов и байтов вместо округленного. Использование этого параметра корректно только в случае, если задано отображение значений счетчиков (см. также параметр **–e**). **–y** Работать с TCP-пакетами с установленным флагом SYN и сброшенным флагом ACK. Параметр игнорируется для пакетов других протоколов и корректен только в случае применения с командами **–a**, **–i** и **–d**.
iptables	**iptables** *command* [*options*] Команда системного администрирования. Настройка правил сетевых фильтров (*netfilter*). В ветви ядра Linux 2.4 возможности брандмауэринга **ipchains** реализованы посредством модуля ядра *netfilter*, который может быть настроен точно так же, как **ipchains**, но при этом комплектуется модулем **iptables**, похожим на **ipchains** и предоставляющим возможности расширения. Правила *iptables* состоят из критериев отбора пакетов и целей, т. е. действий, совершаемых над пакетом, если он соответствует критериям.

iptables

Правила организованы в цепи (*chains*). Вы можете использовать их при создании брандмауэра, сокрытии адресов вашей локальной сети или просто отсечении определенных типов сетевых соединений.

Существует три предопределенных таблицы **iptables**; первая, **filter**, предназначена для фильтрации сетевых пакетов, вторая, **nat** – для трансляции сетевых адресов (Network Address Translation), а последняя, **mangle**, может применяться для внесения специфических изменений в сетевые пакеты. Правила для брандмауэра организуются в цепи, представляющие собой упорядоченные последовательности правил, которые проверяются ядром в процессе обработки пакета. Таблица **filter** содержит три предопределенные цепи: **INPUT**, **OUTPUT** и **FORWARD**. Цепи **INPUT** и **OUTPUT** применяются к входящим или исходящим пакетам. Цепь **FORWARD** предназначена для обработки пакетов, проходящих через данный узел и адресованных другой системе. Таблица **nat** также содержит три предопределенных цепи: **PREROUTING**, **POSTROUTING** и **OUTPUT**. В таблице **mangle** цепей только две: **PREROUTING** и **OUTPUT**.

Модуль *netfilter* проверяет входящие пакеты. После обработки всех правил цепи **PREROUTING**, в случае если пакет просто проходит через данный узел, модуль передает пакет в цепи **INPUT** или **FORWARD**. Исходящие пакеты обрабатываются сначала правилами цепи **OUTPUT**, а затем правилами цепи **POSTROUTING**. Каждая из цепей имеет цель по умолчанию – правило, применяемое, если пакет не соответствует ни одному из критериев цепи. Пользователь может создавать собственные цепи и отводить им роль целей для пакетов, но целей по умолчанию такие цепи иметь не могут. Если пакет не соответствует ни одному критерию в пользовательской цепи, то он возвращается в цепь, из которой произошел вызов, и проверяется на соответствие следующему правилу этой цепи.

Команда **iptables** лишь вносит изменения в правила работающего ядра. При выключении системы эти изменения не сохраняются. Для создания сценариев восстановления может применяться команда **iptables-save**. Восстановление настроек брандмауэра производится с помощью команды **iptables-restore**. Такой сценарий обычно вызывается при загрузке системы. Во многие дистрибутивы входит сценарий инициализации **iptables**, который использует результаты работы **iptables-save**.

iptables

Команды

iptables всегда выполняется с одной из следующих команд:

–A *chain rules*, **––append** *chain rules*

Добавить новые правила к цепи.

–I *chain number rules*, **––insert** *chain number rules*

Вставить правила (*rules*) в цепь (*chain*) после правила с порядковым номером *number*.

–D *chain rules*, **––delete** *chain rules*

Удалить правила из цепи. Правила могут определяться их порядковыми номерами в цепи или их описанием.

–R *chain number rule*, **––replace** *chain number rule*

Заменить правило в цепи. Заменяемое правило определяется порядковым номером *number*.

–C *chain rule*, **––check** *chain rule*

Выяснить, как будет обрабатываться сетевой пакет, соответствующий указанному правилу. Правило должно определять источник, назначение, протокол и интерфейс создаваемого пакета.

–L [*chain*], **––list** *$PARAMETER*

Перечислить правила в цепи либо во всех цепях, если цепь не указана.

–F [*chain*], **––flush** *chain*

Удалить все правила цепи либо правила всех цепей, если цепь не указана.

–Z [*chain*], **––zero** *chain*

Обнулить счетчики байт и пакетов для цепи. Если цепь не указана, происходит обнуление счетчиков для всех цепей. В этом случае, если выполняется также команда **–L**, значения счетчиков перед обнулением отображаются.

–N *chain*, **––new-chain** *chain*

Создать новую цепь. Имя цепи должно быть уникальным. Эта команда предоставляет пользователю возможность создавать собственные цепи.

–X [*chain*], **––delete-chain** *chain*

Удалить указанную цепь, созданную пользователем, либо все пользовательские цепи, если цепь не указана.

–P *chain target*, **––policy** *chain target*

Установить правило по умолчанию для предопределенной цепи; сама по себе цель (*target*) цепью являться не может.

–E *old-chain new-chain*, **––rename-chain** *old-chain new-chain*

Переименовать цепь *old-chain* в *new-chain*.

–h [icmp]

Отобразить краткую справку. Если указан параметр **icmp**, будут перечислены допустимые ICMP-типы.

Цели

Цель определяется именем цепи либо одним из следующих ключевых слов:

ACCEPT

Пропустить пакет.

DROP

Удалить пакет.

QUEUE

Передать пакеты на обработку правилам пользователя.

RETURN

Прекратить опрос правил текущей цепи и вернуть управление в точку предыдущей цепи, из которой произошел вызов. Если **RETURN** является целью правила предопределенной цепи, выполняется правило по умолчанию для этой цепи.

Параметры определения правил

Следующие параметры используются для создания правил, относящихся к предваряющим их командам. Правила состоят из критериев отбора пакетов и, как правило, цели (**–j**) для случая соответствия пакета критерию. Ко многим параметрам в правилах может применяться отрицание – восклицательный знак (**!**) или «не»; тогда критерию будет соответствовать все, кроме указанного параметра.

–p [!] *name*, **––protocol [!]***$PARAMETER*

Отбор пакетов протокола *name*. Значение *name* может быть строковым именем или соответствующим числом (из файла */etc/protocols*). Наиболее часто встречаются значения **tcp**, **udp**, **icmp** или ключевое слово **all**. Число 0 идентично **all**; это значение является значением по умолчанию, если не определяется явно данным параметром.

iptables

Если существуют расширенные правила, связанные с данным протоколом, они будут загружены автоматически; нет необходимости загружать их явно с помощью параметра **–m**.

–s [**!**] *address*[/*mask*] [**!**] [*port*], **––source** [**!**] *address*[/*mask*] [**!**] [*port*]

Отбор пакетов, приходящих с указанного адреса. Адрес может указываться в виде имени узла, имени сети либо в виде IP-адреса. Необязательный параметр *mask* определяет сетевую маску и может представляться в классическом формате (например, /255.255.255.0) или в современной, более краткой форме (например, /24).

–d [**!**] *address*[/*mask*] [**!**] [*port*], **––destination** [**!**] *address*[/*mask*] [*port*]

Отбор пакетов с определенным адресом назначения (*address*). Синтаксис параметра идентичен синтаксису для параметра **–s**.

–j *target*, **––jump** *target*

Переход к специальной цели либо определенной пользователем цепи правил. Если для правила этот параметр не определен, соответствие пакета правилу лишь увеличивает счетчики правила, а пакет передается для проверки на соответствие следующему правилу.

–i [**!**] *name*[+], **––in-interface** *name*[+]

Отбор пакетов, получаемых через интерфейс *name*[+]. *name* – это сетевой интерфейс, используемый системой (например, **eth0** или **ppp0**). Для создания маски применяется символ «+». Так, значению **ppp+** соответствуют все имена интерфейсов, начинающиеся с **ppp**.

–o [**!**] *name*[+], **––out-interface** *name*[+]

Отбор пакетов, отправляемых через интерфейс *name*[+]. См. описание синтаксиса *name* для параметра **–i**.

[**!**] **–f**, [**!**]**––fragment** *$PARAMETER*

Правило отбирает все, кроме первого, фрагменты разбитого на части пакета.

Параметры

–v, **––verbose**

Режим подробной диагностики.

−n, −−numeric

Вывести все IP-адреса и номера портов в численном виде. По умолчанию, где возможно, отображаются имена.

−x, −−exact

Производить подстановку чисел при перечислении (**−L**). Отображать точные значения счетчиков байт и пакетов вместо округленных.

−m *module*, **−−match**

Явным образом загрузить расширение правила, связанное с указанным модулем (*module*). См. ниже раздел «Расширения критериев».

−h [icmp], −−help [icmp]

Вывести справку. Если указан параметр **icmp**, перечислить имена доступных ICMP-типов. **−h** также может указываться совместно с **−m** для получения справки по модулю расширения.

−−line-numbers

Применяется совместно с командой **−L**. Добавлять номер строки в начало каждого правила в перечислении, определяя таким образом позицию правила в цепи.

Расширения критериев

Помимо *netfilter* ядро содержит и другие модули, которые позволяют расширить возможности отбора в правилах. Модули, связанные с определенными протоколами, загружаются автоматически, когда для задания протокола используется команда **−p**. Прочие модули необходимо загружать явно с помощью параметра **−m**.

tcp Загружается, когда единственным заданным протоколом является **−p tcp**.

−−source-port [!] [*port***][:***port***], −−sport [!] [***port***][:***port***]**

Исходные порты. Можно задать включающий диапазон сервисов при помощи двоеточия. Если первый порт не указан, по умолчанию подставляется значение 0. Если второй порт опущен, по умолчанию подставляется значение 65535. Символ «−» также может использоваться для задания диапазона.

−−destination-port [!] [*port***][:***port***], −−dport [!] [***port***][:***port***]**

Целевые порты. Синтаксис идентичен синтаксису для **−−source-port**.

iptables

––tcp-flags [!] *mask comp*

Пакеты с TCP-флагами, определяемыми аргументами *mask* и *comp*. *mask* – это список флагов, которые необходимо проверять, *comp* – список флагов, которые должны быть установлены, чтобы правило сработало. Элементы списков разделяются запятыми. Допустимые флаги: SYN, ACK, FIN, RST, URG, PSH, ALL и NONE.

[!] ––syn

Пакеты с установленным битом SYN и сброшенными битами ACK и FIN. Это пакеты запросов TCP-соединений; блокировка прохождения этих пакетов запрещает входящие соединения. Расширение является сокращением для **––tcp-flags SYN,RST,ACK SYN**.

udp

Загружается, когда единственным заданным протоколом является **–p udp**.

––source-port [!] [*port***][:***port***], ––sport [!] [***port***][:***port***]**

Исходные порты. Синтаксис идентичен синтаксису **––source-port** расширения TCP.

––destination-port [!] [*port***][:***port***], ––dport [!] [***port***][:***port***]**

Целевые порты. Синтаксис идентичен синтаксису **––source-port** расширения TCP.

icmp

Загружается, когда единственным заданным протоколом является **–p icmp**.

––icmp-type [!] *type*

Пакеты указанного ICMP-типа. Тип может определяться ICMP-именем или соответствующим числом. Значения доступны по команде **iptables –p icmp –h**.

mac

Загружается явно, если указан параметр **–m**.

––mac-source [!] *address*

Адрес интерфейса, через который получен пакет. Адрес задается шестнадцатеричными числами через двоеточие (например, **––mac-source 00:60:08:91:CC:B7**).

limit

Загружается явно посредством параметра **–m**. Расширение **limit** указывается для ограничения числа отбираемых пакетов. Такая возможность полезна для применения

совместно с целью **LOG**. Правила, использующие данное расширение, производят отбор пакетов, пока указанный предел не будет достигнут.

––limit *rate*

Отбор адресов с заданной скоростью. Скорость (*rate*) определяется числом и необязательным модификатором **/second**, **/minute**, **/hour** или **/day**. Если параметр не определен, то по умолчанию принимается скорость '3/hour'.

––limit-burst [*number*]

Установить максимально допустимое количество пакетов, отбираемых подряд. Когда ограничение исчерпано, отбор пакетов прекращается до обнуления счетчика. Обнуление происходит в соответствии со скоростью, установленной параметром **––limit**. По умолчанию ограничение имеет значение 5.

multiport

Загружается явно посредством параметра **–m**. Расширение **multiport** позволяет производить отбор групп исходных или целевых портов. Правила могут применяться только совместно с **–p tcp** и **–p udp**. Список, элементы которого разделяются запятыми, может содержать до 15 значений портов.

––source-port [*ports*]

Отбор по исходным портам.

––destination-port [*ports*]

Отбор по целевым портам.

––port [*ports*]

Отбор по совпадению исходного и целевого порта и наличию этого порта в списке.

mark

Загружается явно посредством параметра **–m**. Этот модуль предназначен для использования с дополнительной целью **MARK**:

––mark *value*[/*mask*]

Отбор по заданному положительному значению отметки. Если указана маска, выполняется операция ее логического умножения (AND) на значение отметки перед сравнением.

iptables

owner

Загружается явно посредством параметра **–m**. Расширение **owner** позволяет производить отбор по пользователю, создавшему пакет, групповому процессу либо идентификатору сеанса. Разумно применять только в качестве части цепи **OUTPUT**.

––uid-owner *userid*

Отбор пакетов, созданных процессом, который принадлежит указанному пользователю.

––gid-owner *groupid*

Отбор пакетов, созданных процессом, который принадлежит указанной группе.

––pid-owner *processid*

Отбор пакетов, созданных указанным процессом.

––sid-owner *sessionid*

Отбор пакетов, созданных процессом в указанном сеансе *sessionid*.

state

Загружается явно посредством параметра **–m**. Модуль позволяет производить отбор на основе состояния соединения для пакета.

––state *states*

Отбор пакета в случае его соответствия одному из состояний списка *states* (элементы разделяются запятыми). Доступные значения: **INVALID**, **ESTABLISHED**, **NEW** и **RELATED**.

tos Загружается явно посредством параметра **–m**. Этот модуль позволяет производить отбор по полю Type of Service (Тип сервиса) заголовка пакета.

––tos *value*

Отбор пакета, если его поле TOS имеет указанное значение. Значение может определяться числом или строковым идентификатором Type of Service. Доступные значения можно получить, выполнив команду **iptables –m tos –h**.

Целевые расширения

Целевые расширения являются необязательными дополнительными целями, которые поддерживаются отдельными модулями ядра. Каждая дополнительная цель имеет собственные уникальные параметры.

LOG

Запись информации о пакетах в системный журнал.

––log-level *level*

Задать уровень по имени или номеру (как определяется в файле *syslog.conf*).

––log-prefix *prefix*

Предварять каждую запись в журнале строкой *prefix*. Длина строки *prefix* не может превышать 30 символов.

––log-tcp-sequence

Регистрировать числа TCP-последовательности. Небезопасно, если журнал доступен пользователям для чтения.

––log-tcp-options

Регистрировать параметры из заголовков TCP-пакетов.

––log-ip-options

Регистрировать параметры из заголовков IP-пакетов.

MARK

Служит для отметки пакетов положительными целыми значениями, которые позже могут использоваться как расширения критерия **mark**. Допустимо применение только с таблицей **mangle**.

––set-mark *value*

Пометить пакет указанным значением.

REJECT

Удалить пакет и при необходимости послать отправителю ICMP-уведомление об удалении пакета. Такое уведомление не посылается, если пакет содержал ICMP-сообщение об ошибке, неизвестный ICMP-тип или недопустимый фрагмент, либо если слишком большое количество сообщений ICMP уже было отправлено на данный адрес.

––reject-with *type*

Отправить сообщение ICMP указанного типа. Допустимые типы: **icmp-net-unreachable**, **icmp-host-unreachable**, **icmp-port-unreachable** или **icmp-protocol-unreachable**. Для ICMP-пакетов ping, аргумент *type* может принимать значение **echo-reply**.

iptables

TOS

Установить поле Type of Service в заголовке IP-пакета. **TOS** является допустимой целью только для правил таблицы **mangle**.

––set-tos *value*

Установить поле TOS в значение *value*. Может быть указано 8-битное значение либо идентификатор TOS. Список доступных идентификаторов можно получить по команде **iptables –j TOS –h**.

SNAT

Изменить исходный адрес пакета и всех последующих пакетов текущего соединения. Применение **SNAT** допустимо только для цепи **POSTROUTING** таблицы **nat**.

––to-source *address*[–*address*][*port*–*port*]

Определить новый исходный адрес или диапазон адресов. Если параметром **–p** был определен протокол **tcp** или **udp**, то могут быть также заданы и исходные порты. Если порты не указаны, по возможности происходит отображение в соответствующие порты. В противном случае порты с номерами, меньшими 512, отображаются в другие порты с номерами, меньшими 512, порты с номерами в диапазоне 512–1024 отображаются в другие порты с номерами, меньшими 1024, а порты с номерами, большими 1024 – в другие порты с номерами, большими 1024.

DNAT

Изменение целевого адреса пакета и всех последующих пакетов текущего соединения. Использование **DNAT** допустимо только для цепи **POSTROUTING** таблицы **nat**.

––to-destination *address*[–*address*][*port*–*port*]

Определить новый целевой адрес или диапазон адресов. Синтаксис аргументов идентичен синтаксису для цели **SNAT**.

MASQUERADE

Скрыть адрес пакета, чтобы он считался исходящим из текущей системы. Ответные пакеты в маскированных соединениях автоматически получают нужный целевой адрес. Данная цель является допустимой только для цепей таблицы *nat*, обрабатывающих входящие пакеты, и должна применяться только с динамическими IP-адресами

iptables

(как при коммутируемых соединениях). Для статических IP-адресов следует использовать **DNAT**.

−−to-ports *port*[−*port*]

Указать порт или диапазон портов для применения при сокрытии адресов. Параметр допустим только в случае указания протоколов **tcp** и **udp** посредством команды **−p**. Если данный параметр опущен, порт пакета, для которого производится сокрытие адреса, не будет изменен.

REDIRECT [**−−to-port** *port*]

Переадресовать пакет на локальный порт. Полезно при создании прозрачных прокси-серверов.

−−to-ports *port*[−*port*]

Указать порт или диапазон портов локальной системы для переадресации пакета. Параметр допустимо использовать только в случае указания протоколов **tcp** и **udp** посредством команды **−p**. Если данный параметр опущен, порт пакета, для которого производится переадресация, изменен не будет.

---

iptables-restore [*file*]

iptables-restore

Команда системного администрирования. Восстановление правил брандмауэра. Для восстановления цепей и правил в них **iptables-restore** использует набор команд, созданный **iptables-save**. Команда часто вызывается из сценариев инициализации для восстановления настроек брандмауэра при загрузке. Параметр *file* определяет имя файла, созданного **iptables-save**. Если имя файла не указано, выполняется чтение со стандартного ввода. На момент отправки настоящего издания книги в печать разработка программы еще не была завершена, поэтому она может иметь не перечисленные здесь возможности.

---

iptables-save [*chain*]

iptables-save

Команда системного администрирования. Печать правил брандмауэра, хранящихся в памяти ядра, на стандартный вывод. Если цепь не указана, будут отображены правила для всех цепей. Вывод может быть перенаправлен в файл и впоследствии использован для восстановления значений с помощью **iptables-restore**. Разработка программы не была завершена на момент написания настоящего издания, поэтому она может иметь не перечисленные здесь возможности.

ispell

ispell [*options*] [*files*]

Сравнение слов из указанных файлов с набором слов из словаря. Неопознанные слова отображаются в верхней части экрана в сопровождении возможных вариантов корректного написания; кроме того, возможно редактирование с помощью ряда команд.

Параметры

–b Создать резервную копию исходного файла с именем *filename.**bak***.

–d *file*

Использовать файл *file* в качестве стандартного словаря.

–m Предлагать различные комбинации корня/аффикса.

–n Исходный файл является результатом работы nroff или troff.

–p *file*

Файл *file* является дополнительным пользовательским словарем.

–t Исходный файл имеет формат TEX или LATEX.

–w *chars*

Добавить символы (*chars*) к набору допустимых в слове в дополнение к a–z, A–Z.

–x Не создавать резервную копию исходного файла.

–B Помимо обычных ошибок найти пропущенные пробелы («склейки» слов).

–C Не выдавать сообщение об ошибке при встрече склейки слов.

–L *number*

Показывать указанное количество (*number*) строк контекста.

–M Вывести список диалоговых команд в нижней части экрана.

–N Подавлять вывод диалоговых команд.

–P Предлагать минимальное количество вариантов корня/аффикса.

–S Сортировать предлагаемые замены по их вероятной предпочтительности.

ispell

–T *type*

Считать, что файлы имеют форматирование *type*.

–W *n*

Считать, что все слова, длина которых меньше либо равна *n*, орфографически правильны.

–V Использовать запись (**^L**) для отображения управляющих символов, а префикс **M–** – для записи символов с установленным старшим битом.

Диалоговые команды

? Вывести информацию по применению.

пробел

Принять имеющееся написание слова.

number

Заменить слово другим словом с соответствующим порядковым номером (*number*).

!*command*

Запустить командный интерпретатор и выполнить команду (*command*). Запросить подтверждение выхода из интерпретатора.

a Считать слово правильным, но не добавлять его в пользовательский словарь.

i Считать слово правильным и добавить в пользовательский словарь (с учетом регистра).

l Произвести поиск слова в стандартном словаре.

q Завершить работу без сохранения результатов.

r Заменить слово.

u Считать слово правильным и добавить его в пользовательский словарь (в регистре строчных букв).

x Сохранить изменения и перейти к обработке следующего файла.

^L Обновить изображение на экране.

^Z Приостановить работу **ispell**.

join

join [*options*] *file1 file2*

Слияние строк из двух отсортированных файлов путем сопоставления общего поля. Если вместо имени одного из файлов стоит символ «–», происходит чтение со стандартного ввода.

join

Параметры

–a *filenum*

Для каждой строки файла *filenum*, не имеющей пары, в дополнение к стандартной информации вывести саму строку.

–e *string*

Заменять отсутствующие поля ввода строкой *string*.

–i, ––ignore-case

Игнорировать регистр символов при сравнении ключей.

–1 *fieldnum1*

Поле для сравнения из первого файла (*file1*) – *fieldnum1*. По умолчанию это первое поле строки.

–2 *fieldnum2*

Поле для сравнения из второго файла (*file2*) – *fieldnum2*. По умолчанию это первое поле строки.

–o *fieldlist*

Упорядочить поля вывода в соответствии со списком *fieldlist*. Элементы списка имеют вид *filenum.fieldnum* и разделяются запятыми или пробелами.

–t *char*

Считать символ *char* разделителем полей в строках (по умолчанию это пробелы).

–v *filenum*

Отображать только строки, не поддающиеся слиянию, из указанного файла *filenum*.

––help

Вывести справку и завершить работу.

––version

Вывести номер версии и завершить работу.

kbd_mode

kbd_mode [*option*]

Вывести или установить текущий режим работы клавиатуры. Допустимые режимы: RAW, MEDIUMRAW и XLATE.

Параметры

–a Установить режим XLATE (режим ASCII).

–k Установить режим MEDIUMRAW (режим кодов клавиш).

kbd_mode

–s Установить режим RAW (режим скан-кодов).

–u Установить режим UNICODE (режим UTF-8).

kbdrate

kbdrate [*options*]

Команда системного администрирования. Установка скорости автоповтора для нажатий клавиш, а также времени задержки. Выполнение команды без аргументов устанавливает скорость повтора, равную 10,9 символам в секунду, и задержку, равную 250 миллисекундам. Однако при загрузке Linux устанавливает скорость автоповтора, равную 30 символам в секунду.

Параметры

–s Не выводить сообщения.

–r *rate*

Указать скорость автоповтора (в символах в секунду), значения которой заданы следующими числами: 2,0; 2,1; 2,3; 2,5; 2,7; 3,0; 3,3; 3,7; 4,0; 4,3; 4,6; 5,0; 5,5; 6,0; 6,7; 7,5; 8,0; 8,6; 9,2; 10,0; 10,9; 12,0; 13,3; 15,0; 16,0; 17,1; 18,5; 20,0; 21,8; 24,0; 26,7 или 30,0.

–d *delay*

Указать задержку (в миллисекундах). Может принимать значения: 250, 500, 750 или 1000.

kerneld

kerneld

Команда системного администрирования. Она автоматически загружает модули ядра по мере надобности, уменьшая таким образом расход памяти на загруженные и неиспользуемые модули и избавляя пользователя от необходимости вручную загружать модули с помощью команды **modprobe** или **insmod**. Если к модулю нет обращений более одной минуты, **kerneld** автоматически выгружает его из памяти.

Команда **kerneld** входит в состав пакета модулей и утилит ядра и устанавливается при конфигурировании ядра. Ее функциональность обеспечивается взаимодействием этого пакета и ядра. **kerneld** известны основные типы модулей. Если для работы устройства могут использоваться различные модули (как в случае с сетевыми драйверами), **kerneld** обращается к файлу настройки */etc/conf.modules*, содержащему информацию о путях и псевдонимах для всех загружаемых модулей, чтобы определить, какой именно модуль должен быть загружен.

kerneld	**kerneld** может также применяться для реализации коммутируемой работы с сетью по запросу, как в случаях с соединениями SLIP или PPP. Запросы на соединение с сетью обрабатываются **kerneld**, которая загружает необходимые модули и устанавливает соединение с сервером.
kill	**kill** [*option*] *IDs* Это команда **/bin/kill**; также существует команда интерпретатора с таким же именем. Посылка сигнала принудительного завершения перечисленным процессам (по идентификаторам *ID*). Если не задан определенный сигнал, посылается сигнал TERM. Завершить процесс может либо его владелец, либо привилегированный пользователь. **Параметры** **–l** Перечислить допустимые сигналы. **–p** Вывести идентификатор процесса указанного задания, но не посылать сигнал. Чтобы воспользоваться этим параметром, необходимо указать полное имя команды (например, **/bin/kill –p**). –*signal* Номер сигнала (в соответствии с файлом */usr/include/sys/signal.h*) или его имя (допустимые имена перечисляются по команде **kill –l**). Если номер сигнала равен 9 (KILL), то сигнал не может быть перехвачен процессом, что используется, когда обычная команда **kill** не позволяет завершить работу процесса. Сигнал по умолчанию – TERM.
killall	**killall** [*options*] *names* Принудительное завершение процессов по именам. Если более одного процесса выполняют указанную задачу, завершить все процессы. Имена задач, содержащие символ «/», считаются файлами; завершаются все процессы, выполняющие указанный файл. **Параметры** –*signal* Послать процессам сигнал (по умолчанию – TERM). Сигнал (*signal*) может задаваться числом или именем.

killall

–e Предписание искать точное совпадение для длинных имен (более 15 символов длиной), прежде чем посылать сигнал. Обычно команда **killall** посылает сигнал всем процессам, имена которых совпадают с заданным в пределах первых 15 символов. При указании параметра **–e** имена, не совпадающие полностью с заданным, игнорируются (используйте параметр **–v**, чтобы выводить сообщения для каждого пропущенного имени).

–g Завершить всю группу процессов, к которой принадлежит указанный.

–i Запрашивать подтверждение перед завершением процесса.

–l Перечислить существующие имена сигналов.

–q Режим минимального вывода; не отображать сообщения о процессах, которые не были завершены по сигналу.

–v Режим подробной диагностики: выводить сообщение об успешном завершении процесса и идентификатор процесса.

–V Вывести информацию о версии команды.

–w Выждать, пока все завершенные процессы освободят ресурсы. Следует помнить, что **killall** может ожидать бесконечно долго, если сигнал был проигнорирован или не имел эффекта, либо если процесс остался в состоянии зомби (zombie).[1]

killall5

killall5

Эквивалент **killall** для систем Unix System V. Команда завершает все процессы, кроме тех, которые необходимы для ее работы.

klogd [*options*]

klogd

Команда системного администрирования. Управляет отображением сообщений ядра на консоли, расставляет приоритеты сообщений и ведет журнал посредством **syslogd**. Во многих операционных системах все операции **klogd** выполняет **syslogd**, но в Linux это два разных демона.

[1] Зомби (zombie) – убитый процесс, еще не удаленный из таблицы процессов, т. к. его родительский процесс еще не получил код завершения процесса при помощи системного вызова wait. – *Примеч. науч. ред.*

klogd

Сообщения ядра собираются из файловой системы */proc* и системных вызовов **syslogd**. По умолчанию сообщения на консоли не появляются. Сообщения разбиты на 8 уровней (0–7), и каждое сообщение предваряется номером уровня.

Уровни приоритетов

0 Аварийная ситуация (**KERN_EMERG**).

1 Произошла критическая ошибка (**KERN_ALERT**).

2 Произошла серьезная ошибка (**KERN_CRIT**).

3 Произошла ошибка (**KERN_ERR**).

4 Предупреждение (**KERN_WARNING**).

5 Ситуация штатная, но требует проверки (**KERN_NOTICE**).

6 Уведомление (**KERN_INFO**).

7 Отладочное сообщение (**KERN_DEBUG**).

Параметры

–c *level*

Выводить на консоль все сообщения с более высоким приоритетом (с меньшим числом), чем *level*.

–d Режим отладки.

–f *file*

Записывать все сообщения в указанный файл; подавить нормальное ведение журнала.

–k *file*

Использовать указанный файл как источник имен ядра.

–n Избегать автоматического перехода в фоновый режим. Это необходимо, если **klogd** запускается из **init**.

–o Разовое задание. Расставить приоритеты и записать в журнал все имеющиеся сообщения, а затем завершить работу.

–s Подавить чтение сообщений из файловой системы */proc*.

Файлы

/usr/include/linux/kernel.h, */usr/include/sys/syslog.h*

Исходные тексты, определяющие уровни приоритетов.

/proc/kmsg

Файл, из которого **klogd** извлекает сообщения.

/var/run/klogd.pid

Идентификатор процесса **klogd**.

klogd

---

ksyms [*options*]

ksyms

Команда системного администрирования. Перечислить экспортируемые имена, адреса и определяющие модули ядра (по необходимости).

Параметры

–a Включить имена из выгруженных модулей.

–h Подавить вывод заголовка.

–m Включать адреса и размеры. Полезно только для имен загруженных модулей.

Файл

/proc/ksyms

Еще один источник той же информации.

---

lastlog [*options*]

lastlog

Команда системного администрирования. Вывод времени последнего входа в систему для учетных записей. Выполняется чтение информации из файла */var/log/lastlog*.

Параметры

–t*n*

Данные о входе в систему не позже чем *n* дней назад.

–u*name*

Информация только по указанному пользователю.

---

ld [*options*] *objfiles*

ld

Сборка нескольких объектных файлов в определенном порядке в исполняемый объектный модуль (по умолчанию получает имя *a.out*). **ld** является редактором связей и часто выполняется автоматически по вызову компилятора.

Параметры

–c *file*

Выполнить команды из файла *file*.

ld

–d, **–dc**, **–dp**

Принудительное выделение памяти под общие имена.

–defsym *symbol* = *expression*

Создать глобальное имя *symbol*, имеющее значение *expression*.

–e *symbol*

Определить *symbol* как значение адреса входной точки получаемого файла.

–i Создать связываемый конечный файл; попробовать установить магическое число (magic number) для файла в значение OMAGIC.

–l*arch*

Включить архивный файл *arch* в список связываемых файлов.

–m *linker*

Эмулировать связывание в стиле *linker*.

–n Сделать текст доступным только для чтения; попробовать установить значение NMAGIC.

–noinhibit-exec

Создавать конечный файл, даже если произошла ошибка.

–o *output*

Имя конечного файла *output*, а не *a.out*.

–oformat *format*

Указать формат конечного файла.

–r Создать связываемый конечный файл; попробовать установить магическое число для файла в значение OMAGIC.

–s Не включать в конечный файл информацию об именах.

–shared

Создать разделяемую библиотеку.

–sort-common

Не производить сортировку глобальных общих имен по размеру.

–t Отображать имя каждого из исходных файлов по мере их обработки.

–u *symbol*

Сделать имя *symbol* неопределенным.

–v, ––version

Вывести номер версии **ld**.

––verbose

Вывести информацию о **ld**. Отображать имена исходных файлов в момент обращения к ним.

–warn-common

Отобразить предупреждение при нахождении общих имен, сочетаемых с другими логическими структурами.

–warn-once

Отображать только одно предупреждение для каждого неопределенного имени.

–x При использовании совместно с **–s** или **–S** удалять все локальные имена, начинающиеся с **L**.

–L *dir*

Сначала искать в каталоге *dir*, а затем в стандартных каталогах (этот параметр должен предшествовать параметру **–l**, задающему поиск в том же каталоге.).

–M Выдать карту связей на стандартный вывод.

–Map *file*

Вывести карту связей в файл *file*.

–N Разрешить чтение и запись как данных, так и текста. Отмечать результат, поддерживающий магические числа Unix; не выполнять постраничное выравнивание данных.

–R *file*

Получить имена и адреса, содержащиеся в указанном файле, но подавить перемещение файла и его включение в конечный результат.

–S Не включать отладочную информацию об именах.

–Tbss *address*

Сегмент bss результата начинается по указанному адресу.

–Tdata *address*

Сегмент данных начинается по указанному адресу.

–Ttext *address*

Сегмент текста начинается по указанному адресу.

–Ur Синоним **–r**, за исключением сборки программ на C++, где **–Ur** является предписанием разрешать ссылки конструкторов.

ld

–X При использовании совместно с параметром **–s** или **–S** удалять имена, начинающиеся с **L**.

–V Вывести номер версии и возможные значения эмулируемых редакторов связей для параметра **–m**.

ldconfig

ldconfig [*options*] *directories*

Команда системного администрирования. Изучить библиотеки в указанных каталогах (*directories*), */etc/ld.so.conf*, */usr/lib* и */lib*; по необходимости произвести обновление связей и кэша. Обычно выполняется в стартовых файлах системы или после установки дополнительных разделяемых библиотек.

Параметры

–D Отладка. Запретить обычные действия.

–l Библиотечный режим. Считать, что аргументами являются библиотеки, а не имена каталогов. Способ связывания библиотек вручную.

–n Не обращать внимания на содержимое каталогов */usr/lib* и */lib*, а также файла */etc/ld.so.conf*; не производить кэширование.

–N Не производить кэширование, только связывать.

–p Отображать все каталоги и библиотеки, содержащиеся в кэше. Аргументы игнорируются.

–v Диагностика работы. Включает номер версии и отображение имен каталогов при сканировании, а также имен ссылок при создании.

–X Не производить связывание; только создавать кэш.

Файлы

/lib/ld.so

Редактор связей и загрузчик.

/etc/ld.so.conf

Перечень каталогов, содержащих библиотеки.

/etc/ld.so.cache

Перечень библиотек в каталогах, перечисленных в файле */etc/ld.so.conf*.

ldd [*options*] *programs* **ldd**

Отобразить список разделяемых библиотек, необходимых для работы каждой из перечисленных программ *programs*.

Параметры

–v Информация о номере версии **ldd**.

–V Информация о номере версии редактора связей.

less [*options*] [*filename*] **less**

less – это программа постраничного просмотра файлов или результатов работы какой-либо команды. Программа была написана в качестве протеста против вопиющей примитивности **more** (откуда ее название). Некоторым командам может предшествовать число.

Параметры

–[z]*num*

Установить количество строк прокрутки. По умолчанию это число строк экрана. Отрицательное значение *num* вычитается из текущего количества строк прокрутки.

+[+]*command*

В начале работы выполнить команду *command*. Если команда является числом, перейти к строке с указанным номером. Параметр «++» применяет команду ко всем файлам, перечисленным в командной строке.

–? Вывести справку по **less**. Все прочие параметры игнорируются, листания файла не происходит.

–a Начинать поиск со строки, следующей за последней из отображенных на экране строк. По умолчанию поиск начинается со второй из отображенных строк.

–b*buffers*

Количество буферов на каждый файл (по умолчанию равно 10). Размер одного буфера – 1 Кбайт.

–c Обновлять экран сверху вниз, а не наоборот.

–d Подавить сообщения об ошибках «немого» терминала (dumb terminal).

–e Автоматически завершить работу после двукратного достижения символа EOF.

less

–f Принудительное открытие каталогов и устройств; не выдавать предупреждения при открытии бинарных файлов.

–g Выделить только подстроку, найденную последней командой поиска, а не все найденные строки.

–h*num*

Запретить обратную прокрутку более чем на *num* строк за один раз.

–i Игнорировать регистр символов при поиске, если строка поиска не содержит прописных букв.

–j*num*

Располагать целевую строку в строке экрана с номером *num*. Целевой строкой может быть строка с результатом поиска или строка, к которой совершен переход. Отсчет строк начинается с 1 (самая верхняя строка). Для отрицательного параметра *num* производится отсчет от последней строки экрана.

–k*file*

Прочитать файл определений специальных сочетаний клавиш.

–m Отобразить приглашение в стиле **more** с отображением количества прочитанного текста в процентах.

–n Не вычислять номера строк. Действует совместно с параметрами **–m** и **–M** и с командами **=** и **v** (отключение передачи номера строки редактору).

–o*file*

Если информация поступает с конвейера (pipe), дублировать вывод в файл (*file*) помимо отображения на экране. (Если файл уже существует, запрашивать подтверждение на перезапись.)

–p*pattern*

В начале работы произвести поиск первого соответствия шаблону.

m Приглашение средней длины (определяется параметром **–m**).

M Длинное приглашение (определяется параметром **–M**).

= Сообщение, отображаемое командой **=**.

−q Отключить звуковой сигнал, издаваемый при попытке прокрутить окно за символ EOF или начало файла. Вместо этого по возможности использовать видимый сигнал.

−r Отображать управляющие символы в обычном виде, а не в записи ^*x*. Применение этого параметра может привести к проблемам с отображением информации.

−s При выводе заменять несколько подряд идущих пустых строк одной пустой строкой.

−t*tag*

Редактировать файл, содержащий тег *tag*. Производить поиск в ./*tags* (созданном с помощью **ctags**).

−u Считать забой и возврат каретки отображаемым вводом.

−w Вывести строки за символом EOF пустыми, не помечая символом «~» (тильда).

−x*n*

Установить ширину табуляции в *n* символов. По умолчанию равна 8.

−y*n*

Запретить прямую прокрутку более чем на *n* строк единовременно.

−B Не выделять автоматически буферное пространство для данных, поступающих с конвейера. Если задан параметр **−b**, выделять указанное в этом параметре количество буферов. При необходимости уничтожать информацию предыдущего экрана.

−C Обновить экран, очистив его и отобразив всю информацию заново.

−E Автоматически завершить работу по достижении символа EOF.

−G Не выделять найденные при поиске строки.

−I Игнорировать регистр символов при поиске, даже если строка поиска содержит прописные символы.

−M Приглашение более подробное, чем при использовании **−m**: отображать процент прочитанного текста, номер текущей строки, общее количество строк.

−N Вывести порядковый номер каждой строки.

less

–O*file*

Сходно с **–o**, но без запроса на перезапись файла.

–P[m,M,=]*prompt*

Установить приглашение (определяемое параметром **–m**, **–M** или **=**). По умолчанию выдается короткое приглашение (**–m**).

–Q Запретить использование терминального звукового сигнала.

–S Отбрасывать хвосты длинных строк, а не отображать их на следующей строке.

–T*file*

Используется совместно с режимом **–t** или командой **:t**. Читать указанный файл вместо *./tags*.

–U Считать забой и возврат каретки управляющими символами.

–X Не посылать на терминал termcap-строки инициализации и деинициализации.

Команды

Командам могут предшествовать численные аргументы, которые фигурируют в описаниях команд как *number*.

SPACE, ^V, f, ^F

Прямая (вперед) прокрутка на заданное количество строк (по умолчанию на один экран текста).

z Аналогично SPACE, но позволяет задавать количество строк, и в этом случае делает его количеством строк прокрутки по умолчанию.

RETURN, ^N, e, ^E, j, ^J

Прямая прокрутка. По умолчанию – на одну строку. Отображать все строки, даже если заданное количество строк превышает по совокупности экран текста.

d, ^D

Прямая прокрутка. По умолчанию – на половину экранной страницы. Возможно задание количества строк прокрутки, в этом случае оно становится количеством строк прокрутки по умолчанию.

b, ^B, Esc-v

Обратная прокрутка. По умолчанию – на одну экранную страницу текста.

w Аналогично **b**, но позволяет задавать количество строк (с автоматическим изменением умолчания на это количество).

y, ^Y, ^P, k, ^K

Обратная прокрутка. По умолчанию – на одну строку. Отображать все строки, даже если заданное по умолчанию количество строк превышает по совокупности экранную страницу текста.

u, ^U

Обратная прокрутка. По умолчанию – на половину экранной страницы. Возможно задание количества строк прокрутки (с автоматическим обновлением этого количества по умолчанию).

r, ^R, ^L

Обновить содержимое экранного окна.

R Аналогично **r**, но без сохранения буферизованного ввода.

F Прямая прокрутка. По достижении символа EOF продолжать пролистывание в поисках строк текста. Поведение команды сходно с **tail –f**.

g, <, ESC-<

Перейти к строке. По умолчанию – к строке 1.

G,>, ESC->

Перейти к строке. По умолчанию – к последней строке.

p, %

Перейти к строке, находящейся на соответствующем расстоянии (*number*) от начала текста (в процентах).

{ Если первая строка экранной страницы содержит символ «{», найти парный символ «}». Если символов «{» в строке несколько, необходимо задать порядковый номер (*number*) символа для нахождения соответствующей пары.

} Если последняя строка экранной страницы содержит символ «}», найти парный символ «{». Если символов «}» в строке несколько, необходимо задать порядковый номер символа (*number*) для нахождения соответствующей пары.

(Если первая строка экранной страницы содержит символ «(», найти парный символ «)». Если символов «(» в строке несколько, необходимо задать порядковый номер символа (*number*) для нахождения соответствующей пары.

less

) Если последняя строка экранной страницы содержит символ «)», найти парный символ «(». Если символов «)» в строке несколько, необходимо задать порядковый номер символа (*number*) для нахождения соответствующей пары.

[Если первая строка экранной страницы содержит символ «[», найти парный символ «]». Если символов «[» в строке несколько, необходимо задать порядковый номер символа (*number*) для нахождения соответствующей пары.

] Если последняя строка экранной страницы содержит символ «]», найти парный символ «[». Если символов «]» в строке несколько, необходимо задать порядковый номер символа (*number*) для нахождения соответствующей пары.

ESC-^F

Команда сходна с **{**, но запрашивает два символа, которые при поиске считает символами **{** и **}**.

ESC-^B

Команда сходна с **}**, но запрашивает два символа, которые при поиске считает символами **{** и **}**.

m Команда запрашивает строчный символ для отметки текущего положения в тексте.

' Команда запрашивает строчный символ и переходит к положению в тексте, отмеченному этим символом. Специальные символы:

- **'** Вернуться к положению, предшествовавшему «длинному переходу».
- **^** Начало файла.
- **$** Конец файла.

^X^X

Идентично **'**.

/pattern

Найти следующее соответствие шаблону *pattern*, начав поиск со второй отображаемой строки. Шаблону могут предшествовать специальные символы:

- **!** Найти строки, не соответствующие шаблону.
- ***** Если текущий файл не содержит соответствия шаблону, продолжить поиск в остальных файлах, перечисленных в командной строке.

@ Производить поиск в первом из перечисленных в командной строке файлов независимо от текущего положения окна с текстом.

?pattern

Обратный поиск, начиная с последней строки предыдущей экранной страницы. Если шаблон начинается со специальных символов !, * или @, они интерпретируются так же, как и командой */pattern*.

ESC-/*pattern*

Идентично **/***.

ESC-?*pattern*

Идентично **?***.

n Повторить поиск по последнему шаблону.

N Повторить поиск по последнему шаблону в обратном направлении.

ESC-n

Повторить предыдущую команду поиска, предварив ее символом «*».

ESC-N

Повторить предыдущую команду поиска в обратном направлении, предварив ее символом «*».

Esc-u

Переключение выделения результатов поиска.

:e [*filename***]**

Открыть файл *filename* и добавить имя файла к списку аргументов командной строки. Если имя файла не указано, перечитать текущий файл. Имя файла *filename* может содержать специальные символы.

% Имя текущего файла.

Имя предыдущего файла.

^X^V, E

Идентично **:e**.

:n Прочитать следующий файл из перечисленных в командной строке.

:p Прочитать предыдущий файл из перечисленных в командной строке.

:x Прочитать первый файл из перечисленных в командной строке.

less

:f, =, ^G

Вывести имя файла, положение в списке аргументов командной строки, номер первой строки экранной страницы, общее количество строк, байтовое смещение последней экранной строки и общее количество байт.

– Задание параметра командной строки и переключение состояния соответствующего режима либо, при необходимости, запрос нового значения.

–+ Задание параметра командной строки и задание значения по умолчанию для этого параметра.

–– Задание параметра командной строки и переключение состояния соответствующего режима на противоположное, причем противоположное состояние может быть задано пользователем.

_ Предваряет параметр командной строки. Отобразить текущее значение параметра.

+*command*

Выполнять команду *command* при каждом открытии нового файла.

q, :q, :Q, ZZ

Завершение работы.

v Команда доступна не во всех версиях программы. Вызов редактора, определяемого переменными $VISUAL или $EDITOR, либо вызов **vi**, если редактор не определен.

! [*command*]

Команда доступна не во всех версиях программы. Вызов $SHELL или **sh**. Если задана команда *command*, выполнить ее и завершить работу с интерпретатором. Специальные символы:

% Имя текущего файла.

Имя предыдущего файла.

!! Предыдущая команда интерпретатора.

| *mark-letter* **command**

Команда доступна не во всех версиях программы. Передать фрагмент файла (от первой строки экрана до символа *mark-letter*) через конвейер команде *command*. Символ *mark-letter* может также принимать значения:

^ Начало файла.

$ Конец файла.

., **newline**

Передача в конвейер текущей экранной страницы.

Приглашения

В строке приглашения специальным образом интерпретируются определенные последовательности. Те, что начинаются с символа «%», всегда вычисляются. Те, что начинаются с символа «**?**», вычисляются в случае выполнения определенных условий. В некоторых приглашениях отображаются положения определенных строк на экране. Такие последовательности требуют указания метода определения номера строки. Подробности см. в описании параметра **–P** и на страницах руководства (manpages) команды.

less

---

ln [*options*] *sourcename* [*destname*]
ln [*options*] *sourcenames destdirectory*

ln

Создание псевдонимов (ссылок) для файлов в целях обеспечения доступа к ним по различным именам. В первом варианте создается ссылка *destname* на файл *sourcename*, где *destname* – обычно имя нового файла или (по умолчанию) текущий каталог. Если файл *destname* существует, он перезаписывается. Если каталог *destname* существует, то в нем создается ссылка *sourcename*. Во втором варианте ссылки создаются в каталоге *destdirectory* и получают такие же имена, как и указанные файлы.

Параметры

–b, ––backup

Создавать резервные копии файлов перед удалением оригиналов.

–d, –F, ––directory

Разрешить создание жестких ссылок на каталоги. Операция доступна только привилегированному пользователю.

–f, ––force

Удалять существующие файлы при создании ссылок без запроса подтверждения.

––help

Вывести информацию по использованию **ln** и завершить работу.

–i, ––interactive

Запрашивать подтверждение на перезапись файлов.

ln	**–n, ––no-dereference** Заменять символические ссылки на каталог новой ссылкой вместо разыменования существующей. Удобно применять совместно с **––force**. **–s, ––symbolic** Создать символическую ссылку. Этот параметр разрешает создание ссылки на объекты другой файловой системы, а также отображение имени файла вместе с именем ссылки по команде **ls –l** (иного способа узнать имя связанного файла не существует). **–S** *suffix*, **––suffix** *suffix* Добавлять суффикс *suffix* к именам файлов при создании резервных копий. Суффикс по умолчанию – «~» (тильда). **–v, ––verbose** Диагностика работы. **––version** Вывести информацию о версии **ln** и завершить работу. **–V, ––version-control** *value* Управление созданием резервных копий. Доступные значения: **t, numbered** Нумерация резервных копий. **nil, existing** Простые резервные копии (~), если не задано создание нумерованных резервных копий. **never, simple** Простые резервные копии.
locate	**locate** [*options*] *pattern* Поиск в базе (базах) имен файлов и отображение соответствий. Символы «*», «?», «[» и «]» обрабатываются специальным образом. Символы «/» и «.» не имеют специального значения. В результат попадают все файлы, соответствующие шаблону *pattern*. В том случае если шаблон содержит специальные символы, **locate** отображает только точные совпадения с шаблоном.

locate

Параметры

–d *path*, **––database=***path*

Искать в базах данных, расположенных в путях *path*, где *path* – список, элементы которого разделяются двоеточиями.

–h, **––help**

Вывести справку по использованию **locate** и завершить работу.

––version

Вывести информацию о версии **locate** и завершить работу.

lockfile

lockfile [*options*] *filenames*

Создать файл(ы)-семафор, используемый для ограничения доступа к определенному файлу. Если **lockfile** не удается создать один из указанных файлов, устанавливается пауза в 8 секунд, после которой повторяется попытка создать файл, на котором произошел сбой. Команда обрабатывает флаги (flags) в порядке их перечисления (флаг, указанный после определенного файла, не будет влиять на доступ к этому файлу).

Параметры

–sleeptime

Длительность паузы перед повторной попыткой создать файл семафора. По умолчанию составляет 8 секунд.

–! Инвертировать код возврата. Полезно в сценариях интерпретатора.

–l *lockout_time*

Интервал времени (в секундах) после изменения файла семафора, по истечении которого этот файл будет принудительно удален. См. также **–s**.

–ml, **–mu**

При соответствующих правах доступа, установленных для системной почтовой буферной зоны (spool), либо при соответствующем идентификаторе смены группы, устанавливаемом на выполнение **lockfile**, возможно блокирование доступа и снятие блокировки к системному почтовому ящику с помощью параметров **–ml** и **–mu**, соответственно.

lockfile	**–r** *retries* Повторять попытки создать файлы не более *retries* раз. По умолчанию *retries* равно –1, т. е. количество попыток не ограничено. При завершении по этому ограничению удаляются все созданные командой файлы. **–s** *suspend_time* После принудительного удаления файла блокировки (см. параметр **–l**) по умолчанию устанавливается пауза в 16 секунд, необходимая для предотвращения случайного мгновенного удаления файлов блокировки, только что созданных другой программой. Параметр используется для установки длительности паузы (*suspend_time*).
logger	**logger** [*options*] [*message*. . .] Команда TCP/IP. Добавление записи в системный журнал (посредством **syslogd**). Сообщение (*message*) может быть передано в командной строке, иначе сообщением является информация, поступающая со стандартного ввода. **Параметры** **–f** *file* Прочитать содержимое сообщения из указанного файла. **–i** Включить идентификатор процесса команды **logger**. **–p** *pri* Указать приоритет сообщения (*pri*). По умолчанию это **user.notice**. **–t** *tag* Пометить все строки сообщения в журнале специальным тегом (*tag*).
login	**login** [*name* | *option*] Вход в систему (регистрация). **login** запрашивает имя пользователя (аргумент *name* может быть указан в командной строке) и пароль (если требуется). При удачной регистрации **login** обновляет файлы учетной записи, устанавливает различные переменные окружения, уведомляет пользователя, есть ли для него почтовые сообщения, и выполняет стартовые файлы интерпретатора. Если существует файл */etc/nologin*, в систему может войти только пользователь root. При завершении соединения этот

login

файл будет отображен. Пользователь root может войти в систему только с терминала, включенного в файл */etc/securetty*. Если существует файл ~/*.hushlogin*, выполняется «тихий» вход. Если существует файл */var/adm/lastlog*, отображается время последней регистрации в системе.

Параметры

–f Запретить повторную попытку входа.

–h *host*

Указать имя удаленного узла. Обычно используется серверами. Параметр доступен только пользователю root.

–p Сохранить окружение предыдущего сеанса работы.

---

logname

logname [*option*]

Отобразить регистрационное имя пользователя из файла */var/run/utmp*. Если имя не найдено, завершить работу с сообщением об ошибке.

Параметры

––help

Отобразить информацию по использованию и завершить работу.

––version

Отобразить информацию о версии **logname** и завершить работу.

---

logrotate

logrotate [*options*] *config_files*

Команда системного администрирования. Обработка файлов журналов в соответствии с командами в файлах *config_files*.

Параметры

–d Режим отладки. Файлы журналов не меняются.

–s, ––state *file*

Сохранить информацию о состоянии в указанном файле. По умолчанию – в файле */var/lib/logrotate.status*.

––usage

Справка по применению и информация о правообладании.

logrotate

Команды

compress

Упаковать старые версии файлов журналов с помощью **gzip**.

copytruncate

Скопировать файл журнала, а затем произвести его усечение. Применяется с программами, «журналирование» которых не может быть временно прервано.

create [*permissions*] [*owner*] [*group*]

После обработки заново создать файл журнала с указанными правами доступа, владельцем и группой. Права доступа задаются в восьмеричном виде. Если какой-либо из этих аргументов отсутствует, вместо него используется соответствующий атрибут оригинального файла.

daily

Производить ротацию файлов журналов ежедневно.

delaycompress

Не сжимать файл журнала до следующей ротации.

errors *address*

Отправлять на указанный адрес почтовые сообщения обо всех ошибках.

endscript

Завершить сценарий **postrotate** или **prerotate**.

ifempty

Выполнять ротацию даже пустых файлов журналов. Имеет больший приоритет, чем параметр **notifempty**, принимаемый по умолчанию.

include *file*

Добавить содержимое указанного файла к содержимому текущего. Если файл является каталогом, добавляется содержимое всех файлов каталога.

mail *address*

Отправлять удаляемые журналы на указанный адрес.

monthly

Выполнять ротацию журналов только при первом запуске **logrotate** в текущем месяце.

nocompress

Запретить действие **compress**.

logrotate

nocopytruncate

Запретить действие **copytruncate**.

nocreate

Запретить действие **create**.

nodelaycompress

Запретить действие **delaycompress**.

noolddir

Запретить действие **olddir**.

notifempty

Запретить действие **ifempty**.

olddir *directory*

Переместить журнал в указанный каталог для обработки. Каталог должен находиться на том же физическом устройстве, что и файлы журналов.

postrotate

Начать сценарий, директивы которого будут выполнены после ротации файлов журналов. Сценарий заканчивается директивой **endscript**.

prerotate

Начать сценарий, директивы которого будут выполнены до обработки файлов журналов. Сценарий заканчивается директивой **endscript**.

rotate *number*

Удалить файл журнала после *number* ротаций.

size *n*[**k**|**M**]

Обрабатывать файл журнала, если его размер превысил *n* байт (либо килобайт или мегабайт в зависимости от модификатора).

look [*options*] *string* [*file*]

look

Поиск строк, начинающихся с подстроки *string*, в файле *file* (по умолчанию – */usr/dict/words*).

Параметры

–a Использовать альтернативный словарь */usr/dict/web2*.

–d Сравнивать только алфавитно-цифровые символы.

–f Нечувствительный к регистрам поиск.

look	**–t** *character* Прекратить сравнение после нахождения в строке указанного символа.
lpc	**lpc** [*command*] Команда системного администрирования. Управление принтером. Команда **lpc** без аргументов принимает команды со стандартного ввода. **Команды** **?, help** [*commands*] Перечисление команд либо получение справки по конкретной команде. **abort all** | *printer* Принудительно завершить работу текущего демона печати и отключить печать для указанного принтера. **clean all** | *printer* Удалить файлы, которые не могут быть напечатаны из указанных очередей печати. **disable all** | *printer* Отключить указанные очереди печати. **down all** | *printer message* Отключить очереди печати для определенного принтера и поместить сообщение *message* в файл состояния принтера. **enable all** | *printer* Включить указанные очереди печати. **exit, quit** Завершить работу **lpc**. **restart all** | *printer* Попытаться повторно запустить демоны печати для указанных принтеров. **start all** | *printer* Включить очереди принтера и запустить демоны печати для указанных принтеров. **status all** | *printer* Вернуть состояние указанных принтеров.

lpc

stop all | *printer*

Прекратить работу указанных демонов печати после завершения текущего задания печати.

topq printer [*jobnumbers*] [*users*]

Поместить указанные задания в начало очереди печати в порядке их перечисления.

up all | *printer*

Включить очереди печати и перезапустить демоны для указанных принтеров.

lpd

lpd [*option*] [*port*]

Команда TCP/IP. Демон построчной печати. **lpd** обычно запускается при загрузке системы из файла *rc2*. Демон единожды читает файл настройки принтеров (обычно */etc/printcap*) в поиске существующих в системе печатающих устройств, а затем печатает все задания, оставшиеся в очереди после сбоя. **lpd** принимает запросы на печать файлов из очереди, передачу файлов в буферную зону печати, отображение состояния очереди, удаление заданий из очереди. Для реакции на любой запрос **lpd** выполняет порожденный fork-процесс, продолжая ожидание дальнейших запросов. Если указан порт (*port*), происходит прием запросов через порт, в ином случае для получения корректного порта вызывается функция **getservbyname**.

Файл *lock* в каждом из каталогов буферной зоны предотвращает одновременное выполнение нескольких активных демонов печати. После того как демон заблокировал каталог, он сканирует его в поиске файлов, имена которых начинаются с *cf*. Строки таких файлов содержат имена документов, которые необходимо напечатать, и действия (не относящиеся к печати), которые следует выполнить. Каждая строка начинается с ключевого символа, содержащего информацию о строке или о том, что делать с ее содержимым. Ниже приведен перечень ключевых символов:

C Классификация – строка, используемая на титульной странице в качестве строки классификации.

c Файл **cifplot**.

f Форматированный файл – имя уже форматированного файла для печати.

g Графический файл.

H Имя узла – машина, на которой запущен **lpd**.

lpd

J Имя задания – строка, используемая в качестве имени задания на титульной странице.

L Литерал – эта строка содержит идентификационную информацию из файла паролей и вызывает печать титульного листа.

l Форматированный файл; при печати удалять разрывы страниц и управляющие символы.

M Почтовое сообщение – отправить почтовое сообщение пользователю, когда это задание будет завершено.

n Файл **ditroff**.

P Пользователь – регистрационное имя пользователя, выполнившего **lpd**.

r Файл DVI.

T Название – строка названия для **pr**.

t Файл **troff**.

U Удалить – имя файла, удаляемого по завершении задания.

Параметр

–l Ведение журнала всех корректных запросов.

Файлы

/etc/printcap

Файл описания принтера.

*/var/spool/**

Каталоги буферной зоны.

/var/spool//minfree*

Минимально допустимое свободное пространство.

*/dev/lp**

Печатающие устройства.

/etc/hosts.equiv

Имена узлов, у которых есть доступ к печати.

/etc/hosts.lpd

Имена узлов, у которых есть доступ к печати, но подчиняющихся другой администрации.

lpq [*options*] [*name*] **lpq**

Проверка состояния заданий в очереди буферной зоны печати. Для каждого задания отображается имя владельца, ранг в очереди, имена файлов, номер задания и общий размер файла (в байтах). Если задано имя (*name*), отобразить информацию только для заданий этого пользователя.

Параметры

–l Вывести информацию по каждому из файлов, входящих в задание.

–P*printer*

Определить печатающее устройство. Если этот параметр опущен, **lpq** использует печатающее устройство по умолчанию или то, которое задано переменной окружения PRINTER.

num

Определить состояние задания с указанным номером *num*.

lpr [*options*] *files* **lpr**

Поместить файлы (*files*) в очередь буферной зоны печати.

Параметры

–c Считать, что данные в файлах являются результатом работы **cifplot**.

–d Считать, что данные являются результатом работы TEX и представлены в формате DVI (формат, независимый от устройства вывода).

–f Использовать фильтр, интерпретирующий первый символ каждой строки как стандартный символ управления кареткой.

–g Считать, что данные являются чертежом (например, созданным с помощью **plot**).

–l Использовать фильтр, позволяющий выводить управляющие символы и подавлять разрывы страниц.

–n Считать, что данные являются результатом работы **ditroff** (независимый от устройства **troff**).

–p Использовать **pr** для форматирования файлов.

lpr

–t Считать, что данные являются результатом работы **troff** (система команд форматирования текстов перед выводом).

–v Считать, что данные являются растровым изображением, предназначенным для вывода на устройство типа Benson Varian.

–P*printer*

Вывод на *printer* вместо вывода на принтер, определенный по умолчанию или определенный переменной окружения PRINTER.

–h Не выводить на печать титульную страницу.

–m По завершении задания послать владельцу почтовое сообщение.

–r Удалить файл после помещения в очередь буферной зоны или по завершении задания (нельзя использовать с параметром **–s**).

–s Создавать символические ссылки вместо копирования самих файлов в буферную зону. Это экономит время и дисковое пространство при печати больших файлов. Файлы нельзя изменять или удалять до завершения печати.

–#*num*

Печатать *num* копий указанных файлов.

–C *string*

Заменить имя системы на титульной странице строкой *string*.

–J *name*

Заменить имя печатаемого файла на титульной странице на строку *name*.

–T *title*

Использовать название *title* при разбивке вывода посредством **pr**.

–i [*cols*]

Отступ при печати, задаваемый параметром *cols*. По умолчанию устанавливается равным 8 колонкам.

–w *num*

Установить ширину страницы в *num* символов при разбивке **pr**.

lprm [*options*] [*jobnum*] [*user*]

lprm

Удалить указанные (по номерам) задания из очереди буферной зоны печати. Номера заданий можно узнать с помощью команды **lpq**. Привилегированный пользователь может задать имя *user* для удаления всех заданий, принадлежащих этому пользователю(ям).

Параметры

–P*printer*

Задать имя печатающего устройства (*printer*). Обычно это печатающее устройство по умолчанию либо устройство, определяемое переменной окружения PRINTER.

– Удалить из очереди все задания, принадлежащие пользователю *user*.

lpstat [*options*]

lpstat

Отобразить состояние очереди печати. Если в параметре, принимающем аргумент *list* (список), данный аргумент не указан, то выводится полный объем соответствующей информации. Элементы списка разделяются запятыми либо пробелами, если список заключен в двойные кавычки.

Параметры

–a [*list*]

Показать, принимают ли запросы печатающие устройства или классы из списка.

–c [*list*]

Отобразить информацию о классах печатающих устройств из списка.

–d Показать адрес буферной зоны принтера по умолчанию.

–f [*list*]

Проверить, что **lp** известны формы из списка.

–l Указывается после **–f** для получения описаний доступных форм; после **–p** – для отображения конфигурации принтеров; после **–s** – для описания принтеров, соответствующих указанной таблице символов либо печатающему механизму.

lpstat	**–o** [*list*] Показать состояние запросов вывода. Список содержит имена принтеров, имена классов или идентификаторы запросов. **–p** [*list*] Отобразить состояние принтеров, перечисленных в списке. **–r** Показать, включен ли планировщик задач принтера. **–R** Показать положение задания в очереди печати. **–s** Сводка состояния принтера (отображаются почти все сведения). **–t** Показать всю информацию о состоянии (полные сведения). **–u** [*list*] Отобразить состояние запросов пользователей из списка. Список также может заменяться ключевым словом **all**; в этом случае отображается информация для всех пользователей. **–v** [*list*] Показать имя устройства, связанного с каждым из принтеров в списке.
lptest	**lptest** [*length*] [*count*] Выдача текста для тестирования принтера на стандартный вывод. Выдается стандартная последовательность («волна») всех отображаемых символов набора ASCII со сдвигом на один символ в каждой последующей строке. **Параметры** *length* Длина строки вывода (по умолчанию – 79). *count* Количество отображаемых строк (по умолчанию – 200).
ls	**ls** [*options*] [*names*] Вывод содержимого каталогов. Если не заданы имена (*names*), перечисляются файлы из текущего каталога. Если заданы имена, отображается содержимое указанных каталогов или перечисляются файлы, соответствующие маске имени

name. Имена (*names*) могут содержать специальные символы маски. Параметры команды позволяют отображать разнообразную информацию в различном виде. Самые полезные параметры: **–F**, **–R**, **–l** и **–s**. Некоторые из параметров не имеет смысла использовать совместно (например, **–u** и **–c**).

Параметры

–l, ––format=single-column

Вывести информацию по каждому файлу отдельной строкой.

–a Отобразить все файлы, включая и обычно скрытые файлы, имена которых начинаются с точки.

–b, ––escape

Отображать непечатаемые символы в восьмеричном и алфавитном формате.

–c, ––time-ctime, ––time=status

Перечислить файлы по времени изменения состояния (а не по времени создания/изменения).

––color, ––colour, ––color=yes, ––colour=yes

Отобразить имена файлов различным цветом в зависимости от типов файлов.

––color=no, ––colour=no

Отключить отображение разными цветами. Это режим по умолчанию. Отменяет предыдущий режим включения цветов.

––color=tty, ––colour=tty

Аналогично **––color**, но только для тех случаев, когда стандартным выводом является терминал. Параметр весьма полезен, если **ls** применяется в сценариях интерпретатора и при создании псевдонимов команд, в особенности, если ваше любимое средство постраничного просмотра не поддерживает режим цветного вывода.

–d, ––directory

Отчет только о каталогах, но не о содержимом.

–f Вывести информацию о файлах в каталогах в том же порядке, в каком хранятся файлы, не пытаясь их сортировать.

––full-time

Отображать время в полном формате вместо стандартного (сокращенного).

ls

--help

Вывести справку и завершить работу.

-i, --inode

Отобразить информацию по единицам inode для каждого файла.

-k, --kilobytes

Если отображаются размеры файлов, выводить их в килобайтах. Этот параметр подавляет использование переменной окружения POSIXLY_CORRECT.

-l, --format=long, --format=verbose

Подробный формат описания файла (включает права доступа, информацию о владельце, размер, время изменения и т. д.)

-m, --format=commas

Представить список в виде перечисления имен файлов через запятую.

-n, --numeric-uid-gid

Аналогично **-l**, но вместо имени владельца или группы указывается соответствующий идентификатор.

-p Отметить каталоги, добавляя символ «/» к именам.

-q, --hide-control-chars

Неотображаемые символы заменить на «**?**» (вопросительный знак).

-r, --reverse

Перечислить файлы в обратном порядке (по имени или времени).

-s, --size

Вывести размеры файлов в блоках.

-t, --sort=time

Сортировать файлы по времени изменения (более новые файлы перечисляются первыми).

-u, --time=atime, --time=access, --time=use

Сортировать файлы по времени доступа.

--version

Вывести информацию о версии **ls** на стандартный вывод и завершить работу.

-x, --format=across, --format=horizontal

Перечислить файлы, заполняя последовательно строки.

–A, ––almost-all

Перечислить все файлы, включая и обычно скрытые файлы, имена которых начинаются с точки. При этом каталоги . и .. не отображаются.

–B, ––ignore-backups

Не включать в список файлы, имена которых заканчиваются символом «~», если эти файлы не заданы явно в аргументах.

–C, ––format=vertical

Перечислять файлы в колонках (режим по умолчанию).

–F, ––classify

Отображать типы файлов, отмечая символом «/» каталоги, символом «*» – исполняемые файлы, символом «@» – символические ссылки, символом «|» – файлы FIFO, а символом «=» – сокеты (sockets).

–G, ––no-group

В длинном формате вывода не отображать имя владеющей группы.

–I, ––ignore *pattern*

Не перечислять файлы, имена которых соответствуют маске *pattern*, если обратное не указано явно в командной строке.

–L, ––dereference

Включать в список файл (или каталог), на который ссылается символическая ссылка, а не собственно ссылку.

–N, ––literal

Не заключать имена файлов в кавычки.

–Q, ––quote-name

Заключать имена файлов в двойные кавычки ("); отображать непечатаемые символы алфавитными последовательностями и восьмеричными соответствиями, экранированными символом обратного слэша.

–R, ––recursive

Рекурсивно отображать содержимое подкаталогов, а также указанного (или текущего) каталога.

–S, ––sort=size

Сортировка файлов по размеру – от большего к меньшему.

ls

–T, ––tabsize *n_cols*

Установить размер табуляции в *n_cols* колонок. По умолчанию – 8.

–U, ––sort=none

Не сортировать список файлов. Аналогично **–f**, но отображение в длинном формате.

–X, ––sort=extension

Сортировка файлов по расширениям.

Примеры

Перечислить все файлы в текущем каталоге и отобразить их размеры; использовать многоколоночный вывод и отмечать специальные файлы:

```
ls -asCF
```

Отобразить состояние каталогов */bin* и */etc*:

```
ls -ld /bin /etc
```

Перечислить файлы исходных текстов на языке C в текущем каталоге, начиная с самого старого:

```
ls -rt *.c
```

Сосчитать файлы в текущем каталоге:

```
ls | wc -l
```

lsattr

lsattr [*options*] [*files*]

Отобразить атрибуты файлов файловой системы Linux Second Extended Filesystem (ext2). См. также **chattr**.

Параметры

–a Перечислить все файлы в указанных каталогах.

–d Перечислить атрибуты самих каталогов, а не их содержимого.

–R Рекурсивно обрабатывать каталоги и их содержимое.

–v Перечислить версии файлов.

–V Вывести информацию о версии и завершить работу.

lsmod **lsmod**

Команда системного администрирования. Перечисление загруженных модулей: их имен, размеров (в блоках по 4 Кбайт) и списка связанных модулей, если таковые имеются.

Файл

/proc/modules

Источник той же информации.

---

m4 [*options*] [*macros*] [*files*] **m4**

Макропроцессор для программ на языке C и других файлов.

Параметры

–e, ––interactive

Работа в диалоговом режиме, прерывания игнорируются.

–d*flags*, **––debug=***flags*

Указание уровня отладки.

–l*n*, **––arglength=***n*

Задание длины результатов отладки.

–o *file*, **––error-output=***file*

Копировать вывод в указанный файл. Независимо от имени отображать сообщения об ошибках в стандартный поток ошибок.

–p, ––prefix-built-ins

Предварять префиксом **m4_** все имена встроенных макросов.

–s, ––synclines

Вставлять директивы **#line** для препроцессора C.

–B*n*

Установить размер очереди и буферов накопления аргументов в *n* (по умолчанию – 4096).

–D*name*[=*value*], **––define=***name*[=*value*]

Определить имя *name* со значением *value* или с пустым значением, если *value* опущено.

–E, ––fatal-warnings

Считать все предупреждения неисправимыми ошибками; завершить работу по первому же предупреждению.

m4	**–F***file*, **––freeze-state** *file* Записать состояние **m4** в указанный файл для последующего восстановления. **–G, ––traditional** Игнорировать расширения GNU. **–H***n*, **––hashsize=***n* Установить размер ассоциативного массива таблицы имен в *n* (по умолчанию – 509). **–I***directory*, **––include=***directory* Выполнять поиск включаемых файлов в указанном каталоге. **–Q, ––quiet, ––silent** Подавить вывод предупреждений. **–R***file*, **––reload-state** *file* Загрузить состояние **m4** из файла, прежде чем начать работу. **–U***name*, **––undefine=***name* Сделать имя *name* неопределенным.

mail	**mail** [*options*] [*users*]
	Чтение или отправка почтовых сообщений другим пользователям (*users*). Утилита **mail** позволяет создавать, отправлять, получать, пересылать почтовые сообщения и отвечать на них. Команда **mail** имеет два основных режима работы: режим создания сообщений и командный режим, в котором происходит управление почтовыми сообщениями.
	При том что **mail** является мощным инструментом, применение этой программы может представлять определенные сложности для начинающих пользователей. Большинство дистрибутивов Linux содержат программы **pine** и **elm**, которые гораздо проще в использовании.
	В этом разделе представлены команды, параметры и файлы **mail**. Сначала приведем две самые необходимые команды.
	Для перехода в диалоговый режим чтения почтовых сообщений выполните:
	`mail`
	Чтобы начать написание нового сообщения для адресата *user*, выполните:
	`mail user`

Введите текст сообщения построчно, нажимая клавишу <Enter> в конце каждой строки. Завершите создание сообщения вводом символа «.» (точка) в первой позиции пустой строки.

Параметры командной строки

–b *list*

Установить значение поля *Bcc:* для сообщений. Элементы списка разделяются запятыми.

–c *list*

Установить значение поля *Cc:* для сообщений. Элементы списка разделяются запятыми.

–d Вывести отладочную информацию.

–f [*file*]

Обработать содержимое файла *file* либо *~/mbox* (если имя файла не указано) вместо */var/spool/mail/$user*.

–i Не реагировать на сигналы прерывания от терминала.

–n Не обращаться к */etc/mail.rc* в начале работы.

–p Чтение почты в POP-режиме.

–s *subject*

Установить тему сообщения (*subject*).

–u Обрабатывать содержимое */var/spool/mail/$user*. Режим по умолчанию.

–v Подробная диагностика. Отображать информацию о доставке сообщения на стандартный вывод.

–I Диалоговый режим; действует, даже если стандартный ввод был перенаправлен с терминала.

–N При печати почтового сообщения или переходе в почтовую папку не отображать заголовки сообщений.

–P Отключить режим POP.

Команды режима создания сообщений

~! Выполнить команду интерпретатора.

~? Перечислить команды режима.

~b *names*

Добавить или удалить имена в поле *Bcc*: заголовка.

~c *names*

Добавить или удалить имена в поле *Cc*: заголовка.

mail

~d Читать файл *dead.letter*.

~e Вызвать текстовый редактор.

~f *messages*

Вставить содержимое сообщений *messages* в текущее создаваемое сообщение.

~F *messages*

Аналогично **~f**, но включаются заголовки сообщений.

~h Добавить или изменить заголовки в диалоговом режиме.

~m *messages*

Аналогично **~f**, но преобразовывать отступ в символ табуляции.

~M *messages*

Аналогично **~m**, но включаются заголовки сообщений.

~p Вывести поля заголовка и само отправляемое сообщение.

~q Прервать написание текущего сообщения.

~r *filename*

Добавить к текущему сообщению содержимое файла *filename*.

~s *string*

Изменить поле заголовка *Subject*: на *string*.

~t *names*

Добавить или редактировать имена в списке **To:**

~v Вызвать редактор, определяемый переменной окружения VISUAL.

~| *command*

Передать сообщение команде *command* через конвейер (pipe).

~: *mail-command*

Выполнить почтовую команду.

~~*string*

Вставить строку *string* в текст сообщения, предварив ее символом «~» (тильда). Символ «~» внутри строки должен быть экранирован символом «\».

Командный режим **mail**

?	Перечислить доступные команды (справка).
!	Выполнить команду интерпретатора.
– *num*	Вывести сообщение, отстоящее от текущего на *num*. По умолчанию отображается предыдущее сообщение.
alias (a)	Вывести или создать списки псевдонимов.
alternates (alt)	Указать дополнительные регистрационные почтовые записи на удаленных машинах, принадлежащие текущему пользователю, чтобы почтовый клиент не отвечал на письма с этих адресов.
chdir (c)	Перейти в домашний каталог или в указанный.
copy (co)	Аналогично **save**, но сообщение не помечается для удаления.
delete (d)	Удалить сообщение.
dp	Удалить текущее сообщение и отобразить следующее.
edit (e)	Редактировать сообщение.
exit (ex, x)	Завершение работы **mail** без обновления папки.
file (fi)	Смена папки.
folder (fold)	Прочитать сообщения, сохраненные в определенном файле. Файл может быть: # предыдущий % системный почтовый ящик % *user* системный почтовый ящик пользователя (*user*) & *mbox* +*folder* файл в каталоге *folder*
folders	Перечислить папки.
headers (h)	Перечислить заголовки сообщений в текущем приглашении.
headers+ (h+)	К следующей экранной странице заголовков.
headers– (h–)	К предыдущей экранной странице заголовков.
help	Идентично «**?** ».

mail	**hold (ho)**	Сохранить сообщения в системном почтовом ящике.
	ignore	Добавить список полей к игнорируемым.
	mail *user* **(m)**	Создать сообщение, адресованное пользователю *user*.
	mbox	Режим по умолчанию. Указанные сообщения помещаются в *mbox* при завершении работы.
	next (n)	Набрать следующее сообщение или сообщение, соответствующее аргументу.
	preserve (pr)	То же, что **hold**.
	print [*list*] **(p)**	Отобразить все сообщения из списка (*list*).
	Print [*list*] **(P)**	Аналогично **print**, но отображать также поля заголовков.
	quit (q)	Завершить работу с обновлением папки.
	reply (r)	Послать сообщения каждому адресату из списка рассылки.
	Reply (R)	Ответить только автору сообщения.
	respond	Идентично **reply**.
	retain	При выводе сообщения всегда включать отображение указанных полей заголовка. По команде без аргументов перечисляются заданные поля.
	save (s)	Сохранить сообщение в папке.
	saveignore	Удалять игнорируемые поля при сохранении.
	saveretain	Сохранять указанные поля даже при сохранении через **saveignore**.
	set (se)	Изменить или вывести настройки **mail**.
	shell (sh)	Вызвать новый интерпретатор.
	size	Вывести размеры указанных сообщений.
	source	Выполнить команды из указанного файла.
	top	Вывести несколько первых строк каждого из указанных сообщений.
	type (t)	Идентично **print**.
	Type (T)	Идентично **Print**.
	unalias	Удалить определенные ранее псевдонимы.

undelete (u)	Восстановить удаленные сообщения.
unread (U)	Пометить указанные сообщения как непрочтенные.
unset (uns)	Сбросить параметры **mail**.
visual (v)	Редактировать сообщение в редакторе, определенном переменной окружения VISUAL.
write (w)	Записать сообщение (без заголовка) в файл.
xit (x)	Идентично **exit**.
z	Перейти к следующей экранной странице текста. Переход к предыдущей странице по команде **z–**.

Почтовые параметры

Эти параметры используются в файле *.mailrc*. Действие параметра может быть установлено (**set** *параметр*) или снято (**unset** *параметр*).

append	Добавлять сообщения в конец (а не в начало) файла *mbox*.
ask	Запрашивать тему сообщения.
askbcc	Запрашивать адресатов, которым будет отправлена «скрытая» копия сообщения.
askcc	Запрашивать адресатов, которым будет отправлена обычная копия сообщения.
asksub	Запрашивать значение поля **Subject**:
autoprint	Выводить следующее сообщение после удаления текущего.
chron	Выводить сообщения в хронологическом порядке, последним отображается самое новое сообщение.
debug	Идентично параметру командной строки **–d**.
dot	Интерпретировать символ точки в отдельной строке как EOF.
folder	Определить каталог для хранения почтовых папок.
hold	Оставлять сообщение в системном почтовом ящике при завершении работы.

make	**ignore**	Игнорировать сигналы прерывания, поступающие с терминала. Выводить их как символ @.
	ignoreeof	Не считать символ ^**D** символом EOF.
	metoo	Не удалять отправителя из списка рассылки.
	noheader	Идентично параметру командной строки **–N**.
	nokerberos	Получение почты POP по протоколу POP3, а не KPOP.
	nosave	Не сохранять незавершенные сообщения в файле *dead.letter*.
	pop-mail	Получение почты по протоколу POP3 и сохранение ее в файле *mbox.pop*.
	prompt	Сменить строку приглашения.
	Replyall	Обмен функций **Reply** и **reply**.
	quiet	Не выводить информацию о версии в начале работы.
	searchheaders	При задании спецификатора вида /*x*:*y* отображать все сообщения, содержащие строку *y* в поле заголовка *x*.
	verbose	Идентично параметру командной строки **–v**.
	verbose-pop	Отображать состояние при получении почты POP.

Специальные файлы

calendar	Содержит даты, уведомление о которых отправляется пользователю системой.
.maildelivery	Файл настройки доставки почтовых сообщений.
.mailrc	Файл настроек работы с почтой.
triplog	Список адресатов, получающих автоматические уведомления.
tripnote	Содержание сообщения автоуведомления.

mailq

mailq [*option*]

Команда системного администрирования. Перечисление всех сообщений из почтовой очереди **sendmail**. Эквивалентно **sendmail –bp**.

mailq

Параметр

–v Диагностика работы.

mailstats

mailstats [*options*]

Команда системного администрирования. Отображение форматированной статистики по работе **sendmail**.

Параметры

–C *file*

Использовать указанный файл настройки **sendmail** вместо стандартного *sendmail.cf* (по умолчанию).

–f *file*

Использовать указанный файл статистики **sendmail** вместо файла, определенного в файле настройки **sendmail**.

–o Не отображать имя почтовой программы.

make

make [*options*] [*targets*] [*macro definitions*]

Обновление одной или более целей (*targets*) в соответствии с описаниями зависимостей в специальном файле текущего каталога. По умолчанию этот файл называется *makefile* или *Makefile*. Параметры, цели и макроопределения могут следовать в любом порядке. Макроопределения задаются следующим образом:

```
name=string
```

Более подробную информацию по **make** можно почерпнуть в книге «Managing Projects with make» (Управление проектами и make) Эндрю Орама (Andrew Oram) и Стива Тэлбота (Steve Talbott).

Параметры

–d, ––debug

Вывести подробную отладочную информацию.

–e, ––environment-overrides

Игнорировать макроопределения файла **makefile** в пользу переменных окружения.

–f *makefile*, **––file** *makefile*, **––makefile** *makefile*

Указать имя файла описаний (*makefile*). Имя файла «–» (дефис) означает стандартный ввод.

make

–h, ––help

Вывести информацию о параметрах команды **make**.

–i, ––ignore-errors

Игнорировать ошибочное выполнение команд (аналогично **.IGNORE**)

–j [*jobs*], **––jobs** [*jobs*]

Пытаться выполнить *jobs* заданий одновременно либо, если не задано количество задач, как можно большее их количество.

–k, ––keep-going

Прекратить сборку текущей цели при возникновении ошибки, но продолжить работу с прочими целями, не зависящими от данной.

–l [*load*], **––load-average** [=*load*], **––max-load** [=*load*]

Держаться в пределах загрузки *load* (которая выражается действительным числом). Применяется совместно с **–j**.

–n, ––just-print, ––dry-run, ––recon

Вывести команды, но не выполнять их (проверочное выполнение).

–o *file*, **––old-file=***file*, **––assume-old=***file*

Никогда не пересобирать файл *file* либо пересобирать все остальные файлы, если файл пересобран.

–p, ––print-data-base

Вывести правила и переменные в ходе нормального выполнения.

–q, ––question

Проверка редакции файла; вернуть 0, если файл не следует обновлять, в противном случае – не нуль.

–r, ––no-built-in-rules

Не использовать правила по умолчанию.

–s, ––silent, ––quiet

Не отображать строки команд (то же, что и **.SILENT**).

–t, ––touch

Создать пустые целевые файлы, не пересобирая их.

–v, ––version

Информация о номере версии **make**.

–w, ––print-directory

Отображать имя текущего каталога до и после сборки.

––warn-undefined-variables

Вывести предупреждение, если макрос используется без предварительного определения.

–C *directory*, **––directory** *directory*

Перейти **(cd)** в каталог *directory* перед началом работы **make**. По следующим директивам **–C** будет совершаться переход **(cd)** в соответствующий каталог относительно текущего.

–I *directory*, **––include-dir** *directory*

Включить указанный каталог в список каталогов, содержащих включаемые файлы.

–S, ––no-keep-going, ––stop

Отменить действие предшествующего параметра **–k**. Применяется при рекурсивных сборках **make**.

–W *file*, **––what-if** *file*, **––new-file** *file*, **––assume-new** *file*

Считать файл *file* недавно обновленным.

Строки файла описаний

Инструкции в файле описаний интерпретируются построчно. Если размер инструкции превышает длину строки, укажите обратный слэш (\) в конце строки для продолжения инструкции на следующей строке. Файл описаний может содержать следующие типы строк:

Пустые строки (*blank lines*)

Пустые строки игнорируются.

Комментарии (*comment lines*)

Все символы в строке после символа «#» считаются комментарием и игнорируются.

Описания зависимостей (*dependency lines*)

В зависимости от выбранной цели выполняются соответствующие команды. Варианты записи:

```
targets : dependencies
targets : dependencies ; command
```

Команды выполняются в том случае, когда файлы зависимостей (*dependencies*), имена которых могут являться масками, не существуют или являются более новыми, чем цель сборки. Если предварительные условия отсутствуют,

make

команды выполняются обязательно (при задании любой из целей). Табуляция перед *targets* недопустима.

Правила суффиксов (*suffix rules*)

Правила суффиксов определяют соответствия между файлами предварительных условий и файлами с такими же именами, но отличающимися суффиксами. Варианты записи правил:

```
.suffix.suffix:
.suffix:
```

Во втором варианте основное имя файла определяется именем файла с соответствующим суффиксом.

Команды (*commands*)

Команды группируются после зависимостей и вводятся на строках, начинающихся с табуляции. Если команде предшествует дефис (–), **make** игнорирует ошибочное выполнение команды. Если команде предшествует символ (@), командная строка не отображается при выполнении (если только **make** не выполняется с параметром **–n**).

Макроопределения (*macro definitions*)

Макроопределения имеют следующий вид:

```
name = string
```

или

```
define name
string
endef
```

Пробелы перед символом «=» и после него могут быть опущены.

Предписания включений (*include statements*)

Аналогичны директиве **#include** языка C:

```
include files
```

Внутренние макросы

$? Список предварительных условий, изменившихся позже, чем текущая цель сборки. Может применяться только в обычных строках файла описаний, но не в правилах суффиксов.

$@ Имя текущей цели, за исключением строк файла описаний, относящихся к сборке библиотек, где это будет имя библиотеки. Может присутствовать как в обычных строках файла описаний, так и в правилах суффиксов.

$< Имя текущего предварительного условия, которое изменялось позже текущей цели сборки.

$* Имя (без суффикса) текущего предварительного условия, которое изменялось позже текущей цели сборки. Может использоваться только в правилах суффиксов.

$% Имя соответствующего файла (**.o**), если текущая цель является библиотечным модулем. Может присутствовать как в обычных строках файла описаний, так и в правилах суффиксов.

$^ Список всех зависимостей (без повторений), элементы которого разделяются пробелами.

$+ Список всех зависимостей (с повторениями), элементы которого разделяются пробелами.

Правила шаблонов

Правила шаблонов являются обобщением идеи правил суффиксов. Если и цель и зависимость содержат символ %, GNU-версия **make** подставляет любую часть имени существующего файла. Например, стандартное правило суффиксов:

```
$(cc) -o $@ $<
```

может быть записано следующим правилом шаблонов:

```
%.o : %.c
$(cc) -o $@ $<
```

Модификаторы макросов

D Имя каталога из любого имени внутреннего макроса кроме **$?**. Корректно следующее использование:

```
$(*D) $$(@D) $(?D) $(<D)
$(%D) $(@D) $(^D)
```

F Имя файла (без пути) из любого имени внутреннего макроса кроме **$?**. Корректны следующие записи:

```
$(*F) $$(@F) $(?F) $(<F)
$(%F) $(@F) $(^F)
```

Функции

$(subst *from*, *to*, *string***)**

Заменить все вхождения *from* на *to* в строке *string*.

make

$(patsubst *pattern*, *to*, *string***)**

Аналогично **subst**, но считать символ % внутри шаблона *pattern* метасимволом. Заменяет каждое соответствие шаблону в строке *string* на *to*.

$(strip *string***)**

Удалить все лишние пробелы.

$(findstring *substring*, *mainstring***)**

Вернуть подстроку *substring*, если она найдена в строке *mainstring*, иначе вернуть нуль.

$(filter *pattern*, *string***)**

Вернуть те слова из строки *string*, которые соответствуют хотя бы одному слову шаблона *pattern*. Шаблоны могут включать символ маски %.

$(filter-out *pattern*, *string***)**

Удалить из строки *string* слова, которые соответствуют хотя бы одному слову шаблона *pattern*. Шаблоны могут включать символ маски %.

$(sort *list***)**

Вернуть список *list*, отсортированный в лексикографическом порядке.

$(dir *list***)**

Вернуть имена каталогов (все символы до последнего слэша) всех имен файлов из списка *list*.

$(notdir *list***)**

Вернуть все имена файлов (символы после последнего слэша) из списка *list*.

$(suffix *list***)**

Вернуть суффиксы имен (символы после последней точки) из списка *list*.

$(basename *list***)**

Для каждого элемента списка (имен файлов) вернуть подстроку до суффикса (все символы до последней точки).

$(addsuffix *suffix*, *list***)**

Вернуть список имен файлов, добавив к каждому имени суффикс.

$(addprefix *prefix*, *list***)**

Вернуть список имен файлов, добавив к каждому имени префикс.

$(join *list1*, *list2***)**

Вернуть список, полученный векторной конкатенацией аргументов. Например, **$(join a b, .c .o)** вернет **a.c b.o**.

$(word *n*, *string***)**

Вернуть *n*-ное слово строки *string*.

$(words *string***)**

Вернуть количество слов в строке.

$(firstword *list***)**

Вернуть первое слово из списка *list*.

$(wildcard *pattern***)**

Вернуть список файлов из текущего каталога, соответствующих маске *pattern*.

$(origin *variable***)**

Вернуть одну из следующих строк, описывающих, как была определена переменная *variable*: **undefined**, **default**, **environment**, **environment override**, **file**, **command line**, **override** или **automatic**.

$(shell *command***)**

Вернуть результат выполнения команды. Символы новой строки преобразуются в пробелы. Функция работает аналогично подстановке по обратным кавычкам в большинстве командных интерпретаторов.

Подстановка строк в макросах

$(*macro:s1=s2***)**

Вычислить подстановку по текущему определению макроса $(*macro*), предварительно заменив строкой *s2* каждое вхождение строки *s1*, непосредственно следующее перед пробелом или символом табуляции либо находящееся в конце строки макроопределения.

Специальные имена целей

.DEFAULT:

Команды, ассоциированные с этой целью, выполняются, если **make** не может найти строки описаний в соответствующем файле или правила суффиксов, с помощью которых может быть собрана запрошенная цель.

.EXPORT_ALL_VARIABLES:

Если существует эта цель, экспортировать все макроопределения во все порожденные процессы.

make	**.IGNORE:** Игнорировать ошибки. Действует идентично параметру **–i**. **.PHONY:** Всегда выполнять команды, следующие за целью, даже если цель существует и не требует пересборки. **.PRECIOUS:** Файлы, определяемые для этой цели, не удаляются при получении **make** сигнала принудительного завершения или когда строка команд в файле описаний возвращает ошибку. **.SILENT:** Выполнить команды, не отображая их. Действует идентично параметру **–s**. **.SUFFIXES:** Суффиксы, ассоциированные с этой целью, имеют значение в правилах суффиксов. Если суффиксы не указаны, существующий список правил суффиксов «отключается».
makedbm	**makedbm** [*options*] *infile outfile* Команда NFS/NIS. Создание **dbm**-файла NIS. **makedbm** преобразует файл *infile* в пару файлов в формате **ndbm**, а именно: *outfile.pag* и *outfile.dir*. Каждая из строк исходного файла преобразуется в одну запись **dbm**. Все символы до первой табуляции или символа пробела составляют ключ, а остаток строки – данные. Если строка заканчивается сочетанием \&, продолжение данных этого ключа читается из следующей строки. **makedbm** не считает символ # началом комментария, и его интерпретация предоставляется клиентам NIS. Если в качестве имени *infile* указан символ «–» (дефис), то выполняется чтение со стандартного ввода. **makedbm** создает специальную запись с ключом **yp**_*last_modified* из даты создания файла *infile* (или текущей даты при чтении со стандартного ввода). **Параметры** **–b** Распределить карту на все серверы, используя междоменный сервер имен **named**. **–d yp**_*domain_name* Создать специальную запись с ключом **yp**_*domain_name*.

makedbm

–i yp_*input_file*

Создать специальную запись с ключом **yp**_*input_file*.

–l Преобразовать ключи указанной карты в нижний регистр.

–m yp_*master_name*

Создать специальную запись с ключом **yp**_*master_name*. Если не задано имя узла-хозяина, установить ключ в локальное имя узла.

–o yp_*output_file*

Создать специальную запись с ключом **yp**_*output_file*.

–s Безопасная карта. Устанавливать соединения только с авторизованными NIS-сетями.

–u dbm *filename*

Обратить **dbm**-файл: распечатать построчно базу данных, отделяя ключи от значений пробелом.

Пример

Несложно создать сценарий интерпретатора для преобразования стандартных файлов (например, */etc/passwd*) в формат ключ-значение, используемый **makedbm**. Например, следующая программа на **awk**:

```
BEGIN { FS=":"; OFS = "\t";}
{ print $1, $0}
```

преобразует */etc/passwd* в исходный файл для **makedbm**, после чего можно создать NIS-файл *passwd.byname*. То есть ключом является имя пользователя, а значением – остаток строки из файла */etc/passwd*.

makemap

makemap [*options*] *type name*

Команда системного администрирования. Переход от стандартного ввода к базам данных **sendmail**. Исходные данные должны быть представлены в виде:

key value

Строки комментариев начинаются с символа #; можно производить подстановку параметров по %**n**, а собственно символ % необходимо экранировать (%%). Аргумент *type* может принимать значения **dbm**, **btree** или **hash**, а *name* – это имя файла, к которому **makemap** добавляет стандартные суффиксы.

makemap

Параметры

–d Разрешить дублирование записей. Использование параметра допустимо только с картами типа **btree**.

–f Не преобразовывать символы из верхнего регистра в нижний.

–N Добавлять нулевой байт к каждому ключу.

–o Добавлять данные к существующему файлу, а не перезаписывать его.

–r Если некоторые из ключей уже существуют, заменить их (по умолчанию **makemap** завершает работу, если находит дублируемый ключ).

–s Не производить проверку допустимости.

–v Режим диагностики.

man

man [*options*] [*section*] [*title*]

Отображение информации из системы встроенных руководств. Команда **man** находит и отображает указанную тему (*title*) из определенного раздела (*section*) системы.

Параметры

–7, ––ascii

Считать, что файл в чистом формате ASCII, форматировать его для 7-битного терминала или эмулятора терминала.

–a, ––all

Показать все страницы, соответствующие теме (*title*).

–b Не удалять пустые строки из страниц.

–d, ––debug

Отображать отладочную информацию. Подавить собственно вывод страниц руководств.

–f, ––whatis

Действует идентично команде **whatis**.

–k, ––apropos

Действует идентично команде **apropos**.

–l, ––local-file

Искать страницы руководств среди локальных, а не системных файлов. Если в качестве имени файла задан символ **i**, производить поиск в данных, поступающих со стандартного ввода.

–m *systems*, **––systems=***systems*

Искать в руководствах систем *systems* (перечисляются через запятую).

–p *preprocessor*, **––preprocessor=***preprocessor*

Предварительная обработка страниц руководств с помощью указанного препроцессора перед передачей их **nroff**, **troff** или **groff**. Всегда выполнять сначала команду **soelim**.

–r *prompt*, **––prompt=***prompt*

Установка приглашения для средства постраничного просмотра **less**.

–t, **––troff**

Форматирование страниц руководства командой */usr/bin/groff -Tgv -mandoc*. Подразумевается параметрами **–T** и **–Z**.

–u, **––update**

Выполнять проверку непротиворечивости страниц, помещенных в кэш и реально существующих в системе.

–w, **––where**, **––location**

Отображать пути файлов руководств на стандартный вывод.

–D, **––default**

Использовать для всех параметров значения по умолчанию.

–L *locale*, **––locale=***locale*

Текущий набор параметров локализации имеет значение *locale*. Не вызывать функцию **setlocale()**.

–M *path*, **––manpath=***path*

Искать страницы руководств в пути *path*. Игнорировать параметр **–m**.

–P*pager*, **––pager=***pager*

Предписание использовать программу постраничного просмотра *pager*.

–T *device*, **––troff-device[**=*device***]**

Форматирование **groff** или **troff** для вывода на устройство *device*, например **dvi**, **latin1**, **X75** и **X100**.

–Z, **––ditroff**

Запретить дополнительную обработку после завершения форматирования **groff**.

man	**Имена разделов** Страницы руководств распределены по разделам, каждый из которых предназначен для отдельной категории читателей: **1** Исполняемые программы или команды интерпретатора. **2** Системные вызовы (функции, предоставляемые ядром системы). **3** Библиотечные вызовы (функции, предоставляемые системными библиотеками). **4** Специальные файлы (обычно находящиеся в */dev*). **5** Форматы файлов и соглашения. Например */etc/passwd*. **6** Игры. **7** Макропакеты и соглашения. **8** Команды системного администрирования (обычно доступные только суперпользователю). **9** Подпрограммы ядра (нестандартные).
manpath	**manpath** [*options*] Попытаться определить путь к страницам руководств. Сначала выполняется проверка значения переменной $MANPATH; если она не установлена, читаются файлы */etc/man.conf*, переменные окружения пользователя и текущий рабочий каталог. Команда **manpath** является символической ссылкой на **man**, но большая часть параметров при вызове **manpath** игнорируется. **Параметры** **–d, ––debug** Вывести отладочную информацию. **––help** Отобразить справку и завершить работу.
merge	**merge** [*options*] *file1 file2 file3* Тройственное слияние файлов. Команда **merge** объединяет изменения, посредством которых из файла *file2* получается файл *file3*, и записывает их в *file1*. Она полезна, когда необходимо объединить различные изменения исходного файла. Предположим, исходным является файл *file2*, а *file1* и *file3* являются его модификациями. Тогда слияние объединит все

merge

изменения. Противоречие появляется, когда в файлах *file1* и *file3* изменены одни и те же сегменты строк. Найдя такое противоречие, **merge** выдает предупреждение и заключает противоречие в скобки: <<<<<<< и >>>>>>>. Типичное противоречие выглядит следующим образом:

```
<<<<<<< file1
строки из файла file1
=======
строки из файла file3
>>>>>>> file3
```

В случае обнаружения противоречий редактирование (выбор одного из вариантов) предоставляется пользователю.

Параметры

−e Не уведомлять пользователя о наличии противоречий.

−p Отобразить результаты на стандартный вывод, не перезаписывая файл *file1*.

−q Тихий режим; не предупреждать о наличии противоречий.

−A Обработка противоречий в стиле параметра **−A** программы **diff3**. Переходные изменения (от *file2* к *file3*) записываются в файл *file1* и выполняется самая подробная диагностика.

−E Менее подробное, чем с **−A**, описание противоречий. Режим по умолчанию.

−L *label*

Определить до трех меток, используемых вместо соответствующих имен файлов в описании противоречий. То есть команда:

```
merge -L x -L y -L z file_a file_b file_c
```

создает результат, в котором именами файлов являются *x*, *y* и *z*, а не *file_a*, *file_b* и *file_c*.

−V Вывести номер версии.

mesg [*option*]

mesg

Управление возможностью приема сообщений **write**, посылаемых другими пользователями. Команда без параметра отображает текущее состояние терминала.

mesg

Параметры

n Запретить прием сообщений **write**.

y Разрешить прием сообщений **write** (значение по умолчанию).

mimencode

mimencode [*options*] [*filename*] [**–o** *output_file*]

Кодирование (и раскодирование) данных в формате MIME, который является рекомендованным стандартом для мультимедийных почтовых сообщений в Интернете. По умолчанию **mimencode** читает данные со стандартного ввода и выдает base64-кодированные данные на стандартный вывод.

Параметры

–b Использовать кодирование base64 (по умолчанию).

–o *output_file*

Записать результат в указанный файл, а не на стандартный вывод.

–p Переводить декодируемые последовательности CRLF в соответствии с локальными соглашениями о новых строках и производить обратный перевод при кодировании. Параметр имеет смысл только в случае применения base64-кодирования.

–q Использовать кодирование quoted-printable, а не base64.

–u Декодировать данные, поступающие со стандартного ввода.

mkdir

mkdir [*options*] *directories*

Создание одного или нескольких каталогов. Для того чтобы создать новый каталог, необходимо иметь право на запись в родительский каталог. См. также **rmdir**. Права доступа по умолчанию для вновь создаваемых каталогов – 0777; изменяются в соответствии со значениями **umask** (системным или пользователя).

Параметры

–m, **––mode** *mode*

Установить права доступа *mode* для создаваемых каталогов. Формат аргумента *mode* см. в описании команды **chmod**.

mkdir

–p, ––parents

Создать недостающие в пути родительские каталоги.

––verbose

Уведомить пользователя о каждом создании каталога.

––help

Вывести справку и завершить работу.

––version

Отобразить номер версии и завершить работу.

Примеры

Создать каталог **personal**, доступный только для чтения:

```
mkdir -m 444 personal
```

Последовательность команд:

```
mkdir work; cd work
mkdir junk; cd junk
mkdir questions; cd ../..
```

может быть заменена одной командой:

```
mkdir -p work/junk/questions
```

mke2fs

mke2fs [*options*] *device* [*blocks*]
mkfs.ext2 [*options*] *device* [*blocks*]

Команда системного администрирования. Форматирование устройства под файловую систему *ext2fs*. Можно указать количество блоков на устройстве либо позволить **mke2fs** определить его автоматически.

Параметры

–b *block-size*

Указать размер блока в байтах.

–c Перед форматированием проверить устройство на наличие сбойных блоков.

–f *fragment-size*

Указать размер фрагмента в байтах.

–i *bytes-per-inode*

Создавать inode-блок для каждых *bytes-per-inode* байт дискового пространства. Значение должно быть больше либо равно 1024. По умолчанию равно 4096.

mke2fs

–l *filename*

Прочитать список сбойных блоков из указанного файла.

–m *percentage*

Зарезервировать *percentage* процентов блоков для привилегированного пользователя.

–q Тихий режим.

–v Режим диагностики.

–S Записывать только дескрипторы суперблока и групп блоков. Запретить запись таблицы inode-блоков, а также блочных и inode-битовых карт. Параметр полезен при попытках спасти поврежденную файловую систему.

mkfs

mkfs [*options*] [*fs-options*] *filesys* [*blocks*]

Команда системного администрирования. Создать файловую систему на устройстве (например, на разделе жесткого диска). Аргумент *filesys* является либо именем устройства, либо точкой монтирования. Команда, по существу, представляет собой интерфейс для вызова соответствующего варианта **mkfs** в зависимости от файловой системы, указанной в параметре **–t**. К примеру, для создания второй расширенной файловой системы Linux вызывается **mkfs.ext2** (что идентично **mke2fs**); файловые системы MS-DOS используют **mkfs.msdos**. *fs-options* – это параметры создаваемой системы. *blocks* – размер файловой системы в блоках по 1024 байт.

Параметры

–V Диагностика работы, включая все выполняемые для создания файловой системы команды.

–t *fs-type*

Тип создаваемой файловой системы.

Специфические параметры файловой системы

Эти параметры должны следовать строго после основных. В большинстве программ, создающих файловые системы, поддерживаются следующие параметры:

–c Перед созданием файловой системы произвести проверку на наличие сбойных блоков.

–l *file* Взять список существующих сбойных блоков из указанного файла.

–v Диагностика работы.

mkfs.minix [*options*] *device size*

mkfs.minix

Команда системного администрирования. Создание файловой системы MINIX. См. **mkfs**.

mklost+found

mklost+found

Команда системного администрирования. Создание каталога *lost+found* в текущем рабочем каталоге. Предназначается для второй расширенной файловой системы Linux.

mkraid [*options*] *devices*

mkraid

Команда системного администрирования. Настройка устройств RAID-массивов в соответствии с параметрами, определенными в файле настройки */etc/raidtab*. **mkraid** может применяться для инициализации нового массива либо обновления старых устройств к более новому ядру системы. Инициализация уничтожит все данные на устройствах, используемых для создания массива.

Параметры

–c *file*, **––configfile** *file*

Использовать указанный файл вместо */etc/raidtab*.

–f, **––force**

Инициализировать устройства RAID-массива, даже если они содержат данные.

–h, **––help**

Вывести справку и завершить работу.

–o, **––upgrade**

Обновить старый массив для установления соответствия более новой версии RAID ядра. Сохранить данные массива.

–V, **––version**

Отобразить информацию о версии программы и завершить работу.

mkswap [*option*] *device* [*size*]

mkswap

Команда системного администрирования. Создание раздела подкачки на устройстве *device*. Можно задать размер раздела в блоках, причем каждый блок является страницей размером 4 Кбайт.

mkswap	**Параметр** **–c** Перед созданием раздела подкачки произвести проверку на наличие сбойных блоков.
modprobe	**modprobe** [*options*] [*modules*] Команда системного администрирования. Команда без параметров загружает указанный модуль, а также все модули, необходимые для его работы. Если указанных модулей несколько, каждый последующий загружается только в том случае, если не был загружен предыдущий. **Параметры** **–a** Загрузить все перечисленные модули, а не только первый. **–l** [*pattern*] Перечислить все существующие модули. Этот параметр может использоваться совместно с **–t** для задания типа модуля; также может указываться шаблон (*pattern*) для поиска. **–r** Выгрузить указанные модули, а также все модули, зависимые от них. **–t** *type* Загружать только модули определенного типа. Имена каталогов, в которых хранятся такие модули, содержатся в файле */etc/conf.modules*. **Файлы** */etc/conf.modules* Информация о модулях и их зависимостях, а также о каталогах, соответствующих определенным типам модулей. */sbin/insmod, /sbin/rmmod, /sbin/depmod* Программы, с которыми работает **modprobe**.
more	**more** [*options*] [*files*] Постраничное отображение содержимого указанных файлов на терминале. См. описание команды **less** – альтернативы команде **more**. Некоторые команды могут предваряться числом.

Параметры

+*num*

Начать отображение со строки *num*.

−num *number*

Установить размер экранной страницы в *number* строк.

+/*pattern*

Начать отображение за две строки до соответствия шаблону *pattern*.

−c Обновить экран с первой строки вместо прокрутки.

−d Отображать приглашение «Hit space to continue, Del to abort» в ответ на неправильную команду; не выдавать звуковой сигнал.

−f Считать логические строки, а не экранные. Параметр полезен при отображении строк, длина которых превышает длину экранной строки.

−l Игнорировать символы новой страницы (<Ctrl>+<L>).

−p Постраничный просмотр с полной очисткой экрана вместо прокрутки. Иногда это быстрее.

−r Принудительное отображение управляющих символов в форме ^*x*.

−s Отображать несколько идущих подряд пустых строк как одну строку.

−u Не отображать подчеркивания.

Команды

Все команды **more** основаны на командах **vi**. Многие команды могут предваряться аргументом.

SPACE

Отобразить следующую экранную страницу.

z Отобразить *lines* следующих строк текста и переопределить размер страницы в *lines* строк. По умолчанию страница занимает весь экран.

RETURN

Отобразить *lines* следующих строк текста и переопределить размер страницы в *lines* строк. По умолчанию происходит прокрутка на одну строку.

more

d, ^D

Прокрутить *lines* строк текста и переопределить размер прокрутки в *lines* строк. По умолчанию происходит прокрутка на одну строку.

q, Q, INTERRUPT

Завершить работу.

s Пропустить одну строку текста.

f Пропустить одну экранную страницу текста.

b, ^B

Прокрутка на одну экранную страницу назад.

' Вернуться к точке, из которой начался последний поиск.

= Отобразить номер текущей строки.

/pattern

Поиск по шаблону *pattern*, переход к вхождению с номером *num*, если задан аргумент.

n Повторить последний поиск, совершить переход к вхождению с номером *num*, если задан аргумент.

!*cmd*, **:!***cmd*

Выполнить команду *cmd* в интерпретаторе.

v Вызвать редактор **vi** для редактирования текущего файла, открыть файл на текущей строке.

^L Обновить экран.

:n Перейти к просмотру следующего файла.

:p Перейти к просмотру предыдущего файла.

:f Вывести имя текущего файла и номер строки.

. Повторно выполнить последнюю команду.

Примеры

Постранично просматривать файл *file* в «чистом» режиме и отображать приглашения:

```
more -cd file
```

Удалять подчеркивания из файла *doc* при отображении:

```
nroff doc | more -u
```

Просмотр страниц руководства для команды **grep**; начать со слова «BUGS», удалять лишние пробелы:

```
man grep | more +/BUGS -s
```

mount [*options*] [*special-device*] [*directory*]

mount

Команда системного администрирования. Монтирование файловой системы. **mount** оповещает систему, что монтируемая файловая система присутствует на устройстве *special-device*. Точкой монтирования является каталог *directory*, который должен существовать и быть пустым. После монтирования этот каталог становится корнем смонтированной файловой системы. Команда **mount** без аргументов перечисляет все подмонтированные файловые системы, их точки монтирования, отмечает системы, доступные только для чтения, а также сообщает дату монтирования. Команда **mount** доступна только привилегированному пользователю.

Параметры

–a Монтировать все файловые системы, перечисленные в файле */etc/fstab*. Примечание: это единственный параметр, который не совместим с аргументами *special-device* и *node*.

–f Имитация монтирования. Выполнить все проверочные действия для устройства и каталога, но само монтирование не производить.

–n Не регистрировать монтирование в */etc/fstab*.

–o *option*

Примечание: это единственный параметр **mount**, требующий наличия аргумента *special-device* или *node*. Аргумент *options* может принимать следующие значения:

async

Разрешить асинхронный ввод/вывод для устройства.

auto

Разрешить монтирование с параметром **–a**.

defaults

Использовать значения по умолчанию для всех режимов (**async**, **auto**, **dev**, **exec**, **nouser**, **rw**, **suid**).

dev

Интерпретировать все специальные устройства, существующие в системе.

exec

Разрешить исполнение двоичных файлов.

noauto

Запретить монтирование с параметром **–a**.

mount

nodev

Не интерпретировать специальные устройства, существующие в системе.

noexec

Запретить исполнение файлов.

nosuid

Не принимать во внимание биты **suid** и **sgid**.

nouser

Разрешить доступ к монтируемой файловой системе только привилегированному пользователю.

remount

Считать, что система уже подмонтирована, и перемонтировать ее.

ro Разрешить доступ только для чтения.

rw Разрешить доступ для чтения и записи.

suid

Принимать во внимание биты **suid** и **sgid**.

sync

Синхронный ввод/вывод для устройства.

user

Разрешить непривилегированным пользователям монтировать файловую систему. Обратите внимание, что значениями по умолчанию для такой системы будут **nodev**, **noexec** и **nosuid**, если не указано обратное.

check=relaxed | normal | strict

Позволяет указать, насколько строго должна регулироваться интеграция файловой системы MS-DOS.

conv=binary | text | auto

Определить метод преобразования файлов на файловых системах MS-DOS и ISO 9660.

debug

Включить отладку для файловых систем MS-DOS и **ext2fs**.

errors=continue | remount | ro|panic

Действие, совершаемое при возникновении ошибки. Только для файловых систем **ext2fs**.

–r Монтирование в режиме только для чтения.

mount

–t *type*

Указать тип файловой системы. Возможные значения: **minix**, **ext**, **ext2**, **xiafs**, **hpfs**, **msdos**, **umsdos**, **vfat**, **proc**, **nfs**, **iso9660**, **smbfs**, **ncpfs**, **affs**, **ufs**, **romfs**, **sysv**, **xenix** и **coherent**. Обратите внимание, что **ext** и **xiafs** доступны только для ядер версий меньше 2.1.21 и что вместо **xenix** и **coherent** следует использовать **sysv**.

–v Диагностика монтирования.

–w Монтировать в режиме чтения/записи. Режим по умолчанию.

Файлы

/etc/fstab

Список монтируемых файловых систем и параметров монтирования.

/etc/mtab

Список подмонтированных в настоящий момент систем и параметров монтирования.

mountd

rpc.mountd [*options*]

Команда NFS/NIS. Сервер запросов на монтирование NFS. **mountd** читает файл */etc/exports* и определяет, какие файловые системы доступны для монтирования и какими узлами сети. Также **mountd** отображает информацию о файловых системах, подмонтированных клиентами. См. также **nfsd**.

Параметры

–d, ––debug

Режим отладки. Вывод всей отладочной информации через **syslogd**.

–f *file*, **––exports-file** *file*

Читать права экспорта для файловых систем из указанного файла, а не из */etc/exports*.

–n, ––allow-non-root

Разрешить даже запросы на монтирование, приходящие не через зарезервированные порты.

–p, ––promiscuous

Принимать запросы от любого узла.

mountd

–r, ––re-export

Разрешить повторный экспорт импортированных файловых систем.

–v, ––version

Вывести номер версии.

Файл

/etc/exports

Файл, содержащий информацию о правах на монтирование файловых систем.

mv

mv [*option*] *sources target*

Переместить или переименовать файлы или каталоги. Источник (первая колонка) и приемник (вторая колонка) определяют результат операции (третья колонка):

Источник	Приемник	Результат
Файл	Имя *name* (несуществующее)	Файл получает новое имя *name*
Файл	Существующий файл	Существующий файл перезаписывается исходным
Каталог	Имя *name* (несуществующее)	Каталог получает новое имя *name*
Каталог	Существующий каталог	Исходный каталог перемещается и становится подкаталогом уже существующего каталога
Один или более файлов	Существующий каталог	Файлы перемещаются в указанный каталог

Параметры

–b, ––backup

Создавать резервные копии файлов перед удалением.

–f, ––force

Принудительное перемещение, даже если конечный файл существует; параметр подавляет вывод сообщений об ограничениях прав доступа.

––help

Отобразить справку по использованию и завершить работу.

mv

–i, ––interactive

Запрашивать подтверждение пользователя перед удалением файлов.

–u, ––update

Не удалять файл или ссылку, если время обновления такое же или более позднее, чем у заменяющего файла.

–v, ––verbose

Выводить имя каждого удаляемого файла.

––version

Отобразить информацию о номере версии и завершить работу.

–S *suffix*, **––suffix=***suffix*

Параметр имеет более высокий приоритет, чем переменная окружения SIMPLE_BACKUP_SUFFIX, которая определяет суффикс для создаваемых файлов резервных копий. Если суффикс не установлен ни одним из способов, используется значение по умолчанию – тильда (~).

–V *value*, **––version-control=***value*

Параметр имеет более высокий приоритет, чем переменная окружения VERSION_CONTROL, которая определяет тип создаваемых резервных копий. Допустимые значения:

t, numbered

Всегда создавать пронумерованные резервные копии.

nil, existing

Создавать нумерованные резервные копии только для файлов, у которых они уже есть, в противном случае создавать обычные резервные копии. Значение по умолчанию.

never, simple

Всегда создавать простые резервные копии.

named

named [*options*]

Команда TCP/IP. Сервер доменных имен Интернета. **named** используется библиотеками разрешения для обеспечения доступа к распределенной базе данных имен Интернета. При вызове без аргументов **named** читает файл */etc/named.boot* в поисках начальных данных, а затем отслеживает запросы, приходящие на определенный порт. Более подробную информацию можно найти в RFC 1034 и RFC 1035.

named

В разных дистрибутивах Linux существует несколько исполняемых файлов **named**, поведение которых различается. Если ваша версия не работает так, как описано здесь, не тревожьтесь – к ней наверняка имеется документация.

Параметры

–d *debuglevel*

Отображение отладочной информации. *debuglevel* – это число, отражающее уровень подробности.

–p *port*

Использовать указанный порт. По умолчанию – 42.

[–b] *bootfile*

Файл, используемый вместо *named.boot*. Параметр **–b** является необязательным и предназначен только для задания имен, начинающихся с дефиса.

Файл

/etc/named.boot

Файл читается при запуске **named**.

namei

namei [*options*] *pathname* [*pathname*. . .]

Следовать по пути, пока не найдена конечная точка (файл, каталог, файл символьного устройства и т. д.). Найдя символическую ссылку, **namei** отображает ее и следует по ссылке, отступами указывая контекст поиска. Если превышено максимально допустимое количество символических ссылок в системе, **namei** выводит соответствующее сообщение.

Параметры

–m Отображать права доступа для файлов в стиле **ls**; например «rwxr-xr-x».

–x Отличать точки монтирования символом **D** вместо **d**.

Символы типов файлов

В каждую из строк вывода **namei** вставляет следующие символы для обозначения типа файла:

– Обычный файл.

? Ошибка какого-либо типа.

b Блочное устройство.

namei

c Символьное устройство.

d Каталог.

f: Имя пути, которое **namei** исследует в данный момент.

l Символическая ссылка (отображается как сама ссылка, так и объект ссылки).

s Сокет (socket).

netdate [*options*] [*protocol*] *hostname*...

netdate

Команда TCP/IP. Позволяет привести системное время в соответствие с временем одного из узлов, перечисленных в списке *hostname*. **netdate** пытается определить, какой из узлов является наиболее надежным источником информации. Если **netdate** выполняется непривилегированным пользователем, отображается текущее время без попытки переустановить системные часы. Можно задать используемый протокол: **udp** (по умолчанию) или **tcp** – один или несколько раз для различных узлов.

Параметры

–l *time*

Самый надежный узел выбирается из списка путем сортировки узлов по группам на основе возвращаемого по запросам времени. Затем второй раз опрашивается первый узел из самой многочисленной группы. Различия между временем этого узла и локальным временем записываются при каждой сверке. Затем два интервала времени сравниваются. Если разница между ними больше, чем *time* (5 секунд по умолчанию), кандидатура узла с неточным временем отклоняется.

–v Отображать отсортированные по группам узлы.

netstat [*options*]

netstat

Команда TCP/IP. Отображает состояния сетевых соединений. Для всех активных сокетов отображаются протоколы, число байт в очереди приема, число байт в очереди посылки, номер порта, удаленные адрес и порт, а также состояние сокета.

Параметры

–a Отобразить состояние всех сокетов, а не только активных.

netstat	**–c** Отображать информацию постоянно с обновлением раз в секунду. **–i** Включать статистику по сетевым устройствам. **–n** Отображать численные сетевые адреса. **–o** Отображать дополнительную информацию, например, имя пользователя. **–r** Отображать таблицы маршрутизации. **–t** Перечислять только TCP-сокеты. **–u** Перечислять только UDP-сокеты. **–v** Вывести номер версии и завершить работу. **–w** Перечислять только простые сокеты. **–x** Перечислять только доменные гнезда Unix.
newgrp	**newgrp** [*group*] Изменить групповую принадлежность пользователя на указанную группу. Если группа не указана, изменить на регистрационную группу пользователя. Впоследствии новая группа используется при проверке прав доступа.
newusers	**newusers** *file* Команда системного администрирования. Создание или обновление системных учетных записей пользователей на основе записей из указанного файла. Каждая строка в файле должна иметь формат строк файла */etc/passwd*, с тем исключением, что пароли представляются в открытом виде, а идентификаторы групп могут определяться числами или именами. В процессе обновления поле возраста пароля игнорируется, если запись для пользователя уже существует в файле паролей */etc/shadow*. Если имя группы или численный идентификатор еще не существует, то они будут созданы. Если не существует домашний каталог, он также будет создан.
nfsd	**rpc.nfsd** [*options*] Команда системного администрирования. Демон, запускающий демоны сервера NFS, которые обрабатывают запросы клиентов на монтирование файловых систем. Эти демоны являются процессами пользовательского уровня. Параметры абсолютно идентичны параметрам **mountd**.

nice [*options*] *command* [*arguments*] **nice**

Выполнить команду *command* (с аргументами) с более низким приоритетом (т. е. проявление вежливости по отношению к другим пользователям). Команда без аргументов отображает текущий приоритет планирования (степень вежливости (niceness)). Если **nice** является порожденным процессом, выводится приоритет планирования родительского процесса. Степень вежливости варьируется от –20 (наивысший приоритет выполнения) до 19 (самый низкий приоритет).

Параметры

––help

Вывести справку по использованию и завершить работу.

–n *adjustment*, **–***adjustment*, **––adjustment=***adjustment*

Выполнить команду с приращением приоритета *adjustment* (от 1 до 19). По умолчанию – 10. Привилегированный пользователь может повышать приоритет, указывая отрицательное приращение (например, –5).

––version

Вывести номер версии и завершить работу.

nm [*options*] [*objfiles*] **nm**

Вывести в алфавитном порядке таблицу (список) имен для перечисленных объектных файлов. Если объектные файлы не указаны, производится работа с файлом *a.out*. Результат содержит значение каждого имени, тип, размер, имя и т. д. Также может отображаться ключевой символ категории.

Параметры

–a, ––debug-syms

Вывести отладочные имена.

–f *format*

Указать формат вывода (**bsd**, **sysv** или **posix**). По умолчанию принимается **bsd**.

–g, ––extern-only

Вывести только внешние имена.

–n, –v, ––numeric-sort

Сортировка внешних имен по адресам.

nm	**–p, ––no-sort** Запретить сортировку имен. **–r, ––reverse-sort** Обратный порядок сортировки – алфавитной или по адресам. **––size-sort** Сортировка по размеру. **–u, ––undefined-only** Перечислять только неопределенные имена. **–A, –o, –print-file-name** Отображать для каждого символа имя содержащего файла. **–C, ––demangle** Транслировать имена низкого уровня в читаемые имена. **–D, ––dynamic** Вывести только имена динамических объектов. Полезно, например, при работе с разделяемыми библиотеками. **–P, ––portability** Идентично **–f posix**. **–V, ––version** Вывести номер версии **nm** в стандартный поток ошибок.
nohup	**nohup** *command* [*arguments*] Продолжить выполнение указанной команды с соответствующими аргументами даже после завершения работы интерпретатора (сделать команды иммунными к завершению работы интерпретатора, т. е. **no hangup**). Вывод на терминал по умолчанию добавляется к файлу *nohup.out*. Современные командные интерпретаторы сохраняют процессы, работающие по умолчанию в фоновом режиме; эта команда необходима только в классическом интерпретаторе Борна.
nslookup	**nslookup** [–*option*...] [*host_to_find* | – [*server*]] Команда TCP/IP. Опрос серверов доменных имен Интернета. **nslookup** работает в двух режимах: диалоговом и неинтерактивном. В диалоговом режиме пользователь может запрашивать у серверов DNS информацию о различных узлах и доменах или отображать список узлов домена. Диалоговый режим инициируется при выполнении команды **nslookup** без

параметров (используется сервер имен по умолчанию), если в качестве первого аргумента указаны либо дефис (а второго – имя узла), либо адреса сервера доменных имен Интернета. Неинтерактивный режим используется для отображения имени и запрошенной информации по узлу или домену. Этот режим инициируется, если имя узла или домена является первым аргументом команды. Любая из пар *keyword=value*, перечисленных в описании диалоговой команды **set**, может быть параметром командной строки, причем слово должно предваряться символом «–» (дефис). Необязательным вторым аргументом является сервер имен.

Параметры

Все параметры в описании команды **set** могут указываться в командной строке посредством записи **–***keyword*[=*value*].

Диалоговые команды

exit

Завершение работы с **nslookup**.

finger [*name*] [> | >>*filename*]

Соединиться с сервером finger текущего узла. Существует возможность добавления вывода к файлу (записи в файл).

help, **?**

Вывести краткую сводку команд.

host [*server*]

Получить информацию об узле *host*, используя текущий (по умолчанию) или указанный (*server*) сервер имен.

ls –[ahd] *domain* [> | >>*filename*]

Отображение доступной информации о домене с необязательным сохранением результатов в файле *filename*. Параметр **–a** определяет перечисление псевдонимов узлов в домене, параметр **–h** выводит информацию о процессоре и операционной системе домена, а параметр **–d** – содержимое зональных переходов.

lserver *domain*

Изменить сервер по умолчанию на сервер *domain*. Использовать начальный сервер для поиска информации о сервере *domain*.

root

Изменить сервер по умолчанию на сервер для корневого пространства доменных имен.

nslookup

server *domain*

Изменить сервер по умолчанию на *domain*. Использовать текущий сервер по умолчанию для поиска информации о сервере *domain*.

set *keyword*[=*value*]

Изменить информацию состояния, влияющую на поиск. Допустимые ключевые слова:

all

Вывести текущие значения часто используемых аргументов **set**.

class=*name*

Установить класс запросов в IN (Интернет), CHAOS, HESIOD или ANY. По умолчанию устанавливается IN.

domain=*name*

Изменить имя домена по умолчанию на *name*.

[no]debug

Включение и выключение режима отладки.

[no]d2

Включение и выключение режима исчерпывающей отладки.

[no]defname

Добавлять в каждый запрос имя домена по умолчанию.

[no]ignoretc

Игнорировать ошибку усечения.

[no]recurse

Предписание серверу имен опрашивать или не опрашивать другие серверы, если на нем нет требуемой информации.

[no]search

Для *defname* производить поиск всех имен в родительских или в текущем доменах.

[no]vc

Всегда использовать виртуальный канал при посылке запросов к серверу.

port=*port*

Назначить указанный порт *port* для подключения к серверу.

querytype=*value*

См. **type=***value*.

retry=*number*

Установить количество повторных попыток.

root=*host*

Изменить имя корневого сервера на указанное.

srchlist=*domain*

Список для поиска.

timeout=*number*

Установить интервал ожидания ответа в *number* секунд.

type=*value*

Изменить тип информации, возвращаемой по запросу, на один из следующих:

A	Интернет-адрес узла.
ANY	Любая доступная информация.
CNAME	Каноническое имя для псевдонима.
HINFO	Процессор и тип операционной системы узла.
MD	Адресат почтовых сообщений.
MG	Член почтовой группы.
MINFO	Информация о почтовом ящике или списке рассылки.
MR	Имя домена переименования почты.
MX	Система обмена почтой.
NS	Сервер имен указанной зоны.
PTR	Имя узла или указатель на другую информацию.
SOA	Домен start-of-authority.
TXT	Текстовая информация.
UINFO	Пользовательская информация.
WKS	Поддержка традиционных сервисов.

view filename

Сортировка и просмотр выдачи последней команды (или команд) **ls** с помощью **more**.

passwd

passwd [*user*]

Создать или изменить пароль для пользователя *user*. Только владелец пароля или привилегированный пользователь могут изменять его. Владельцу не надо указывать свое имя.

paste

paste [*options*] *files*

Объединить соответствующие строки из одного или нескольких файлов (*files*) в вертикальные колонки, разделяемые табуляцией. См. также **cut**, **join** и **pr**.

Параметры

– Взять имена файлов со стандартного ввода.

–d*char*, **––delimiters=***char*

Разделять колонки символом *char*, а не символом табуляции. Примечание: колонки могут разделяться различными символами, перечисленными в параметре **–d**.

––help

Отобразить справку по использованию и завершить работу.

––version

Вывести информацию о номере версии и завершить работу.

–s, ––serial

Объединить последовательные строки из одного файла.

Примеры

Создать трехколоночный файл *file* из файлов *x*, *y* и *z*:

```
paste x y z > file
```

Перечислить пользователей в две колонки:

```
who | paste - -
```

Объединить строки попарно:

```
paste -s -d"\t\n" list
```

patch

patch [*options*] [*original* [*patchfile*]]

Наложение «заплаты» *patchfile* на файл *original*. Исходный файл заменяется более новой исправленной версией и переименовывается в *original.orig* или в *original~*.

Параметры

+ [*options*] [*original2*]

Повторное наложение исправлений (заплат) с другими параметрами или другим исходным файлом.

–b, **––backup**

Создать резервные копии изменяемых файлов.

–z *suffix*, ***––suffix***=*suffix*

Сохранить исходный файл под именем *original.suffix*.

–B *prefix*, **––prefix**=*prefix*

Добавить *prefix* к имени резервной копии исходного файла.

–c, **––context**

Интерпретировать *patchfile* как контекстный diff-файл (context diff).

–d *dir*, **––directory**=*dir*

Перейти в указанный каталог, прежде чем применять исправления.

–D *string*, **––ifdef**=*string*

Отметить все изменения следующим образом:

```
#ifdef
string
#endif
```

–e, **––ed**

Считать, что *patchfile* содержит команды редактора **ed**.

–E, **––remove-empty-files**

Если в результате работы **patch** появились пустые файлы, удалить их.

–f, **––force**

Принудительно наложить все изменения (заплаты), даже некорректные. Если исходный файл не существует, изменения не накладываются. Изменять файлы с неправильно указанной версией, считать наложенные изменения необратимыми.

–i *file*, **––input**=*file*

Читать файл изменений из указанного файла, а не со стандартного ввода.

patch

–t, ––batch

Пропускать наложение изменений, если исходный файл не существует.

–F *num*, **––fuzz=***num*

Задать максимальное количество строк, которые могут игнорироваться при решении о выборе места вставки кода. По умолчанию равно 2. Имеет смысл только с контекстными diff-файлами (context diffs).

–l, ––ignore-whitespace

Игнорировать пробелы при поиске по шаблонам.

–n, ––normal

Считать *patchfile* обычным diff-файлом (normal diff).

–N, ––forward

Игнорировать обратные (reversed) или уже наложенные изменения.

–o *file*, **––output=***file*

Вывод результата в указанный файл.

–p[*num***], ––strip[=***num***]**

Задать количество удаляемых из имени каталогов. При *num* = 0 удаляются все, остается только имя файла. Значение, равное 1, удаляет первый символ «/»; каждое последующее значение является предписанием удалять на один каталог больше, считая слева.

–r *file*, **––reject-file=***file*

Поместить отклоненные блоки (которые **patch** не смог наложить на исходный файл) в указанный файл. По умолчанию это файл *original.rej*.

–R, ––reverse

Обратное изменение: попытка отката нанесенных «заплатой» повреждений.

–s, ––silent, ––quiet

Подавить диагностику.

–u, ––unified

Интерпретировать *patchfile* как унифицированный контекстный diff-файл.

–V *method*, **––version-control=***method*

Определить способ создания резервных копий исходных файлов (игнорируется в пользу заданного параметра **–B**).

patch

t, numbered

Нумеровать резервные копии.

nil, existing

Создавать резервные копии файлов в соответствии с существующей схемой, по умолчанию создаются обычные резервные копии. Это режим **patch** по умолчанию.

never, simple

Обычные резервные копии.

Переменные окружения

TMPDIR

Указать каталог для временных файлов. По умолчанию – */tmp*.

SIMPLE_BACKUP_SUFFIX

Суффикс, добавляемый к именам резервных копий файлов вместо *.orig* или ~.

VERSION_CONTROL

Указать метод именования резервных копий (см. **–V**).

---

pathchk

pathchk [*option*] *filenames*

Определить корректность и переносимость имен файлов (*filenames*). В частности, определяется, возможен ли поиск в указанных каталогах и допустима ли длина имен файлов.

Параметры

–p, ––portability

Проверка переносимости на все POSIX-системы.

––help

Вывести справку и завершить работу.

––version

Отобразить информацию о версии программы и завершить работу.

---

pcnfsd

/usr/sbin/rpc.pcnfsd

Команда NFS/NIS. Сервер авторизации и запросов на печать NFS. **pcnfsd** является RPC-сервером, поддерживающим клиентов ONC на PC-системах, и использует файл настройки */etc/pcnfsd.conf* (если такой существует). Он отвечает на запросы

pcnfsd

RPC, направляемые программе с номером 150001. Текущая реализация демона **pcnfsd** (на момент написания этой книги) поддерживает как первую, так и вторую версию протокола **pcnfsd**. Запросы, на которые реагирует **pcnfsd**, разделяются на три категории: авторизация, печать и все прочие. Административное значение имеют только запросы на авторизацию и печать.

Авторизация

При получении запроса PCNFSD_AUTH или PCNFSD2_AUTH **pcnfsd** регистрирует пользователя, проверяя имя пользователя и пароль, возвращая соответствующий идентификатор пользователя, идентификатор группы, имя домашнего каталога и значение **umask**. Помимо этого происходит добавление записи в базу данных *wtmp*. Если нет необходимости записывать вход пользователей таким образом, добавьте строку

```
wtmp off
```

в файл */etc/pcnfsd.conf*.

Печать

pcnfsd поддерживает модель печати, основанную на использовании NFS для передачи данных для печати от клиента серверу. Клиент посылает запрос PCNFSD_PR_INIT или PCNFSD2_PR_INIT, а сервер возвращает путь к буферному каталогу, который может использоваться клиентом и который экспортируется NFS. Команда **pcnfsd** создает подкаталог для каждого клиента; родительским каталогом является, как правило, */usr/spool/pcnfs*, а именем подкаталога – имя узла системы клиента. Для того чтобы задать другой родительский каталог, следует добавить строку

```
spooldir path
```

в файл */etc/pcnfsd.conf*. После того как клиент подмонтирует буферный каталог и скопирует в него данные для печати, **pcnfsd** посылает запрос PCNFSD_PR_START или PCNFSD2_PR_START. **pcnfsd** создает команду, основываясь на сервисах печати операционной системы сервера, и выполняет ее, учитывая информацию о пользователе PC. Каждый запрос на печать включает имя устройства вывода. **pcnfsd** интерпретирует принтер либо как конечный файл, передаваемый в системный буфер печати, либо как виртуальное печатающее устройство. Виртуальные принтеры определяются строками следующего вида в файле */etc/pcnfsd.conf*:

pcnfsd

`printer` *name alias-for command*

где *name* – это имя определяемого принтера, *alias-for* – имя, соответствующее реальному принтеру, а *command* – команда, выполняемая при печати файла на устройстве *name*.

perl

perl

Мощный язык для обработки текстов, сочетающий многочисленные удобства программирования в интерпретаторе, на языках C, **awk** и **sed**, а также собственные широкие возможности. Более подробно о языке Perl можно узнать из книги Рэндала Шварца (Randal L. Schwartz) «Learning Perl»[1] и книги Ларри Уолла (Larry Wall), Тома Кристиансена (Tom Christiansen) и Рэндала Шварца (Randal L. Schwartz) «Programming Perl, Second Edition».[2]

pidof [*options*] *programs*

pidof

Отобразить идентификаторы процессов указанных программ. **pidof** по существу является символической ссылкой на **killall5**.

Параметры

–o *pids*

Опускать процессы с перечисленными идентификаторами.

–s Возвращать идентификатор только одного процесса.

–x Возвращать идентификаторы процессов интерпретаторов, выполняющих указанные сценарии.

ping [*options*] *host*

ping

Команда системного администрирования. Проверка доступности удаленного узла и его реакции на запросы. **ping** предназначается для измерения, управления и тестирования сети. Неблагоразумно запускать **ping** во время нормальной работы или в автоматизированных сценариях из-за создаваемой ею нагрузки на сеть.

---

[1] Р. Шварц «Изучаем Perl», издательство «BHV-Киев», 2000 г.

[2] Ларри Уолл, Том Кристиансен и Рэндал Шварц «Программирование на Perl», 3 издание, издательство «Символ-Плюс», IV кв. 2001 г.

ping

Параметры

–c *count*
Завершить работу после посылки (и получения) *count* пакетов ECHO_RESPONSE.

–d Установить режим SO_DEBUG для используемого сокета.

–f Лавинный **ping**: отправлять такое же количество пакетов, какое возвращается, либо 100 пакетов в секунду, в зависимости от того, что больше. Этот параметр может создавать огромную нагрузку на сеть, а потому должен применяться с большой осторожностью. Доступен только привилегированному пользователю.

–i *wait*
Выждать *wait* секунд между посылкой пакетов. По умолчанию интервал равен одной секунде. Параметр несовместим с **–f**.

–l *preload*
Послать *preload* пакетов как можно быстрее, а затем вернуться к обычному поведению. Параметр доступен только привилегированному пользователю.

–n Численный вывод. Не разыменовывать IP-адреса.

–p *digits*
Указать до 16 байт, используемых для заполнения пакетов данных. Полезно для диагностики искажения данных при прохождении их по сети. *digits* записываются в шестнадцатеричной системе. Например, если указать **–p ff**, посылаемые пакеты будут заполняться единицами.

–q Подавить вывод, т. е. отображать только строки статистики в начале и в конце работы.

–r Посылать пакеты напрямую адресату в смежной сети, в обход обычных таблиц маршрутизации.

–s *packetsize*
Указать количество посылаемых байт. По умолчанию – 56, что равно 64 байт ICMP-данных (добавляются 8 байт данных заголовка ICMP).

–v Диагностика: перечислить получаемые ICMP-пакеты (помимо ECHO_RESPONSE).

–R Установить режим IP-записи маршрута, при котором маршрут пакета записывается в заголовке. Запись маршрута отображается при указании параметра **–v** и в возвращаемых пакетах, если целевой узел сохраняет запись маршрута между ответами либо при указании параметра **–l**.

in.pop2d **pop2d**

Команда системного администрирования. Разрешить пользователям работу через порт 109 и доступ к содержимому почтовых ящиков в каталоге */var/spool/mail*. **pop2d** запрашивает пароль и имя пользователя перед открытием доступа к сообщениям и позволяет получать отдельные сообщения. См. также **pop3d**.

Команды

Каждая команда вводится на отдельной строке.

HELO

Запросить имя пользователя и пароль.

FOLD

Открыть папку */var/spool/mail/$USER*.

HOST

Открыть папку */var/spool/pop/$USER*.

READ

Прочитать сообщение.

RETR

Получить сообщение.

ACKS

Сохранить последнее полученное сообщение и перейти к следующему.

ACKD

Удалить последнее полученное сообщение и перейти к следующему.

NACK

Сохранить последнее полученное сообщение и приготовиться к его повторной пересылке.

QUIT

Завершить работу.

in.pop3d **pop3d**

Команда системного администрирования. **pop3d** является более современной версией **pop2d**. Работает аналогично, но команды несколько отличаются.

pop3d	**Команды** **USER** Запросить имя пользователя. **PASS** Запросить пароль. **STAT** Отобразить количество сообщений в почтовом ящике и их суммарный размер. **LIST** Отобразить размер каждого сообщения. **DELE** Удалить сообщение. **NOOP** Выполнить пустую операцию. **LAST** Вывести количество прочитанных сообщений среди последних полученных. **RSET** Сброс: снять все отметки удаления. **TOP** Отобразить первую часть сообщения. **QUIT** Завершить работу.
portmap	**rpc.portmap** [*option*] Команда NFS/NIS. Связывание номера программы RPC с IP-портом. **portmap** является сервером, который переводит номера программ RPC в номера IP-портов. Работа сервера необходима для выполнения RPC-вызовов. При запуске RPC-сервер сообщает **portmap** номер порта приема запросов, а также номера программ, которые обслуживает сервер. Когда клиент выполняет вызов RPC по определенному номеру программы, **portmap** преобразует номер программы в номер порта, через который должны быть отправлены RPC-пакеты. **portmap** должен быть первым из запускаемых RPC-серверов.

portmap

Параметр

–d Выполнять **portmap** в режиме отладки. Работа в режиме демона в этом случае невозможна.

powerd

powerd *device*

Команда системного администрирования. Отслеживать состояние источника бесперебойного питания, подключенного через устройство *device*. При падении уровня питания дает **init** сигнал выполнить функции **powerwait** и **powerfail**. При восстановлении питания дает **init** сигнал выполнить функцию **powerokwait**.

pppd

pppd [*options*] [*tty*] [*speed*]

Команда системного администрирования. PPP (протокол точка-точка, Point-to-Point Protocol) обеспечивает передачу пакетов данных через последовательное соединение. **pppd** производит настройку терминала (*tty*) для PPP (поиском в */dev*) или управляющего терминала (по умолчанию). Можно также задать скорость передачи в бодах (*speed*).

Параметры

asyncmap *map*

Указать, какие управляющие символы не могут передаваться по соединению. Аргумент *map* является 32-битным шестнадцатеричным числом, в котором каждый бит соответствует экранируемому символу. Например, бит 00000001 соответствует символу 0x00; бит 80000000 – символу 0x1f или _. Можно указать несколько символов.

auth

Требовать подтверждения подлинности узлов перед началом передачи пакетов.

connect *command*

Установить соединение указанной командой, которая может являться исполняемой программой или сценарием интерпретатора.

debug, –d

Увеличить уровень отладки.

pppd

defaultroute

Добавить новый маршрут по умолчанию, в котором шлюзом является вторая сторона. Маршрут удаляется по завершении соединения.

–detach

Работа в приоритетном режиме. По умолчанию **pppd** выполняет fork-ветвление и работает в фоновом режиме.

disconnect *command*

Завершить соединение указанной командой, которая может являться исполняемой программой или сценарием интерпретатора.

domain *d*

Указать домен с именем *d*.

escape *character-list*

Экранировать все символы из списка *character-list* (элементы разделяются запятыми и являются шестнадцатеричными числами). Запрещено экранирование символов 0x20–0x3f или 0x5e.

file *file*

Указать файл, содержащий параметры для работы.

lock

Разрешить доступ к устройству только для команды **pppd**.

mru *bytes*

Не принимать пакеты, размер которых превышает *bytes* байт.

name *name*

Указать имя локальной машины.

netmask *mask*

Указать маску сети (например, 255.255.255.0).

passive, –p

Не завершать работу, если вторая сторона не отвечает на попытки установить соединение. Вместо этого переходить к ожиданию корректных пакетов от второй стороны.

silent

Воздержаться от посылки пакетов после получения первого.

pppd

[*local_IP_address*]:[*remote_IP_address*]

Указать IP-адрес локального или удаленного интерфейса в виде символического имени или последовательности чисел.

Файлы

/var/run/pppn.pid

Идентификатор процесса **pppd**. Число *n* в имени *pppn.pid* является номером PPP-интерфейса, соответствующего процессу **pppd**.

/etc/ppp/ip-up

Исполняемый файл или сценарий, вызываемый при установлении PPP-соединения.

/etc/ppp/ip-down

Исполняемый файл или сценарий, вызываемый, когда PPP-соединение завершено.

/etc/ppp/pap-secrets

Содержит имена пользователей, пароли и IP-адреса для PAP-авторизации.

/etc/ppp/options

Файл системных значений по умолчанию. Параметры, описанные в этом файле, устанавливаются раньше, чем перечисленные в командной строке.

~/.ppprc

Файл значений по умолчанию для пользователя. Чтение этих параметров происходит после чтения файла системных значений по умолчанию, но до чтения параметров командной строки.

/etc/ppp/options.ttyname

Имя последовательного порта по умолчанию.

pr

pr [*files*]

Преобразование текста с разбивкой на страницы, колонки, с добавлением заголовков. Если в качестве имени указан символ «–» (дефис), выполняется чтение со стандартного ввода.

Параметры

+*beg_pag*[:*end_pag*], **––pages=**[*beg_pag*[:*end_pag*]]

Начать вывод со страницы *beg_pag* и закончить страницей с номером *end_pag*, если задан соответствующий аргумент.

pr

–*num_cols*, **––columns=***num_cols*

Задать количество колонок вывода. По возможности выводить равное число строк в колонках.

–a, **––across**

Горизонтальная, а не вертикальная печать колонок.

–c, **––show-control-chars**

Преобразовывать управляющие символы в запись с символом «^» (например, **^C**), а прочие непечатаемые символы – в восьмеричные ASCII-коды, предваряемые обратным слэшем.

–d, **––double-space**

Удвоение пробелов.

–e[*tab-char*[*width*]], **––expand-tabs=**[*tab-char*[*width*]]

Преобразовывать символы табуляции (или *tab-chars*) в пробелы. Если задан аргумент *width*, табуляция преобразуется в *width* пробелов (по умолчанию – 8).

–f, **–F**, **––form-feed**

Разделять страницы символом новой страницы, а не символом новой строки.

–h *header*, **––header=***header*

Использовать строку *header* в качестве заголовка текста вместо имени файла.

–i[*out-tab-char*[*out-tab-width*]],

––output-tabs[*out-tab-char*[*out-tab-width*]]

Заменять пробелы табуляциями на выводе. Могут быть заданы альтернативный символ табуляции (по умолчанию просто <Tab>) и ширина (по умолчанию – 8).

–J, **––join-lines**

Произвести слияние полных строк, игнорируя указание параметра **–W**.

–l *lines*, **––length=***lines*

Установить длину страницы в *lines* строк (по умолчанию – 66). Если количество строк меньше 10, опускать заголовки и нижние колонтитулы.

–m, **––merge**

Отображать по одному файлу в колонке.

–n[*delimiter*[*digits*]], **––number-lines**[=*delimiter*[*digits*]]

Нумеровать колонки либо строки (при использовании параметра **–m**). Разделитель *delimiter* (по умолчанию – символ табуляции) добавляется к каждому числу, длина которого может быть ограничена количеством символов *digits* (по умолчанию – 5).

–o *width*, **––indent=***width*

Установить ширину левого поля страницы.

–r, **––no-file-warnings**

При невозможности открыть исходный файл продолжать работу без предупреждения.

–s[*delimiter*], **––separator**[=*delimiter*]

Разделять колонки символом *delimiter* (по умолчанию – символом табуляции), а не пробелами.

–S[*string*], **––sep-string**[=*string*]

Использовать указанную строку в качестве разделителя колонок. По умолчанию это табуляция совместно с **–J** либо пробел.

–t, **––omit-header**

Подавить отображение заголовков, нижних колонтитулов и заполнителей остатков страниц.

–T, **––omit-pagination**

Аналогично **–t**, но не отображать также символы начала новой страницы.

–v, **––show-non-printing**

Преобразовывать неотображаемые символы в восьмеричные ASCII-коды, предваряемые символом \.

–w *page_width*, **––width=***page_width*

Установить ширину страницы в *page_width* символов для многоколоночного вывода. По умолчанию – 72.

–W *page_width*, **––page-width=***page_width*

Установить ширину страницы в *page_width* символов. По умолчанию – 72.

––help

Вывести справку и завершить работу.

––version

Отобразить информацию о версии программы и завершить работу.

praliases

praliases [*option*]

Команда системного администрирования. **praliases** перечисляет текущие почтовые псевдонимы **sendmail**, которые обычно определяются в файле */etc/aliases* или */etc/aliases.db*.

Параметр

–f *file*

Читать псевдонимы из указанного файла, а не из стандартного файла псевдонимов **sendmail**.

printenv

printenv [*variables*]

Отображение значений всех переменных окружения либо значений указанных переменных.

printf

printf *formats* [*strings*]

Вывести строки в соответствии с заданными форматами. Форматы могут содержать обычные символы, escape-символы языка C или аргументы преобразования (наиболее распространенные приведены ниже).

Аргументы

%s

Вывести следующую строку текста.

%*n*$s

Вывести *n*-ную строку текста.

%[-]*m*[.*n*]s

Отобразить следующую строку текста, используя поле в *m* символов шириной. Возможно ограничение печатаемых символов строки первыми *n* символами. Строки выравниваются по правому краю поля, если не задан флаг выравнивания по левому полю (символ «–»).

Примеры

```
printf '%s %s\n' "My files are in" $HOME
printf '%-25.15s %s\n' "My files are in" $HOME
```

ps

ps [*options*]

Сводка по активным процессам. Элементы списков в параметрах разделяются запятыми или заключаются в двойные

кавычки. Обратите внимание, что символ «–» (дефис) перед параметром не является обязательным.[1] Что касается влияния параметров команды на объем выдаваемой информации, то **e** подробнее, чем **a**, а **l** подробнее, чем **f**.

Параметры

pids

Отобразить информацию только по указанным процессам, идентификаторы которых перечисляются через запятую.

a Отобразить все процессы.

c Извлечь имя команды из структуры **task_struct**.

e Включить окружение.

f Выдача в формате дерева потомков.

h Не отображать заголовки колонок.

j Формат заданий.

l Подробный формат выдачи.

m Формат памяти.

n Вывести численные идентификаторы пользователя и значения WCHAN.

r Исключить ожидающие процессы.

s Формат сигналов.

––sort*delimiter*[+|–]*key*[,[+|–]*key*[,. . .]]

Аналогично **O**, но используется с многобуквенными ключами сортировки. См. ниже раздел «Ключи сортировки».

t*tty*

Отобразить только процессы, работающие на терминале *tty*.

u Включать имя пользователя и время запуска процесса.

v Формат **vm**.

---

[1] Вообще говоря, не совсем так. В руководстве необходимость дефиса перед параметром определяется несколько иначе, а именно: параметры Unix можно группировать и их необходимо предварять дефисом; параметры BSD можно группировать и их необходимо использовать без дефиса; параметры в стиле GNU («длинные параметры») необходимо предварять двумя дефисами. Кроме того, параметры разных типов можно использовать одновременно. – *Примеч. науч. ред.*

ps

w Длинный формат. Не усекать слишком длинные строки.

x Отобразить процессы, не имеющие ассоциированного терминала.

O[+|−]*key*[,[+|−]*key*[,. . .]]

Сортировать процессы (см. ниже раздел «Ключи сортировки»).

+ Порядок сортировки по умолчанию.

− Обратить порядок сортировки.

S Включить информацию об использовании процессорного времени и ошибках обращения к памяти для порожденных процессов.

Ключи сортировки

c, cmd

Имя исполняемого файла.

C, cmdline

Вся командная строка.

f, flags

Флаги.

g, pgrp

Групповой идентификатор процесса.

G, tpgid

Групповой идентификатор ассоциированного терминала.

j, cutime

Суммарное время пользователя.

J, cstime

Суммарное время системы.

k, utime

Время пользователя.

K, stime

Системное время.

m, min_flt

Количество некритичных сбоев при обращении к страницам памяти.

M, maj_flt

Количество критичных сбоев при обращении к страницам памяти.

n, cmin_flt

Общее количество некритичных сбоев при обращении к страницам памяти.

N, cmaj_flt

Общее количество критичных сбоев при обращении к страницам памяти.

o, session

Идентификатор сеанса.

p, pid

Идентификатор процесса.

P, ppid

Идентификатор родительского процесса.

r, rss

Размер резидентной части.

R, resident

Резидентные страницы.

s, size

Объем используемой памяти в килобайтах.

S, share

Количество совместно используемых страниц.

t, tty

Терминал.

T, start_time

Время начала работы процесса.

U, uid

Идентификатор пользователя.

u, user

Имя пользователя.

v, vsize

Используемая виртуальная память в байтах.

y, priority

Приоритет планирования ядра.

ps

Поля

PRI

Приоритет планирования выполнения процесса. Чем больше значение, тем ниже приоритет.

NI Значение приоритета процесса. Чем больше значение, тем меньше расход процессорного времени.

SIZE

Размер виртуального образа.

RSS

Занимаемый резидентный объем (физической памяти) в килобайтах.

WCHAN

Функция ядра, выполняемая процессом.

STAT

Состояние:

R Работающий процесс.

T Остановленный.

D Ждущий непрерываемый процесс.

S Ждущий процесс.

Z «Зомби».

W Нет резидентных страниц (второе поле).

N Положительное значение приоритета (**nice**) (третье поле).

TT Ассоциированный с процессом терминал.

PAGEIN

Количество критичных сбоев при обращении к памяти.

TRS

Объем памяти, постоянно занятый под сегмент текста.

SWAP

Использование файла подкачки в килобайтах.

SHARE

Объем разделяемой памяти.

psupdate [*mapfile*] **psupdate**

Команда системного администрирования. Обновление базы данных **psupdate** (в некоторых системах */boot/psupdate*, на других – */etc/psdatabase*), содержащей информацию о системном файле карты загрузки образа ядра. Если не указан файл карты (*mapfile*), **psupdate** использует файл по умолчанию (в зависимости от дистрибутива */usr/src/linux/vmlinux* или */usr/src/linux/tools/zSystem*).

pwck [*option*] [*files*] **pwck**

Команда системного администрирования. Удалить некорректные или дублирующиеся записи из файлов */etc/passwd* и */etc/shadow*. Перед удалением **pwck** запрашивает подтверждение (yes/no). При отрицательном ответе работа завершается. Также можно проверять альтернативные файлы (*files*). При наличии исправимых ошибок пользователю будет рекомендовано применить команду **usermod**.

Параметр

–n Пакетный режим. Не запрашивать подтверждение и не удалять записи. Вернуть соответствующий код завершения.

Код завершения

0 Успешное завершение.

1 Синтаксическая ошибка.

2 Найдены некорректные пароли.

3 Невозможно открыть файл паролей.

4 Невозможно заблокировать файл паролей.

5 Невозможно произвести запись файла паролей.

pwconv **pwconv**

pwunconv

Команда системного администрирования. Преобразовывать нешифрованные записи из */etc/passwd* в шифрованные записи в */etc/shadow*. В файле */etc/passwd* заменять шифрованный пароль символом «x». Шифрование (*shadowing*) паролей позволяет обезопасить их от взлома. **pwconv** записывает в */etc/shadow* дополнительную информацию об окончании

pwconv	действия пользовательской записи, которую он берет из */etc/login.defs*. При добавлении новых записей в */etc/passwd* можно повторно выполнить команду **pwconv** для переноса новой информации в */etc/shadow*. Уже зашифрованные записи игнорируются. **pwunconv** производит восстановление записей (расшифровку в */etc/passwd*) и удаляет файл */etc/shadow*. В процессе преобразования теряется некоторая информация по датам окончания действия пользовательской записи.
pwd	**pwd** Вывод полного пути текущего рабочего каталога. См. также описание команды интерпретатора **dirs (bash, csh/tcsh)**.
quota	**quota** [*options*] [*user*|*group*] Отобразить расходование ресурсов диска группой или пользователем и общее количество ресурсов диска, отведенное пользователю или группе. Команда без аргументов отображает информацию по ограничению ресурсов для пользователя, который ее выполнил. Команда отображает ограничения для всех файловых систем, перечисленных в */etc/fstab*. **Параметры** **–g** Если аргументом является имя пользователя, отобразить статистику для групп, к которым принадлежит этот пользователь. **–q** Отображать информацию только по файловым системам, на которых произошло превышение ограничений. **–u** Поведение по умолчанию. Если указывается совместно с **–g**, то отображается информация по ограничениям и для пользователя, и для всей группы. **–v** Отображать ограничения для файловых систем, даже если дисковое пространство не используется.
raidstart	**raidstart** [*options*] [*devices*] **raidstop** [*options*] [*devices*] Команда системного администрирования. Запуск или останов работы RAID-устройств в соответствии с параметрами, определенными в файле настройки */etc/raidtab*. Параметр **–a** (или **––all**) позволяет выполнить команду для всех устройств, определенных в файле настройки. Устройства при этом можно не указывать.

raidstart

Параметры

–a, ––all

Выполнить команду для всех устройств, определенных в файле настройки.

–c *file*, **––configfile** *file*

Использовать в качестве файла настройки указанный файл, а не */etc/raidtab*.

–h, ––help

Отобразить справку по применению и завершить работу.

–V, ––version

Вывести версию программы и завершить работу.

---

ramsize [*option*] [*image* [*size* [*offset*]]]

ramsize

Команда системного администрирования. Команда без параметров выводит информацию об использовании RAM-дисков. Пара байт по смещению 504 в образе ядра обычно задает размер оперативной памяти. Если указан образ ядра (*image*), выводится информация, расположенная по этому адресу. Для изменения информации необходимо указать новый размер *size* (в килобайтах). Можно также задать другое смещение (*offset*). Обратите внимание, что **rdev –r** является синонимом **ramsize**.

Параметр

–o *offset*

Синоним задания смещения в качестве аргумента.

---

ranlib *filename*

ranlib

Создать индекс для указанного архива. Действует идентично **ar –s**.

---

rarp [*options*]

rarp

Команда системного администрирования. Управление таблицей (обычно */proc/net/rarp)* протокола обратного разрешения адресов (Reverse Address Resolution Protocol, RARP).

rarp

Параметры

–a [*hostname*]

Просмотр всех записей. Если указано имя узла, то просмотр только записей, имеющих отношение к этому узлу. Может быть задан список узлов.

–d *hostname*

Удалить записи, имеющие отношение к узлу *hostname* или, возможно, к узлам из списка.

–s *hostname hw_addr*

Добавить новую запись для узла *hostname* с аппаратным адресом *hw_addr*.

–t *type*

Проверять только записи указанного типа при отображении или изменении таблицы. Аргумент *type* может принимать значения **ether** (по умолчанию) или **ax25**.

–v Режим диагностики.

rcp

rcp [*options*] *file1 file2*

rcp [*options*] *file* . . . *directory*

Копирование файлов с одной машины на другую. Каждый файл или каталог являются конечным именем на удаленной машине в форме *rname@rhost*:*path* или локальным именем файла.

Параметры

–k Пытаться получить квитанции удаленного узла, запрашивая их в области, определяемой по **krb_realm-ofhost**.

–p Сохранить информацию о времени изменения исходных файлов и правах доступа к ним.

–r Если какой-либо из исходных файлов является каталогом, **rcp** копирует все его содержимое.

–x Включить шифрование по DES всех передаваемых посредством **rcp** данных.

rdate

rdate [*options*] [*host*. . .]

Команда TCP/IP. Получить значения текущего времени и даты от узла (узлов) сети и, при необходимости, синхронизировать время локальной системы.

rdate

Параметры

–p Вывести полученные данные.

–s Синхронизировать системное время с полученным. Действие доступно только привилегированному пользователю.

rdev

rdev [*options*] [*image* [*value* [*offset*]]]

Команда системного администрирования. Команда без аргументов отображает строку в синтаксисе */etc/mtab*, описывающую корневую файловую систему. В противном случае изменяет значения байт в образе ядра, которые определяют размер RAM-диска (по умолчанию располагаются по смещению 504), режим VGA (по умолчанию – 506) и корневое устройство (по умолчанию – 508). Необходимо указать изменяемый образ ядра. Можно указать значение *value* и иное смещение *offset*.

Параметры

–o *offset*

То же, что и задание смещения в качестве аргумента. Смещение задается в десятичном виде.

–r Эмуляция **ramsize**.

–s Эмуляция **swapdev**.

–v Эмуляция **vidmode**.

–R Эмуляция **rootflags**.

rdist

rdist [*options*] [*names*]

Команда системного администрирования. Клиентская часть программы удаленного распределения файлов. **rdist** обеспечивает синхронизацию версий файлов на множественных узлах. Управление обновлением файлов и/или каталогов производится командами из файла *distfile*. Альтернативный *distfile* может быть указан параметром **–f** или **–c**.

Параметры

–a *num*

Не обновлять файловую систему, если на ней менее *num* свободных байт.

rdist

–c *name* [*login*@]*host*[:*dest*]

Микровариант файла *distfile*. Аргумент *login* – имя пользователя для регистрации, *host* – целевой узел, *name* – имя передаваемого локального файла, а *dest* – конечное имя передаваемого файла в целевой системе.

–d *var=value*

Определить переменную *var* со значением *value*. Параметр имеет больший приоритет, чем определения переменных в файле *distfile*.

–f *file*

Считать данные из указанного файла (по умолчанию – *distfile*). Если имя указано символом «–» (дефис), выполняется чтение со стандартного ввода.

–l *options*

Указать параметры регистрации на локальной машине.

–m *machine*

Обновлять только для указанной машины *machine*. Параметр может указываться в нескольких экземплярах для различных машин.

–n Подавить нормальное выполнение. Отобразить команды, которые были бы выполнены.

–o *options*

Задать один или более режимов, разделяемых запятыми.

chknfs

Не работать с файлами, которые находятся на файловых системах NFS.

chkreadonly

Перед обновлением файлов проверить, доступна ли система для записи.

chksym

Не обновлять файлы, которые существуют на локальном узле, но являются символическими ссылками на удаленном узле.

compare

Сравнивать файлы и использовать сравнения в качестве критериев принятия решения при обновлении (вместо возраста файлов).

follow

Интерпретировать символические ссылки, копируя файлы, на которые они указывают, вместо создания символических ссылок на удаленной машине.

ignlnks

Игнорировать ссылки, которые невозможно разрешить.

nochkgroup

Не обновлять групповые права владения файлом, если файл не заменяется полностью.

nochkmode

Не обновлять права доступа к файлу, если файл не заменяется полностью.

nochkowner

Не обновлять информацию о владельце файла, если файл не заменяется полностью.

nodescend

Запретить рекурсивный обход каталогов.

noexec

Запретить распределение исполняемых файлов в формате *a.out*.

numchkgroup

Проверять группу-владельца по идентификатору, а не по имени.

numchkowner

Проверять владельца по идентификатору, а не по имени.

quiet

Не отображать команды по мере их выполнения.

remove

Удалять файлы, которые существуют на удаленном узле, но не существуют на локальном.

savetargets

Сохранять обновленные файлы в виде *name.old*.

verify

Отобразить список файлов на удаленной машине, которые требуют обновления, но не обновлять их.

rdist

whole

Сохранять структуру каталогов, создавая необходимые подкаталоги. К примеру, если файл */foo/bar* распределяется в каталог */baz*, то результатом будет файл */baz/foo/bar*, а не */baz/bar* (как по умолчанию).

younger

Не обновлять файлы, которые новее файлов основного узла.

–p *path*

Указать путь поиска **rdistd** на удаленной машине.

–t *seconds*

Указать время ожидания (по умолчанию – 900 секунд), по истечении которого и при отсутствии ответа удаленного сервера **rdist** разорвет соединение.

–A *num*

Указать минимальное число inode-блоков, необходимых **rdist**.

–D Режим отладки.

–F Выполнять все команды последовательно, без fork-ветвления.

–L *options*

Указать параметры регистрации на удаленной машине.

–M *num*

Ограничить количество одновременно выполняющихся порожденных процессов **rdist**. По умолчанию равно 4.

–P *path*

Указать путь к **rsh** на локальной машине.

rdistd

rdistd *options*

Команда системного администрирования. Запуск сервера **rdist**. Обратите внимание, что параметр **–S** является обязательным, если только вы не выполняете простой запрос информации о версии при помощи параметра **–V**.

Параметры

–D Режим отладки.

–S Запуск сервера.

–V Вывести номер версии сервера и завершить работу.

reboot [*options*] **reboot**

Команда системного администрирования. Размонтировать файловые системы, остановить работу системы и перезагрузить машину. Команда немедленно останавливает все процессы, поэтому ее следует использовать только в однопользовательском режиме. Во всех режимах работы, кроме 0 и 6, **reboot** запускает **shutdown –nf**.

Параметры

–d
Подавить запись в */var/log/wtmp*.

–f
Выполнить **reboot**, даже если должна быть вызвана команда **shutdown**.

–n
Подавить обычный вызов **sync**.

–w
Подавить нормальное выполнение; выполнять только запись в */var/log/wtmp*.

---

renice [*priority*] [*options*] [*target*] **renice**

Управление приоритетом выполнения запущенных процессов. Может применяться к процессу, группе процессов или пользователю (*target*). Привилегированный пользователь также может изменять приоритеты выполнения чужих процессов. Для обычных пользователей приоритет (*priority*) должен лежать в интервале от 0 до значения переменной окружения PRIO_MAX (обычно 20), причем чем больше значение, тем ниже приоритет. Привилегированный пользователь может устанавливать отрицательное значение приоритета (вплоть до PRIO_MIN), чтобы ускорить выполнение заданий.

Параметры

+num

Указать приращение приоритета процесса (не путать с конкретным значением приоритета).

–num

Указать отрицательное приращение приоритета процесса (не путать с конкретным значением приоритета).

–g Считать параметры *target* идентификаторами групп процессов.

renice	**–p** Считать параметры *target* идентификаторами процессов (по умолчанию). **–u** Считать параметры *target* именами пользователей.
reset	**reset** Очистка экрана (сброс терминала).
rev	**rev** [*file*] Выдача строк файла в обратном порядке на стандартный вывод. Порядок символов в каждой строке также обращается. Если файл не задан, выполняется чтение со стандартного ввода.
rexecd	**rexecd** *command-line* Команда TCP/IP. Сервер программы **rexec**, обеспечивающий удаленное выполнение с проверкой подлинности, основанной на именах пользователей и паролях. **rexecd** выполняется сервером **inetd** и должна иметь соответствующую запись в файле настройки */etc/inetd.conf*. При получении запроса выполняются следующие действия: 1. Сервер читает символы, поступающие из сокетов, до получения нулевого байта. Полученная строка интерпретируется как десятичное число в ASCII-записи. 2. Если полученное число не равно нулю, оно интерпретируется как номер порта вторичного потока (используемого как **stderr**). Создается еще одно соединение с клиентской машиной через указанный порт. 3. Через тот же сокет принимается имя пользователя. Длина имени ограничена 16 символами. 4. Через тот же сокет принимается незашифрованный пароль (не более 16 символов), завершающийся нулевым символом. 5. Через тот же сокет принимается команда для интерпретатора, завершающаяся нулевым символом. Длина команды ограничена сверху размером, установленным системой для списка аргументов. 6. Затем выполняется проверка пользователя, аналогичная производимой во время входа в систему, и, если авторизация прошла успешно, происходит смена каталога на

rexecd

домашний каталог пользователя и установка соответствующих значений пользователя и группы.

7. Нулевой байт возвращается по соединению, связанному с потоком ошибок **stderr**, и командная строка передается обычному командному интерпретатору пользователя. Сетевые соединения, установленные **rexecd**, автоматически доступны в этой копии интерпретатора.

Диагностика

Username is too long

Длина имени пользователя превышает 16 символов.

Password too long

Длина пароля превышает 16 символов.

Command too long

Длина команды превышает размер списка аргументов.

Login incorrect

Нет записи в файле паролей для указанного пользователя.

Password incorrect

Представлен неверный пароль.

No remote directory

Невозможно перейти (по команде **chdir)** в домашний каталог.

Try again

Произошла ошибка при выполнении системного вызова **fork** сервера.

<shellname>:. . .

Произошла ошибка при выполнении системного вызова **fork** сервера. Невозможно запустить стандартный интерпретатор пользователя.

---

rlogin

rlogin *rhost* [*options*]

Удаленная регистрация в системе. Команда соединяет терминал локального узла с удаленной системой *rhost*. Тип удаленного терминала такой же, как и тип локального. Терминал или размер окна тоже копируются в удаленную систему, если это допустимо.

Параметры

–8 Разрешить передачу 8-битных данных.

rlogin

–e*c* Указать escape-символ (по умолчанию – «~»).

–d Режим отладки.

–k Попытка получать квитанции удаленного узла, запрашивая их в области, определяемой по **krb_realm-ofhost**.

–l *username*

Указать иное имя пользователя (*username*) для регистрации в удаленной системе. По умолчанию это локальное регистрационное имя пользователя.

–x Включить шифрование по DES всех данных, передаваемых в сеансе работы **rlogin**.

–E Отключить распознавание escape-символов.

–K Подавить опознавание по протоколам Kerberos.

–L Разрешить работу в сеансе **rlogin** без пост-обработки (в режиме **litout**).

rlogind

rlogind [*options*]

Команда TCP/IP. Сервер программы **rlogin**, обеспечивающий возможность удаленного входа в систему с определением подлинности пользователей на основе выделенного номера порта и списка узлов, которым разрешен доступ. **rlogind** выполняется сервером **inetd** по запросу на удаленный вход в систему и работает по следующей схеме:

- Сервер проверяет исходный порт клиента. Если порт лежит вне диапазона 0–023, сервер завершает соединение.
- Сервер проверяет исходный адрес клиента и выясняет соответствующее имя узла. Если имя узла не может быть определено, используется IP-адрес узла в записи через точку.

Процесс регистрации (login) передает скорость терминала клиента и тип терминала из переменной окружения TERM.

Параметры

–a Проверять имя узла.

–l Не авторизовать узлы по файлу *.rhosts*, не принадлежащему пользователю root.

–n Подавлять сообщения keep-alive (сохранение соединения).

rm [*options*] *files* **rm**

Удаление указанных файлов. Чтобы удалить файл, пользователь должен иметь разрешение на запись в каталог, содержащий этот файл, но права доступа к файлу значения не имеют. Если у пользователя нет разрешения на запись в файл, будет выполнен запрос на подтверждение удаления.

Параметры

–d, ––directory

Удалять каталоги, даже если они не пусты. Операция доступна только привилегированному пользователю.

–f, ––force

Удалять файлы, защищенные от записи, без подтверждения.

––help

Отобразить справку и завершить работу.

–i, ––interactive

Запрашивать подтверждение на удаление файла.

–r, –R, ––recursive

Если файл является каталогом, удалить все его содержимое, включая и подкаталоги, рекурсивно. Осторожно: использование этого параметра небезопасно.

–v, ––verbose

Включить режим диагностики (**rm** отображает имя файла перед его удалением).

––version

Вывести информацию о версии программы и завершить работу.

–– Отметка конца списка параметров. Параметр удобен, если необходимо задать имя, начинающееся с символа «–» (дефис).

---

rmail *user*... **rmail**

Команда TCP/IP. Обработка почты, полученной по **uucp**, сокращение строки **From** в формате **mail** в единственную строку вида **return-path!sender** с последующей передачей сообщения **sendmail**. **rmail** был разработан для использования с **uucp** и **sendmail**.

rmdir

rmdir [*options*] *directories*

Удаление указанных каталогов (но не их содержимого). Удаляемые каталоги должны быть пустыми (если они не пусты, можно использовать команду **rm –r**). См. также **mkdir**.

Параметры

––help

Отобразить справку и завершить работу.

––ignore-fail-on-non-empty

При работе пропускать каталоги, которые не могут быть удалены.

–p, ––parents

Удалить указанные каталоги и все вышестоящие родительские каталоги, которые становятся при этом пустыми. Удобно при удалении дерева подкаталогов.

––verbose

Режим диагностики работы; вывести соответствующее сообщение для каждого обработанного каталога.

––version

Вывести информацию о версии программы и завершить работу.

rmmod

rmmod [*option*] *modules*

Команда системного администрирования. Выгрузка указанных модулей ядра. Команда выполняется только в том случае, если указанные модули не используются и нет зависящих от них модулей.

Параметр

–r Рекурсивное удаление связанных модулей (всех модулей, которые используют указанный).

rootflags

rootflags [*option*] *image* [*flags* [*offset*]]

Команда системного администрирования. Установка флагов образа ядра (*image*). Команда без аргументов отображает флаги (*flags*) для образа ядра. Флаги являются двухбайтным целым числом, расположенным по смещению 498 в образе ядра (*image*). В настоящее время единственное, за что отвечают флаги, – это монтирование файловой системы в режиме толь-

rootflags

ко для чтения, если значение *flags* не равно нулю. Флаги можно изменить указанием соответствующего образа, значения флагов и смещения, по которому записываются новые данные (по умолчанию – 498). Обратите внимание, что **rdev –R** является синонимом **rootflags**. Если используется **LILO**, применение **rootflags** не требуется. Флаги могут быть установлены в приглашении **LILO** при загрузке.

Параметр

–o *offset*

Идентично указанию смещения в качестве аргумента.

route

route [*option*] [*command*]

Команда TCP/IP. Ручная работа с таблицами маршрутизации, за которые обычно отвечает **routed**. **route** принимает две команды: **add** – для добавления маршрута, и **del** – для его удаления. Эти две команды имеют следующий синтаксис:

add [–net | –host] *address* **[gw** *gateway***]**
[netmask *mask***] [mss** *tcp-mss***] [dev** *device***]**
del *address*

Адрес (*address*) интерпретируется как прямой маршрут, если не указан параметр **–net** или адрес существует в файле */etc/networks*. **–host** позволяет указать, что адрес является прямым маршрутом, независимо от того, содержится он в */etc/networks* или нет. Ключевое слово *default* предписывает использовать этот маршрут для всех запросов, если другой маршрут не найден. Можно также задавать шлюз (*gateway*), через который будут отправляться пакеты на этот адрес, маску сети (*netmask*), TCP *mss* и устройство, с которым ассоциируется маршрут (*device*). Изменять таблицы маршрутизации может только привилегированный пользователь.

Если команда не указана, **route** выводит таблицы маршрутизации.

Параметр

–n Отображать адреса в численном виде; не выполнять поиск имен узлов (такая возможность полезна, если сервер доменных имен (DNS) функционирует некорректно).

routed

routed [*options*] [*logfile*]

Команда TCP/IP. Демон сетевой маршрутизации. **routed** запускается привилегированным пользователем при загрузке системы для управления таблицами межсетевой маршрутизации. Демон маршрутизации работает на основе информационного протокола маршрутизации Xerox NS для обновления записей таблицы маршрутизации ядра. При запуске **routed** выполняет системный вызов **ioctl** SIOCGIFCONF для поиска активных интерфейсов прямых соединений. **routed** посылает REQUEST-пакет каждому из интерфейсов, а затем переходит в цикл ожидания пакетов REQUEST и RESPONSE от других узлов. При получении пакета REQUEST **routed** формулирует ответ на основе информации собственных внутренних таблиц. Создаваемый пакет RESPONSE содержит список известных маршрутов. Любой полученный RESPONSE-пакет используется для соответствующего обновления таблиц маршрутизации.

При обновлении **routed** регистрирует изменения во внутренних таблицах, обновляет таблицы маршрутизации ядра и создает пакет RESPONSE, сообщающий об изменениях всем узлам и сетям, связанным с ним напрямую.

Параметры

–d Режим отладки. Дополнительная информация записывается в *logfile*.

–g Добавить маршрут к адресату по умолчанию.

–q Действует противоположно параметру **–s**.

–s Получить информацию о маршрутизации от **routed**, независимо от того, является он активным демоном межсетевой маршрутизации или нет.

–t Предотвратить переход **routed** в режим фонового выполнения и отсоединение от управляющего терминала. При использовании этого параметра прерывание с клавиатуры принудительно завершает **routed**.

rpcgen

rpcgen [*options*] *file*

Разбор содержимого файла, написанного на языке RPC. Результатом работы является эквивалентная программа на языке C. Код заголовка-эквивалента *file.x* помещается в файл *file.h*, подпрограммы XDR – в *file_xdr.c*, код сервера – в *file_svc.c*, а код клиента – в файл *file_clnt.c*. Строки исходных файлов, предваряемые символом %, игнорируются. По умолчанию **rpcgen** создает код, совместимый с SunOS 4.1.

rpcgen

Параметры

–a Создать все файлы (код клиента и сервера).

–5 Создать код, совместимый с SVR4.

–c Создать подпрограммы XDR. Не может применяться совместно с другими параметрами.

–C Код на ANSI C (по умолчанию).

–D*name*[=*value*]

Определить имя *name* и присвоить ему указанное значение либо 1 (по умолчанию).

–h Создать заголовочный файл. При использовании с параметром **–T** включить поддержку координирующих таблиц RPC. Не может применяться совместно с другими параметрами.

–I Создать **inetd**-совместимый сервер.

–K *secs*

Определить интервал ожидания сервера после ответа на запрос и перед завершением работы. По умолчанию –120. Значение –1 предотвращает завершение программы.

–l Создать код клиента. Не может применяться совместно с другими параметрами.

–m Создать только код сервера, подавив создание «головной» программы. Не может использоваться совместно с другими параметрами.

–N Новый стиль. Допустимы множественные аргументы процедур. Не гарантируется обратная совместимость.

–o [*file*]

Выдача результатов в указанный файл или на стандартный вывод.

–Ss Создать только каркас кода сервера.

–t Создать координирующую таблицу RPC. Не может применяться совместно с другими параметрами.

–T Включить поддержку координирующих таблиц RPC.

rpcinfo

rpcinfo [*options*] [*host*] [*program*] [*version*]

Команда NFS/NIS. Отображение информации RPC. Программа (*program*) может задаваться именем или номером. Если указана версия (*version*), **rpcinfo** пытается вызвать эту версию указанной программы. В противном случае пытается найти все зарегистрированные версии указанной программы (вызовом версии 0) и вызвать каждую из них.

rpcinfo

Параметры

–b *program version*

Послать RPC-сообщение указанной программе с версией *version*, используя UDP, и отобразить список ответивших узлов.

–d *program version*

Удалить указанную версию программы из списка зарегистрированных. Операция может выполняться только пользователем, добавившим программу, или привилегированным пользователем.

–n *portnum*

Число *portnum* задает номер порта для параметров **–t** и **–u** вместо порта, указанного демоном portmapper.

–p [*host*]

Послать запрос демону portmapper указанного узла и получить список всех зарегистрированных программ RPC. Если узел (*host*) не задан, используется значение, возвращаемое вызовом **hostname**.

–t *host program* [*version*]

Произвести RPC-вызов программы (*program*) на указанном узле (*host*), используя TCP, и сообщить о получении ответа либо об отсутствии такового.

–u *host program* [*version*]

Произвести RPC-вызов программы (*program*) на указанном узле (*host*), используя UDP, и сообщить о получении ответа либо об отсутствии такового.

Примеры

Для отображения всех сервисов RPC, зарегистрированных на локальной машине, введите:

```
$ rpcinfo -p
```

Для отображения всех сервисов RPC, зарегистрированных на машине с именем **klaxon**, введите:

```
$ rpcinfo -p klaxon
```

Для перечисления всех машин локальной сети, на которых работает сетевая информационная служба (NIS), используйте команду:

```
$ rpcinfo -b ypserv version | uniq
```

где *version* – текущая версия NIS, информацию о которой можно получить, указав параметр **–p**.

rpm [*options*] **rpm**

Менеджер пакетов Red Hat (Red Hat Package Manager). Свободно распространяемая система работы с пакетами дистрибутивов программного обеспечения. Пакеты RPM собираются, устанавливаются и опрашиваются с помощью команды **rpm**. Подробная информация по работе с **rpm** приведена в главе 5 «Менеджеры пакетов Red Hat и Debian».

rsh [*options*] *host* [*command*] **rsh**

Выполнение указанных команд на удаленном узле либо (если команды опущены) запуск диалогового интерпретатора на удаленном узле посредством **rlogin**.

Параметры

–d Включение отладки сокетов.

–k Попытаться получить квитанции для удаленного узла, запрашивая их в области, определяемой **krb_realmofhost(3)**.

–l *username*

Регистрация под именем пользователя *username*. По умолчанию это имя пользователя, выполняющего **rsh**.

–n Перенаправить ввод **rsh** из специального файла */dev/null*. (Это необходимо делать при переводе **rsh** в режим фоновой работы для перенаправления ввода с терминала.)

–x Включить шифрование по DES для обмена любыми данными.

–K Не производить опознавание по протоколам Kerberos.

rshd [*options*] **rshd**

Команда TCP/IP. Сервер удаленных интерпретаторов для таких программ, как **rcmd** и **rcp**, которым требуется выполнение неинтерактивных интерпретаторов на удаленных машинах. **rshd** запускается сервером **inetd** и должен быть представлен соответствующей записью в файле */etc/inetd.conf*.

Все параметры в точности совпадают с параметрами **rlogind**, за исключением **–L**, который присутствует только в **rshd**.

Параметр

–L Вести журнал всех удачных и неудачных соединений посредством **syslogd**.

rstat	**rstat** *host* Команда TCP/IP. Статистика состояния системы указанного узла: текущее время, время непрерывной работы, средние показатели загрузки (среднее число заданий в очереди процессов). Происходит опрос демона **rstat_svc** удаленного узла.
run-parts	**run-parts** [*options*] [*directory*] Команда системного администрирования. Выполнить в алфавитном порядке все сценарии из указанного каталога. Исключать файлы с именами, содержащими не буквенно-цифровые символы (кроме подчеркиваний и дефисов). **Параметры** **––** Интерпретировать последующие аргументы как имена файлов. **––test** Вывести информацию о сценариях, которые будут выполнены, но не выполнять их. **––umask**=*umask* Указать значение *umask* (по умолчанию – 022).
runlevel	**runlevel** Команда системного администрирования. Отобразить предыдущий и текущий режимы работы системы.
ruptime	**ruptime** [*options*] Команда TCP/IP. Отображение информации о длительности непрерывной работы каждого из узлов локальной сети, а также о пользователях, работающих с каждым из узлов в настоящий момент. Машины, доклад о состоянии которых не приходил в течение 11 минут, считаются неработающими. Список сортируется по именам узлов. **Параметры** **–a** Включить в перечисление пользователей с периодом бездействия, превышающим один час. **–l** Отсортировать список по показателю средней загрузки. **–r** Отсортировать в обратном порядке.

ruptime

–t Отсортировать по показателю продолжительности непрерывной работы.

–u Отсортировать по количеству пользователей, работающих в настоящий момент в системе.

rusers

rusers [*options*] [*host*]

Команда TCP/IP. Перечислить пользователей, работающих с узлом *host* либо со всеми машинами локальной сети, в формате выдачи **who** (имя узла, имена пользователей).

Параметры

–a Включить узлы, на которых в данный момент нет работающих пользователей.

–l Более подробная информация: терминал, дата, время, время бездействия, удаленный узел.

rwall

rwall *host* [*file*]

Команда TCP/IP. Посылка сообщения всем пользователям, работающим на определенном узле (*host*). Если указано имя файла, посылается его содержимое; в ином случае происходит чтение со стандартного ввода.

rwho

rwho [*option*]

Перечислить пользователей, в данный момент работающих на всех машинах в локальной сети (аналог **who**).

Параметр

–a Включать в перечисление даже пользователей, бездействующих более одного часа.

rwhod

rwhod

Команда TCP/IP. Сервер состояния системы, который ведет базу данных, используемую программами **rwho** и **ruptime**. Работа сервера основана на способности передавать широковещательные сообщения по сети. Как поставщик информации **rwhod** периодически определяет состояние системы и создает пакет состояния, который затем посылается в сеть. Как потребитель информации **rwhod** принимает пакеты состояний от других серверов, проверяет их и записывает в коллекцию файлов, расположенную в каталоге */var/spool/rwho*.

rwhod	Сообщения, получаемые **rwhod**, игнорируются, если получены не через порт сервера. Пакеты состояний генерируются приблизительно раз в три минуты.
script	**script** [*option*] [*file*] Выполнить fork-ветвление текущего интерпретатора и создать запись клавиатурных нажатий сеанса работы. Результат сохраняется в файле *file*. Если имя файла не указано, выполняется сохранение в файл *typescript*. Запись сценария прекращается, когда завершается работа порожденного интерпретатора, обычно по команде **exit** или в результате нажатия клавиш <Ctrl>+<D>. **Параметр** **–a** Добавить команды к файлу *file* или *typescript*, а не перезаписывать его содержимое.
sed	**sed** [*options*] [*command*] [*files*] Потоковый редактор. Позволяет редактировать один или более файлов в пакетном режиме. Подробно **sed** описан в главе 12 «Редактор sed».
sendmail	**sendmail** [*flags*] [*address...*] Команда системного администрирования. **sendmail** является агентом передачи почты (Mail Transfer Agent, MTA). Программа принимает сообщения от почтового клиента пользователя, интерпретирует почтовый адрес, преобразует в адрес, доступный программе доставки, а затем переправляет почту соответствующей программе доставки. **Флаги командной строки** **–b***x* Установить один из следующих режимов работы *x*: **a** Режим ARPAnet. **d** Режим демона. **i** Инициализация базы данных псевдонимов. **m** Доставка почты (режим по умолчанию). **p** Вывод почтовой очереди. **s** SMTP на вводе.

t Режим проверки.

v Только проверка адреса; без сбора и доставки почтовых сообщений.

–C *file*

Использовать указанный файл настройки.

–d *level*

Установить уровень отладки.

–F *name*

Указать полное имя пользователя.

–f *name*

Указать имя отправителя.

–h *cnt*

Установить значение счетчика повторов (сколько раз сообщение было обработано **sendmail**).

–n Не создавать псевдонимы и не пересылать письма.

–o *x value*

Установить значение параметра *x* равным *value*. Параметры описаны ниже.

–p *protocol*

Получение сообщений по протоколу *protocol*.

–q [*time*]

Обрабатывать сообщения из очереди немедленно либо через указанные интервалы времени (*time*). Например, при указании параметра **–q30m** проверка будет производиться каждые полчаса.

–r *name*

Устаревшая форма параметра **–f**.

–t Прочесть поля **To:**, **Cc:** и **Bcc:** заголовка и послать копии сообщений по всем найденным адресам.

–v Режим диагностики.

–X *file*

Вести журнал рабочей нагрузки в файле *file*. Не следует применять для обычного ведения журналов.

Параметры настройки

Следующие параметры могут устанавливаться флагом **–o** в командной строке либо строкой **O** в файле настройки:

sendmail

7 Преобразовать все входящие сообщения в 7-битный формат.

a*min*

Если установлен параметр **D**, в течение *min* минут ожидать завершения пересборки файла псевдонимов перед возвращением предупреждения об устаревшей базе данных псевдонимов.

A*file*

Использовать альтернативный файл псевдонимов.

b*minblocks*[/*maxsize*]

Определить минимально допустимое количество свободных блоков (*minblocks*) и при необходимости ограничить максимальный размер сообщения (*maxsize*). Если аргумент *maxsize* не указан, символ / также можно опустить.

B*char*

Задать символ замены немаскированного пробела.

c Не производить немедленного соединения с почтовыми программами, если таковое считается «накладным».

C*num*

Ввести контрольные точки в очередь при отправке сообщения нескольким адресатам. Команда **sendmail** перезаписывает перечень адресатов после обработки каждой группы из *num* адресатов.

d*x* Установить режим доставки *x*. Режимы доставки: **d** – отложенная, **i** – диалоговая (синхронная), **b** – фоновая (асинхронная) и **q** – только очередь, т. е. отправлять при следующей обработке очереди.

D При необходимости попытаться автоматически пересобрать базу данных псевдонимов.

e*x* Установить режим обработки ошибок *x*. Существующие режимы: **m** – сообщение об ошибке посылается отправителю, **w** – сообщение об ошибке отображается для отправителя, **p** – сообщение об ошибке выдается на терминал (по умолчанию), **q** – сообщения об ошибках удаляются и **e** – специальная обработка для BerkNet.

E*text*

Указать заголовок для сообщений об ошибках. *text* является либо непосредственно отображаемым текстом, либо именем файла. Имя файла должно быть абсолютным и начинаться с символа «/».

f Сохранить строки **From** в стиле Unix в начале сообщений.

F*mode*

Установить значения по умолчанию для прав доступа к временным файлам. Если аргумент опущен, по умолчанию задаются права 0644.

G Сравнить локальные почтовые имена с разделом GECOS в файле паролей.

g *n* Идентификатор группы по умолчанию, используемый для вызова почтовых программ.

H*file*

Файл справки SMTP.

h *num*

Максимально допустимое число переходов для сообщения.

i Не считать точку в отдельной строке признаком конца сообщения.

I *arg*

Использовать просмотр DNS и производить соответствующую настройку. Поместить сообщения в очередь, если не удалось установить соединение. Аргументы *arg* идентичны флагам resolver без префикса RES_. Каждый из флагов может предваряться плюсом или минусом, что включает или отключает соответствующий параметр сервера имен. Между **I** и первым флагом обязателен пробел.

j Использовать формат MIME для сообщений об ошибках.

J*path*

Установить альтернативный путь поиска файла *.forward*.

k*num*

Указать размер кэша соединений.

K*time*

Интервал, по истечении которого, при отсутствии активности соединения, последнее разрывается.

l Не пропускать поле заголовка **Errors-To:**

L*n* Указать уровень подробности для ведения журнала.

m Послать копию адресату **me** (отправителю), если определено соответствующее расширение псевдонима.

sendmail

M*X value*

Определить значение макроса в командной строке. Присвоить значение *value* макросу *X*.

n При выполнении **newaliases** проверять правую часть псевдонимов.

o Считать, что заголовок сообщения может быть определен в старом стиле, иначе сообщение гарантированно имеет заголовок современного стиля (т. е. адреса разделяются запятыми, а не пробелами).

p*what,what,. . .*

Настройка приватности SMTP-демона. Аргументы *what* разделяются запятыми и могут принимать следующие значения:

public

Полностью открытый SMTP (по умолчанию).

needmailhelo

Потребовать отправления HELO или ELHO перед отправкой почтовых сообщений.

needexpnhelo

Потребовать отправления HELO или ELHO перед ответом на запрос расширения адреса.

needvrfyhelo

Аналогично предыдущему, но для проверочных запросов.

noexpn

Запретить запросы расширения адресов.

novrfy

Запретить проверочные запросы.

authwarnings

Вставить специальное поле в заголовок сообщения, уведомляющего адресата, что письмо может не являться аутентичным.

goaway

Все вышеперечисленные режимы (кроме **public**).

restrictmailq

Разрешать просмотр почтовой очереди только пользователям той же группы, к которой принадлежит владелец каталога почтовой очереди.

restrictqrun

Ограничить допуск к почтовой очереди ее владельцем и привилегированным пользователем.

P*user*

Отправить копии всех недоставленных сообщений указанному пользователю (обычно это postmaster – администратор почтовой системы).

q*fact*

Множитель (коэффициент) высоких загрузок очередей.

Q*queuedir*

Выбор каталога для почтовой очереди.

R Не отсекать адреса маршрутов.

S*file*

Сохранить статистику в указанном файле.

s Всегда создавать файл очереди, даже в тех случаях, когда это необязательно.

T*time*

Установить интервал времени, в течение которого предпринимаются попытки доставки сообщения.

t*stz*, *dtz*

Установить имя часового пояса.

U*database*

Выполнять поиск информации о перенаправлении сообщений в указанной пользовательской базе данных.

u*N*

Установить пользовательский идентификатор на выполнение почтовых программ.

v Работа в режиме диагностики.

V*host*

Указать полное доменное имя резервного MX-узла.

w Использовать обычную запись вместо записи MX.

x*load*

Поместить сообщения в очередь, если загрузка превышает указанное значение.

X*load*

Не устанавливать SMTP-соединения, если загрузка превышает указанное значение.

sendmail

y*factor*

Установить штрафной коэффициент *factor* за длинные списки получателей сообщений.

Y Выполнить каждое задание из очереди как отдельный процесс. Это позволяет ограничить размер процессов для систем с небольшим объемом памяти.

z*factor*

Множитель увеличения приоритетов. Это значение определяет вес приоритета сообщения. Значение по умолчанию для **sendmail** составляет 1800.

Z*inc*

Увеличивать приоритет сообщений в очереди на *inc* после завершения каждого задания. Значение по умолчанию равно 90 000.

Файлы поддержки sendmail

/usr/lib/sendmail

Исполняемый файл **sendmail**.

/usr/bin/newaliases

Ссылка на */usr/lib/sendmail*. Выполняется пересборка базы данных псевдонимов.

/usr/bin/mailq

Вывод содержимого очереди сообщений.

/etc/sendmail.cf

Файл настройки в текстовом виде.

/etc/sendmail.hf

Файл справки SMTP.

/usr/lib/sendmail.st

Файл статистики. Может не существовать.

/etc/aliases

Файл псевдонимов в текстовом виде.

/etc/aliases.{pag,dir}

Файл псевдонимов в формате **dbm**.

/var/spool/mqueue

Каталог почтовой очереди и временных файлов.

/var/spool/mqueue/qf

Контрольные файлы сообщений.

sendmail

/var/spool/mqueue/df

Файлы данных.

/var/spool/mqueue/lf

Файлы блокировок.

/var/spool/mqueue/tf

Временные версии файлов *af*, необходимые во время пересборки файла очереди.

/var/spool/mqueue/nf

Используется при создании уникальных идентификаторов.

/var/spool/mqueue/xf

Запись текущего сеанса.

setfdprm

setfdprm [*options*] *device* [*name*]

Загрузка параметров диска для автоматической настройки накопителя на гибких магнитных дисках.

Параметры

–c *device*

Сбросить параметры устройства.

–n *device*

Отключить настройку формата для указанного устройства.

–p *device* [*name* | *parameter*]

Установка и блокировка параметров устройства *device*. Можно использовать *name* для указания конфигурации либо определить отдельные параметры. Возможны следующие значения параметров: **dev**, **size**, **sect**, **heads**, **tracks**, **stretch**, **gap**, **rate**, **spec1** или **fmt_gap**. Исходные значения можно найти в файле */etc/fdprm*.

–y *device*

Включить настройку формата для указанного устройства *device*.

setsid

setsid *command* [*arguments*]

Команда системного администрирования. Выполнить указанную команду с необязательными аргументами *arguments* в отдельном сеансе.

sh	**sh** [*options*] [*file* [*arguments*]] Стандартный командный интерпретатор, принимающий и выполняющий все прочие команды. В системах Linux **sh** является синонимом для интерпретатора **bash**. Подробно он описан в главе 7 «bash: командный интерпретатор Борна».
shar	**shar** [*options*] *files* **shar –S** [*options*] Создание архивов интерпретатора (или shar-файлов), которые представляются в текстовом формате и могут быть включены в почтовые сообщения. Файлы архивов могут быть распакованы выполнением их в */bin/sh*. На системе получателя могут также потребоваться и другие команды, например **compress**, **gzip** и **uudecode**. Создаваемый конечный архив направляется на стандартный вывод, если не задан параметр **–o**. **Параметры** **–a, ––net-headers** Разрешить автоматическое создание заголовков. Может применяться только совместно с параметром **–n**. **–b** *bits*, **––bits-per-code=***bits* Передать параметр **–b** *bits* программе **compress** (если выполняется сжатие). По умолчанию принимается значение, равное 12. Параметр **–b** автоматически включает и параметр **–Z**. **–c, ––cut-mark** Начать shar-файл строкой «Cut here» (вырезать отсюда). **–d** *delimiter*, **––here-delimiter=***delimiter* Использовать для разделения файлов внутри архива разделитель *delimiter*, а не значение SHAR_EOF. **–f, ––basename** При восстановлении файлов давать им только простые имена, что удобно при создании shar-архива из нескольких каталогов или из другого каталога. (Если **shar** передается имя каталога, то содержимое последнего будет восстановлено независимо от наличия параметра **–f**.) **–g** *level*, **––level-for-gzip=***level* Задать степень сжатия *–level* при вызове **gzip**. Значение по умолчанию равно 9. По умолчанию параметр **–g** автоматически включает параметр **–z**.

−−help

Отобразить краткую справку на стандартный вывод и завершить работу.

−l *nn*, **−−whole-size-limit=***nn*

Ограничить размер конечного файла значением *nn* Кбайт, но не разрывать исходные файлы. Требуется использование параметра **−o**.

−m, **−−no-timestamp**

Не выполнять команды **touch** для восстановления информации о времени извлекаемых из архива файлов.

−n *name*, **−−archive-name=***name*

Имя архива, включаемое в заголовок shar-файлов. Параметр необходим при задании параметра **−a**.

−−no-i18n

Не создавать архивы интерпретатора с поддержкой вывода сообщений на языке пользователя; формировать сообщения на английском языке, принятые по умолчанию. По умолчанию **shar** создает архивы с предпочтительными для распаковщика сообщениями (язык которых определяется переменными LANG/LC_MESSAGES).

−o *prefix*, **−−output-prefix=***prefix*

Сохранять архив в файлах от *prefix*.01 до *prefix*.*nn* (вместо выдачи на стандартный вывод). Этот параметр должен применяться, если указаны **−l** или **−L**.

−p, **−−intermix-type**

Разрешить использование позиционных параметров. Параметры **−B**, **−T**, **−z** и **−Z** могут быть вложенными, тогда файлы, перечисленные за параметрами, обрабатываются в указанном режиме.

−−print-text-domain-dir

Вывести имя каталога, в котором **shar** производит поиск файлов сообщений на различных языках, и завершить работу.

−q, **−−quiet**, **−−silent**

Выключить режим диагностики.

−s *who@where*, **−−submitter=***who@where*

Подставить имя и адрес отправителя (иначе **shar** определит имя и адрес автоматически).

shar

––version

Вывести номер версии программы на стандартный вывод и завершить работу.

–w, ––no-character-count

После распаковки не проверять каждую строку командой **wc –c**. По умолчанию проверка выполняется.

–x, ––no-check-existing

Перезаписывать существующие файлы. По умолчанию производится проверка, и существующие файлы сохраняются. Если **–c** передается в качестве параметра сценарию при распаковке (**sh** *archive* **–c**), происходит безусловная перезапись файлов. См. также **–X**.

–z, ––gzip

Обработать файлы **gzip** и **uuencode**, прежде чем поместить их в архив. Распаковка должна осуществляться с помощью **uudecode** и **gunzip** (или **zcat**).

–B, ––uuencode

Считать все файлы двоичными; выполнить **uuencode** перед упаковкой файлов. Это увеличивает размер архива. Для извлечения файлов необходимо запустить **uudecode**.

–D, ––no-md5-digest

Не использовать подпись «md5sum» для проверки распакованных файлов. По умолчанию проверка производится.

–F, ––force-prefix

Принудительное добавление префиксного символа к каждой строке, даже если это не требуется. Может незначительно увеличить размер архива, в особенности при использовании совместно с параметром **–B** или **–Z**.

–L *nn*, **––split-size-limit=***nn*

Ограничить размер архива *nn* килобайтами и разделять файлы по необходимости. Части архива, созданные в этом режиме, должны распаковываться в корректном порядке. Требуется использование параметра **–o**.

–M, ––mixed-uuencode

Упаковка файлов в смешанном режиме (по умолчанию). Различаются текстовые и двоичные файлы; двоичные файлы кодируются **uuencode** перед помещением в архив.

–P, ––no-piping

Использовать временные файлы вместо конвейеров.

shar

–Q, ––quiet-unshar

Отключить диагностику.

–S, ––stdin-file-list

Читать список файлов, помещаемых в архив, со стандартного ввода, а не из командной строки. Ввод должен иметь формат, схожий с форматом выдачи команды **find**, т. е. одно имя в строке.

–T, ––text-files

Считать все файлы текстовыми.

–V, ––vanilla-operation

Создавать shar-файлы, для распаковки которых понадобятся **sed** и **echo** в окружении распаковщика.

–X, ––query-user

Запрашивать подтверждение пользователя на перезапись существующих файлов при распаковке.

–Z, ––compress

Сжимать и кодировать **uuencode** все файлы перед помещением в архив.

showmount

showmount [*options*] [*host*]

Команда NFS/NIS. Отображение информации о сервере NFS. Эта информация поддерживается сервером **mountd** на узле *host*. Значением по умолчанию для *host* является значение, возвращаемое командой **hostname**. Команда без параметров перечисляет клиентов, имеющих подмонтированные каталоги на узле *host*. **showmount** обычно находится в каталоге */usr/sbin*, который не входит в путь поиска по умолчанию.

Параметры

–a, ––all

Вывести информацию о подмонтированных системах в виде:

hostname:*directory*

где *hostname* – это имя клиента, а *directory* – корень подмонтированной файловой системы.

–d, ––directories

Перечислить каталоги, удаленно подмонтированные клиентами.

showmount	**–e, ––exports** Вывести список экспортированных файловых систем. **–h, ––help** Краткая справка. **––no-headers** Не отображать заголовки. **–v, ––version** Информация о текущем номере версии программы.
shutdown	**shutdown** [*options*] *when* [*message*] Команда системного администрирования. Принудительно завершить все процессы в системе. Аргумент *when* может быть задан конкретным моментом времени (в формате *hh*:*mm*), количеством минут ожидания (в форме +*m*) или ключевым словом **now**. Широковещательное сообщение (*message*) уведомляет всех пользователей, что им необходимо завершить работу с системой. Все процессы получают сигнал SIGTERM, позволяющий им мягко завершить работу. Для собственно останова системы вызывается */etc/init*, т. е. система переводится в уровень выполнения (runlevel) 1. Команда **shutdown** доступна только привилегированному пользователю. Широковещательные сообщения (посылаемые по умолчанию или задаваемые в командной строке) отображаются с равным интервалом во время паузы перед остановом системы; чем ближе время останова, тем короче интервал между сообщениями. **Параметры** **–c** Отменить запущенную последовательность останова. **–f** Быстрая перезагрузка без запуска **fsck**. **–h** Полный останов (halt) системы по завершении последовательности. **–k** Вывести широковещательное сообщение, но не производить останов системы. **–n** Выполнить останов без вызова **init**. **–r** Перезагрузить систему после останова. **–t** *sec* Пауза в *sec* секунд между принудительным завершением процессов и сменой уровня выполнения системы.

size [*options*] [*objfile*. . .] **size**

Отобразить размер каждого из разделов объектного файла в байтах и общий размер файла. Если объектный файл не указан, используется *a.out*.

Параметры

–d Отобразить размеры в десятичной и шестнадцатеричной системах.

––format *format*

Имитация команды **size** System V (**––format sysv**) или BSD (**––format berkeley**).

–o Отобразить размеры в восьмеричной и шестнадцатеричной системах.

––radix *num*

Указать основание для отображения размеров: шестнадцатеричное и десятеричное (*num* = **10** или **16**) или восьмеричное (*num* = **8**).

–x Отобразить размеры в шестнадцатеричной и десятичной системах.

–A Имитация команды **size** System V.

–B Имитация команды **size** BSD.

---

slattach [*options*] [*tty*] **slattach**

Команда TCP/IP. Настройка последовательных линий в качестве сетевых интерфейсов для работы по протоколу PPP. Операции с сетевыми интерфейсами доступны только привилегированному пользователю.

Параметры

–c *command*

Выполнить указанную команду, после того как соединение будет разорвано.

–d Режим отладки.

–e Завершить работу сразу после инициализации линии.

–h Завершить работу при разрыве соединения.

–l Создать блокирующий файл в стиле UUCP в каталоге */var/spool/uucp*.

–L Разрешить работу по трехлинейному каналу.

slattach

–m Не выполнять инициализацию линии для работы в 8-битном открытом режиме.

–n Аналог **mesg –n**.

–p *protocol*

Указать протокол: **slip**, **adaptive**, **ppp** или **kiss**.

–q Тихий режим; не выводить сообщения.

–s *speed*

Указать скорость линии.

sleep

sleep *amount* [*units*]

Задержка на указанное количество времени перед выполнением следующей команды. По умолчанию аргумент *units* (единицы времени) задан в секундах.

Время	Единицы
s	секунды
m	минуты
h	часы
d	дни

sort

sort [*options*] [*files*]

Сортировка строк в указанных файлах. Выполняется сравнение указанных полей для каждой пары строк либо, если поля не заданы, посимвольное сравнение в машинно-определяемой последовательности. См. также описания команд **uniq**, **comm** и **join**.

Параметры

–b Игнорировать пробелы и символы табуляции в началах строк.

–c Проверить, не являются ли указанные файлы отсортированными, и в случае положительного результата завершить работу.

–d Сортировать в словарном порядке.

–f «Свертка» – регистр символов игнорируется.

−−help

Вывести информацию по использованию и завершить работу.

−i Игнорировать непечатаемые символы (лежащие за пределами интервала ASCII 040–176).

−m Объединить исходные файлы (т. е. сортировать их как один файл).

−n Сортировать в арифметическом порядке.

−o*file*

Записать результат в файл *file*.

−r Изменить порядок сортировки.

−t*c*

Разделять поля символом *c* (по умолчанию – табуляцией).

−u Идентичные строки исходного файла фигурируют в результате сортировки лишь один раз.

−z*recsz*

Отвести по *recsz* байт под каждую строку файла. Этот параметр предотвращает аварийное завершение **sort** в некоторых случаях.

+*n* [−*m*]

Пропустить *n* полей при сортировке и учитывать поля до поля с номером *m*. Если *m* не указано, учитывать поля до конца строки. Адреса представляются в виде *a.b*, что означает символ *b* поля *a*. Если *b* не указано, сортировка начинается с первого символа поля.

−k *n*[,*m*]

Аналогично +. Пропустить *n*−1 полей и обрабатывать поля до поля *m*−1 (т. е. сортировка начинается на поле *n*; поля нумеруются, начиная с 1).

−−version

Вывести информацию о номере версии и завершить работу.

−M Считать первые три буквы сокращением названия месяца (JAN, FEB и т. д.). При сравнении считать, что JAN < FEB и любое правильное название месяца меньше неправильного.

−T *tempdir*

Имя каталога для хранения временных файлов.

sort

Примеры

Перечислить файлы в порядке убывания числа строк в них:

```
wc -l * | sort -r
```

Расположить список слов в алфавитном порядке, удалив дубликаты, и отобразить количество вхождений каждого слова:

```
sort -fd wordlist | uniq -c
```

Отсортировать файл с паролями по значению в третьем поле (идентификатор пользователя):

```
sort +2n -t: /etc/passwd
```

split

split [*option*] [*infile*] [*outfile*]

Разбить файл *infile* на сегменты равного размера. *infile* не изменяется, а результаты записываются в файл *outfile**aa***, *outfile**ab*** и т. д. (По умолчанию – **xaa**, **xab** и т. д.). Если имя исходного файла опущено или указан символ «–» (дефис), происходит чтение со стандартного ввода. См. также **csplit**.

Параметры

–*n*, **–l** *n*, **––lines=***n*

Разделить файл *infile* на сегменты по *n* строк (по умолчанию *n*=1000).

–b *n***[bkm]**, **––bytes=***n***[bkm]**

Разбить файл *infile* на сегменты из *n* байт. Могут быть заданы альтернативные размеры блоков:

b 512 байт.

k 1 Кбайт.

m 1 Мбайт.

–C *bytes***[bkm]**, **––line-bytes=***bytes***[bkm]**

Ограничить размер файла указанным числом байт; заполнять файл полными строками.

––help

Отобразить справку по использованию и завершить работу.

––verbose

Режим диагностики; вывести сообщение при создании секции.

split

––version

Вывести информацию о номере версии и завершить работу.

– Читать данные со стандартного ввода.

Примеры

Разбить файл *bigfile* на сегменты по 1000 строк:

```
split bigfile
```

Объединить четыре файла, затем разбить результат на файлы по 10 строк, имеющие имена *new.aa*, *new.ab* и т. д. Обратите внимание, что без параметра «–» (стандартный ввод) имя **new.** трактовалось бы как несуществующий исходный файл:

```
cat list[1-4] | split -10 - new.
```

stat

stat *filename* [*filenames*. . .]

Для указанных файлов отобразить информацию по единицам inode в удобном для восприятия виде. Сообщения об ошибках «Can't stat file» или «Can't lstat file» обычно означают, что файл не существует. Ошибка «Can't readlink file» обычно означает сбой при чтении символической ссылки.

Результат работы

Пример вывода команды:

```
stat /

  File: "/"
  Size 1024          Filetype: Directory
  Mode: (0755/drwxr-xr-x)  Uid: (   0/  root) Gid: (   0/  system)
Device 3,3           Inode: 2       Links: 21
Access: Tue Apr 11 04:02:01 2000(00000.11:47:35)
Modify: Wed Nov 17 11:46:38 1999(00146.03:02:58)
Change: Wed Nov 17 11:46:38 1999(00146.03:02:58)
```

strace

strace [*options*] *command* [*arguments*]

Отслеживание системных вызовов и сигналов в процессе выполнения указанной команды с необязательными аргументами. **strace** позволяет определить, как передаются данные между программой и ядром системы. Команда **strace** без аргументов отображает в стандартный поток ошибок по одной строке на каждый системный вызов. Для каждого вызова

strace

отображается его имя, аргументы и возвращаемое значение, а также все сообщения об ошибках. Выводится имя сигнала и строка описания.

Параметры

–a *n*

Выравнивать возвращаемые значения в колонке *n*.

–c Производить подсчет всех вызовов и сигналов и выдать сводку по завершении работы программы.

–d Режим отладки. Отладочная информация для **strace** выдается в стандартный поток ошибок (**stderr**).

–e *keyword*[=[**!**]*values*

Передать **strace** выражение, определяющее, какие типы вызовов и сигналов отслеживаются и как. Значения выражений передаются в виде списка, элементы которого разделяются запятыми. Восклицательный знак (**!**) перед списком является отрицанием. Доступны ключевые слова **all** и **none**, а также нижеперечисленные:

abbrev=*names*

Сокращать вывод крупных структур для системных вызовов, перечисленных в *names*.

read=*descriptors*

Отображать все данные, которые прочитаны из перечисленных файловых дескрипторов.

signal=*symbols*

Отслеживать сигналы с перечисленными именами (к примеру, **signal**=SIGIO,SIGHUP).

trace=*values*

Отслеживать перечисленные значения. Значения могут включать имена системных вызовов либо один из следующих наборов системных вызовов:

file Вызовы с именем файла в качестве аргумента

ipc Взаимодействие процессов

network Сетевые

process Управление процессами

signal Имеющие отношение к сигналам

verbose=*names*

Для указанных системных вызовов отображать структуры полностью. По умолчанию имеет значение **none**.

write=*descriptors*

Отображать все данные, записываемые в файлы с перечисленными дескрипторами.

–f Отслеживать fork-ветвление и порождаемые процессы.

–ff Записывать системные вызовы для порожденных процессов в различные файлы, имеющие имена вида *filename.pid*, при использовании параметра **–o**.

–h Вывести справку и завершить работу.

–i Вывести состояние указателя команд для каждого системного вызова.

–o *filename*

Запись вывода в файл с именем *filename*, а не в стандартный поток ошибок **(stderr)**. Если имя файла начинается с символа конвейера «|», оставшаяся часть имени считается командой, на ввод которой передается вывод **strace**.

–O *n*

Принудительно вычитать указанное число микросекунд из времени, которое, по оценке **strace**, заняло выполнение системного вызова. Используется для компенсации времени, которое уходит на расчет данных по вызову.

–p *pid*

Начать работу с уже выполняющимся процессом (по идентификатору). Может быть указано несколько параметров **–p**, и **strace** в этом случае будет отслеживать сразу несколько процессов. Для завершения работы необходимо нажать клавиши <Ctrl>+<c>.

–q Тихий режим. Не отображать сообщения о начале и конце процесса слежения.

–r Относительное время. Вычислять временные интервалы между системными вызовами.

–s *n*

Вывести только *n* первых символов строки. По умолчанию *n*=32.

–S *value*

Сортировать вывод параметра **–c** по указанному значению. Допустимые значения: **calls**, **name**, **time** или **nothing**. По умолчанию используется **time**.

–T Вывести количество времени, затраченное на выполнение системного вызова.

–t Выводить время в каждой строке вывода.

strace

–tt Выводить время на каждой строке вывода, включая микросекунды.

–ttt

Выводить на каждой строке время в виде количества секунд, прошедших с начала Эпохи (1 января 1970 года).

–u *username*

Выполнить команду от имени указанного пользователя. Параметр необходим при испытании работы программ **setuid** и **setgid**.

–V Вывести номер версии программы и завершить работу.

–v Подробный режим. Выдавать информацию о структурах в полном объеме.

–x Отображать все не-ASCII строки в шестнадцатеричном виде.

–xx

Вывести все строки в шестнадцатеричном виде.

strfile

strfile [*options*] *input_file* [*output_file*]

unstr [**–c** *delimiter*] *input_file*[.*ext*] [*output_file*]

strfile создает файл произвольного доступа для хранения строк. Исходный файл должен содержать группы строк, разделенные строкой, которая содержит единственный символ % (или любой другой заданный символ-разделитель). **strfile** создает конечный файл, содержащий структуру-заголовок и таблицу смещений каждой из групп строк, которые и позволяют организовать произвольный доступ к строкам. Конечный файл, если он не задан, получает имя *sourcefile.dat*. **unstr** обращает действие **strfile**, отображая строки, содержащиеся в исходном файле в том порядке, в каком они перечислены в заголовке файла данных. Конечным файлом по умолчанию считается стандартный вывод. Команда **unstr** также может применяться для глобальной замены символа-разделителя в исходном файле.

Параметры

Из нижеперечисленных параметров только **–c** может использоваться с **unstr**. Все прочие параметры относятся только к **strfile**.

–c *delimiter*

Произвести замену символа-разделителя на *delimiter*. Параметр можно применять как с **strfile**, так и с **unstr**.

strfile

–i Игнорировать регистр символов при сортировке строк.

–o Лексикографическая сортировка строк.

–r Случайный доступ к строкам.

–s Тихое выполнение; не выводить статистику при завершении.

–x Установить бит STR_ROTATED в поле заголовка *str_flags*.

strings

strings [*options*] *files*

Поиск в перечисленных файлах строк, состоящих по меньшей мере из четырех отображаемых символов и одного неотображаемого.

Параметры

–, –a, ––all

Сканировать объектные файлы полностью. По умолчанию выполняется сканирование инициализированных и загруженных разделов объектных файлов.

–f, ––print-file-name

Предварять каждую найденную строку именем файла.

–*min-len*, **–n** *min-len*, **––bytes=***min-len*

Вывести строки, содержащие по меньшей мере *min-len* символов.

–t *base*, **––radix=***base*

Предварять каждую найденную строку ее смещением от начала файла в формате, определяемом параметром *base*:

d Десятичный

o Восьмеричный

x Шестнадцатеричный

––target=*format*

Указать альтернативный формат объектного кода в качестве значения по умолчанию.

–o То же, что и **–t o**.

––help

Отобразить справку и завершить работу.

–v, ––version

Вывести информацию о версии программы и завершить работу.

strip	**strip** [*options*] *files* Удалить информацию об именах из объектных файлов *files*, уменьшив таким образом размеры файлов и сократив объем используемого дискового пространства. **Параметры** **–F** *format*, **––target=***format* Исходный файл имеет указанный формат. **–O** *format*, **––output-target=***format* Записать конечный файл в указанном формате. **–R** *section*, **––remove-section=***section* Удалить раздел *section*. **–s**, **––strip-all** Удалить все имена. **–S**, **–g**, **––strip-debug** Удалить отладочную информацию. **–x**, **––discard-all** Удалить неглобальные имена. **–X**, **––discard-locals** Удалить локальные имена, созданные компилятором. **–v**, **––verbose** Режим диагностики.
stty	**stty** [*options*] [*modes*] Установить режимы ввода/вывода терминала текущего стандартного устройства ввода. Команда без параметров отображает текущие настройки, отличающиеся от принимаемых по умолчанию (при вызове **stty sane**), при этом символом «^» обозначается клавиша <Ctrl>, а последовательность «^`» подразумевает пустое значение. Большинство режимов могут быть инвертированы добавлением символа «–» (заключенного в квадратные скобки). Соответствующие описания также взяты в квадратные скобки. Некоторые аргументы используют расширения, не соответствующие стандарту POSIX; такие аргументы отмечены символом «*».

Параметры

−a, −−all

Перечислить значения всех настроек.

−g Вывести значения в шестнадцатеричной системе счисления.

Режимы управления

n Установить скорость передачи терминала в *n* бод (например, 2400).

[−]clocal

[Включение] отключение управления модемом.

[−]cread

[Отключение] включение приемника.

cs*bits*

Установить размер символа равным *bits*, причем аргумент может принимать только значения 5, 6, 7 или 8.

[−]cstopb

[Один] два стоповых бита на символ.

[−]hup

[Не] обрывать соединение при последнем закрытии.

[−]hupcl

То же самое, что и **[−]hup**.

ispeed *n*

Установить скорость ввода терминала в *n* бод.

ospeed *n*

Установить скорость вывода терминала в *n* бод.

[−]parenb

[Отключить] включить генерацию четности и контроль по четности.

[−]parodd

Использовать контроль по [четности] нечетности.

[−]crtscts*

[Отключить] включить квитирование по RTS/CTS.

Режимы управления данными

Доступны следующие режимы управления данными посредством комбинирования флагов *ortsfl*, *ctsflow* и *rtsflow*:

stty

Настройки флагов	Режим управления данными
ortsfl rtsflow ctsflow	Включить однонаправленное управление данными
ortsfl rtsflow –ctsflow	Послать сигнал RTS при готовности к посылке данных
ortsfl –rtsflow ctsflow	Режим не имеет влияния
ortsfl –rtsflow –ctsflow	Включить двунаправленное управление данными
–ortsfl rtsflow ctsflow	Включить двунаправленное управление данными
–ortsfl rtsflow –ctsflow	Режим не имеет влияния
–ortsfl –rtsflow ctsflow	Прекращать передачу при сбросе CTS
–ortsfl –rtsflow –ctsflow	Отключить аппаратный контроль передачи данных

Режимы ввода

[–]brkint

[Не] выдавать сигнал INTR при прерывании (break).

[–]icrnl

[Не] преобразовывать CR в NL при вводе.

[–]ignbrk

[Не] игнорировать прерывание (break) при вводе.

[–]igncr

[Не] игнорировать CR при вводе.

[–]ignpar

[Не] игнорировать ошибки четности.

[–]inlcr

[Не] преобразовывать NL в CR при вводе.

[–]inpck

[Отключить] включить контроль по четности для ввода.

[–]istrip

[Не] усекать вводимые символы до 7 бит.

[–]iuclc*

[Не] преобразовывать верхний регистр символов в нижний при вводе.

[–]ixany*

Разрешить любому [XON] символу продолжать вывод.

[–]ixoff [–] tandem

[Не] посылать символы START/STOP, когда очередь почти пуста/полна.

[–]ixon

[Отключить] включить начальный контроль вывода по START/STOP.

[–]parmrk

[Не] сообщать об ошибках четности.

[–]imaxbel*

При переполнении буфера ввода [производить сброс данных из буфера] издавать звуковой сигнал и не сбрасывать буфер ввода.

Режимы вывода

bs*n* Выбрать стиль задержки при забое (0 или 1).

cr*n* Выбрать стиль задержки при переводе каретки (от 0 до 3).

ff*n* Выбрать стиль задержки для символов новой страницы (0 или 1).

nl*n* Выбрать стиль задержки для переводов строки (0 или 1).

tab*n*

Выбрать стиль задержки для горизонтальных табуляций (от 0 до 3).

vt*n* Выбрать стиль задержки для вертикальных табуляций (0 или 1).

[–]ocrnl*

[Не] преобразовывать CR в NL при выводе.

[–]ofdel*

Установка символа заполнения в [NULL] DEL.

[–]ofill*

Задержка вывода [по таймеру] символами заполнения.

[–]olcuc*

[Не] производить преобразование нижнего регистра в верхний при выводе.

[–]onlcr*

[Не] преобразовывать NL в CR-NL при выводе.

stty

[−]onlret*

При работе с терминалом NL выполняет [не выполняет] функцию CR.

[−]onocr*

[Не] выводить символы CR в нулевой строке.

[−]opost

[Не] выполнять последующую обработку вывода.

Локальные режимы

[−]echo

[Не] отображать каждый вводимый символ.

[−]echoe, [−]crterase

[Не] отображать символ ERASE как строку BS-пробел-BS.

[−]echok

[Не] отображать NL после символа KILL.

[−]echonl

[Не] отображать символ NL.

[−]icanon

[Отключить] включить канонический ввод (с обработкой ERASE, KILL, WERASE и RPRINT).

[−]iexten

[Отключить] включить расширенные функции для ввода данных.

[−]isig

[Отключить] включить отслеживание символов INTR, SUSPEND и QUIT.

[−]noflsh

[Включить] отключить сброс буфера после получения INTR или QUIT.

[−]tostop*

[Не] посылать сигнал SIGTTOU при посылке фоновым процессом сообщения на терминал.

[−]xcase*

[Не] изменять регистр символов для локального вывода.

[−]echoprt, [−]ptrerase*

При удалении символов отображать их, заключив в символы \ и /.

[–]echoctl, [–]ctlecho*

Не отображать управляющие (control) символы буквально. Использовать запись с символом «^» (например, **^Z**).

[–]echoke, [–]crtkill*

Удалять символы, как предписывается установками **echoprt** и **echoe** (установками **echoctl** и **echok** по умолчанию).

Настройки управления

ctrl-char c

Установить управляющий символ в *c*. Параметр *ctrl-char* может принимать значения: **dsusp** (сбросить ввод и послать stop), **eof**, **eol**, **eol2** (альтернативный конец строки), **erase**, **intr**, **lnext** (обновление строки), **quit**, **start**, **stop**, **susp**, **swtch** или **werase** (удаление предыдущего слова). Аргумент *c* может быть буквальным управляющим символом, символом в записи с «^» (например, **^Z**), в шестнадцатеричной записи (начинается с 0x), в восьмеричной записи (начинается с 0) либо в десятичной. Управляющий символ отключается значениями «^–» или **undef**.

min *n*

Установить минимальное количество символов, удовлетворяющих операции чтения до истечения установленного интервала времени; при работе в режиме **–icanon**.

time *n*

Установить количество десятых долей секунды, задающее длительность ожидания при чтении установленного количества символов (**min**); при работе в режиме **–icanon**.

line *i*

Установить режим работы линии в *i* (1–126).

Комбинированные режимы

cooked

То же, что и **–raw**.

[–]evenp, [–]parity

То же, что **[–]parenb** и **cs[8]7**.

[–]parity

То же, что **[–]parenb** и **cs[8]7**.

ek Сбросить символы ERASE и KILL в значения по умолчанию: <Ctrl>+<h> и <Ctrl>+<u>.

stty

[–]lcase

[Отключить] включить **xcase, iuclc** и **olcuc.**

[–]LCASE

Идентично **[–]lcase.**

[–]nl

[Отключить] включить **icrnl** и **onlcr. –nl** также отключает **inlcr, igncr, ocrnl** и **onlret, icrnl, onlcr.**

[–]oddp

Идентично **[–]parenb, [–]parodd** и **cs7[8].**

[–]raw

[Отключить] включить прямой ввод и вывод (без символов ERASE, KILL, INTR, QUIT, EOT, SWITCH или постобработки вывода).

sane

Сбросить все настройки в непротиворечивые значения.

[–]tabs*

[Расширять в пробелы] сохранять табуляцию при выводе.

[–]cbreak

Идентично **–icanon.**

[–]pass8

Идентично **–parenb –istrip cs8.**

[–]litout

Идентично **–parenb –istrip cs8.**

[–]decctlq*

Идентично **–ixany.**

crt Идентично **echoe echoctl echoke.**

dec

Идентично **echoe echoctl echoke –ixany.** Дополнительно установить значение INTERRUPT в **^C**, ERASE – в DEL, а KILL – в **^U.**

Специальные настройки

ispeed *speed*

Задать скорость ввода.

ospeed *speed*

Задать скорость вывода.

stty

rows *rows**

Задать количество строк.

cols *columns*, **columns** *columns**

Задать количество колонок.

size*

Отобразить текущие настройки строк и колонок.

line *discipline**

Установить режим работы линии.

speed

Отобразить скорость терминала.

su

su [*options*] [*user*] [*shell_args*]

Создать интерпретатор с эффективным идентификатором другого пользователя (*user*). Если имя пользователя не указано, создается интерпретатор привилегированного пользователя (т. е. происходит превращение в суперпользователя). Сеанс завершается символом *EOF*. Можно запускать интерпретатор с различными параметрами (*shell_args*), т. е. если выполняется **sh**, то можно задать **–c** *command* для выполнения команды *command* в **sh** или **–r** для создания ограниченного интерпретатора.

Параметры

–, –l, ––login

Выполнить полную регистрацию в системе (т. е. переход в окружение пользователя *user*).

–c *command*, **––command=***command*

Выполнить команду в интерпретаторе и немедленно завершить работу в нем. Если команда состоит более чем из одного слова, ее необходимо заключить в кавычки:

```
su -c 'find / -name \*.c -print' nobody
```

–f, ––fast

Выполнить интерпретатор с параметром **–f**. В **csh** и **tcsh** этот параметр подавляет чтение файла *.cshrc*. В **bash** параметр подавляет расширение имен по маске.

–m, –p, ––preserve-environment

Не сбрасывать переменные окружения.

su	**–s** *shell*, **––shell=***shell* Выполнить интерпретатор *shell*, а не заданный в */etc/passwd*, если не задано ограничение на запуск соответствующего интерпретатора. **––help** Отобразить справку и завершить работу. **––version** Вывести информацию о версии программы и завершить работу.
sum	**sum** [*options*] *file* Подсчет и отображение контрольной суммы файла и количества блоков (по 1 Кбайт). Команда полезна для проверки целостности при передаче данных. **Параметры** **–r** Режим по умолчанию. Использовать BSD-версию алгоритма подсчета контрольной суммы. **–s, ––sysv** Применить альтернативный алгоритм подсчета контрольной суммы (из System V). Размер блока – 512 байт. **––help** Отобразить информацию по использованию **sum** и завершить работу. **––version** Вывести номер версии **sum** и завершить работу.
swapdev	**swapdev** [*option*] [*image* [*swapdevice* [*offset*]]] Команда системного администрирования. Команда без аргументов отображает информацию по использованию устройства подкачки. Если задан только образ ядра (*image*), то отображается информация из этого образа. Для изменения информации необходимо указать новое устройство подкачки (*swapdevice*). Также возможно задание смещения (*offset*) для изменения информации в образе ядра. Обратите внимание, что **rdev –s** является синонимом **swapdev**. **Параметр** **–o** *offset* Синоним задания смещения в качестве аргумента.

swapoff **−a** | *device* . . .

swapoff

Команда системного администрирования. Сделать устройство (или устройства, список разделяется пробелами) недоступным для подкачки.

Параметр

−a Использовать устройства, помеченные в */etc/fstab* флагом **sw**. Эти устройства заменяют аргумент *device*.

swapon [*options*] *device*. . .

swapon

Команда системного администрирования. Сделать устройство *device* (или список устройств, элементы разделяются пробелами) доступным для подкачки.

Параметры

−a Использовать устройства, помеченные в */etc/fstab* флагом **sw**. Эти устройства заменяют аргумент *device*.

−p *priority*

Установить приоритет раздела подкачки. Разделы с более высоким приоритетом используются раньше прочих.

sync

sync

Команда системного администрирования. Сброс буферов файловой системы на диск. **sync** вызывает системную функцию **sync()**. Перед остановом системы вызов **sync** необходим для обеспечения целостности структуры файловой системы, поэтому **shutdown** автоматически вызывает **sync**. Выполнение **sync** может занять несколько секунд, так что систему необходимо приостановить на короткое время **(sleep)**, если вы собираетесь остановить систему **(halt)** или произвести перезагрузку **(reboot)** вручную. Следует помнить, что предпочтительным способом перезагрузки и останова является вызов **shutdown**, поскольку тогда синхронизация **(sync**-ing) и прочие вспомогательные действия выполняются автоматически.

sysklogd

sysklogd

Команда системного администрирования. Программа Linux, предоставляющая функциональность **syslogd** и работающая в точности так же, как и BSD-версия **syslogd**. Различия для пользователя незаметны. Тем не менее, **sysklogd** очень сильно

sysklogd

отличается построением и предоставляет несколько расширенный синтаксис вызова. Запускается, как **syslogd**. См. также **klogd**.

Параметры

–d Включить отладку.

–f *configfile*

Указать альтернативный файл настройки.

–h Передавать сообщения от удаленных узлов на пересылающие узлы.

–l *hostlist*

Указать список узлов, записи для которых должны вестись по именам узлов, а не полным доменным именам. При перечислении имена узлов разделяются двоеточиями.

–m *markinterval*

Указать количество минут между интервалами отметки.

–n Не переходить автоматически в режим фонового выполнения. Необходимо при запуске **syslogd** через **insit**.

–p *socket*

Направлять журнал в указанный *socket*, а не в */dev/log*.

–r Получать сообщения из сети, используя доменный сокет Интернета с сервисом **syslog**.

–s *domainlist*

Удалять перечисленные имена доменов *domainlist* перед записью в журнал. При перечислении имена доменов разделяются двоеточием **(:)**.

syslogd

syslogd

Команда TCP/IP. Ведение журнала системных сообщений в наборе файлов, описываемым файлом настройки */etc/syslog.conf*. Каждое сообщение записывается отдельной строкой. Сообщение может содержать код приоритета – число, заключенное в угловые скобки и расположенное в начале строки. Приоритеты определены в файле *<sys/syslog.h>*. **syslogd** читает доменный интернет-сокет, указанный в файле */etc/services*. Чтобы остановить работу **syslogd**, необходимо послать программе сигнал принудительного завершения.

systat [*options*] *host* **systat**

Команда системного администрирования. Запрос информации о статусе сети или системы на удаленном узле *host* посредством обращения к сервисам **netstat**, **systat** или **daytime**.

Параметры

–n, ––netstat

Опрашивать **netstat**-сервис узла.

–p *port*, **––port** *port*

Прямой запрос через указанный порт узла.

–s, ––systat

Опрашивать **systat**-сервис системы.

–t, ––time

Опрашивать **daytime**-сервис системы.

tac [*options*] [*file*] **tac**

Отображение содержимого файла в обратном порядке (команда получила свое имя от **cat**). Если не указано имя или представлен символ «–» (дефис), выполняется чтение со стандартного ввода. По умолчанию меняется порядок строк – последняя строка отображается первой.

Параметры

–b, ––before

Отображать разделитель (символ новой строки по умолчанию) перед отделяемым элементом.

–r, ––regex

Считать разделитель регулярным выражением.

–s *string*, **––separator=***string*

Задать альтернативный разделитель (по умолчанию это символ новой строки).

––help

Вывести справку и завершить работу.

––version

Отобразить информацию о версии программы и завершить работу.

tail

tail [*options*] [*file*]

Выдача последних десяти строк указанного файла (или стандартного ввода, если указано имя «–») на стандартный вывод.

Параметры

–*n*[*k*]

Начать печать с *n*-го от конца файла элемента. Аргумент *k* определяет элементы: **l** (строки, по умолчанию), **b** (блоки) и **c** (символы).

–*k* Аналогично –*n*, но используется значение счетчика по умолчанию, равное 10.

+*n*[*k*]

Аналогично –*n*, но начать с *n*-го элемента от начала файла.

+*k*

Аналогично –*k*, но считать от начала файла.

–c *num*{**bkm**}, **––bytes** *num*{**bkm**}

Вывести *num* последних байт файла. Можно указать альтернативный размер блока:

b 512 байт

k 1 Кбайт

m 1 Мбайт

–f Не завершать работу при достижении конца файла; продолжать чтение по мере роста файла. Работа завершается комбинацией <Ctrl>+<C>.

–n *num*, **––lines** *num*

Вывести *num* последних строк.

–q, **––quiet**, **––silent**

Подавить отображение имени файла.

––version

Вывести информацию о номере версии и завершить работу.

Примеры

Отобразить 20 последних строк, содержащих вхождение символов **.Ah**:

```
grep '\.Ah' file | tail -20
```

Показать 10 последних символов содержимого переменной **name**:

```
echo "$name" | tail -c
```

tail

Вывести два последних блока файла **bigfile**:

```
tail -2b bigfile
```

talk *person* [*ttyname*]

talk

Общение с другим пользователем. Имя *person* является либо регистрационным именем пользователя на той же самой машине, либо адресом вида *user@host* (имя пользователя на другом узле). Для общения с пользователем, имеющим в текущий момент несколько запущенных терминалов в системе, существует параметр *ttyname* – имя терминала. После установки связи обе стороны могут подавать реплики параллельно, при этом они будут отображаться в разных окнах (вернее, в разных половинах окна). Экран обновляется комбинацей <Ctrl>+<L>. Для завершения работы с **talk** введите символ прерывания; курсор переместится к последней строке экрана и терминал будет восстановлен.

talkd [*option*]

talkd

Команда TCP/IP. Сервер удаленного общения пользователей. **talkd** уведомляет пользователя, когда кто-то другой посылает запрос на разговор. Клиент **talk** начинает «рандеву», посылая серверу сообщение CTL_MSG, имеющее тип LOOK_UP. По этому сообщению сервер производит поиск приглашения для клиента в соответствующих таблицах. Если поиск не возвращает результатов, вызывающий посылает сообщение ANNOUNCE, которое передается вызываемому в качестве запроса беседы. Когда вызываемый отвечает, локальный сервер возвращает адрес «встречи» и устанавливается потоковое соединение, по которому и происходит общение.

Параметр

–d Запись отладочной информации в файл журнала **syslogd**.

tar [*options*] [*tarfile*] [*other-files*]

tar

Копирование файлов в архив или извлечение файлов из архива. Если какой-либо из файлов является каталогом, **tar** работает со всей структурой подкаталогов этого каталога. Необязательно предварять параметры символом «–» (хотя это и не возбраняется). Исключением из этого правила являются

tar

параметры с длинными именами (например, **--modification-time**). В этом случае точный синтаксис выглядит следующим образом:

```
tar --long-option -function-options files
```

К примеру:

```
tar --modification-time -xvf tarfile.tar
```

Функциональные параметры

Можно использовать только один из этих параметров, и он обязательно должен предшествовать всем остальным:

-c, --create

Создать новый архив.

-d, --compare

Сравнить файлы, хранимые в архиве *tarfile*, с файлами *other-files*. Сообщать обо всех различиях: отсутствующих файлах, различных размерах, различных атрибутах файла (таких как права доступа и время изменения).

-r, --append

Добавить файлы *other-files* в конец существующего архива.

-t, --list

Перечислить имена файлов *other-files*, если они содержатся в архиве (при отсутствии аргумента *other-files* отображаются имена всех файлов архива) .

-u, --update

Добавить файлы, которых нет в архиве, или измененные файлы.

-x, --extract, --get

Извлечь из архива файлы *other-files* (либо все, если аргумент опущен).

-A, --catenate, --concatenate

Добавить второй tar-файл в конец первого.

Параметры

n Выбор устройства *n*, причем *n* лежит в диапазоне 0–9999. Значение по умолчанию находится в файле */etc/default/tar*.

[*drive*][*density*]

Определить диск (0–7) и плотность информации устройства (**l**, **m** или **h**, что соответствует низкой, средней или высокой плотности).

––atime-preserve

Восстанавливать исходное время доступа к файлу при извлечении.

–b, **––block-size=***n*

Установить размер блока в n×512 байт.

––checkpoint

Перечислить имена обнаруженных каталогов.

––exclude=*file*

Исключить указанный файл из всех списков файлов.

–f *arch*, **––file=***filename*

Сохранить файлы в архиве *arch* или извлечь их из архива *arch*. Имейте в виду, что имя *filename* может иметь вид *hostname*:*filename* (*имя узла*: *имя файла*).

––force-local

Интерпретировать имена файлов вида *hostname:filename* как локальные.

–g, **––listed-incremental**

Создать инкрементную резервную копию нового стиля.

–h, **––dereference**

Разыменовывать символические ссылки.

–i, **––ignore-zeros**

Игнорировать блоки нулевого размера (т. е. символы EOF).

––ignore-failed-read

Игнорировать невозможность чтения файлов, которые требуется добавить в архив. По умолчанию при нахождении такого файла происходит завершение работы.

–k, **––keep-old-files**

При извлечении файлов не перезаписывать файлы с совпадающими именами. Выводить сообщение об ошибке.

–l, **––one-file-system**

Не включать в архив файлы, расположенные в других файловых системах.

tar

–m, ––modification-time

Не восстанавливать время изменения файлов; обновлять его временем извлечения из архива.

––null

Имена файлов, предоставляемые параметром **–T**, могут завершаться нулевым символом. Параметр имеет больший приоритет, чем **–C**.

––old, ––portability, ––preserve

Эквивалент одновременного указания параметров **–s** и **–p**.

–p, ––same-permissions, ––preserve-permissions

Восстанавливать информацию о правах доступа для извлекаемых файлов.

––remove-files

Удалять исходные файлы после помещения их в архив.

––rsh-command=*command*

Не соединяться с удаленным узлом посредством **rsh**; вместо этого выполнять указанную команду.

–s, ––same-order, ––preserve-order

Извлечь файлы из архива в том порядке, в каком они хранятся.

––totals

Суммарный объем в байтах.

––use-compress-program=*program*

Произвести сжатие либо распаковку архивов указанной программой.

v, ––verbose

Диагностика. Выводить имена файлов по мере добавления или извлечения.

–w, ––interactive

Ожидать подтверждение пользователя **(y)** перед выполнением каких-либо действий.

–z, ––gzip, ––ungzip

Сжимать файлы **gzip** перед архивацией либо распаковывать их командой **gunzip** перед извлечением.

–C, ––directory=*directory*

Перед началом работы перейти в указанный каталог.

–F, ––info-script, ––new-volume-script=*script*

Подразумевает включение параметра **–M** (многотомные архивы). Выполнять указанный сценарий для каждого файла.

–G, ––incremental

Создать инкрементную резервную копию старого стиля.

–K *file*, **––starting-file** *file*

Предписание **tar** начать работу с файла *file* в архиве.

–L, ––tape-length=*length*

Ограничить объем данных, записываемых на каждую ленту, значением *length* × 1024 байт.

–M, ––multivolume

Работа с многотомными архивами. С параметром **–c** создает многотомный архив.

–N *date*, **––after-date** *date*

Игнорировать файлы старше указанной даты.

–O, ––to-stdout

Отображать имена извлекаемых файлов на стандартный вывод.

–P, ––absolute-paths

Не удалять первый символ «/» из имен исходных файлов.

–R, ––record-number

Вывести количество записей в архиве.

–S, ––sparse

Подвергать короткие файлы более эффективной обработке.

–T *filename*, **––files-from** *filename*

Взять список имен файлов для обработки из файла *filename*.

–V *name*, **––label=***name*

Дать имя *name* текущему тому архива.

–W, ––verify

После создания архива проверить его корректность.

–X *file*, **––exclude** *file*

Взять список исключенных из обработки файлов из *file*.

tar

–Z, ––compress, ––uncompress

Сжимать файлы программой **compress** перед архивацией и распаковывать **uncompress** перед извлечением.

Примеры

Создать архив содержимого */bin* и */usr/bin* **(c)**, отобразить работу команды **(v)** и сохранить архив на ленте */dev/rmt0*:

```
tar cvf /dev/rmt0 /bin /usr/bin
```

Перечислить содержимое ленты в формате, подобном формату выдачи **ls –l**:

```
tar tvf /dev/rmt0
```

Извлечь каталог */bin*:

```
tar xvf /dev/rmt0 /bin
```

Создать архив текущего каталога и сохранить его в файле *backup.tar*:

```
tar cvf - `find . -print` > backup.tar
```

Параметр «–» является предписанием **tar** сохранять архив на стандартный вывод, который потом перенаправляется в файл.

tcpd

tcpd

Команда TCP/IP. Прием входящих запросов TCP/IP (например, запросов **telnet**, **ftp**, **finger**, **exec**, **rlogin**). Обеспечение сервисов проверки и ведения журналов, передача запроса соответствующему демону.

tcpdchk

tcpdchk [*options*]

Команда TCP/IP. Прочитать параметры настройки обертки TCP (TCP wrapper) из файлов */etc/hosts.allow* и */etc/hosts.deny*); перечислить проблемы, которые могут возникнуть в связи с текущими настройками, предложить возможные варианты решения.

Параметры

–a Включить список правил. Не требовать получения ключевого слова ALLOW перед открытием доступа к локальному узлу.

tcpdchk

–d Читать файлы *./hosts.allow* и *./hosts.deny* вместо */etc/hosts.allow* и */etc/hosts.deny*.

–i *conf-file*

Указать нахождение файла *inetd.conf* или *tlid.conf*. Это файлы, автоматически используемые **tcpdchk** при оценке файлов обертки TCP.

–v Режим диагностики.

---

tcpdmatch [*options*] *daemon client*

tcpdmatch

Команда TCP/IP. Предсказание ответа обертки TCP на определенный запрос. Необходимо указать, какому из демонов (*daemon*) адресован запрос (возможно использование синтаксиса вида *daemon@host* для запросов к удаленным машинам) и клиента (*client*), от которого исходит запрос (задание определенного пользователя, а также масок возможно при помощи синтаксиса *user@client*). Действия обертки TCP определяются на основе содержимого файлов */etc/hosts.allow* и */etc/hosts.deny*.

Параметры

–d Читать файлы *./hosts.allow* и *./hosts.deny* вместо */etc/hosts.allow* и */etc/hosts.deny*.

–i *conf-file*

Указать нахождение файла *inetd.conf* или *tlid.conf*. Это файлы, автоматически используемые **tcpdmatch** при определении файлов обертки TCP.

---

tcsh [*options*] [*files* [*arguments*]]

tcsh

Расширенная версия командного интерпретатора C shell. Более подробно интерпретатор описан в главе 8 «Интерпретаторы csh и tcsh».

---

tee [*options*] *files*

tee

Вывод другой команды посылается как на стандартный вывод, так и в файлы *files* (аналогия – развилка дороги).

Параметры

–a, ––append

Добавлять данные в файлы, не перезаписывая их.

tee

–i, ––ignore-interrupts

Игнорировать сигналы прерываний.

––help

Отобразить информацию по использованию и завершить работу.

––version

Вывести информацию о версии и завершить работу.

Пример

`ls -l | tee savefile` *Просмотреть листинг и сохранить его*

telinit

telinit [*option*] [*runlevel*]

Команда системного администрирования. Предписание **init** изменить уровень выполнения (*runlevel*). **telinit** является просто ссылкой на **init** – прародителя всех процессов.

Параметр

–t *seconds*

Послать сигнал SIGKILL через *seconds* секунд после посылки SIGTERM. По умолчанию – через 20 секунд.

Уровни выполнения

В зависимости от дистибутива уровни выполнения могут различаться, тем не менее, нижеперечисленные являются стандартными:

0 Останов системы.

1, s, S

Однопользовательский режим.

6 Перезагрузка системы.

a, b, c

Обрабатывать только записи */etc/inittab*, отмеченные уровнями **a**, **b** и **c**.

q, Q

Повторное чтение */etc/inittab*.

Информация об уровнях выполнения системы содержится в файле */etc/inittab*.

telnet [*options*] [*host* [*port*]]

telnet

Доступ к удаленной системе. **telnet** является интерфейсом пользователя для протокола связи между узлами Telnet. Если при запуске **telnet** не указывается узел (*host*), начинается работа в командном режиме, что отражается приглашением **telnet>**, и принимаются к выполнению команды, перечисленные ниже после параметров. При запуске с аргументами **telnet** выполняет команду **open** (см. соответствующее описание) с этими аргументами. Аргумент *host* является именем узла, а *port* – это номер порта (по умолчанию – порт Telnet).

Параметры

–a Автоматическая регистрация в удаленной системе.

–d Включение отладки уровня сокета (socket-level).

–e [*escape_char*]

Установить начальный escape-символ **telnet** в *escape_char*. Если символ не указан, в сеансе работы не будет предварительно заданного escape-символа.

–l *user*

При подключении к удаленной системе, понимающей установку ENVIRON, послать имя пользователя (*user*) для инициализации переменной USER.

–n *tracefile*

Открыть файл *tracefile* для записи информации о маршрутах.

–r Эмуляция **rlogin**: escape-символом по умолчанию является тильда; символ точки, предваренный escape-символом, вызывает рассоединение **telnet** с удаленным узлом, а **^Z**, предваренный тильдой, приостанавливает выполнение **telnet**. По символу «**]**» (escape-символ **telnet** по умолчанию) выдается обычное приглашение **telnet**. Эти команды распознаются только в начале строки.

–8 8-битные операции.

–E Отключить действие escape-символа.

–L Задать 8-битный путь для данных вывода.

–S *tos*

Установить IP-параметр TOS (type-of-service, тип сервиса) для Telnet-соединения в значение *tos*.

telnet

Команды

CTRL-Z

Приостановить выполнение **telnet**.

![*command***]**

Выполнить одну команду в интерпретаторе локальной системы. Если команда опущена, вызывается диалоговый интерпретатор.

?[*command***]**

Получение справки о команде. «**?**» без аргументов приводит к выдаче конспекта справки. Если команда задана, отображается справка только по этой команде.

close

Завершить сеанс Telnet и вернуться в командный режим.

display *argument* . . .

Отобразить значения всех или некоторых переменных **set** и **toggle**.

environ [*arguments* **[. . .]]**

Управление переменными, значения которых могут быть посланы с параметром TELNET ENVIRON. Допустимые аргументы **environ**:

? Получение справки по команде **environ**.

define *variable value*

Определить переменную и ее значение.

undefine *variable*

Удалить определение переменной из списка переменных окружения.

export *variable*

Сделать значение переменной видимым для удаленной системы.

unexport *variable*

Сделать значение переменной невидимым для удаленной системы без специального запроса.

list

Перечислить значения существующих переменных.

logout

Если удаленный узел поддерживает команду **logout**, завершить сеанс **telnet**.

mode [*type*]

В зависимости от состояния сеанса Telnet аргумент *type* может быть одним из следующих параметров:

? Вывести справку по команде **mode**.

character

Отключить режим TELNET LINEMODE либо, если этот режим не поддерживается удаленной системой, перейти в «посимвольный» режим.

[–]edit

Попытаться [отключить] включить режим EDIT параметра TELNET LINEMODE.

[–]isig

Попытаться [отключить] включить режим TRAPSIG параметра TELNET LINEMODE.

line

Включить режим LINEMODE либо, если этот режим не поддерживается удаленной системой, попытаться вернуться к «построчному» режиму.

[–]softtabs

Попытаться [отключить] включить режим SOFT_TAB параметра LINEMODE.

[–]litecho

[Отключить] включить режим LIT_ECHO.

open [**–l** *user*] *host* [*port*]

Открыть соединение с указанным узлом. Если не задан номер порта, производить соединение через стандартный порт Telnet.

quit

Закрыть все сеансы Telnet и завершить работу.

status

Отобразить текущее состояние **telnet**, включая систему, с которой установлено соединение, а также текущий режим.

send *arguments*

Передача одной или нескольких специальных последовательностей удаленной системе. Ниже перечислены возможные аргументы:

? Вывести справку по команде **send**.

abort

Послать последовательность Telnet ABORT.

telnet

ao

Послать последовательность Telnet AO, предписывающую удаленной системе сбросить весь вывод на терминал пользователя.

ayt

Послать последовательность Telnet AYT (Are You There).

brk

Послать последовательность Telnet BRK (Break).

do *cmd*

dont *cmd*

will *cmd*

wont *cmd*

Послать предписание Telnet DO *cmd*, где *cmd* является числом от 0 до 255 или именем определенной команды **telnet**. Если аргумент *cmd* имеет значение **?** или **help**, отображается справка (включающая список имен команд).

ec Послать Telnet EC (Erase Character) – предписание удаленной системе стереть последний введенный символ.

el Послать Telnet EL (Erase Line) – предписание удаленной системе стереть последнюю введенную строку.

eof

Послать предписание Telnet EOF (End Of File).

eor

Послать предписание Telnet EOR (End Of Record).

escape

Послать текущий escape-символ Telnet (изначально –^).

ga Послать предписание Telnet GA (Go Ahead).

getstatus

Если удаленная система поддерживает команду Telnet STATUS, **getstatus** посылает запрос на получение текущего состояния параметров сервера.

ip Послать Telnet IP (Interrupt Process) – предписание удаленной системе прекратить выполнение текущего задания.

nop

Послать предписание Telnet NOP (No OPeration).

susp

Послать предписание Telnet SUSP (SUSpend Process).

synch

Послать Telnet SYNCH – предписание удаленной системе удалить весь ввод, который не был прочитан.

set *argument value*

unset *argument value*

Установка любой из переменных **telnet** в указанное значение либо в **TRUE**. Специальное значение **off** отключает функции, связанные с переменной. **unset** отключает любые указанные функции. Значения переменных могут быть получены при помощи команды **display**. Доступные переменные:

? Отобразить доступные команды **set** и **unset**.

ayt Если **telnet** находится в режиме LOCALCHARS, указанный символ становится альтернативным символом AYT.

echo

Значение клавиатурного сочетания (изначально – ^E), которое в построчном режиме переключает локальное отображение вводимых символов.

eof

Если **telnet** находится в режиме LINEMODE или в старом построчном режиме, ввод этого символа в качестве первого символа в строке приведет к немедленной посылке его удаленной системе.

erase

Если **telnet** находится в режиме LOCALCHARS и действует в посимвольном режиме, то при вводе этого символа удаленной системе будет посылаться предписание TELNET EC.

escape

Это escape-символ Telnet (изначально – ^[), который инициирует командный режим Telnet в ходе сеанса работы с удаленной системой.

flushoutput

Если **telnet** находится в режиме LOCALCHARS, ввод символа **flushoutput** приведет к посылке удаленной системе предписания Telnet AO.

forw1, forw2

Если Telnet находится в режиме LOCALCHARS, эти символы считаются альтернативным символом конца строки.

telnet

interrupt

Если Telnet AO работает в режиме LOCALCHARS, ввод этого символа приводит к посылке удаленной системе предписания Telnet IP.

kill

Если Telnet IP находится в режиме LOCALCHARS и работа происходит посимвольно, то ввод этого символа приводит к посылке удаленной системе предписания TELNET EL.

lnext

Если Telnet EL работает в режиме LINEMODE или старом «построчном» режиме, этот символ интерпретируется терминалом как **lnext**.

quit

Если Telnet EL работает в режиме LOCALCHARS, ввод символа **quit** приводит к посылке удаленной системе предписания Telnet BRK.

reprint

Если Telnet BRK работает в режиме LINEMODE или в старом «построчном» режиме, этот символ интерпретируется терминалом как **reprint**.

rlogin

Включение режима **rlogin**. Действует идентично параметру командной строки **–r**.

start

Если включен параметр Telnet TOGGLE-FLOW-CONTROL, этот символ интерпретируется терминалом как **start**.

stop

Если включен параметр Telnet TOGGLE-FLOW-CONTROL, этот символ интерпретируется терминалом как **stop**.

susp

Если Telnet работает в режиме LOCALCHARS либо включен режим LINEMODE, ввод этого символа приводит к посылке удаленной системе предписания Telnet SUSP.

tracefile

Файл, в который записывается вывод команды **netdata**.

worderase

Если Telnet BRK находится в режиме LINEMODE или старом «построчном» режиме, то этот символ интерпретируется терминалом как символ **worderase**. Значение по умолчанию совпадает со значением по умолчанию для терминала.

slc [*state*]

Установить состояние специальных символов для включенного режима Telnet LINEMODE.

? Вывести справку по команде **slc**.

check

Проверить текущие установки специальных символов. При нахождении несоответствий преобразовывать локальные символы таким образом, чтобы они соответствовали настройкам удаленной системы.

export

Переключиться на локальные значения по умолчанию для специальных символов.

import

Переключиться на значения специальных символов, принятые по умолчанию на удаленной системе.

toggle *arguments* [. . .]

Переключение различных флагов, управляющих реакцией Telnet на события. Флаги могут явно устанавливаться в **true** или **false** при помощи ранее описанных команд **set** и **unset**. Существующие флаги:

? Отобразить доступные команды **toggle**.

autoflush

Если и **autoflush** и LOCALCHARS имеют значение **true**, то при получении символов **ao** или **quit** Telnet отказывается отображать какие-либо данные на терминале пользователя до тех пор, пока удаленная система не подтвердит, что эти предписания Telnet были обработаны.

autosynch

Если и **autosynch** и LOCALCHARS имеют значение **true**, то при вводе символов **intr** или **quit** предписания дополняются Telnet SYNCH. Начальное значение для этого флага **toggle** – **false**.

telnet

binary

Включение или отключение режима Telnet BINARY для ввода и вывода.

inbinary

Включение или отключение режима Telnet BINARY для ввода.

outbinary

Включение или отключение режима Telnet BINARY для вывода.

crlf

Если этот флаг имеет значение **true**, возвраты каретки посылаются в виде **CR-LF**. В противном случае возвраты каретки посылаются как **CR-NUL**. Начальное значение флага – **false**.

crmod

Управление режимом возврата каретки. Начальное значение – **false**.

debug

Управление отладкой уровня сокета. Начальное значение – **false**.

localchars

Если флаг имеет значение **true**, то символы **flush**, **interrupt**, **quit**, **erase** и **kill** распознаются локальной системой, а затем переводятся в соответствующие управляющие предписания Telnet. Начальное значение флага – **true**.

netdata

Управление отображением сетевых данных. Начальное значение – **false**.

options

Управление отображением некоторых из внутренних процессов протокола, имеющих отношение к настройкам Telnet. Начальное значение флага – **false**.

prettydump

Если флаги **netdata** и **prettydump** установлены (имеют значение **true**), то вывод команды **netdata** преобразуется в более удобный для пользователя формат. Символы разделяются пробелами, а escape-последовательности Telnet предваряются звездочками.

telnet

skiprc
Флаг отвечает за обработку файла ~/*.telnetrc*. Начальное значение флага – **false**, т. е. файл обрабатывается.

termdata
Управление выводом шестнадцатеричных данных на терминал. Начальное значение флага – **false**.

z Приостановить работу **telnet**; команда доступна только в интерпретаторе **csh**.

telnetd

telnetd [*options*]

Команда TCP/IP. Сервер протокола Telnet. **telnetd** запускается интернет-сервером по запросам на соединение с портом Telnet (по умолчанию это порт 23). Создает псевдотерминальное устройство для клиента, для чего инициирует процесс регистрации, работающий в связке с подчиненной частью псевдотерминала, обеспечивающей потоки **stdin**, **stdout** и **stderr**. **telnetd** работает с основной частью псевдотерминала, обеспечивая протокол Telnet и обмен символами между удаленным клиентом и процессом регистрации.

Параметры

–debug [*port*]
Автономное (не посредством **inetd**) выполнение **telnetd**. Может быть указан альтернативный TCP-порт для работы **telnetd**.

–D *modifier*(*s*)
Режим отладки, в котором **telnet** выводит информацию о текущем соединении, благодаря чему пользователь может узнать, что делает **telnet**. В режиме отладки доступны следующие модификаторы (*modifiers*):

exercise
Реализация пока отсутствует.

netdata
Отобразить поток данных, получаемый **telnetd**.

options
Вывести информацию об обмене параметрами сеанса Telnet.

ptydata
Отобразить данные, которые записываются в устройство псевдотерминала.

report
Более подробный вариант **options**.

test

test *expression*
[*expression*]

В большинстве интерпретаторов существует идентичная встроенная команда.

Вычислить значение выражения и, если результат равен **true**, вернуть нулевой код завершения, иначе вернуть ненулевой код. В сценариях интерпретатора может использоваться альтернативная форма записи команды – [*expression*]. Эта команда в основном применяется в сценариях для записи условных операторов.

Операции с файлами

Синтаксис для всех нижеперечисленных параметров: **test** *option file*. Если указанный файл не существует, возвращается значение **false** (ложь). В противном случае файл проверяется в соответствии с параметром.

–b Является ли файл специальным файлом блочного устройства?

–c Является ли файл специальным файлом символьного устройства?

–d Является ли файл каталогом?

–e Существует ли файл?

–f Является ли файл обычным?

–g Установлен ли бит смены идентификатора группы?

–k Установлен ли бит «липкости» файла?

–L Является ли файл символической ссылкой?

–p Является ли файл именованным каналом?

–r Доступен ли файл для чтения текущему пользователю?

–s Является ли файл непустым?

–S Является ли файл сокетом (socket)?

–t [*file-descriptor*]
Связан ли файл с указанным файловым дескриптором (по умолчанию это стандартный вывод, чему соответствует значение 1) с каким-либо терминалом?

–u Установлен ли бит смены идентификатора пользователя?

–w Доступен ли файл для изменения текущему пользователю?

–x Является ли файл исполняемым?

–O Является ли эффективный идентификатор пользователя процесса владельцем файла?

–G Является ли эффективный идентификатор группы процесса владельцем файла?

Сравнения файлов

Синтаксис для записи сравнения файлов следующий: **test** *file1 option file2*. Если задан единственный аргумент и его длина больше нуля, возвращается **true** (истина).

–nt Является ли *file1* более новым, чем *file2*? Выполняется проверка по времени изменения, а не создания файлов.

–ot Является ли *file1* более старым, чем *file2*? Выполняется проверка по времени изменения, а не создания файлов.

–ef Идентичны ли числа устройств и единиц inode для файлов?

Строковые операции

Синтаксис для строковых операций: **test** *option string*.

–z Равна ли нулю длина строки?

–n Содержит ли строка хотя бы один символ?

= *string*

Проверка равенства двух строк.

!= *string*

Проверка неравенства двух строк.

Проверка выражений

Обратите внимание, что выражение может быть комбинацией вышеописанных проверок.

! *expression*

Ложно ли выражение?

expression **–a** *expression*

Оба ли выражения истинны?

expression **–o** *expression*

Истинно ли хотя бы одно из выражений?

Целочисленные сравнения

Синтаксис целочисленных сравнений: **test** *integer1 option integer2*. В качестве целого числа может быть задано значение **–l** *string* (длина строки).

test	**–eq** Числа равны. **–ne** Числа не равны. **–lt** *integer1* меньше, чем *integer2*. **–le** *integer1* меньше либо равно *integer2*. **–gt** *integer1* больше, чем *integer2*. **–ge** *integer1* больше либо равно *integer2*.
tftp	**tftp** [*host* [*port*]] Пользовательский интерфейс к протоколу TFTP (Trivial File Transfer Protocol), позволяющему передавать файлы между удаленными машинами. Может быть задан удаленный узел (*host*); в этом случае он становится узлом по умолчанию для последующих операций. **Команды** При запуске **tftp** выдается приглашение: `tftp>` и воспринимаются следующие команды: **?** [*command-name*. . .] Вывести справочную информацию. **ascii** Сокращение для перехода в режим ASCII (**mode ASCII**). **binary** Сокращение для перехода в двоичный режим (**mode binary**). **connect** *hostname* [*port*] Установить имя узла (*hostname*) и, при необходимости, порт для передачи файлов. **get** *filename* **get** *remotename localname* **get** *filename1 filename2 filename3 . . . filenameN* Получение файла или набора файлов из ранее указанного удаленного источника. **mode** *transfer-mode* Установить режим передачи данных. Режим может иметь значение **ASCII** или **binary** (двоичный). По умолчанию принимается **ASCII**.

tftp

put *filename*

put *localfile remotefile*

put *filename1 filename2 . . . filenameN remote-directory*

Передать файл или набор файлов в указанный файл или каталог удаленного узла.

quit

Завершить работу с **tftp**.

rexmt *retransmission-timeout*

Установить интервал (в секундах), в течение которого предпринимаются попытки повторной передачи пакета.

status

Вывести информацию о состоянии: установлено ли **tftp**-соединение с удаленным узлом (т. е. задан ли узел для последующих операций), текущий режим передачи файлов, включены ли режимы диагностики и отслеживания маршрута, а также значения **retransmission timeout** и **total transmission timeout**.

timeout *total-transmission-timeout*

Установить интервал (в секундах), в течение которого предпринимаются попытки передачи файла.

trace

Управление отслеживанием маршрутов пакетов.

verbose

Управление режимом диагностики.

tftpd

tftpd [*homedir*]

Команда TCP/IP. Сервер протокола Trivial File Transfer Protocol. **tftpd** обычно запускается сервером **inetd** и работает с портом, указанным в описании интернет-сервиса **tftp** (в файле */etc/inetd.conf*). По умолчанию запись для **tftp** в */etc/inetd.conf* закомментирована, и символ комментария необходимо удалить, чтобы **tftpd** включился в работу. Перед тем как ответить на запрос, сервер пытается сменить текущий каталог на *homedir*; значение по умолчанию – **tftpboot**.

tload

tload [*options*] [*tty*]

Отображение средней загрузки системы в виде графика. Если задан терминал (*tty*), то отображать на нем.

tload	**Параметры** **–d** *delay* Задержка (в секундах) между обновлениями. **–s** *scale* Указать масштаб (количество символов между отметками координатных линеек). Чем меньше число, тем больше масштаб.
top	**top** [*options*] Отображение постоянно обновляемой информации о запущенных задачах, наиболее интенсивно использующих процессор. Дескрипторы полей см. в описании команды **ps**. **Параметры** **–b** Работа в пакетном режиме; программа не реагирует на клавиатурные команды. Полезно при необходимости выдать результат работы **top** в файл или на ввод другой команды. **–c** Отображать командную строку задания, а не только имя команды. **–d** *delay* Задать временной интервал между обновлениями информации. **–i** Подавить отображение информации для бездействующих процессов и процессов-зомби. **–n** *num* Завершить работу после *num* обновлений информации. **–p** *pid* Отображать информацию только для перечисленных процессов (по идентификаторам). **–q** Обновление без задержки. Привилегированный пользователь может выполнять эту команду в режиме наивысшего приоритета работы. **–s** Безопасный режим. Отключаются некоторые (опасные) диалоговые команды. **–S** Режим накопления. Вывести суммарное время использования процессорных ресурсов для каждого процесса, включая в подсчет отработавшие порожденные процессы.

Диалоговые команды

пробел

Обновить данные немедленно.

c Отобразить полную командную строку или имя команды.

f, F Добавить или удалить поля в отображение.

h, ? Отобразить справку по командам, а также состояния безопасного и накопительного режимов.

k Ввести идентификатор процесса, который необходимо завершить, и послать этому процессу сигнал (по умолчанию – 15).

i Управление отображением бездействующих процессов и процессов-зомби.

l Управление отображением средних показателей загрузки и информации о времени непрерывной работы системы.

m Управление отображением информации об использовании памяти.

n, #

Количество отображаемых процессов. Если указано значение 0, то отображается столько, сколько умещается на экранной странице (режим по умолчанию).

o, O

Изменение порядка отображения полей.

q Завершение работы.

r Применить к процессу команду **renice**. Идентификатор процесса и значение **renice** вводятся пользователем. В безопасном режиме команда недоступна.

s Изменить задержку между обновлениями. Новое значение задержки в секундах вводится пользователем. В безопасном режиме команда недоступна.

t Управление отображением процессов и информации об использовании процессора.

A Сортировать задания по старшинству, в порядке увеличения возраста (времени работы).

^L Обновить экран.

M Сортировать задания по размеру памяти, постоянно выделенной под каждое из них.

N Сортировать задания по численным идентификаторам процессов.

top	**P** Сортировать процессы по использованию процессорных ресурсов (режим по умолчанию). **S** Управление режимом накопления (см. параметр **–S**). **T** Сортировать процессы по времени/суммарному времени выполнения. **W** Записать текущие настройки в файл ~/*.toprc*. Это предпочтительный способ записи настроек **top**.
touch	**touch** [*options*] *files* Обновить время доступа и изменения для указанных файлов (*files*) текущим значением. С помощью **touch** можно заставить команды работать с файлами определенным образом. В частности, работа **make** или **find** может зависеть от времени доступа и изменения файла. Если файл не существует, **touch** создает его, при этом размер файла равен нулю. **Параметры** **–a, ––time=atime, ––time=access, ––time=use** Обновить только время доступа к файлу. **–c, ––no-create** Не создавать несуществующие файлы. **–d** *time*, **––date** *time* Заменить текущее значение времени на указанное (*time*). Время может задаваться в различных форматах и содержать названия месяцев, информацию о часовых поясах, строки a.m. и p.m. и т. д. **–m, ––time=mtime, ––time=modify** Обновить только время изменения файла. **–r** *file*, **––reference** *file* Изменить информацию о времени на информацию для файла *file*. **–t** *time* Использовать указанное время вместо текущего. Аргумент должен иметь формат [[*cc*]*yy*]*mmddhhmm*[.*ss*], указывая месяц, дату, часы, минуты, при необходимости век, год, а также секунды. **––help** Вывести справку по использованию программы. **––version** Вывести информацию о номере версии программы.

tr [*options*] [*string1* [*string2*]]

tr

Преобразование символов – стандартный ввод копируется на стандартный вывод после замены символов строки *string1* на символы строки *string2* или удаления символов из *string1*.

Параметры

–c, ––complement

Дополнить символы строки *string1* с учетом ASCII-диапазона 001–377.

–d, ––delete

Удалить символы строки *string1* из вывода.

–s, ––squeeze-repeats

Сократить последовательности повторяющихся символов в строке *string2* до одного символа.

–t, ––truncate-set1

Перед преобразованием произвести усечение *string1* до длины *string2*.

––help

Отобразить справку и завершить работу.

––version

Вывести информацию о версии программы и завершить работу.

Специальные символы

Квадратные скобки [ ] необходимо указывать, если они есть в описании.

\a **^G** (сигнал)

\b **^H** (забой)

\f **^L** (новая страница)

\n **^J** (новая строка)

\r **^M** (возврат каретки)

\t **^I** (табуляция)

\v **^K** (вертикальная табуляция)

nnn

Символ с восьмеричным ASCII-кодом *nnn*.

**** Символ «\».

tr

char1–char2

Все символы диапазона *char1–char2*. Если *char1* не предшествует *char2* в наборе символов, это вызывает ошибку.

[*char1–char2*]

То же, что и *char1–char2*, если конструкция [*char1–char2*] используется в обеих строках.

[*char**]

Буквальное включение содержимого скобок. Количество вхождений символа расширяется до длины строки *string1*.

[*char*number*]

Буквальное включение содержимого скобок. Количество вхождений символа расширяется до указанного. Например, **[x*4]** – это **xxxx**.

[:*class*:]

Буквальное включение содержимого скобок. Расширение до полного набора символов указанного класса.

alnum Буквы и цифры.

alpha Буквы.

blank Пробелы.

cntrl Управляющие символы.

digit Цифры.

graph Все отображаемые символы кроме пробела.

lower Символы нижнего регистра.

print Все отображаемые символы.

punct Символы пунктуации.

space Пробелы и символы табуляции (горизонтальной или вертикальной).

upper Символы верхнего регистра.

xdigit Шестнадцатеричные цифры.

[=*char*=]

Класс символов, к которому принадлежит символ *char*.

Примеры

В содержимом файла изменить символы верхнего регистра на символы нижнего:

```
cat file | tr '[A-Z]' '[a-z]'
```

tr

Преобразовать пробелы в символы новой строки (код ASCII = 012):

```
tr ' ' '\012' < file
```

Удалить пустые строки из файла **file** и сохранить результат в файле **new.file** (можно также использовать **011**, чтобы заменить последовательные табуляторы одним):

```
cat file | tr -s "" "\012" > new.file
```

Удалить двоеточия из файла **file**; результат сохранить в файле **new.file**:

```
tr -d : < file > new.file
```

traceroute

traceroute [*options*] *host* [*packetsize*]

Команда TCP/IP. Отслеживание маршрута пакетов, посылаемых определенному узлу сети (*host*). Для выполнения поставленной задачи **traceroute** отправляет пробные UDP-пакеты с небольшим временем жизни TTL (time-to-live), а затем ожидает получения ICMP-ответа «time exceeded» (время истекло) от шлюза. *host* может задаваться именем конечного узла или IP-адресом. Аргумент *packetsize* определяет размер посылаемых пробных пакетов в байтах. По умолчанию размер равен 38.

Параметры

–d Отключение отладки уровня сокета (socket).

–g *addr*

Включение параметра IP LSRR (Loose Source Record Route) в дополнение к проверке по TTL. Позволяет выяснить, по какому маршруту идут пакеты на IP-адрес *addr* от конечного узла.

–l Включать значение time-to-live для каждого из полученных пакетов.

–m *max_ttl*

Установить максимальное время жизни исходящих пакетов в *max_ttl* транзитных участков (hops). По умолчанию равно 30.

–n Показать численные, а не буквенные адреса узлов на транзитных участках. Полезно при нерабочем сервере доменных имен.

traceroute	**–p** *port* Установить основной адрес UDP-порта, который используется для отправки пробных пакетов. По умолчанию равно 33 434 (десятичное). **–q** *n* Установить количество пробных пакетов для каждого значения time-to-live в *n*. По умолчанию равно 3. **–r** Посылать пакеты напрямую адресату в смежной сети в обход обычных таблиц маршрутизации. **–s** *src_addr* Использовать указанный адрес в качестве исходного IP-адреса, добавляемого во все исходящие пробные пакеты. **–t** *tos* Установить значение типа сервиса (type-of-service) в исходящих пакетах (по умолчанию – 0). Значение должно быть десятичным целым из диапазона 0–255. **–v** Диагностика – перечислить получаемые ICMP-пакеты (кроме TIME_EXCEEDED и PORT_UNREACHABLE). **–w** *wait* Установить время ожидания реакции на отправку пробного пакета в *wait* секунд (по умолчанию равно 3).
troff	**troff** См. **groff.**
true	**true** Команда, возвращающая код успешного завершения (нулевой). См. также **false**.
tune2fs	**tune2fs** [*options*] *device* Команда системного администрирования. Настройка различных параметров работы файловой системы Linux ext2fs. Необходимо указать устройство, на котором находится файловая система; в момент изменения параметров файловая система не должна быть подмонтирована в режиме чтения/записи.

Параметры

–c *mount-counts*

Указать максимально допустимое количество монтирований между проверками файловой системы.

–c *behavior*

Задать реакцию ядра на наличие ошибок. Аргумент *behavior* может принимать следующие значения:

continue

Продолжать работу.

remount-ro

Перемонтировать файловую систему-нарушительницу в режиме «только для чтения».

panic

Паника: ядро прекращает работу.

–g *group*

Разрешить использование зарезервированного пространства указанной группе *group* (имя группы или идентификатор).

–i *interval*[**d** | **w** | **m**]

Задать максимально допустимый интервал между проверками файловой системы. Единицами времени могут быть дни (**d**), недели (**w**) или месяцы (**m**). Значение интервала 0 указывает на то, что проверки не будут зависеть от времени.

–l Отобразить содержимое суперблока.

–m *percentage*

Указать (в процентах) количество зарезервированных (только для привилегированного пользователя) блоков.

–r *num*

Указать количество блоков, зарезервированных только для привилегированного пользователя.

–u *user*

Разрешить пользователю *user* (имя или идентификатор) доступ к зарезервированным блокам.

tunelp

tunelp *device* [*options*]

Команда системного администрирования. Настройка параметров работы принтера. Команда без параметров выведет информацию по устройству (устройствам).

Параметры

–a [on | off]

Позволяет указать, надо завершать работу при ошибке принтера или нет. По умолчанию работа не завершается.

–c *n*

Количество попыток повторной посылки символа в устройство. (По умолчанию равно 250). После обнуления счетчика сделать паузу перед повторением попыток.

–i *irq*

Назначить указанное прерывание *irq* для работы с параллельным портом. Игнорируются параметры **–t** и **–c**. При значении 0 восстанавливается непрерываемая работа (polling) с устройством.

–o [on | off]

Переключение завершения работы, если устройство недоступно или нет бумаги в устройстве.

–q [on | off]

Переключение отображения текущих настроек IRQ.

–r Сброс порта.

–s Отображение текущего состояния принтера.

–t *time*

Длительность паузы (в jiffies) между повторными посылками символа в устройство. Мгновение (jiffy) определяется либо как один квант системных часов, либо один цикл времени AC. Приблизительно 0,01 секунды.

–w *time*

Указать время задержки (в мгновениях) перед повторной посылкой тактового сигнала.

–C [on | off]

Включение или отключение повышенной «внимательности» демона печати при выявлении ошибок печатающего устройства.

ul [*options*] [*names*] **ul**

Преобразовать символы подчеркивания в подчеркивания символов. Корректная обработка операции зависит от конкретного типа терминала. Некоторые из терминалов не поддерживают подчеркивание символов.

Параметры

–i Преобразовать символ «–» (дефис) в отдельной строке в подчеркивание символов вместо преобразования символа _ (символ подчеркивания).

–t *terminal-type*

Указать тип терминала. По умолчанию тип терминала – TERM.

umount [*options*] [*special-device*/*directory*] **umount**

Команда системного администрирования. Размонтировать файловую систему. **umount** уведомляет систему, что удаляемая файловая структура, которая была подмонтирована на устройстве *special-device*, будет впредь недоступна. В качестве аргумента **umount** может быть указана точка монтирования. Процессы ввода/вывода для файловой системы завершаются, и файловая структура отмечается как корректная.

Параметры

–a Размонтировать все файловые системы, перечисленные в /*etc*/*mtab*.

–n Размонтировать, но не записывать изменения в /*etc*/*mtab*.

–t *type*

Размонтировать только файловые системы указанного типа.

uname [*options*] **uname**

Вывод информации об операционной системе и архитектуре машины. Команда без параметров отображает только имя операционной системы (Linux).

uname	**Параметры** **–a, ––all** Сводная информация о системе (подразумеваются все соответствующие параметры). **–m, ––machine** Вывести аппаратные свойства системы. **–n, ––nodename** Отобразить полное сетевое имя машины. **–r, ––release** Вывести номер версии ядра. **–s, ––sysname** Отобразить имя операционной системы (Linux). **–p, ––processor** Вывести тип процессора (параметр доступен не во всех версиях программы). **–v** Вывести информацию о параметрах сборки ядра. **––help** Вывести справку по **uname**. **––version** Вывести номер версии **uname**.
uncompress	**uncompress** [*options*] *files* Распаковка файлов, которые были сжаты программой **compress** (имена файлов заканчиваются суффиксом *.Z*). Доступные параметры рассмотрены в описании команды **compress**. Для **uncompress** действительны все параметры кроме **–r** и **–b**.
unexpand	**unexpand** [*options*] [*files*] Преобразование строк, начинающихся с пробелов. Два и более пробелов/табуляторов заменяются символами табуляции. Если имя файла опущено или задано символом «–» (дефис), выполняется чтение со стандартного ввода. **Параметры** **–a, ––all** Преобразовать все, а не только начальные подстроки из пробелов и символов табуляции.

unexpand

–*nums*, **–t** *nums*, **––tabs** *nums*

nums – это список целых чисел, разделяемых запятыми. Числа определяют длину табуляторов. Если задано одно число, размер табулятора устанавливается равным этому числу. По умолчанию равно 8 пробелам. Если выбран вариант параметра **–t** или **––tabs**, список может быть разделен пробелами. Подразумевается наличие параметра **–a**.

––help

Отобразить справку и завершить работу.

––version

Вывести информацию о версии программы и завершить работу.

uniq

uniq [*options*] [*file1* [*file2*]]

Удалить соседние идентичные строки из файла *file1*, записав по одной копии каждой строки в файл *file2* (или на стандартный вывод). Часто используется в качестве фильтра. Из параметров **–d** и **–u** может быть задан только один. См. также **comm** и **sort**.

Параметры

–*n*, **–f** *n*, **––skip-fields=***n*

При сравнении игнорировать *n* первых полей строки. Поля разделяются пробелами или символами табуляции.

+*n*, **–s** *n*, **––skip-chars=***n*

Игнорировать *n* первых символов поля.

–c, **––count**

Вывести каждую строку один раз, подсчитывая количество ее вхождений в текст.

–d, **––repeated**

Отображать один раз не уникальные строки, а только те, которые имеют идентичных соседей.

–i, **––ignore-case**

При сравнении строк игнорировать регистр символов.

–u, **––unique**

Вывести только уникальные строки (не оставлять копии повторяющихся).

uniq

–w *n*, **––check-chars=***n*

Сравнивать только *n* первых символов строк (отсчет начинается после пропуска указанного количества полей и символов).

––help

Вывести справку по использованию и завершить работу.

––version

Отобразить информацию о номере версии и завершить работу.

Примеры

Записать по копии каждой из строк файла **list** в файл **list.new**:

```
uniq list list.new
```

Показать имена, имеющие более одного вхождения:

```
sort names | uniq -d
```

unshar

unshar [*options*] [*files*]

Распаковать архив интерпретатора (shar-файл). **unshar** сканирует почтовые сообщения в поисках начала shar-файла. Затем архив передается в копию интерпретатора для распаковки. Команда принимает множественные имена файлов. Если файлы не указаны, выполняется чтение со стандартного ввода.

Параметры

–c, **––overwrite**

Разрешить перезапись существующих файлов.

–d *directory*, **––directory=***directory*

Перейти в указанный каталог, прежде чем распаковывать файлы.

–e, **––exit-0**

Последовательно распаковать множественные архивы, содержащиеся в одном файле; для различения архивов используется тот факт, что многие архивы **shar** завершаются командой **exit 0** в начале строки. (Параметр **–e** по существу эквивалентен **–E «exit 0»**.)

unshar

−E *string*, **−−split-at=***string*

Аналогично **−e**, но считать разделителем архивов строку *string*.

−f, **−−force**

Идентично **−c**.

−−help

Вывести справку по использованию на стандартный вывод и завершить работу.

−−version

Вывести информацию о версии программы на стандартный вывод и завершить работу.

update

update [*options*]

Команда системного администрирования. **update** является демоном, управляющим частотой записи содержимого буферов файловой системы на диск. **update** известен также под именем **bdflush**. Демон порождает пару процессов (fork-ветвлением) для вызова системных функций *flush*() и *sync*(). При выполнении команды непривилегированным пользователем демон не создается. Вместо этого происходит вызов *sync*() и завершение работы. По умолчанию команда выполняется каждые пять секунд, вызывая *flush*() для использованных буферов. Если это не срабатывает, **update** выполняет вызов *sync*() для буферов каждые 30 секунд. Не все перечисленные параметры доступны во всех версиях **update**.

Параметры

−d Отображение параметров ядра. Демон не запускается.

−f *seconds*

Вызывать *flush*() с интервалом в *seconds* секунд (по умолчанию – 5).

−h Краткая справка по команде.

−s *seconds*

Вызывать *sync*() с интервалом в *seconds* секунд (по умолчанию – 30).

−S Всегда вызывать *sync*() вместо *flush*().

update	**–0** *percent* Произвести сброс буферов на указанное количество процентов при заполнении буферного кэша. **–1** *blocks* Максимальное количество записываемых (в одном вызове) блоков. **–2** *buffers* Количество пустых буферов, к которому следует стремиться при перераспределении данных в кэше. **–3** *blocks* Производить сброс буферов, если количество занятых блоков превышает указанное при перераспределении. **–4** *percent* Процентный объем буферного кэша, просматриваемый при поиске свободных кластеров. **–5** *seconds* Время, в течение которого сохраняется буфер данных. По истечении этого времени происходит сброс. **–6** *seconds* Время, в течение которого сохраняется обычный буфер. По истечении указанного времени происходит сброс. **–7** *constant* Константа времени, используемая для среднего показателя загрузки. **–8** *ratio* Насколько низкой может стать средняя загрузка перед уменьшением количества буферов.
uptime	**uptime** Отобразить текущее время, время работы (непрерывной) системы, количество пользователей, работающих с системой в настоящий момент, средние показатели загрузки системы. Эти данные также выводятся в первой строке вывода команды **w**.
useradd	**useradd** [*options*] [*user*] Команда системного администрирования. Создание новой учетной записи пользователя или обновление информации в

существующей записи. Если отсутствует параметр **–D**, необходимо указать имя пользователя (*user*). **useradd** создает новые записи в системных файлах. Домашний каталог и начальные файлы настроек могут быть созданы при необходимости.

Параметры

–c *comment*

Поле комментария.

–d *dir*

Домашний каталог. По умолчанию создается каталог с именем *user* в каталоге *home*, определяемом параметром **–D**.

–e *date*

Дата прекращения действия доступа пользователя к системе. Задается в формате ММ/ДД/ГГГГ. Год также может задаваться двумя цифрами. Значение хранится в виде количества дней, считая от первого января 1970 года. Этот параметр подразумевает использование зашифрованных паролей (shadow passwords).

–f *days*

Запретить возможность доступа пользователя к системе на указанное количество дней после устаревания пароля. Действие параметра отключается, если указано значение –1. Параметр подразумевает применение зашифрованных паролей.

–g *group*

Имя начальной группы или число-идентификатор. Если не задана другая группа по умолчанию с помощью параметра **–D**, то принимается группа по умолчанию, имеющая идентификатор, равный 1.

–G *groups*

Дополнительные группы пользователей в виде списка имен или идентификаторов, разделяемых запятыми (пробелы недопустимы).

–k [*dir*]

Копировать начальные файлы в домашний каталог пользователя. Имеет смысл только совместно с параметром **–m**. Файлы по умолчанию копируются из каталога */etc/skel*, если не указан иной каталог (*dir*).

useradd	**–m** Создать домашний каталог пользователя, если он еще не существует. По умолчанию домашний каталог не создается. **–o** Принудительное использование указанного в параметре **–u** неуникального идентификатора пользователя (*uid*). (Не очень здравая мысль.) **–s** *shell* Интерпретатор по умолчанию. **–u** *uid* Численный идентификатор пользователя. Значение должно быть уникальным, если не указан параметр **–o**. Значение по умолчанию является самым маленьким значением, превосходящим 99 и любой другой идентификатор пользователя. **–D** [*options*] Установить или отобразить значения по умолчанию. Если указаны параметры (*options*), устанавливать соответствующие значения, иначе отображать существующие. *–b dir* Префикс, используемый при создании домашних каталогов. Если при создании записи не указан параметр **–d**, то к префиксу *dir* добавляется имя пользователя (*user*). *–e date* Дата окончания доступа. Подразумевается применение зашифрованных паролей. *–f days* Количество дней после устаревания пароля, на которое прекращается доступ пользователя к системе. Подразумевается, что пароли зашифрованы. *–g group* Имя или идентификатор начальной группы. *–s shell* Интерпретатор по умолчанию.
userdel	**userdel** [*option*] *user* Команда системного администрирования. Удаление всей информации о пользователе *user* из системных файлов учета.

Параметр **userdel**

–r Удалить домашний каталог указанного пользователя вместе со всем содержимым.

usermod [*options*] *user* **usermod**

Команда системного администрирования. Изменение информации об учетной записи указанного пользователя.

Параметры

–c *comment*

Поле комментария.

–d *dir*

Домашний каталог.

–e *date*

Дата прекращения доступа пользователя к системе. Задается в формате ММ/ДД/ГГГГ. Год также может задаваться двумя цифрами. Значение хранится в виде количества дней, считая от первого января 1970 года. Этот параметр подразумевает применение зашифрованных паролей (shadow passwords).

–f *days*

Запретить доступ пользователю к системе на указанное количество дней после устаревания пароля. Действие параметра отключается, если указано значение –1. Параметр подразумевает, что пароли зашифрованы.

–g *group*

Имя начальной группы или число-идентификатор.

–G *groups*

Дополнительные группы пользователей в виде списка имен или идентификаторов, разделяемых запятыми (пробелы не допустимы). Пользователь удаляется из всех групп, не указанных в этом списке.

–l *name*

Регистрационное имя пользователя (*name*). Не может быть изменено, если пользователь работает в системе.

–o Принудительное использование указанного в параметре **–u** неуникального идентификатора пользователя (*uid*).

–s *shell*

Интерпретатор по умолчанию.

usermod	**–u** *uid* Численный идентификатор пользователя. Значение должно быть уникальным, если не указан параметр **–o**. Все файлы в домашнем каталоге пользователя автоматически сменят владельца. Файлы вне домашнего каталога будут иметь прежнего владельца. В момент изменения не должны быть запущены какие-либо процессы этого пользователя.
users	**users** [*file*] Перечислить (через пробел) все зарегистрированные сеансы работы с узлом. Перечисление может включать различные сеансы работы одного и того же пользователя. Для сбора статистики отводится указанный файл (*file*) либо */etc/utmp* (по умолчанию).
usleep	**usleep** [*microseconds*] **usleep** [*options*] Пауза длиной в указанное количество микросекунд (по умолчанию равно 1). **Параметры** **––help** Вывести справку по команде и завершить работу. **––usage** Вывести справку по использованию команды. **–v, ––version** Отображение номера версии **usleep**.
uudecode	**uudecode** [**–o** *outfile*] [*file*] Прочитать файл, закодированный **uuencode**, и воссоздать исходный файл, сохранив права доступа и имя, указанное при кодировании (см. **uuencode**). Параметр **–o** задает альтернативное имя раскодированного файла.
uuencode	**uuencode** [**–m**] [*file*] *name* Кодирование файла данных. Кодирование использует только отображаемые ASCII-символы, сохраняет права доступа и имя (*name*) файла. Когда файл *file* декодируется посредством

uudecode, результат записывается в файл с именем *name*. При запуске без аргумента *file* **uuencode** производит чтение со стандартного ввода, так что единственный аргумент будет воспринят как имя, которое будет дано файлу при раскодировании. Параметр **−m** является предписанием использовать кодирование base64.

uuencode

Пример

Обычно закодированные файлы получают идентифицирующее их расширение, например *.uue*. В следующем примере файл *flower12.jpg* получает имя *rose.jpg* и после кодирования сохраняется в файле *rose.uue*:

```
% uuencode flower12.jpg rose.jpg > rose.uue
```

Закодировать файл *flower12.jpg* и отправить его почтой:

```
% uuencode flower12.jpg flower12.jpg | mail ellen@oreilly.com
```

vacation
vacation [*options*] [*user*]

vacation

Автоматически возвращать почтовое уведомление о том, что адресат находится в отпуске.

vacation без параметров предназначена для инициализации данного механизма. В ходе процесса выполняются следующие шаги:

1. Создается файл *.forward* в домашнем каталоге, содержащий команду:

   ```
   \user, "|/usr/bin/vacation user"
   ```

   где *user* – регистрационное имя пользователя. Выполнение файла позволяет доставить почту пользователю и обработать входящую почту с помощью **vacation**.

2. Создаются файлы *.vacation.pag* и *.vacation.dir*. Эти файлы содержат информацию об отправителях писем, что позволяет отправлять им уведомление о том, что вы в отпуске, лишь раз в неделю.

3. Вызывается редактор для изменения содержимого файла *.vacation.msg*. Содержимое файла отправляется человеку, который прислал вам письмо. В теле файла **$subject** заменяется содержимым строки **Subject** полученного почтового сообщения.

Чтобы прекратить работу **vacation**, удалите или переименуйте файл *.forward*.

vacation

Параметры

Параметры **–a** и **–r** указываются в теле файла *.forward*; см. пример.

–a *alias*

Почта, адресованная пользователю *alias*, реально является почтой для пользователя *user*; на такие сообщения необходимо отвечать автоматически.

–i Повторно инициализировать файлы *.vacation.pag* и *.vacation.dir*. Используйте параметр непосредственно перед уходом в очередной отпуск.

–r *interval*

По умолчанию каждому адресату отправляется не более одного письма в неделю. Данный параметр позволяет изменить интервал. Интервал – это число, которое может модифицироваться буквами **s**, **m**, **h**, **d** или **w**, означающими, соответственно, секунды, минуты, часы, дни и недели. Если интервал определен ключевым словом **infinite**, каждый адресат получает только одно уведомление.

Пример

Отправлять каждому адресату не более одного уведомления за три недели:

```
$ cd
$ vacation -I
$ cat .forward
\jp, "|/usr/bin/vacation -r3w jp"
$ cat .vacation.msg
From: jp@wizard-corp.com (J. Programmer, via the vacation program)
Subject: Меня нет на работе. . .
Привет. Я сейчас на заслуженном отдыхе после окончания первой
версии программы whizprog. Я прочту Ваше письмо на тему
"$SUBJECT" и отвечу, когда вернусь.
Счастливо.
```

vi

vi [*options*] [*files*]

Экранный текстовый редактор, основанный на **ex**. Более подробно описан в главе 11 «Редактор vi».

vidmode

vidmode [*option*] *image* [*mode* [*offset*]]

Команда системного администрирования. Установка видеорежима для образа (*image*) ядра. Команда без аргументов

vidmode

отображает текущее значение режима (*mode*). Аргумент *mode* – это однобайтовое значение, хранимое в образе ядра по смещению 506. Режим может быть изменен указанием образа ядра и нового значения, а также смещения в байтах, по которому необходимо записать новую информацию (по умолчанию – 506). Обратите внимание, что **rdev –v** является синонимом **vidmode**. При использовании **LILO vidmode** не требуется. Видеорежим может быть указан в приглашении **LILO** при загрузке системы.

Режимы

–3 Ввод режима пользователем.

–2 Расширенный VGA.

–1 Обычный VGA.

0 Аналогично вводу пользователем режима **0**.

1 Аналогично вводу пользователем режима **1**.

2 Аналогично вводу пользователем режима **2**.

3 Аналогично вводу пользователем режима **3**.

n Аналогично вводу пользователем режима **n**.

Параметр

–o *offset*

Идентично указанию смещения в качестве аргумента команды.

---

w [*options*] [*user*]

w

Вывод статистики по расходованию ресурсов системы пользователями, работающими с ней в настоящий момент, и их процессами. Данная команда, по существу, является сочетанием команд **uptime**, **who** и **ps –a**. Для получения результатов по определенному пользователю необходимо указать его имя (*user*).

Параметры

–f Управление отображением поля From (имя удаленного узла).

–h Подавить вывод заголовков и информации **uptime**.

–s Краткий формат выдачи.

w

–u Игнорировать имя пользователя, вычисляя временные показатели процессов и процессора системы.

–V Вывести информацию о версии программы.

Файл

/var/run/utmp

Список пользователей, работающих в системе.

wall

wall [*file*]

Команда системного администрирования. Посылка сообщения всем пользователям. **wall** читает сообщение со стандартного ввода, пока не встретится символ EOF. Затем сообщение (предваряемое строкой «Broadcast Message from...») отправляется всем пользователям, работающим в системе. Если задан файл (*file*), посылается содержимое этого файла, а не данные, поступающие со стандартного ввода.

wc

wc [*options*] [*files*]

Вывод количества символов, слов и строк в каждом файле. Вывод общего количества строк во всех указанных файлах. Если имена файлов опущены, читать данные со стандартного ввода. См. другие примеры в описаниях команд **ls** и **sort**.

Параметры

–c, ––bytes, ––chars

Производить подсчет символов.

–l, ––lines

Производить подсчет строк.

–w, ––words

Выполнять подсчет слов.

––help

Отобразить справку и завершить работу.

––version

Вывести информацию о версии программы и завершить работу.

Примеры

Подсчет количества пользователей, работающих в системе в настоящий момент:

```
who | wc -l
```

wc

Подсчет количества слов в трех файлах с эссе:

```
wc -w essay.[123]
```

Подсчет строк в файле, имя которого задается переменной **$file** (не отображать имя файла):

```
wc -l < $file
```

whatis *keyword*

whatis

Поиск краткого описания в базе данных *whatis* по каждому из указанных ключевых слов и отображение одной строки описания на стандартный вывод для каждого соответствия. Действие команды подобно действию **apropos**, но производится поиск только полностью совпадающих слов. Эквивалентно **man −f**.

whereis [*options*] *files*

whereis

Поиск исполняемого файла, исходных текстов, страниц руководства указанной команды/файла. Из имен в аргументах удаляются пути и одно завершающее расширение вида *.ext* (например, *.c*). Префиксы *s.* управления исходными текстами также удаляются. **whereis** производит поиск указанной программы в стандартных каталогах Linux (*/bin*, */etc*, */usr/bin*, */usr/local/bin* и т. д.).

Параметры

−b Поиск только исполняемых файлов.

−f Прекратить работу со списком для предыдущего каталога и сигнализировать о начале имен файлов. Параметр **−f** должен указываться, если задан любой из параметров **−B**, **−M** или **−S**.

−m Поиск только страниц руководства.

−s Поиск только исходных текстов.

−u Поиск необычных объектов. Объект считается необычным, если для него не существует файла указанного типа. Так, команда **whereis −m −u *** перечислит файлы из текущего каталога, для которых отсутствует документация.

−B *directory*

Изменить или ограничить пути поиска двоичных файлов.

whereis

–M *directory*

Изменить или ограничить пути поиска документации.

–S *directory*

Изменить или ограничить пути поиска файлов с исходными текстами.

Пример

Найти все файлы в каталоге */usr/bin*, которые не документированы в секции */usr/man/man1*, но имеют исходные тексты в */usr/src*:

```
% cd /usr/bin
% whereis -u -M /usr/man/man1 -S /usr/src -f *
```

which

which [*options*] [**––**] [*command*] [...]

Перечислить полные пути файлов, которые будут выполнены при запуске команд *commands*. **which** выполняет поиск по переменной окружения пользователя $PATH. В интерпретаторах **csh** и **tcsh** существует встроенная команда **which**, которая вызывается без параметров. Чтобы иметь возможность указать параметры, следует задавать полный путь команды (т. е. */usr/bin/which*).

Параметры

–a, ––all

Вывести все соответствия, а не только первое.

–i, ––read-alias

Читать псевдонимы со стандартного ввода и выдавать соответствия на стандартный вывод. Полезно для создания псевдонима **which**.

––skip-alias

Игнорировать параметр **––read-alias**, если он указан. Полезно для поиска нормальных исполняемых файлов при псевдониме **which**, определенном с параметром **––read-alias**.

––skip-dot

Пропускать каталоги, имена которых начинаются с точки.

––skip-tilde

Пропускать каталоги, имена которых начинаются с тильды (~), а также исполняемые файлы из домашнего каталога, определенного в переменной $HOME.

which

––show-dot

Если соответствующая критерию поиска команда найдена в каталоге, имя которого начинается с точки, вывести имя не полностью, а в виде *./cmdname*.

––show-tilde

Вывести «~» вместо части имени, соответствующей пути домашнего каталога пользователя. Параметр игнорируется для пользователя **root**.

––tty-only

Не обрабатывать последующие параметры, если работа происходит без терминала.

–v, –V, ––version

Вывести информацию о версии программы и завершить работу.

Пример

```
$ which cc ls
/usr/bin/cc
ls:     aliased to ls -sFC
```

who

who [*options*] [*file*]
who am i

Отобразить сведения о пользователях, работающих в системе. Команда без параметров перечисляет имена пользователей, работающих с системой в данный момент, их терминалы, время пребывания в системе и имя узла, с которого они соединились с системой. Возможно задание имени системного файла (по умолчанию – */etc/utmp*) для получения дополнительной информации.

Параметры

am i

Вывести имя пользователя, выполняющего команду.

––help

Отобразить справку по команде **who** и завершить работу.

–i, –u, ––idle

Включить в информацию время бездействия каждого пользователя. Время «.» (точка) означает активность пользователя в пределах последней минуты; время **old** указывает на отсутствие активности в течение суток.

who

–l, ––lookup

Попытаться отобразить канонические имена узлов с помощью DNS.

–m Идентично **who am i**.

–q, ––count

«Быстрый» режим. Отобразить только имена пользователей и их общее количество.

––version

Вывести информацию о версии программы и завершить работу.

–w, –T, ––mesg, ––message, ––writable

Отобразить коммуникационное состояние пользователя:

+ **mesg y** (принимаются сообщения **write**).
– **mesg n** (сообщения **write** игнорируются).
? не найдено устройство терминала.

–H, ––heading

Вывести заголовки столбцов.

Пример

Следующий результат был получен в 8 часов утра 17 апреля:

```
$ who -uH
NAME    LINE   TIME           IDLE    PID    COMMENTS
Earvin  ttyp3  Apr 16 08:14   16:25   2240
Larry   ttyp0  Apr 17 07:33     .    15182
```

Поскольку Эрвин (Earvin) бездействовал с 16 часов предыдущего дня, разумно предположить, что он еще не пришел на работу. Просто он оставил активный сеанс. Ларри (Larry), напротив, работает на своем терминале.

whoami

whoami

Отобразить идентификатор пользователя, выполняющего команду. Эквивалентно **id –un**.

write

write *user* [*tty*]

message

Начать интерактивную беседу с пользователем *user* или ответить на реплику. Сеанс работы **write** завершается по вводу символа *EOF*. Если пользователь работает более чем на одном

терминале, можно указать номер терминала (*tty*). См. также **talk**. Выполните команду **mesg**, чтобы запретить другим пользователям посылать вам сообщения.

write

xargs [*options*] [*command*]

xargs

Выполнить указанную команду (с произвольными начальными аргументами) и прочитать дополнительные аргументы команды со стандартного ввода. **xargs** в нескольких пакетах передает прочитанные аргументы команде *command*, позволяя ей обработать больше аргументов, чем это возможно (за один раз) в обычных условиях. Как правило, эти аргументы являются длинным списком имен файлов (созданным, например, **ls** или **find**), который передается в **xargs** через конвейер.

Параметры

–0, ––null

Считать, что имена файлов завершаются символом NULL. Не трактовать кавычки и обратные слэши специальным образом.

–e[*string*], **––eof**[=*string*]

Считать символом EOF символ «_» или указанную строку *string*.

––help

Вывести справку по параметрам **xargs** и завершить работу.

–i[*string*], **––replace**[=*string*]

Заменить все вхождения символов {} или строки *string* именами, прочитанными со стандартного ввода. Немаскированные пробелы не считаются признаками концов аргументов. Подразумеваются параметры **–x** и **–l 1**.

–l[*lines*], **––max-lines**[=*lines*]

Запретить более 1 непустой строки либо *lines* непустых строк в командной строке. Подразумевается параметр **–x**.

–n *args*, **––max-args=***args*

Ограничить максимальное количество аргументов числом *args*. Имеет меньший приоритет, чем параметр **–s**.

–p, ––interactive

Запрашивать подтверждение перед выполнением каждой командной строки. Подразумевается параметр **–t**.

xargs

–P *max*, **––max-procs=***max*

Ограничить количество одновременно выполняемых заданий числом *max*. По умолчанию имеет значение, равное 1. Максимальное значение, равное 0, снимает ограничение на количество заданий.

–r, ––no-run-if-empty

Не выполнять команду, если она состоит только из пробелов.

–s *max*, **––max-chars=***max*

Ограничить длину командной строки количеством символов, равным *max*.

–t, ––verbose

Режим диагностики (отображать командную строку в стандартный поток ошибок перед выполнением).

–x, ––exit

Если превышен максимальный размер (заданный параметром **–s**), завершить работу.

––version

Вывести номер версии **xargs** и завершить работу.

Примеры

Производить поиск по шаблону *pattern* во всех файлах системы:

```
find / -print | xargs grep pattern > out &
```

Выполнить **diff** для пар файлов (например, **f1.a** и **f1.b**, **f2.a** и **f2.b . . .**):

```
echo $* | xargs -n2 diff
```

Предыдущая строка могла бы использоваться внутри сценария, вызываемого с именами файлов в качестве аргументов. В следующем примере указанный файл отображается целиком, по одному слову на строке (идентично работе команды **deroff –w**):

```
cat file | xargs -n1
```

Переместить файлы из каталога *olddir* в каталог *newdir*, показывая каждую команду переноса:

```
ls  olddir | xargs -i -t mv olddir/{} newdir/{}
```

yacc [*options*] *file* **yacc**

Построение таблиц для последующего грамматического разбора на основе файла (*file*), содержащего описание контекстно-свободной грамматики. Результат записывается в файл *y.tab.c*. Имя команды расшифровывается как **y**et **a**nother **c**ompiler-**c**ompiler (еще один компилятор компиляторов). См. также **flex**, **bison** и книгу «lex & yacc» Джона Левина (John Levine), Тони Мэйсона (Tony Mason) и Дага Брауна (Doug Brown).

Параметры

–b *prefix*

Добавлять указанный префикс к имени файла вместо *y*.

–d Создать файл *y.tab.h*, содержащий директивы **#define**, которые связывают коды лексем **yacc** с именами лексем, определяемыми пользователем.

–l Исключить директивы **#line** из кода в файле *y.tab.c* (обычно применяется после завершения отладки).

–t Компиляция отладочного кода времени выполнения.

–v Создать файл *y.output*, содержащий диагностику и примечания к таблицам разбора.

yes [*strings*] **yes**

yes [*option*]

Отображать аргументы командной строки, разделенные пробелами. Последовательность завершается символом новой строки. Команда **yes** работает, пока не поступит сигнал принудительного завершения. Если команда выполняется без аргументов, то выводится символ **y**, за которым следует символ новой строки (последовательность повторяется, пока **yes** не будет принудительно завершена). Команда полезна в сценариях и для фоновых задач; ее вывод может передаваться через конвейер в диалоговую программу.

Параметры

––help

Вывести справку и завершить работу.

––version

Вывести информацию о версии программы и завершить работу.

ypbind

ypbind [*options*]

Команда NFS/NIS. Связывающий процесс NIS. **ypbind** – это демон, который обычно запускается при загрузке системы. Его задачей является запоминание информации, которая позволяет процессам клиентов на одном из узлов обмениваться данными с процессом **ypserv**. Данные, хранимые **ypbind**, называются *связками* (bindings). Они ассоциируют имя домена с интернет-адресом NIS-сервера и номером порта, на котором процесс **ypserv** этого узла осуществляет прием запросов клиентов. Данные хранятся в файле */var/yp/bindings/domainname.version*.

Параметры

–ypset

Может использоваться для изменения связки. Применение параметра очень опасно и допустимо только при отладке работы сети с удаленной машины.

–ypsetme

Запросы **ypset** могут выдаваться только с этой машины. Защита обеспечивается проверкой IP-адресов, что совершенно не помогает в случае посылки пакетов не заслуживающими доверия личностями. Не рекомендуется использовать этот параметр.

ypcat

ypcat [*options*] *mname*

Команда NFS/NIS. Вывод значения из базы данных NIS по имени *mname* (имя или псевдоним карты).

Параметры

–d *domain*

Указать домен вместо домена по умолчанию.

–k Отображать ключи для карт, в которых ключи имеют пустые значения или не являются частью значения.

–t Не переводить *mname* в имя карты.

–x Отображать таблицу псевдонимов, в которой перечисляются известные псевдонимы (*mnames*) и имена связанных карт. Аргумент *mname* при этом не требуется.

ypchfn [*options*] [*user*] **ypchfn**

Команда NFS/NIS. Изменение информации, которая хранится в файле */etc/passwd* и отображается по finger-запросам. Распространение изменений через NIS. Команда без параметров работает в диалоговом режиме, запрашивая данные у пользователя. Чтобы оставить поле пустым, следует ввести ключевое слово **none**. Привилегированный пользователь может изменять информацию для любого пользователя. См. также **yppasswd** и **ypchsh**.

Параметры

–f Работа в стиле **ypchfn** (по умолчанию).

–l Работа в стиле **ypchsh**.

–p Работа в стиле **yppasswd**.

---

ypchsh [*options*] [*user*] **ypchsh**

Команда NFS/NIS. Изменение командного интерпретатора по умолчанию и распространение этого изменения через NIS. Если для интерпретатора нет соответствующей записи в файле */etc/shells*, выдается предупреждение. Привилегированный пользователь может изменять интерпретатор для любого пользователя. См. также **yppasswd** и **ypchfn**.

Параметры

–f Работа в стиле **ypchfn**.

–l Работа в стиле **ypchsh** (по умолчанию).

–p Работа в стиле **yppasswd**.

---

ypinit [*options*] **ypinit**

Команда NFS/NIS. Сборка и установка базы данных NIS на NIS-сервер. **ypinit** может использоваться для настройки основного сервера или подчиненного сервера, а также подчиненного сервера, копирующего карты. Выполнять **ypinit** может только привилегированный пользователь.

Параметры

–c *master_name*

Установить базу данных подчиненного узла, копирующего данные. Аргумент *master_name* задает имя узла NIS-сервера либо основного сервера всех карт, либо сервера, на котором хранится стабильная база с самыми последними данными.

ypinit	**–m** Указать, что локальный сервер будет являться NIS-сервером. **–s** *master_name* Установить базу данных подчиненного сервера. Аргумент *master_name* задает имя узла NIS-сервера либо основного сервера, либо сервера, на котором хранится стабильная база данных с самыми последними данными.
ypmatch	**ypmatch** [*options*] *key* . . . *mname* Команда NFS/NIS. Вывести значения перечисленных ключей из NIS-карты, заданной именем *mname* (имя или псевдоним карты). **Параметры** **–d** *domain* Указать домен для использования вместо домена по умолчанию. **–k** Для каждого значения Вывести собственно ключ с символом двоеточия (:). **–t** Не переводить псевдоним в имя карты. **–x** Отобразить таблицу псевдонимов, в которой перечисляются известные псевдонимы (*mnames*) и имена связанных карт. Аргумент *mname* при этом не требуется.
yppasswd	**yppasswd** [*options*] [*user*] Команда NFS/NIS. Создание или изменение пароля для пользователя *user* и распространение нового пароля через NIS. Привилегированный пользователь может изменить пароль для любого пользователя. См. также **ypchfn** и **ypchsh**. **Параметры** **–f** Работа в стиле **ypchfn**. **–l** Работа в стиле **ypchsh**. **–p** Работа в стиле **yppasswd** (по умолчанию).
yppasswdd	**rpc.yppasswdd** [*option*] Команда NFS/NIS. Сервер изменения файла паролей NIS. **yppasswdd** обрабатывает запросы на изменение пароля по команде **yppasswd**. Изменение записи пароля происходит

yppasswdd

только в том случае, если пароль, представленный **yppasswd**, совпадает с зашифрованным паролем этой записи, а идентификаторы пользователя и группы соответствуют значениям из */etc/passwd*. Затем происходит обновление файла */etc/passwd* и карт паролей локального сервера.

Параметр

–s Поддержка функций шифрования паролей (shadow passwords).

yppoll

yppoll [*options*] *mapname*

Команда NFS/NIS. Определение версии NIS-карты на NIS-сервере. **yppoll** получает от процесса **ypserv** номер ранга и имя узла NIS-сервера, являющегося хозяином указанной карты.

Параметры

–h *host*

Запросить параметры карты у процесса **ypserv** на указанном узле (*host*). Если узел не указан, используется имя узла NIS-сервера для локальной машины (возвращаемое командой **ypwhich**).

–d *domain*

Указать домен для использования вместо домена по умолчанию.

yppush

yppush [*options*] *mapnames*

Команда NFS/NIS. Принудительное распределение измененной NIS-карты. **yppush** копирует новую версию NIS-карты (*mapname*) с сервера-хозяина карт на подчиненные серверы. Сначала создается список узлов NIS-серверов посредством чтения **ypservers** для указанной NIS-карты (с параметром **–d** для указания имени домена). Ключи полученной карты представляют собой ASCII-запись имен машин, на которых работают серверы NIS. Запрос «transfer map» (передача карты) направляется каждому из серверов NIS на перечисленных узлах вместе с информацией, необходимой агенту передачи для последующего обратного вызова **yppush**. После завершения этой операции агент передачи отправляет **yppush** данные о состоянии, а результаты – на стандартный вывод (**stdout**). **yppush** обычно выполняется из */var/yp/Makefile*.

yppush

Параметры

–d *domain*

Указать домен.

–v Диагностика: вывести сообщения при вызове и получении ответа каждого из серверов.

ypserv

ypserv [*options*]

Команда NFS/NIS. Процесс NIS-сервера. **ypserv** является процессом-демоном, активизируемым, как правило, при запуске системы. Он работает только на серверах NIS, обладающих полной базой данных NIS. Основной его задачей является поиск информации в своих локальных базах данных или базах данных NIS. Действия, выполняемые **ypserv**, определяются спецификацией протокола NIS, что для программистов эквивалентно информации, содержащейся в заголовочном файле <*rpcvc/yp_prot.h*>. Работа с **ypserv** происходит посредством вызовов RPC.

Параметры

–d Предписание NIS-сервису опрашивать DNS в поисках более подробной информации об узлах.

–localonly

Запрет **ypserv** отвечать на внешние запросы.

Файлы и каталоги

/var/yp/[*domainname*]*/*

Расположение баз данных NIS.

/var/yp/Makefile

Файл сборки, ответственный за создание баз данных *yp*.

ypset

ypset [*options*] [*server*]

Команда NFS/NIS. Указание **ypbind** на определенный сервер. **ypset** предписывает **ypbind** получать NIS-службы для указанного домена от процесса **ypserv**, запущенного на сервере (*server*). *server* является связываемым сервером NIS и может быть задан как именем, так и IP-адресом.

ypset

Параметры

–d *domain*

Использовать указанный домен вместо домена по умолчанию.

–h *host*

Установить связывание **ypbind** для узла *host* вместо локального. Узел может быть задан именем или IP-адресом.

---

ypwhich [*options*] [*host*]

ypwhich

Команда NFS/NIS. Возвращает имя узла NIS-сервера или хозяина карт. Команда без аргументов вызывает NIS-сервер для текущей машины. Если задан узел (*host*), он опрашивается на предмет получения используемого сервера-хозяина NIS.

Параметры

–d *domain*

Использовать указанный домен вместо домена по умолчанию.

–m *map*

Найти NIS-сервер, являющийся хозяином карт. Узел не может быть указан в параметре **–m**. *map* может являться именем или псевдонимом карты.

–t *mapname*

Запретить перевод псевдонимов.

–x Вывести таблицу псевдонимов карт. Не сочетается с прочими параметрами.

---

ypxfr [*options*] *mapname*

ypxfr

Команда NFS/NIS. Передача NIS-карты с сервера на локальный узел при помощи обычных служб NIS. **ypxfr** создает временную карту в каталоге */etc/yp/domain* (где *domain* – домен по умолчанию для локального узла), последовательно заполняет ее записями карты, получает от сервера параметры карты и загружает их. При работе в диалоговом режиме **ypxfr** отображает вывод на терминал. Однако при работе без управляющего терминала весь вывод добавляется к файлу */etc/admin/nislog*, если таковой существует.

–b Сохранить при передаче флаг разрешения в файле карты.

ypxfr	**–C** *tid prog ipaddr port* Этот параметр предназначен для использования только **ypserv**. Когда **ypserv** вызывает **ypxfr**, он указывает, что **ypxfr** необходимо вызвать процесс **yppush** для узла с IP-адресом *ipaddr*, с именем программы *prog*, зарезервированным портом приема *port* и предписанием ожидать ответа на передачу *tid*. **–c** Не посылать запрос «Clear current map» (очистки текущей карты) локальному процессу **ypserv**. **–d** *domain* Использовать указанный домен, а не домен по умолчанию. **–f** Принудительно передать карту, даже если версия узла-хозяина более старая, чем локальная. **–h** *host* Получить карту с указанного узла, независимо от того, кто является его реальным хозяином. Если узел не задан, **ypxfr** запрашивает у NIS-службы имя хозяина и пытается получить карту от него. *host* может задаваться как именем, так и интернет-адресом вида *h.h.h.h*. **–S** Использовать только NIS-серверы, работающие от пользователя **root** через зарезервированный порт. **–s** *domain* Указать исходный домен, из которого должна быть получена карта для междоменной синхронизации (например, карта *services.byname*).
zcat	**zcat** [*options*] [*files*] Чтение одного или нескольких файлов (*files*), сжатых программами **compress** или **gzip**, и отображение их содержимого на стандартный вывод. Чтение производится со стандартного ввода, если имена файлов не указаны или в качестве одного из имен указан символ «–». Чтение ввода заканчивается по символу EOF. **zcat** работает идентично команде **gunzip –c** и распознает параметры **–fhLV**, описанные для **gzip/gunzip**.
zcmp	**zcmp** [*options*] *files* Чтение сжатых файлов и передача их в несжатом виде команде **cmp** вместе со всеми параметрами командной строки. Если не задан второй файл для сравнения, используется файл с именем *file.gz*.

zdiff [*options*] *files* **zdiff**

Чтение сжатых файлов и передача их в несжатом виде программе **diff** вместе со всеми параметрами командной строки. Если не задан второй файл для сравнения, используется файл с именем *file*.**gz**.

zdump [*options*] [*zones*] **zdump**

Команда системного администрирования. Перечисление всех известных часовых поясов либо, если задан аргумент, только указанных поясов. Включается название для каждого пояса и его текущее поясное время.

Параметры

–c *year*

Задать год, чтобы ограничить подробный вывод. Имеет смысл только с параметром **–v**.

–v Подробный режим. Включить дополнительную информацию по каждому поясу.

zforce [*names*] **zforce**

Переименовывание всех файлов, сжатых **gzip**, таким образом, чтобы имена имели вид *filename*.***gz***. Операция не производится, если файл уже имеет расширение **.gz**.

zgrep [*options*] [*files*] **zgrep**

Распаковка файлов и передача их **grep** вместе со всеми параметрами командной строки. Если имена файлов не указаны, производится чтение со стандартного ввода (и попытка распаковать поступающие данные). Вызов может производиться командами **zegrep** или **zfgrep**, и в этом случае данные будут передаваться **egrep** и **fgrep**, соответственно.

zic [*options*] [*files*] **zic**

Команда системного администрирования. Создание из указанных файлов информационных файлов преобразования времени. Если задано имя «–», выполняется чтение со стандартного ввода.

zic

Параметры

–d *directory*

Поместить созданные файлы в указанный каталог. По умолчанию это каталог */usr/local/etc/zoneinfo*.

–l *timezone*

Указать часовой пояс (*timezone*) для местного времени. **zic** производит связывание информации часового пояса *timezone* с поясом **localtime**.

–p *timezone*

Установка правил по умолчанию для обработки переменных окружения POSIX-формата в имя часового пояса *timezone*.

–s Сохранять значения времени, только если отрицание не меняет их.

–v Режим диагностики. Включаются дополнительные проверки и вывод предупреждений.

–y *command*

Производить определение типа года с помощью указанной команды. По умолчанию – **yearistype**.

–L *file*

Взять информацию о корректировочных секундах из файла *file*.

Исходные файл(ы) для **zic** должны содержать последовательности строк с правилами, строк часовых поясов и строк связей. Также может быть указано необязательное имя файла, содержащего информацию о корректировочных секундах. Строки правил определяют, каким образом должно вычисляться время. Описывается изменение времени, переход на летнее время, военное время и любые другие параметры, влияющие на время определенного часового пояса. Строки часовых поясов указывают, какие из правил действительны для конкретного часового пояса. Строки связей предназначены для объединения сходных часовых поясов. Корректировочные строки содержат информацию о точном времени добавления или вычитания корректировочной секунды. Любая из этих строк состоит из набора полей. Поля разделяются произвольным числом пробелов. Строки комментариев начинаются с символа #. Типы полей разных строк приводятся ниже.

Поля строк правил

Формат строки правила следующий:

```
Rule NAME FROM TO TYPE IN ON AT SAVE LETTERS
```

NAME

Имя набора правил.

FROM

Первый год, к которому применяется правило. Работа выполняется с датами Григорианского календаря. Вместо указания конкретного года можно задать *minimum* или *maximum* – минимальный и максимальный года, представимые целым числом.

TO

Последний год, для которого действует данное правило. Синтаксис такой же, как и для поля FROM.

TYPE

Тип года, к которому следует применять правило. Маска «–» подразумевает все года. Заданный тип года проверяется командой параметра **–y** либо по умолчанию с помощью **yearistype** *год тип*. Нулевой код завершения означает, что год действительно имеет указанный тип; код завершения, равный 1, означает, что год имет другой тип (см. параметр **–y**).

IN Месяц, в который должно применяться правило.

ON День, в который должно применяться правило. Использование пробелов не допускается. Например:

1 Первое число.

firstSun

Первое воскресенье месяца.

Sun>=3

Первое воскресенье, которое приходится на 3 число либо до него.

AT Время, после которого правило вступит в действие. Например, для часа дня можно задать **13**, **13:00** или **13:00:00**. Ко времени могут добавляться суффиксы (без предваряющих пробелов):

s Местное стандартное время.

u, g, z Универсальное время.

w Время настенных часов (по умолчанию).

zic

SAVE

Количество времени, добавляемое к местному стандартному. Форматируется так же, как **AT**, но без суффиксов.

LETTERS

Буквы, используемые для сокращенного указания часового пояса (например, **S** для **EST**). Если сокращение отсутствует, используйте символ «–».

Поля строк часовых поясов

Формат строки часового пояса следующий:

```
Zone NAME GMTOFF RULES/SAVE FORMAT [UNTIL]
```

NAME

Имя часового пояса.

GMTOFF

Разница в часах с поясом GMT (среднее время по Гринвичу). Форматируется так же, как **AT**. Отрицательное время вычитается из GMT; по умолчанию время добавляется.

RULES/SAVE

Либо имя правила для данного часового пояса, либо количество времени, добавляемого к местному стандартному. Символ «–» является указанием на часовой пояс для местного стандартного времени.

FORMAT

Формат сокращений для часовых поясов. Переменная часть определяется параметром %**s**.

UNTIL

Дата, при наступлении которой изменяется правило для часового пояса. В следующей строке должна определяться новая информация часового пояса, т. е. строка «Zone» и поле NAME должны быть опущены.

Поля строк связей

Формат строки связей:

```
Link LINK-FROM LINK-TO
```

LINK-FROM

Имя связываемого часового пояса.

LINK-TO

Альтернативное имя для часового пояса, указанного в **LINK-FROM**.

zic

Поля строк корректировки

Формат строки корректировки:

```
Leap YEAR MONTH DAY HH:MM:SS CORR R/S
```

YEAR MONTH DAY HH:MM:SS

Точное время добавления корректировочной секунды.

CORR

+ или − используются для указания на добавление или вычитание корректировочной секунды.

R/S

Сокращение от Rolling или Stationary, указывающее, что корректировочная секунда добавляется к местному времени по настенным часам или к времени GMT.

zmore

zmore [*files*]

Аналог **more**. Поэкранный просмотр запакованных файлов. Работает с файлами, сжатыми **compress**, **gzip** или **pack**, а также с распакованными файлами.

Команды

пробел

Перейти к следующей странице текста.

i[*number*]

Перейти к следующей странице текста или к следующей порции из *number* строк. Установить значение пролистывания по команде *i* в *number* строк.

d, **^D**

Вывести *i* следующих либо 11 строк.

*i***z** Вывести *i* следующих строк либо полную экранную страницу.

*i***s** Пропустить *i* строк. Вывести следующую полную экранную страницу.

*i***f** Пропустить *i* экранных страниц. Вывести следующую полную экранную страницу.

q, **Q**, **:q**, **:Q**

Перейти к следующему файлу или завершить работу, если текущий файл является последним.

zmore	**e, q** При отображении приглашения «--More--(Next file: *file*)» завершать работу **zmore**. **s** Пропустить следующий файл и продолжить просмотр. **=** Вывести номер строки. *i/expr* Прямой поиск *i*-го вхождения (во всей цепочке файлов) соответствия регулярному выражению *expr*. Отобразить вхождение и две предшествующие строки контекста. *i***n** Повторение прямого поиска *i*-го вхождения по последнему регулярному выражению. **!***command* Выполнить указанную команду в интерпретаторе. Если команда не указана, выполняется последняя выполненная команда. Для перехода в интерпретатор без указания команды введите «**!**». **.** Повторить последнюю команду.
znew	**znew** [*options*] [*files*] Распаковка файлов **.Z** и повторное сжатие их в формате **.gz**. **Параметры** **–9** Наилучший (и самый медленный) метод сжатия данных. **–f** Принудительное повторное сжатие, даже если уже существует *filename*.**gz**. **–t** Проверить корректность создания файлов в формате **.gz** перед удалением файлов **.Z**. **–v** Диагностика работы. **–K** Если размер исходного файла меньше размера полученного **.gz**, не удалять исходный файл. **–P** Передать (через конвейер) данные программе преобразования. Это сокращает занимаемое при работе дисковое пространство.

Глава 4

Методы загрузки

В этой главе рассматриваются различные методы загрузки системы Linux. В зависимости от аппаратной базы и необходимости совместной работы с другими операционными системами можно настроить систему так, чтобы Linux загружался автоматически либо предоставить пользователю выбор между различными операционными системами во время загрузки. Такая возможность обычно называется *двойной загрузкой*, но на деле можно выбирать между произвольным числом систем (например, Linux и Windows 95/98/NT/2000). Рассмотрены следующие темы:

- Процесс загрузки
- LILO: загрузчик Linux
- Loadlin: загрузка из MS-DOS
- Двойная загрузка Linux и Windows NT/2000
- Параметры, передаваемые ядру при загрузке
- **initrd**: использование RAM-диска

Процесс загрузки

Если Linux уже установлен и работает, процесс загрузки достаточно прост. Но наличие большого количества аппаратных и программных средств предоставляет разнообразные возможности для настройки этого процесса. Существует три традиционных варианта загрузки:

- С дискеты. При этом прочие системы могут, как и прежде, загружаться с жесткого диска.

- С помощью LILO (LInux LOader, загрузчик Linux).[1] По всей видимости, это самый распространенный метод загрузки, который позволяет загружать и Linux, и другие операционные системы.
- С помощью Loadlin, программы MS-DOS, позволяющей загружать Linux из DOS.

Существуют и другие загрузчики, которые способны загружать Linux, но мы не будем говорить о них в этой главе. Также не будем подробно останавливаться на загрузке с дискеты. Имеет смысл лишь отметить, что какой бы способ загрузки ни был выбран, необходимо обеспечить наличие загрузочной дискеты, незаменимой при возникновении аварийных ситуаций и особенно в тех случаях, когда читатель собирается экспериментировать с различными файлами и параметрами, описанными в данной главе, поскольку любая ошибка может привести к тому, что нельзя будет загрузиться с жесткого диска.

На машинах с Intel-архитектурой первый сектор любого диска называется *загрузочным сектором* (*boot sector*); он содержит таблицу разделов этого диска и, по необходимости, код для загрузки операционной системы. Загрузочный сектор первого жесткого диска называется *главной загрузочной записью* (MBR, *master boot record*), поскольку при загрузке системы BIOS передает управление программе, расположенной в этом секторе наряду с таблицей разделов. Эта программа называется *первичным загрузчиком* (*boot loader*); она запускает операционную систему. При установке Linux необходимо либо заменить первичный загрузчик, либо загружать Linux с дискеты.

В системе Linux каждый диск и каждый раздел считаются отдельным устройством. Так, к примеру, весь первый жесткий диск системы известен как */dev/hda*, а весь второй жесткий диск (если он есть) – как */dev/hdb*. Первый раздел первого диска – это */dev/hda1*, а второй раздел – */dev/hda2*. Первый раздел второго жесткого диска – */dev/hdb1* и т. д. Если установлены диски SCSI, а не IDE, то система имен работает таким же образом, с той единственной разницей, что устройства называются */dev/sda*, */dev/sda1* и т. д. Таким образом, запись */dev/hda2* указывает, что раздел Linux – это второй раздел первого жесткого диска (как в примерах этой главы).

Оставшаяся часть главы посвящена описанию различных методов загрузки системы Linux, а также параметрам, которые можно изменять с целью настройки и применяемого загрузчика, и ядра системы. Как LILO, так и Loadlin обеспечивают передачу параметров загрузчику и позволяют задавать параметры загружаемого ядра.

---

[1] LILO является стандартным загрузчиком для машин с архитектурой i386. На системах Alpha эквивалентный загрузчик называется MILO (Mini Loader, мини-загрузчик), а на системах Sparc – SILO.

LILO: загрузчик Linux

Приняв решение установить LILO, необходимо выбрать один из вариантов настройки загрузчика. Те, кто предпочитают с помощью двойной загрузки работать с Linux или Windows 95/98, могут установить LILO в главную загрузочную запись (MBR) и посредством его соответствующей настройки обеспечить возможность загрузки нужной системы. Двойная загрузка в комбинации Linux и Windows NT не столь прозрачна, поскольку Windows NT устанавливает в MBR свой собственный загрузчик и ожидает, что именно он будет основным. Следовательно, необходимо добавить Linux в варианты загрузчика Windows NT, а LILO установить в раздел с Linux в качестве вторичного загрузчика. В результате загрузчик Windows NT передает управление LILO, который и загружает Linux. То же относится к системе Windows 2000, в которой используется загрузчик NT. Подробности приведены в разделе «Двойная загрузка Linux и Windows NT/2000» далее в этой главе.

В дополнение к загрузке Linux LILO может загружать и прочие операционные системы, например MS-DOS, Windows 95/98 или OS/2. Все наиболее распространенные дистрибутивы Linux позволяют настроить и установить LILO в процессе установки самой системы; при необходимости его можно установить и позже. LILO может устанавливаться в главную загрузочную запись (MBR) жесткого диска или в качестве вторичного загрузчика в раздел, содержащий систему Linux. LILO состоит из нескольких частей – собственно загрузчика, файла настройки (*/etc/lilo.conf*), файла карты (*/boot/map*), содержащего информацию о расположении ядра, и программы **lilo** (*/sbin/lilo*), которая читает файл настройки и использует прочитанную информацию для создания или обновления файла карты и установки необходимых файлов LILO.

Если LILO устанавливается в MBR, он заменяет первичный загрузчик MS-DOS. Если вы столкнулись с проблемами в ходе его установки или просто хотите удалить LILO из системы и восстановить исходный загрузчик, можно воспользоваться одним из следующих вариантов:

- Загрузить Linux с дискеты и восстановить загрузочный сектор из резервной копии:

  ```
  % /sbin/lilo -u
  ```

- Загрузить DOS и запустить специальную команду **fdisk**, которая реконструирует MBR:

  ```
  c:> fdisk /mbr
  ```

Необходимо помнить, что LILO выступает в двух ипостасях: как собственно загрузчик и как команда **lilo**. Команда **lilo** конфигурирует и устанавливает загрузчик и обновляет его при необходимости. Загрузчик является кодом, который выполняется во время загрузки машины и запускает Linux или другую операционную систему.

Конфигурационный файл LILO

Информацию, необходимую для установки загрузчика, команда **lilo** получает из файла настроек */etc/lilo.conf*. Среди прочего, **lilo** создает файл карты, в котором содержится информация обо всех секторах диска, необходимых для загрузки.

Обратите внимание, всякий раз, когда вы изменяете */etc/lilo.conf*, пересобираете или перемещаете ядро системы, необходимо перезапускать **lilo**, чтобы обновить файл карты и загрузчик.

Конфигурационный файл начинается с раздела глобальных параметров, которые описаны ниже. Глобальные параметры не зависят от загружаемой системы. Вот пример раздела глобальных параметров (символ # является началом комментария):

```
boot = /dev/hda          # Загрузочное устройство /dev/hda
map = /boot/map          # Сохранить карту в файле /boot/map
install = /boot/boot.b   # Файл, устанавливаемый как новый загрузочный сектор
prompt                   # Всегда выдавать приглашение загрузчика
timeout = 30             # Установить ожидание в 3 (30 десятых долей секунды)
```

За разделом глобальных параметров следуют параметры загрузки каждого ядра Linux и других операционных систем – по одному разделу на каждый вариант. Каждый из разделов носит название раздела *образа*, поскольку описывает загрузку одного из образов ядра (в данном случае образ – синоним двоичного файла, содержащего код ядра) либо другой операционной системы. Каждый раздел образа для Linux начинается со строки **image=**. Например, раздел для Linux может выглядеть так:

```
image = /boot/vmlinuz    # Файл образа ядра
   label = linux         # Метка загрузки, вводимая в ответ на приглашение загрузчика
   root = /dev/hda2      # Расположение корневой файловой системы
   vga = ask             # Всегда предлагать пользователю выбор VGA-режима
   read-only             # Монтировать файловую систему в режиме "только для чтения"
                           и выполнять проверку fsck
```

Эквивалентный раздел для не-Linux системы начинается со строки **other=** вместо **image=**. Например:

```
other = /dev/hda1        # Местоположение раздела
   label = dos
   table = /dev/hda      # Расположение таблицы разделов
```

Итак, следует размещать параметры LILO, одинаковые для всех вариантов загрузки, в разделе глобальных параметров в файле */etc/lilo.conf*, а специфичные для каждого из вариантов – в соответствующем разделе образа. Если параметр дублируется в разделе глобальных параметров и в разделе образа, используется значение этого параметра из раздела образа.

Приведем пример полного файла */etc/lilo.conf* для системы, раздел */dev/hda2* которой содержит Linux.

```
## Глобальные параметры
boot = /dev/hda2
map = /boot/map
delay = 30
timeout = 50
prompt
vga=ask

## Раздел образа: обычная загрузка Linux
image = /boot/vmlinuz
  label = linux
  root = /dev/hda2
  install = /boot/boot.b
  map = /boot/map
  read-only

## Раздел образа: для тестирования нового ядра Linux
image=/testvmlinuz
  label = testlinu
  root = /dev/hda2
  install = /boot/boot.b
  map = /boot/map
  read-only
  optional         # Пропустить образ, если он недоступен во время создания карты

## Раздел образа: загрузка DOS
other = /dev/hda1
  label = dos
  oader = /boot/chain.b
  table = /dev/hda         # Текущая таблица разделов диска

## Раздел образа: загрузка Windows 95
other = /dev/hda1
  label = win95
  loader = /boot/chain.b
  table = /dev/hda
```

Глобальные параметры

Помимо описанных ниже, в качестве глобальных параметров для LILO также могут использоваться параметры ядра **append**, **read-only**, **read-write**, **root** и **vga** (описанные далее в разделе «Параметры, передаваемые ядру»).

backup=*backup-file*

Создавать резервную копию загрузочного сектора в файле *backup-file*, тогда как обычно резервная копия получает имя */boot/boot.nnnn*, где *nnnn* – число, соответствующее типу дискового устройства.

boot=*boot-device*

Установить имя устройства, содержащего загрузочный сектор. По умолчанию **boot** – это устройство, которое подмонтировано в качестве корневой файловой системы в настоящее время (например */dev/hda2*). Указание всего диска в качестве загрузочного устройства (например */dev/hda*) является для LILO предписанием устанавливать загрузчик в главную загрузочную запись. Альтернативой является установка в произвольный раздел диска, например в */dev/hda2*.

compact

Объединять запросы на чтение соседних секторов диска в целях ускорения процесса загрузки. Применение параметра **compact** рекомендуется, в частности, при загрузке с дискет. Но он может конфликтовать с параметром **linear**.

default=*name*

Использовать образ *name* в качестве образа загрузки по умолчанию. Если параметр **default** отсутствует, выбирается первый из образов, описанных в файле настройки.

delay=*tsecs*

Параметр задает время ожидания (в десятых долях секунды) перед началом загрузки образа по умолчанию. Если установлен параметр **serial**, значение **delay** не может быть меньше 20. По умолчанию ожидания не происходит.

disk=*device-name*

Определить параметры диска *device-name*, если LILO не в состоянии автоматически распознать их. В обычных условиях загрузчик самостоятельно определяет параметры устройства, и эти ухищрения не требуются. Задание параметра **disk** состоит из нескольких строк, например, таких:

```
disk = /dev/sda
  bios = 0x80      # Обычно значение для первого диска - 0x80, для второго - 0x81
  sectors = ...
  heads = ...
```

Обратите внимание, что этот параметр не совпадает с параметрами геометрии диска, которые можно определить при помощи параметров команды **hd**. Дело в том, что **disk** предоставляет информацию LILO, а **hd** – ядру системы. Далее кратко описаны ключевые слова, которые можно указывать с параметром **disk**. Подробную информацию можно найти в руководстве пользователя LILO, которое входит в комплект поставки загрузчика.

bios=*bios-device-code*

Число, используемое BIOS для адресации устройства. Смотрите предыдущий пример.

cylinders=*cylinders*

Количество цилиндров диска.

heads=*heads*

Количество головок диска.

inaccessible

Указание LILO, что BIOS не может получить доступ к диску; предотвращает последующие проблемы с загрузкой системы, если LILO считает, что BIOS имеет доступ к диску.

partition=*partition-device*

Начинает новый раздел. Раздел содержит единственную переменную, **start=***partition-offset*, которая определяет адрес первого сектора раздела (отсчет от нуля).

```
partition=/dev/sda1
  start=2048
```

sectors=*sectors*

Количество секторов на одной дорожке диска.

disktab=*disktab-file*

Этот параметр не используется по причине появления параметра **disk=**.

fix-table

Параметр позволяет **lilo** поправить адреса вида сектор/дорожка/цилиндр в таблице разделов. Иногда это бывает необходимо, если разделы не выровнены по границам дорожек, а на диске присутствует еще одна операционная система, например MS-DOS. Подробнее параметр описан на страницах руководства по файлу *lilo.conf*.

force-backup=*backup-file*

Аналогично **backup**, но приводит к перезаписыванию старой резервной копии, если таковая существует.

ignore-table

Указание **lilo** игнорировать поврежденные таблицы разделов.

install=*boot-sector*

Установить указанный файл в качестве нового загрузочного сектора. При отсутствии этого параметра используется файл */boot/boot.b*.

lba32

Генерировать 32-битные логические адреса блоков вместо адресов вида сектор/головка/цилиндр, позволяя загружать систему с любого раздела на жестком диске размером больше 8,4 Гбайт. Этот параметр позволяет преодолеть «барьер» в 1024 цилиндра. Требуется версия LILO не ниже 21-4 и поддержка интерфейса пакетных вызовов EDD в BIOS.[1]

---

[1] Если BIOS новее 1998 г., он должен поддерживать этот интерфейс.

linear

Использовать линейную адресацию дискового пространства, не зависящую от геометрии диска. Если LILO не может самостоятельно определить геометрию диска, можно указать параметр **linear**; если это не помогает, необходимо задать геометрию с помощью параметра **disk=**. Однако режим **linear** не всегда работает с дискетами и может конфликтовать с параметром **compact**.

lock

Указание LILO запоминать командную строку загрузки и каждый раз загружать систему, выполняя эту строку, пока она не будет изменена вводом новой строки в ответ на приглашение загрузчика. Параметр **lock** полезен, когда необходимо вводить набор параметров при каждой загрузке.

map=*map-file*

Определить расположение файла карты загрузки (по умолчанию – */boot/map*).

message=*message-file*

Определить файл, содержимое которого будет отображаться перед выдачей приглашения загрузчика. В файле могут содержаться символы новой страницы (<CTRL>+<L>) для очистки экрана. Если файл приветствия (*message-file*) изменяется или переносится в другой каталог, необходимо заново создать файл карты, выполнив команду **lilo**. Максимальная длина файла приветствия составляет 65 535 байт.

nowarn

Отключить предупреждающие сообщения.

optional

Указание **lilo**: не включать образ в список доступных, если он отсутствует в момент создания файла карты. Действует аналогично параметру **optional** для отдельного образа, но распространяется на весь файл настройки.

password=*password*

Определить пароль, который должен будет ввести пользователь, выбрав загрузку определенного образа. Пароль хранится в файле настройки в открытом виде (не шифруется), поэтому необходимо изменить права доступа к файлу таким образом, чтобы его мог читать/изменять только суперпользователь. Действует аналогично параметру **password** для отдельного образа, но распространяется на все образы, и у всех вариантов загрузки один и тот же пароль.

prompt

Автоматически выдать приглашение загрузчика, не ожидая нажатия клавиш <Shift>, <Alt> или <Scroll Lock>. Полезно помнить, что установка параметра **prompt** без параметра **timeout** предотвращает автоматическую загрузку системы без участия пользователя.

restricted

Применяется совместно с параметром **password** для указания загрузчику, что пользователь должен вводить пароль только в том случае, когда он определяет параметры загрузки в командной строке приглашения LILO. Действует аналогично параметру **restricted** для отдельного образа, но распространяется на все образы.

serial=*parameters*

Параметр позволяет загрузчику принимать ввод из последовательного порта, а не только с клавиатуры. Посылка сигнала разрыва в последовательный порт идентична нажатию клавиши <Shift>, влияющей на поведение загрузчика. Все загрузочные образы должны быть защищены паролями, если доступ организован небезопасным способом (например, через модем). Установка режима **serial** автоматически увеличивает значение **delay** до 20 (т. е. до двух секунд), если это значение было меньше. Формат строки *parameters* следующий:

```
port[,bps[parity[bits]]]
```

Например, можно инициализировать порт COM1 с параметрами по умолчанию таким образом:

```
serial=0,2400n8
```

Параметры:

port

Номер последовательного порта. По умолчанию равен 0, что соответствует порту COM1 (*/dev/ttys0*). Может принимать значения от 0 до 3, что соответствует четырем возможным последовательным портам.

bps

Скорость передачи информации в бодах. Корректные значения *bps*: 110, 300, 1200, 2400, 4800, 9600, 19200 и 38400. Значение по умолчанию равно 2400.

parity

Проверка четности. Значения *n* и *N* означают отсутствие контроля, *e* или *E* означают контроль по четности, а *o* или *O* – контроль по нечетности. Загрузчик, впрочем, игнорирует четность ввода и отсекает старший (восьмой) бит.

bits

Указание на количество бит в символе (7 или 8). По умолчанию – 8 без контроля по четности, иначе – 7.

timeout=*tsecs*

Ожидание ввода с клавиатуры (в десятых долях секунды). Если по истечении заданного времени не нажата ни одна клавиша, происходит автоматическая загрузка образа по умолчанию. Параметр **timeout** также ис-

пользуется для ограничения времени ожидания ввода пароля. По умолчанию ожидание не ограничено.

verbose=*level*

Включить подробную диагностику работы. Чем больше число *level*, тем подробнее диагностика. Если задан параметр **–v** в командной строке **lilo**, уровень диагностики увеличивается на единицу. Максимальный уровень равен 5.

Параметры загрузочного образа

Для каждого варианта загрузки могут быть заданы следующие параметры:

alias=*name*

Создать дополнительный синоним образа, который можно использовать наравне с именем, которое определено параметром **label**.

image=*pathname*

Определить файл или устройство, содержащее образ ядра системы Linux. С параметра **image** должен начинаться каждый раздел, описывающий загрузку Linux. *См.* также параметр **range**.

label=*name*

Определить имя образа загрузки, которое можно вводить в ответ на приглашение загрузчика. По умолчанию этим именем является имя файла образа (без пути к нему).

loader=*chain-loader*

Определение загрузчика, которому передается управление при выборе загрузки операционной системы, отличной от Linux. По умолчанию это */boot/chain.b*. Если система расположена не на дискете и не на первом диске, такой загрузчик должен быть определен обязательно.

lock

Аналогично ключевому слову **lock**, описанному в разделе глобальных параметров. Может также указываться для отдельного образа.

optional

Пропускать образ загрузки, если он недоступен в момент создания файла карты с помощью команды **lilo**. Параметр полезен для описания тестовых ядер системы, которые не всегда могут быть в наличии.

password=*password*

Защитить загрузку образа паролем *password*. Пароли не шифруются в файле настроек, поэтому только суперпользователь должен иметь права на чтение этого файла.

range=*sectors*

Применяется в комбинации с параметром **image**, если образ определен как устройство (например **image**=*/dev/fd0*) для указания диапазона секторов, информация о которых будет занесена в файл карты загрузки. Сек-

тора (*sectors*) могут задаваться диапазоном *start-end* или как *start+number*, где *start* и *end* – номера секторов (отсчет от нуля), а *number* – количество включаемых секторов после сектора *start*. Если задан только один аргумент *start*, включается только этот сектор. Пример:

```
image = /dev/fd0
range = 1+512      # 512 секторов, начиная с сектора 1
```

restricted

Пароль требуется только при указании параметров в командной строке загрузчика.

table=*device*

Задать (для ОС, отличных от Linux) устройство, на котором располагается таблица разделов. Если параметр **table** опущен, загрузчик не передает информацию о разделах стартуемой операционной системе. Обратите внимание, что необходимо повторно выполнить */sbin/lilo* при изменении таблицы разделов. Этот параметр не может указываться совместно с **unsafe**.

unsafe

Может использоваться в разделе образа не-Linux операционной системы для указания, что загрузочный сектор не должен читаться во время создания файла карты. Если задан параметр **unsafe**, некоторые проверки не выполняются. Но этот параметр может пригодиться для запуска **lilo** без использования дискеты в том случае, если загрузочный сектор находится на диске фиксированного формата. Этот параметр нельзя указывать одновременно с **table**.

Параметры, передаваемые ядру

Следующие параметры могут быть заданы как в файле */etc/lilo.conf*, так и в ответ на приглашение загрузчика:

append=*string*

Добавляет параметр, заданный в строке *string*, к строке параметров, передаваемых загрузчиком ядру. Обычно указывается для уточнения определенных свойств аппаратной базы. К примеру, если в системе установлено более 64 Мбайт оперативной памяти (больше, чем может определить BIOS), то **append** используется таким образом:

```
append = "mem=128M"
```

initrd=*filename*

Определить файл, загружаемый в устройство */dev/initrd* при загрузке с RAM-диском. См. также параметры **load_ramdisk** (в разделе «Параметры, передаваемые ядру при загрузке»), **prompt_ramdisk**, **ramdisk_size** и **ramdisk_start** (в разделе «initrd: использование RAM-диска»).

literal=*string*

Действует аналогично **append**, но строка *string* заменяет все прочие параметры, передаваемые ядру.

noinitrd

Сохранять содержимое */dev/initrd* для однократного чтения после загрузки ядра.

prompt_ramdisk=*n*

Указание ядру выдавать (или не выдавать) приглашение вставить дискету, содержащую образ RAM-диска, который используется для продолжения загрузки. *n* принимает значения:

0 – Не выдавать приглашение. Обычно применяется для дискет, содержащих и ядро, и образ RAM-диска.

1 – Выдавать приглашение. Режим по умолчанию.

ramdisk=*size*

Не используется; в настоящее время возможно применение только с ядрами версии ниже 1.3.48. Для более новых ядер см. параметры **load_ramdisk** (в разделе «Параметры, передаваемые ядру при загрузке»), **prompt_ramdisk, ramdisk_size** и **ramdisk_start** далее в этом разделе.

ramdisk_size=*n*

Определяет объем памяти (в килобайтах), отводимый под RAM-диск. По умолчанию равен 4096, т. е. 4 Мбайт.

ramdisk_start=*offset*

Применяется для дискет, содержащих и ядро, и RAM-диск. Смещение в килобайтах (*offset*) определяет адрес начала образа RAM-диска на дискете.

root=*root-device*

Определяет устройство, монтируемое в качестве корневой файловой системы. Если указано специальное имя **current**, выбирается устройство, на котором подмонтирована корневая файловая система в настоящий момент. Значения по умолчанию определяются в образе ядра.

vga=*mode*

Определяет текстовый режим VGA, выбираемый при загрузке. По умолчанию режим *mode* задается настройками ядра. Имена режимов не чувствительны к регистру.

ask

Предоставить пользователю возможность выбирать текстовый режим. Нажатие клавиши <Enter> в ответ на запрос выдает список доступных режимов.

extended (или **ext**)

Выбрать текстовый режим – 80×50 знакомест.

normal

Выбрать обычный режим – 80×25 знакомест.

number

Установить текстовый режим, соответствующий числу (*number*). Список чисел и соответствующих режимов можно получить, загрузившись с параметром **vga=ask** и нажав клавишу <Enter>.

Команда lilo

Для установки и обновления загрузчика (при изменении ядра или файла настройки */etc/lilo.conf*) следует выполнить команду **lilo**.

Полный путь команды обычно */sbin/lilo*. Синтаксис команды:

```
lilo [options]
```

Некоторые из параметров соответствуют ключевым словам файла */etc/lilo.conf*.

Ключевое слово конфигурации	Параметр
boot=*bootdev*	**–b** *bootdev*
compact	**–c**
delay=*tsecs*	**–d** *tsecs*
default=*label*	**–D** *label*
disktab=*file*	**–f** *file*
install=*bootsector*	**–i** *bootsector*
lba32	**–L**
linear	**–l**
map=*mapfile*	**–m** *mapfile*
fix-table	**–P fix**
ignore-table	**–P ignore**
backup=*file*	**–s** *file*
force-backup=*file*	**–S** *file*
verbose=*level*	**–v**

По возможности эти параметры лучше описывать в файле настройки; использование их в командной строке **lilo**, а не в */etc/lilo.conf*, уже устарело. В следующем разделе описаны параметры, которые можно задать только в командной строке **lilo**; все прочие параметры описаны в предыдущем разделе.

Параметры команды lilo

Далее перечислены командные параметры **lilo**. Если параметров несколько, они задаются раздельно:

```
% lilo -q -v
```

–C *config-file*

Использовать альтернативный файл настройки. На основании данных файла настройки (по умолчанию */etc/lilo.conf*) **lilo** определяет файлы, входящие в карту загрузки.

–I *label*

Печатает на стандартный вывод путь к ядру, к которому относится имя *label*, либо сообщение об ошибке, если такая запись в файле настройки не найдена. К примеру:

```
% lilo -I linux
/boot/vmlinuz-2.0.34-0.6
```

–q Перечислить файлы, занесенные в текущую карту загрузки. **lilo** ведет специальный файл (по умолчанию */boot/map*), содержащий информацию о расположении загружаемых образов ядра. Выполнение **lilo** с этим параметром приводит к печати имен файлов из карты на стандартный вывод. Например (звездочкой обозначается образ по умолчанию):

```
% lilo -q
linux      *
test
```

–r *root-directory*

Предписание **lilo** выполнить команду **chroot** *root-directory* раньше всех остальных операций. Применяется для восстановления установки с дискеты – можно загрузиться с дискеты, но использовать файлы настроек с диска. К примеру, если выполнить следующие команды:

```
% mount /dev/hda2 /mnt
% lilo -r /mnt
```

то **lilo** будет обращаться к необходимым файлам на диске.

–R *command-line*

Установить команду для загрузчика, которая выполнится при следующем его запуске, а затем будет «забыта». Этот вариант, как правило, используется в сценариях перезагрузки системы перед вызовом **shutdown –r**.

–t Режим проверки. Реальной записи нового загрузочного сектора или создания файла карты не происходит. Можно использовать параметр совместно с **–v**, чтобы узнать, какие действия выполнит **lilo** при обычном запуске.

–u *device-name*

Удаляет **lilo**, восстанавливая загрузочный сектор из резервной копии */boot/boot.nnnn*, проверив время ее создания. *device-name* – это имя устройства, на котором установлен LILO, например */dev/hda2*.

–U *device-name*

Действует аналогично **–u**, но не проверяет время создания.

–V Отображение номера версии **lilo**.

Loadlin: загрузка из MS–DOS

Loadlin – это программа-загрузчик Linux, которая располагается на обычном MS-DOS разделе. Для работы загрузчика необходимо использовать реальный режим DOS, а не окно сеанса DOS в Windows. Установка загрузчика не требуется, нужна лишь копия исполняемого файла *loadlin.exe* из пакета Loadlin на разделе MS-DOS.[1] Понадобится также сжатое ядро Linux (например *vmlinuz*), которое можно загружать с дискеты, с RAM-диска или раздела DOS. Например, так:

```
C:> loadlin c:\vmlinuz root=/dev/hda2
```

Здесь происходит загрузка образа ядра *vmlinuz* и передача ядру параметра **root=/dev/hda2**, который указывает на корневой раздел Linux (*/dev/hda2*). Подробно работа с RAM-диском описана далее в этой главе в разделе «initrd: использование RAM-диска».

Если предполагается, что Loadlin будет работать с Windows 95/98, обратитесь к руководству пользователя Loadlin («Loadlin User Guide») и руководству «Linux+Win95 mini-HOWTO». Имейте в виду, что если установлена файловая система FAT32, обычные методы работы с Loadlin и Windows 95 неприменимы.В этом случае, либо если точно не известно, какая установлена файловая система (FAT16 или FAT32), прежде чем делать что-либо, обратитесь к указанному mini-HOWTO.

Loadlin может быть запущен из командной строки DOS либо из файла CONFIG.SYS, либо AUTOEXEC.BAT. Подобно LILO, Loadlin принимает как параметры, управляющие его работой, так и параметры, которые передаются ядру.

Возможны два варианта использования Loadlin:

```
LOADLIN @params
LOADLIN [zimage_file] [options] [boot_params]
```

Использование файла параметров

В первом варианте использования Loadlin *params* – это имя файла DOS, который содержит желаемые параметры работы Loadlin. В комплект поставки Loadlin входит образец такого файла, *test.par*, который можно взять за основу для создания собственного файла параметров. Для того чтобы задать имя ядра Linux (параметр **image=**), необходимо сделать его первым параметром в

[1] Loadlin можно загрузить с любого из крупных сайтов, посвященных Linux, например, с сайта Metalab *http://metalab.unc.edu/pub/Linux*.

файле. Комментарии начинаются с символа #. Описания параметров в файле могут дополняться или переопределяться в командной строке. Так, чтобы переопределить параметры **vga**, заданные в файле параметров, выполните следующую команду:

```
C:> LOADLIN @myparam vga=normal
```

Параметры в командной строке

Во втором варианте использования Loadlin *zimage_file* – это имя образа ядра, за которым следует перечень параметров Loadlin и/или параметров ядра. Команда **LOADLIN** без параметров отображает справку по применению параметров Loadlin и некоторых параметров, передаваемых ядру. Справка достаточно велика, поэтому ее целесообразно просматривать с помощью команды **more**:

```
C:> LOADLIN | more
```

Параметры Loadlin:

–clone

Пропустить определенные проверки – перед выполнением команды с этим параметром прочтите руководство пользователя LOADLIN.

–d ***file***

Режим отладки. Действует аналогично **–t**, но результат записывается не только на стандартный вывод, но и в файл.

–dskreset

Сброс дисков после загрузки, но перед стартом Linux.

–noheap

Режим только для «суровых» Linux-хакеров; отключение использования установочного резерва памяти.

–t Режим проверки. Выполняется прогон загрузки, но Linux не грузится. Также включает режим **–v**.

–txmode

Установка текстового режима 80×25 знакомест при запуске.

–v Подробная диагностика. Отображает параметры и информацию о настройках на стандартный вывод.

–wait=*nn*

После загрузки выждать *nn* квантов времени (DOS), прежде чем загружать Linux.

В дополнение к этим параметрам Loadlin в справке по программе описан также ряд параметров ядра, которые можно задавать. Реально параметров существует намного больше. Более полный перечень можно найти в руководстве «Bootprompt-HOWTO».

Двойная загрузка Linux и Windows NT/2000

Как уже говорилось ранее, при запуске Windows NT система считает, что основным загрузчиком должен быть именно загрузчик Windows NT; поэтому нормальный способ обеспечения двойной загрузки заключается в том, чтобы добавить Linux в качестве одного из вариантов загрузки в меню загрузчика Windows. Информация в этом разделе также применима к операционной системе Windows 2000, в которой используется загрузчик NT.

Для осуществления этого необходимо предоставить загрузчику NT копию загрузочного сектора Linux. Ниже описывается, как сделать это на машине с Windows NT и файловой системой NTFS (обратите внимание, Windows NT должна быть уже установлена). Более подробную информацию и некоторые альтернативы можно почерпнуть в руководстве «NT OS loader+Linux mini-HOWTO».

Для этого понадобится загрузочная дискета Linux, чтобы при необходимости можно было загрузить Linux до изменения загрузчика NT. Также нужна чистая, отформатированная в DOS дискета для переноса загрузочного сектора на раздел Windows NT. Если LILO уже установлен, может понадобиться изменить файл */etc/lilo.conf*, как показано ниже. В противном случае можно установить LILO в процессе инсталляции дистрибутива Linux либо с помощью сценария *QuickInst*, который входит в комплект поставки загрузчика. После установки LILO можно приступать к настройке двойной загрузки системы.

Помните, что нижеследующие инструкции предполагают существование Linux-раздела в */dev/hda2*. Если Linux установлен на другом разделе, измените инструкции соответствующим образом.

1. Задайте раздел, содержащий корневую файловую систему, в качестве устройства загрузки. Если */etc/lilo.conf* редактируется вручную, строка должна выглядеть так:

   ```
   boot=/dev/hda2
   ```

   и будет являться копией параметра **root=**.

2. Выполните команду **lilo**, чтобы установить LILO в раздел с Linux.

3. Теперь, для того чтобы перезагрузить Linux, необходимо воспользоваться загрузочной дискетой, потому что загрузчик NT еще не настроен для загрузки Linux.

4. В Linux выполните команду **dd**, для того чтобы создать копию загрузочного сектора.

   ```
   % dd if=/dev/hda2 of=/bootsect.lnx bs=512 count=1
   ```

   Эта команда копирует один блок размером 512 байт из исходного файла */dev/hda2* в конечный файл */bootsect.lnx*. (Имя конечного файла может быть любым, не обязательно *bootsect.lnx*.)

5. Скопируйте *bootsect.lnx* на дискету, отформатированную в DOS.

```
% mount -t msdos /dev/fd0 /mnt
% cp /bootsect.lnx /mnt
% umount /mnt
```

6. Перезагрузитесь в Windows NT и скопируйте загрузочный сектор с дискеты на диск. Например, с помощью команды копирования:

```
C:> copy a:\bootsect.lnx c:\bootsect.lnx
```

Неважно, где на диске будет расположен загрузочный сектор, потому что это расположение указывается загрузчику NT на шаге 8.

7. Измените атрибуты файла *boot.ini*[1], сняв атрибуты «системный» и «только для чтения», чтобы его было можно редактировать.

```
C:> attrib -s -r c:\boot.ini
```

8. Измените файл *boot.ini* с помощью текстового редактора, добавив следующую строку:

```
C:\bootsect.lnx="Linux"
```

В результате Linux будет добавлен в меню загрузки, а загрузчик Windows NT «узнает», где находится загрузочный сектор. Строка может находиться в любом месте раздела **[operating systems]** файла *boot.ini*. Расположение строки в файле определяет и расположение ее в меню загрузки, которое отображает загрузчик NT. К примеру, если добавить строку в конец секции, получится приблизительно следующее (вторая строка **multi(0)** разбита на две строки, чтобы соответствовать формату книги):

```
[boot loader]
timeout = 30
default=multi(0)disk(0)rdisk(0)partition(1)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Server Version 4.00"
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Server Version 4.00
  [VGA mode]" /basevideo /sos
C:>bootsect.lnx="Linux"
```

Для того чтобы сделать Linux системой, загружаемой по умолчанию, измените значение строки `default=` на следующее:

```
default=C:\bootsect.lnx
```

9. Повторно выполните команду **attrib**, чтобы восстановить атрибуты файла («системный» и «только для чтения»):

```
C:> attrib +s +r c:\boot.ini
```

---

[1] *boot.ini* является аналогом */etc/lilo.conf* в Windows NT. В этом файле определяются операционные системы, которые можно загружать с помощью загрузчика NT.

Теперь можно завершить работу с Windows NT и перезагрузить машину; загрузчик NT предложит меню, которое выглядит приблизительно так:

```
OS Loader V4.00

Please select the operating system to start:

Windows NT Workstation Version 4.00
Windows NT Workstation Version 4.00 [VGA mode]
Linux
```

Если выбрать Linux, то загрузчик NT прочитает загрузочный сектор Linux и передаст управление LILO, установленному на разделе Linux.

Если впоследствии пользователь изменит */etc/lilo.conf* или пересоберет ядро системы, то ему необходимо будет выполнить **lilo**, создать новый файл *bootsect.lnx* и обновить его на разделе с Windows NT. Таким образом, надо будет повторить шаги 2–6.

Если возникают какие-либо проблемы или просто требуется удалить LILO, можно выполнить всю процедуру в обратном порядке: загрузить Windows NT, изменить атрибуты файла *boot.ini*, удалить запись для Linux, сохранить файл, восстановить его атрибуты, а затем удалить загрузочный сектор Linux с раздела Windows NT.

Параметры, передаваемые ядру при загрузке

В разделах этой главы, посвященных LILO и Loadlin, описывались отдельные параметры, которые можно определять при загрузке. Таких параметров существует довольно много. В данном разделе рассматриваются способы передачи параметров загрузки ядру системы, а также описаны наиболее часто используемые параметры ядра (они влияют только на само ядро, а потому действуют независимо от того, какой применяется загрузчик).

Как это обычно бывает с Unix-системами, существует несколько вариантов загрузки. Если используется Loadlin, параметры можно передать ядру в командной строке или в файле.

Если же выбран загрузчик LILO, то можно добавлять или переопределять параметры, описанные в */etc/lilo.conf*, в ходе процесса загрузки следующим образом:

- Если в */etc/lilo.conf* присутствует ключевое слово **prompt**, LILO всегда отображает приглашение командной строки и ожидает ввода. В этот момент пользователь может выбирать, какую систему загружать, и (если это Linux) определять дополнительные параметры.
- Если ключевое слово **prompt** отсутствует, после отображения слова LILO в процессе загрузки можно нажать клавишу <Ctrl>, <Shift> или <Alt>, и приглашение появится. Также можно нажать клавишу <Scroll Lock>

до появления слова LILO, чтобы не ловить потом нужный момент, сосредоточившись над клавиатурой.

- В командной строке приглашения загрузчика определите систему, которую собираетесь загрузить, либо нажмите клавишу <Tab>, чтобы получить список доступных систем. Затем можно ввести имя загрузочного образа. Например:

```
LILO boot: <нажимаем Tab>
linux   test   dos
boot: linux
```

  Можно также добавлять параметры загрузки:

```
boot: linux single
```

- Если ничего не ввести, LILO будет ждать в течение времени, заданного параметром **delay**, а затем загрузит операционную систему по умолчанию с параметрами по умолчанию в соответствии с определениями в */etc/lilo.conf*.

Некоторые из параметров загрузки уже упоминались ранее. Прочие являются специфичными для аппаратной части и слишком многочисленны, чтобы их здесь рассматривать. Полный перечень параметров и подробное описание процесса загрузки можно найти в руководстве «Bootprompt-HOWTO». Часть полезных параметров, не рассмотренных ранее, освещены ниже, а оставшиеся – в упомянутом руководстве. Большинство этих параметров служат для передачи информации и инструкций ядру, а не загрузчику.

debug

Режим отладки, все сообщения ядра отображаются на консоли.

hd=*cylinders,heads,sectors*

Определение параметров геометрии жесткого диска для ядра. Полезно, если Linux испытывает сложности с самостоятельным определением геометрии, в особенности для дисков с IDE-интерфейсом, у которых более 1024 цилиндров.

load_ramdisk=*n*

Предписание ядру загружать (или не загружать) образ RAM-диска для использования во время установки Linux. Корректные значения *n*:

0 Не загружать образ. Режим по умолчанию.

1 Загружать образ с дискеты в RAM-диск.

mem=*size*

Определяет количество оперативной памяти, доступной в системе. Используется, если BIOS не отображает корректно объем памяти, определяя только 64 Мбайт, в то время как в системе установлено больше. *size* – это число с модификатором **M** или **k** (не чувствительны к регистру):

```
mem=128M
```

Чтобы не вводить параметр **mem** при каждой загрузке, его часто указывают в командной строке, сохраняемой с параметром **lock**, или добавляя к параметрам ядра с помощью **append**.

noinitrd

Наличие этого ключевого слова запрещает загрузку в два этапа и сохраняет содержимое */dev/initrd*, чтобы данные можно было читать после загрузки ядра. */dev/initrd* можно прочесть лишь один раз, затем его содержимое возвращается системе.

number

Стартует Linux в режиме (runlevel), определяемом числом *number*. Режим – это состояние, в которое можно загрузить систему, например, режим многопользовательской работы или режим конфигурирования системы с запущенной системой X Window. Режим может быть представлен числом от 1 до 6; по умолчанию обычно 3. Режимы и соответствующие им состояния системы определяются в файле */etc/inittab*. Более подробно режимы описаны на страницах руководства (manpage) по файлу */etc/inittab*.

ro Монтировать корневую файловую систему в режиме «только для чтения». Применяется, в частности, при проверке целостности файловой системы, когда изменение данных нежелательно.

rw Монтировать корневую файловую систему в режиме чтения и записи (read-write). Если не задан параметр **ro** или **rw**, используется значение по умолчанию (обычно **rw**), хранимое в образе загружаемого ядра.

single

Загрузка Linux в однопользовательском режиме. Режим предназначен для задач системного администрирования и восстановления. Производится минимальная инициализация системы и вход в систему суперпользователя. Никакие другие пользователи в систему не допускаются.

initrd: использование RAM-диска

Современные дистрибутивы Linux используют модульное ядро, что позволяет подключать модули без пересборки ядра. Если корневая файловая система расположена на устройстве, доступ к которому обеспечивается модульным драйвером (как это бывает, например, со SCSI-дисками), возможно использование **initrd**, двухэтапного процесса загрузки, позволяющего сначала создать содержащую необходимые модули (например, драйвер SCSI) временную корневую файловую систему на виртуальном диске в памяти (RAM-диск), а затем загрузить модули и смонтировать реальную корневую файловую систему. RAM-диск, содержащий временную файловую систему, – это специальный файл устройства */dev/initrd*.

Прежде чем использование **initrd** станет возможным, необходимо пересобрать ядро с поддержкой RAM-дисков (CONFIG_BLK_DEV_RAM=y) и под-

держкой загрузки RAM-дисков (CONFIG_BLK_DEV_INITRD=y). Затем необходимо подготовить обычную корневую файловую систему и создать образ RAM-диска. Возможно, ваш дистрибутив Linux включает утилиты, выполняющие некоторые установки за вас; например, в дистрибутив Red Hat входит утилита **mkinitrd,** предназначенная для создания **initrd**-образа. Более подробную информацию можно почерпнуть на страницах руководства (man-page) по **initrd** и в файле *initrd.txt* (путь может отличаться, но обычно это нечто вроде */usr/src/linux/Documentation/initrd.txt*).

Если система Linux подготовлена к использованию **initrd**, то можно выполнить одно из следующих действий (в зависимости от применяемого загрузчика):

- Если загрузчиком является LILO, добавьте параметр **initrd** в соответствующий раздел образа:

  ```
  image = /vmlinuz
    initrd = /boot/initrd      # Файл, загружаемый в качестве содержимого /dev/initrd
  ```

  Выполните команду **/sbin/lilo** и можете перезагружать систему с помощью **initrd.**

- Если используется Loadlin, добавьте параметр **initrd** к командной строке:

  ```
  loadlin c:\linux\vmlinuz initrd=c:\linux\initrd
  ```

Глава 5

Менеджеры пакетов Red Hat и Debian

В этой главе описаны две распространенные системы управления программными пакетами для Linux: Red Hat Package Manager (RPM) и Debian GNU/Linux Package Manager.

В большинстве случаев при установке приложения на Linux-системе пользователю доступен пакет с исполняемыми файлами или исходными текстами этого приложения вместо *.tar.gz*-архива (или вместе с ним). Пакет – это файл, который содержит файлы, необходимые для установки и работы конкретной программы. Обратите внимание, что несмотря на то что пакет содержит файлы, требующиеся для установки приложения, для работы последнего могут потребоваться и другие файлы или пакеты, отсутствующие в системе и не входящие в данный пакет, например, дополнительные библиотеки (или даже их конкретные версии). Такие требования называются *зависимостями*.

Системы управления пакетами предоставляют много преимуществ. Как пользователь вы можете получить список пакетов, установленных в системе, и версии этих пакетов. Как системный администратор вы должны применять специальные инструменты для установки и управления пакетами в системе. А если вы еще и разработчик, то вам необходимо знать, как собирать пакеты для их распространения.

Помимо этого, менеджеры пакетов:

- Предоставляют средства для установки, обновления, удаления приложений и отслеживания версий программ в системе.
- Позволяют устанавливать новые приложения или обновлять их по сети.
- Позволяют определить, к какому пакету принадлежит тот или иной файл или какие файлы содержатся в пакете.

- Ведут базу данных по пакетам, установленным в системе, и их состоянию, так что пользователь всегда может определить, какие версии пакетов у него установлены.
- Обеспечивают автоматическую проверку зависимостей, чтобы предотвратить установку несовместимых пакетов.
- Обеспечивают поддержку PGP, MD5 или других средств проверки подлинности пакетов.
- Предоставляют инструментарий для создания пакетов.

Каждый пользователь может получить список пакетов и делать запросы к базе данных. При этом установка, обновление и удаление пакетов обычно требуют привилегий суперпользователя по той причине, что пакеты традиционно устанавливаются в области файловой системы, которые доступны для записи только пользователю root. Иногда пользователь может указать альтернативный каталог для установки пакета, например, свой домашний каталог или каталог проекта, права на запись в который у него есть.

Как RPM, так и менеджер пакетов Debian создают резервные копии файлов перед обновлением пакета. Это позволяет не только совершать откат при возникновении проблем, но и восстанавливать конфигурационные файлы пакета, которые были изменены до обновления.

Менеджер пакетов Red Hat

Менеджер пакетов Red Hat (RPM) является свободно распространяемым средством управления пакетами для распространения и установки программного обеспечения. Помимо Red Hat и систем, основанных на Red Hat, RPM также используется в таких дистрибутивах Linux, как SuSE и Caldera.

Работать с RPM достаточно просто. Все функции сосредоточены в одной команде – **rpm**. Например, для того чтобы узнать, установлен ли в системе редактор Emacs, можно выполнить такую команду:

```
% rpm -q emacs
emacs-20.4-4
```

Кроме того, существует графический интерфейс к RPM – X-программа GNOME-RPM, которая может выполняться даже в отсутствие среды GNOME. В этом разделе описано использование команды **rpm** и команды **gnorpm**, запускающей GNOME-RPM.

Команда rpm

Пакеты RPM создаются, устанавливаются и опрашиваются командой **rpm**. Имена пакетов RPM обычно заканчиваются расширением *.rpm*. Эта команда имеет набор режимов работы, у каждого из которых есть свои параметры. Формат команды **rpm** следующий:

```
rpm [options] [packages]
```

За несколькими исключениями (отмеченными в следующих списках параметров) первый параметр определяет режим работы **rpm** (а именно: установка, опрос, обновление, сборка пакета и т. д.), а все последующие параметры влияют на этот режим.

В описании параметров, относящихся к пакетам, мы будем иногда указывать *package-name* (имя пакета), а иногда *package-file* (файл пакета). Имя пакета – это имя программы или приложения, например **gif2png**. Файл пакета – это имя файла RPM, т. е. *gif2png-2.2.5-1.i386.rpm*.

В RPM доступен файл настройки, содержащий часто применяемые параметры. Общесистемным файлом обычно является */etc/rpmrc*, а пользователи могут создавать собственные файлы *$HOME/.rpmrc*. Указав параметр **––showrc**, можно просмотреть значения, которые используются RPM в работе и могут быть установлены в файле *rpmrc*:

```
rpm --showrc
```

Программа **rpm** содержит реализацию FTP- и HTTP-клиентов, что дает возможность указания URL-адресов (*ftp://* или *http://*) пакетов для установки их через Интернет. URL может заменять любое вхождение параметра *package-file* в командах, описанных далее.

Опрос базы данных может производиться любым пользователем. Большая часть прочих функций требует привилегий суперпользователя.

Общие параметры

Следующие параметры могут применяться во всех режимах:

––dbpath *path*

path задает путь к базе данных RPM.

––ftpport *port*

Порт *port* отводится для доступа по протоколу FTP.

––ftpproxy *host*

Узел *host* выступает в качестве прокси-сервера при передаче данных. Параметр необходимо указывать при доступе по протоколу FTP из защищенной брандмауэром системы, в которой применяется прокси-сервер.

––help

Получить подробную справку по использованию (выполнение **rpm** без параметров приводит к получению краткой справки).

––justdb

Обновлять только базу данных; не изменять никакие файлы.

––pipe *command*

Передать вывод **rpm** на ввод команды *command*.

--quiet

Отображать только сообщения об ошибках.

--rcfile *filename*

Использовать указанный файл в качестве файла настроек, заменяющего общесистемный файл */etc/rpmrc* или *$HOME/.rpmrc*.

--root *dir*

Выполнять все операции в каталоге *dir*.

--version

Отобразить номер версии **rpm**.

-vv

Отобразить отладочную информацию.

Параметры install, upgrade и freshen

Установка или обновление RPM-пакета. Команда **install** имеет следующий формат:

```
rpm -i [install-options] package_file ...
rpm --install [install-options] package_file ...
```

Для того чтобы установить новую версию пакета и удалить при этом существующую, выполните команду **upgrade**:

```
rpm -U [install-options] package_file ...
rpm --upgrade [install-options] package_file ...
```

При использовании **-U**, если пакет отсутствует в системе, **rpm** ведет себя так, как если бы был указан параметр **-i** и просто устанавливает пакет. Обратного можно добиться с помощью команды **freshen**. **rpm** произведет обновление пакета только в том случае, если будет найдена предыдущая версия. Формат команды **freshen**:

```
rpm -F [install-options] package_file ...
rpm --freshen [install-options] package_file ...
```

Следующие параметры доступны для установки и обновления:

--allfiles

Установка или обновление всех файлов.

--badreloc

Применяется совместно с **--relocate**, для того чтобы выполнить принудительное перемещение, даже если пакет не является перемещаемым.

--excludedocs

Не устанавливать файлы документации.

--excludepath *path*

Не устанавливать файлы, имена которых начинаются с *path*.

––force

Принудительная установка. Эквивалента использованию **––replacepkgs**, **––replacefiles** и **––oldpackage**.

–h, ––hash

Отобразить 50 символов «#» в течение времени распаковки архива. Применяется совместно с **––version** для достижения наиболее эстетичного отображения.

––ignorearch

Производить установку, даже если пакет с исполняемыми файлами предназначен для другой архитектуры.

––ignoreos

Устанавливать пакет с исполняемыми файлами, даже если он предназначен для другой операционной системы.

––ignoresize

Не проверять наличие необходимого дискового пространства перед установкой.

––includedocs

Устанавливать файлы документации. Имеет смысл только в том случае, если в файле *rpmrc* указан параметр **excludedocs: 1**.

––nodeps

Не производить проверку зависимостей данного пакета от других.

––noorder

Не упорядочивать пакеты в целях удовлетворения зависимостей при установке.

––noscripts

Не выполнять никакие до- и пост-установочные сценарии.

––notriggers

Не выполнять никакие сценарии, вызываемые установкой пакета.

––oldpackage

Разрешить при обновлении заменять более новый пакет старым.

––percent

Отображать процентный прогресс при распаковке файлов.

––prefix *path*

Установить префикс установки в *path* для перемещаемых пакетов.

––replacefiles

Устанавливать пакеты, даже если произойдет замена файлов, принадлежащих другим пакетам.

--replacepkgs

Устанавливать пакеты, даже если некоторые из них уже установлены.

--test

Только проверка действий, выполняемых при установке; реальной установки не происходит.

Параметры запросов к базе данных RPM

Команда **query** имеет следующий синтаксис:

```
rpm -q[information-options] [package-options]
rpm -query[information-options] [package-options]
```

Существует два подмножества параметров этой команды: *параметры выбора пакетов*, которые определяют, какие пакеты необходимо отобрать, и *параметры выбора информации*, определяющие, какую информацию о пакетах требуется получить.

Параметры выбора пакетов

package_name

Получить информацию по указанному пакету.

-a, **--all**

Информация по всем установленным пакетам.

-f *file*, **--file** *file*

Выяснить, какому пакету принадлежит указанный файл.

-g *group*, **--group** *group*

Выяснить, какие пакеты принадлежат указанной группе.

-p *package_file*

Получить информацию по удаленному пакету.

--querybynumber *num*

Получить информацию по записи из базы данных, имеющей указанный номер. Команда полезна, в основном, для отладки.

-qf, **--queryformat** *num*

Указать формат для отображения запрошенной информации, используя теги для представления различных типов данных (например NAME, FILENAME, DISTRIBUTION). Спецификация формата является вариацией задания форматирования **printf**. (Используйте параметр **--querytags**, описанный в разделе «Различные параметры», чтобы получить список доступных тегов).

--specfile *specfile*

Получить информацию по указанному файлу описания пакета, считая его пакетом.

--triggeredby *pkg*

Перечислить пакеты, которые требуют установки пакета *pkg*.

--whatrequires *capability*

Перечислить пакеты, для работы которых необходима указанная возможность (*capability*).

--whatprovides *capability*

Перечислить пакеты, которые предоставляют указанную возможность.

Параметры выбора информации

-c, --configfiles

Перечислить файлы настройки в пакете.

--changelog

Отобразить журнал изменений пакета.

-d, --docfiles

Перечислить файлы документации в пакете.

--dump

Отобразить информацию по каждому файлу пакета. Этот параметр должен использоваться совместно по крайней мере с одним из параметров **-l**, **-c** или **-d**. Вывод включает следующую информацию в указанном порядке:

```
path size mtime md5sum mode owner group isconfig isdoc rdev symlink
```

--filesbypkg

Перечислить все файлы каждого пакета.

-i Отобразить информацию по пакету, включая имя, версию и описание.

-l, --list

Перечислить все файлы пакета.

--last

Перечислить пакеты, упорядочив их по времени установки. Пакеты, установленные последними, находятся в начале списка.

--provides

Перечислить возможности, обеспечиваемые данным пакетом.

-R, --requires

Перечислить все пакеты, которые необходимы для работы данного.

-s, --state

Перечислить файлы пакета с указанием состояния. Возможные состояния файла: **normal** (нормальное), **not installed** (не установлен) или **replaced** (заменен).

--scripts

Перечислить все специфичные для пакета сценарии командного интерпретатора, выполняемые при установке и удалении пакета.

Параметры удаления пакетов

Команда **uninstall** применяется следующим образом:

```
rpm -e package_name
rpm --erase package_name
```

Параметры команды:

--allmatches

Удалить все версии пакета. Должен быть указан только один пакет, иначе команда завершится с ошибкой.

--nodeps

Не проверять зависимости перед удалением пакета.

--noscripts

Не выполнять никакие сценарии, которые в обычном порядке выполнялись бы до или после удаления пакета.

--notriggers

Не выполнять никакие сценарии, выполнение которых связано с удалением этого пакета.

--test

Не производить действительное удаление; осуществить только проверку выполняемых действий.

Параметры проверки

Команда **verify** имеет следующий формат:

```
rpm -V|-y|--verify[package-selection-options]
```

В режиме проверки происходит сравнение информации об установленных файлах пакета с информацией о файлах, находящихся в исходном пакете, и отображение несоответствий. Сравнивается такая информация, как размер, сумма MD5, права доступа, тип, владелец и группа каждого файла. Удаленные файлы не обрабатываются.

Параметры выбора пакета включают все параметры, доступные в режиме запросов к базе данных, а также следующие:

--nofiles

Пропускать отсутствующие файлы.

--nomd5

Игнорировать ошибки несовпадения сумм MD5.

--nopgp

Игнорировать ошибки несовпадения PGP-подписи.

Вывод форматируется в виде строки из восьми символов, за которой может следовать символ «c», указывающий на файл настройки, а затем имя файла. Каждый из восьми символов строки представляет результат сравнения одной характеристики файла с соответствующей характеристикой файла, полученной из базы данных RPM. Точка (.) означает, что тест успешно пройден. Следующие символы означают несоответствие, найденное в процессе одной из проверок:

5 Сумма MD5

D Устройство

G Группа

L Символическая ссылка

M Режим (права доступа и тип файла)

S Размер файла

T Время изменения

U Пользователь

Параметры пересборки базы данных

Команда пересборки базы данных RPM имеет следующий формат:

```
rpm --rebuilddb [options]
```

Также можно создать новую базу данных:

```
rpm --initdb [options]
```

В режиме пересборки базы данных доступны параметры **--dbpath** и **--root**, описанные ранее в разделе «Общие параметры».

Параметры проверки подписи

Пакеты RPM могут быть подписаны PGP-подписями. Информация о настройках PGP читается из файла */etc/rpmrc*. Режим проверки подписи включается следующим образом:

```
rpm --checksig package_file...
rpm -K package_file...
```

Параметры, доступные в этом режиме:

--nogpg

Не проверять подписи GPG.

--nomd5

Не проверять подписи MD5.

--nopgp

Не проверять подписи PGP.

Два других параметра позволяют добавлять подписи к пакетам:

--addsign *binary-pkgfile*...

Сгенерировать и добавить новые подписи к уже существующим в указанных пакетах.

--resign *binary-pkgfile*...

Сгенерировать и добавить новые подписи к указанным пакетам, удаляя все существующие подписи.

Различные параметры

В **rpm** доступно несколько дополнительных параметров:

--querytags

Перечислить теги, доступные для использования с параметром **--queryformat** в режиме запросов.

--setugids *packages*

Установить владельца и группу для указанных пакетов в значения из базы данных.

--setperms *packages*

Установить права доступа для указанных пакетов в значения из базы данных.

--showrc

Отобразить значения, которые будут использованы **rpm** для всех параметров, устанавливаемых в файле *.rpmrc*.

Параметры FTP/HTTP

Следующие параметры доступны для применения с адресами *ftp://* и *http://* в режимах установки, обновления и запросов:

--ftpport *port*

Указанный порт назначается для FTP-соединения через прокси-сервер FTP вместо стандартного порта. Идентично указанию макроса **_ftpport**.

--ftpproxy *host*

Указанный узел используется в качестве прокси-сервера для передачи по протоколу FTP данных, проходящих через брандмауэр. Идентично указанию макроса **_ftpproxy**.

--httpport *port*

Указанный порт выделяется для HTTP-соединения через прокси-сервер HTTP вместо стандартного порта. Идентично указанию макроса **_httpport**.

––httpproxy *host*

Указанный узел используется в качестве прокси-сервера для передачи данных по протоколу HTTP. Идентично указанию макроса **_httpproxy**.

Параметры режима создания пакетов

Режим создания пакетов имеет следующий формат вызова:

```
rpm -[b|t]step [build-options] spec-file ...
```

Параметр **–b** является указанием создавать пакет прямо из файла описания пакета, **–t** – указанием открыть архив tar/gzip и использовать файл описания пакета из этого архива. В обоих вариантах следующие односимвольные ключи считаются **step**-аргументами:

p Выполнить стадию подготовки, распаковав исходные файлы и наложив «заплаты» (patches).

l Произвести проверку списка, подставляя макросы в разделе files файла описания пакета и проверяя существование каждого файла.

c Выполнить стадию сборки. Следует за стадией подготовки; обычно эквивалентно выполнению команды **make**.

i Выполнить стадию установки. Следует за стадиями подготовки и сборки; обычно эквивалентно выполнению команды **make install**.

b Создать пакет с исполняемыми файлами. Операция производится после стадий подготовки, сборки и установки.

s Создать пакет с исходными текстами. Операция выполняется после стадий подготовки, сборки и установки.

a Создать пакет с исполняемыми файлами и пакет с исходными текстами. Операция производится после стадий подготовки, сборки и установки.

Следующие дополнительные параметры доступны при создании файла **rpm**:

––buildarch *arch*

––buildos *os*

Для использования с RPM версии ниже 3.0. Позволяют производить сборку пакетов для архитектуры *arch* или операционной системы *os*. В версии 3.0 заменены параметром **––target**.

––buildroot *dir*

При сборке пакета заменить значение тега **BuildRoot** значением *dir*.

––clean

Удалить файлы сборки после создания пакета.

––rmsource

Удалить исходные файлы и файл описания пакета после завершения сборки. Может выступать в качестве самостоятельного параметра **rpm** для удаления файла независимо от создания пакетов.

--short-circuit

Может применяться вместе с параметрами **-bc** и **-bi** для пропуска предыдущих стадий.

--sign

Добавить подпись PGP к пакету.

--target *platform*

При создании пакета установить значения макросов **_target**, **_target_arch** и **_target_os** в значения, определяемые параметром *platform*.

--test

Проверка выполнения. Реального прохода через стадии не происходит. Параметр полезен для тестирования файлов описания пакета.

--timecheck

Установить время проверки файла (максимальный возраст файла, добавляемого в пакет, в секундах). Чтобы запретить проверку, установите значение в 0.

Еще два параметра могут использоваться самостоятельно для перекомпиляции или пересборки пакета:

--rebuild *source-pkgfile*...

Действует аналогично параметру **--recompile**, но создается также новый пакет с исполняемыми файлами. Каталог сборки, исходные файлы и файл описания пакета удаляются, когда пакет собран.

--recompile *source-pkgfile*...

Установить указанный пакет с исходными текстами, выполнить стадии подготовки, компиляции и установки пакета.

Примеры использования RPM

Найти в базе данных RPM все пакеты, имеющие отношение к редактору Emacs:

```
% rpm -q -a | grep emacs
```

Найти удаленный из системы пакет, отобразив информацию о нем и содержащихся в нем файлах:

```
% rpm -qpil ~/downloads/bash2-doc-2.03-8.i386.rpm
```

Установить пакет (требуются привилегии суперпользователя):

```
% rpm -i sudo-1.5.3-6.i386.rpm
```

GNOME-RPM

GNOME-RPM – это графический интерфейс пользователя к программе **rpm**, который выполняется под управлением системы X Window. Можно запус-

кать **gnorpm**, даже если GNOME не используется. При запуске **gnorpm** открывает окно, которое позволяет управлять пакетами **rpm** посредством графического интерфейса. Формат вызова **gnorpm** следующий:

```
gnorpm [options]
```

Параметры gnorpm

В **gnorpm** доступны следующие параметры:

––geometry=*geom*

Определить геометрические параметры главного окна в стандартном для системы X формате (т. е. $w \times h + x + y$).

–i *pkgfiles*, **––install** *pkgfiles*

Установить указанные пакеты.

–p *pkgs*, **––packages** *pkgs*

Пакеты находятся в файлах, а не в базе данных **rpm**, т. е. еще не были установлены.

–q *pkgs*, **––query** *pkgs*

Создать окно запроса для указанных установленных пакетов.

–qp *pkgfiles*, **––query ––packages** *pkgfiles*

Создать окно запроса для указанных файлов пакета. Идентично указанию параметров **–q** и **–p**.

–U *pkgfiles*, **––upgrade** *pkgfiles*

Обновить указанные пакеты.

–K *pkgfiles*, **––checksig** *pkgfiles*

Проверить подписи указанных пакетов.

–y *pkgs*, **––verify** *pkgs*

Проверить указанные пакеты.

–?, ––help

Отобразить справку и завершить работу.

––root=*dir*

Считать *dir* корнем файловой системы.

––usage

Отобразить краткую справку по использованию и завершить работу.

Окно GNOME-RPM

Главное окно GNOME-RPM делится на пять частей. Наверху расположено меню с тремя кнопками:

Packages (Пакеты)

Содержит пункты Query (Запрос), Uninstall (Удаление) и Verify (Проверка).

Operations (Действия)

Содержит пункты Find (Поиск), Web find (Поиск в Web), Install (Установка) и Preferences (Настройки).

Help

Вызов справки для GNOME-RPM.

Под меню располагается панель инструментов, на которой присутствуют кнопки Install (Установить), Unselect (Отменить выделение), Uninstall (Удалить), Query (Сделать запрос), Verify (Проверить), Find (Искать) и Web find (Искать в Web). В самом низу окна расположена строка состояния.

Оставшаяся часть окна представляет собой главную панель. Слева расположена панель пакетов, в которой отображено дерево пакетов. Щелчок по папке делает ее текущей, двойной щелчок отображает содержимое папки (т. е. пакеты, которые в нее входят) в правой части панели. Щелчок по пакету делает его текущим; выбрав пакет, можно воспользоваться меню и кнопками панели инструментов для совершения действий над пакетом. Можно выбрать одновременно несколько пакетов и работать со всей группой. Щелчок правой кнопкой мыши по пиктограмме пакета делает пакет текущим, если он еще не является таковым, и приводит к появлению меню с пунктами Query (Запрос), Uninstall (Удаление) и Verify (Проверка).

Более подробно работа с программой описана в документации и справочной системе GNOME-RPM.

Менеджер пакетов Debian

Система Debian GNU/Linux включает набор инструментов для управления пакетами, предназначенных для создания, установки и работы с пакетами, содержащими готовые исполняемые файлы. Имена пакетов Debian обычно имеют суффикс *.deb*. Инструменты Debian для работы с пакетами:

dpkg

До недавнего времени **dpkg** являлся самым важным инструментом для работы с пакетами Debian и до сих пор является в этом процессе основным. Применяется для установки или удаления программных пакетов либо в качестве интерфейса к **dpkg-deb**.

dpkg-deb

Инструмент для работы с пакетами более низкого уровня. Служит для создания и работы с пакетами Debian. Принимает и исполняет команды от **dpkg** либо может использоваться самостоятельно.

dselect

Интерактивный интерфейс к **dpkg**.

apt-get

Доступная в настоящее время часть инструмента Advanced Package Tool (APT), который все еще находится в разработке и создается как современ-

ное, удобное для пользователя средство управления пакетами. Может выполняться из командной строки или использоваться в виде метода из **dselect**. Одной из возможностей, предоставляемых **apt-get**, является способность программы получать и устанавливать файлы из Интернета при указании URL *ftp://* или *http://*. Помимо этого, **apt-get** можно использовать для обновления всех пакетов, установленных в системе, за один раз.

Файлы

Ниже перечислены важные файлы, с которыми работают инструменты управления пакетами Debian:

control

Есть в каждом пакете. Этот файл описывает зависимости этого пакета от других, содержит имя и версию пакета, описание пакета, данные о сотруднике, ведущем этот пакет, требования к дисковому пространству и т. д.

conffiles

Есть в каждом пакете и содержит перечень файлов настройки, связанных с пакетом.

preinst, postinst, prerm, postrm

Сценарии, которые могут включаться в пакет и выполняться, соответственно, до установки, после установки, до удаления и после удаления пакета.

/var/lib/dpkg/available

Содержит информацию о пакетах, доступных в системе.

/var/lib/dpkg/status

Содержит информацию о состоянии пакетов, доступных в системе.

/etc/apt/sources.list

Предназначенный для APT список источников, используемый для поиска пакетов. Источники перечислены в порядке предпочтения по одному на строке.

/etc/apt/apt.conf

Главный файл настроек APT.

Состояние пакета и состояние выбора

Ниже перечислены возможные состояния пакета:

config-files

В системе присутствуют только конфигурационные файлы программного пакета.

half-configured

Пакет распакован и его настройка была начата, но не завершилась.

half-installed

Установка была начата, но не завершилась.

installed

Пакет распакован и настроен.

not-installed

Пакет не установлен.

unpacked

Пакет распакован, но не настроен.

Возможные состояния выбора пакета:

deinstall

Пакет назначен для удаления (удаляются все файлы, кроме файлов настроек).

install

Пакет назначен для установки.

purge

Пакет назначен для уничтожения (удаляются все файлы, включая файлы настроек).

Флаги пакетов

Существуют два флага, которые могут быть установлены для пакета. А именно:

hold

Пакет не будет обрабатываться **dpkg**, если не указан параметр **––force-hold**.

reinst-required

Пакет поврежден и должен быть установлен повторно. Такой пакет не может быть удален, если не указан параметр **––force-reinstreq**.

Сценарии

В дополнение к командам, описанным в следующем подразделе, несколько сценариев интерпретатора и сценариев на Perl существуют в системе управления пакетами и предназначаются для создания пакетов.

dpkg-buildpackage

Сценарий интерпретатора, позволяющий автоматизировать процесс создания пакета.

dpkg-distaddfile

Сценарий на Perl, позволяющий добавлять записи для файлов к *debian/files*.

dpkg-genchanges

Сценарий на Perl, создающий файл управления загрузкой на основе информации из распакованного, прошедшего сборку дерева исходных текстов и файлов, полученных в результате сборки.

dpkg-gencontrol

Сценарий на Perl, который читает информацию из распакованного дерева исходных текстов и отображает управляющий файл для пакета с исполняемыми файлами на стандартный вывод.

dpkg-name

Сценарий интерпретатора, переименовывающий пакеты Debian, расширяя их имена до полных.

dpkg-parsechangelog

Сценарий на Perl, выполняющий чтение и разбор журнала изменений в распакованном дереве исходных текстов и выдающий информацию на стандартный вывод в машинночитаемом виде.

dpkg-scanpackages

Создает файл *Packages* на основе дерева пакетов с исполняемыми файлами. Файл *Packages* используется **dselect** для формирования списка пакетов, доступных для установки. Сценарий на Perl.

dpkg-shlibdeps

Ищет зависимости указанных исполняемых файлов с разделяемыми библиотеками. Сценарий на Perl.

dpkg-source

Сжатие и распаковка исходных архивов Debian. Сценарий на Perl.

Перечень команд менеджера пакетов Debian

apt-cdrom

apt-cdrom [*options*] *command*

Добавить новый компакт-диск к списку возможных источников пакетов для APT. В настоящее время единственной командой является **add**; присутствие этой команды обязательно (кроме случая вызова с параметром **--help**). База данных идентификаторов дисков хранится в файле */var/state/apt/cdroms.list*.

Параметры

Параметры можно указывать в командной строке либо задавать в файле настройки. Булевские параметры, заданные в файле настройки, могут быть отменены различными способами в командной строке, в частности, заданием **--no-***opt* или **-***opt*=**no**, где *opt* – символ или полное имя параметра.

apt-cdrom	**–a, ––thorough** Тщательный поиск пакетов. Может требоваться для некоторых старых компакт-дисков с Debian. **–c, ––config-file** Указать файл настройки, который будет прочитан после стандартного файла настройки. **–d, ––cdrom** Указать точку монтирования компакт-диска, которая должна присутствовать в файле */etc/fstab*. Параметр настройки – Acqцв::cdrom::mount. **–f, ––fast** Быстрое копирование. Считать, что все файлы корректны и не требуют проверки. Используйте этот параметр только в том случае, если ранее не наблюдалось ошибок при работе с этим диском. Параметр настройки – APT::CDROM::Fast. **–h, ––help** Отобразить справку и завершить работу. **–m, ––no-mount** Не производить монтирование и размонтирование устройства. Параметр настройки – APT::CDROM::NoMount. **–n, ––just-print, ––recon, ––no-act** Проверить все, но не вносить никаких изменений. Параметр настройки – APT::CDROM::NoAct. **–o, ––option** Установить параметр настройки. Формат команды: **–o** *group::tool=option* (например APT::CDROM=Fast). **–r, ––rename** Запрашивать новую метку для переименования диска. Параметр настройки –APT::CDROM::Rename. **–v, ––version** Отобразить информацию о версии и завершить работу.
apt-get	**apt-get** [*options*] *command* [*package* . . .] Инструмент командной строки для работы с пакетами. Предполагается, что он в итоге станет основой APT.

Команды

autoclean

Аналогично **clean**, но удаляются только файлы пакетов, которые более не могут быть загружены из сети.

clean

Очистка локального хранилища от полученных файлов пакетов.

check

Обновление кэша пакетов и проверка пакетов на целостность.

dist-upgrade

Аналогично **upgrade**, но при работе особое внимание уделяется зависимостям пакетов.

dselect-upgrade

Используется в паре с **dselect**. Отслеживает изменения поля состояния доступных пакетов, внесенные **dselect**, и выполняет действия, необходимые для приведения файлов в соответствие с их показателями состояния.

install *package*...

Установка одного или более пакетов. Необходимо указывать имена пакетов, а не полные имена файлов. Пакеты, необходимые для работы тех, которые указаны в списке *package*..., также устанавливаются. Добавление дефиса к имени пакета означает требование удалить пакет, если он в настоящий момент установлен.

remove *package*...

Удаление одного или нескольких пакетов. Необходимо указывать имена пакетов, а не полные имена файлов. Если к имени пакета добавлен плюс, пакет устанавливается.

source *package*...

Найти указанные пакеты с исходными текстами и скопировать их в текущий каталог. При использовании команды с параметром **--compile** компилируются пакеты, содержащие исполняемые файлы. Если задан параметр **--download-only**, пакеты не распаковываются.

update

Синхронизировать файлы обзоров пакетов с их источниками. Операция должна производиться перед выполнением команд **upgrade** и **dist-upgrade**.

apt-get

upgrade

Инсталлировать самые новые версии всех пакетов, установленных в настоящий момент. Предварительно выполнить команду **update**.

Параметры

Параметры могут указываться в командной строке или задаваться в файле настройки. Булевские параметры, заданные в файле настройки, могут быть отменены различными способами при вызове в командной строке, в частности, заданием **––no–***opt* или **–***opt***=no**, где *opt* – символ или полное имя параметра.

–b, ––compile, ––build

Скомпилировать пакеты после их получения.

–c, ––config-file

Задание файла настройки, который должен читаться после стандартного файла настройки.

–d, ––download-only

Получить файлы пакетов, но не распаковывать и не устанавливать их. Параметр настройки – APT::Get::Download-only.

–f, ––fix-broken

Попытаться исправить ошибки в системе с нарушенными зависимостями пакетов. Параметр может использоваться самостоятельно или с командой. Параметр настройки – APT::Get::Fix-Broken.

––force-yes

Автоматическое разрешение любых действий APT, вплоть до тех, которые могут повредить файловую систему. Использовать с большой осторожностью и только в случае крайней необходимости. Параметр настройки – APT::Get::force-yes.

–h, ––help

Отобразить справку и завершить работу.

––ignore-hold

Игнорировать флаг **hold** в пакетах. Может использоваться с командой **dist-upgrade** для удаления многочисленных флагов **hold**, которые более не нужны. Параметр настройки – APT::Get::Ignore-Hold.

–m, ––ignore-missing, ––fix-missing

Игнорировать отсутствующие или поврежденные пакеты, либо пакеты, которые не могут быть получены. При использовании совместно с **–f** могут возникать проблемы.

–no-download

Отключить получение пакетов. Совместно с **–ignore-missing** является указанием APT использовать только те пакеты, которые уже получены.

––no-upgrade

Не обновлять пакеты. Совместно с командой **install** предотвращает обновление уже установленных пакетов. Параметр настройки – APT::Get::no-upgrade.

–o, ––option

Установить параметр настройки. Формат:
–o *group::tool=option* (например APT::Get=force-yes).

––print-uris

Отображать URI-адреса файлов, но не копировать их. Отображается размер, имя конечного файла, путь и хеш MD5. Параметр настройки – APT::Get::Print-URIs.

–q, ––quiet

«Тихий» режим. Не отображать индикаторы хода работы, а только сообщения журнала. Для получения еще более «тихого» режима необходимо добавить **q**.

–s, ––simulate, ––just-print, ––dry-run, ––recon, ––no-act

Проверка. Выполнить все действия, но не вносить реальные изменения в систему. Параметр настройки – APT::Get::Simulate.

–u, ––show-upgraded

Перечислить пакеты, которые будут обновлены. Параметр настройки – APT::Get::Show-Upgraded.

–v, ––version

Отобразить номер версии и завершить работу.

–y, ––yes, ––assume-yes

Автоматически подтверждать все запросы и работать в неинтерактивном режиме. Работа завершается аварийно при возникновении ошибки. Параметр настройки – APT::Get::Assume-Yes.

dpkg

dpkg [*options*] *action*

Инструмент для сборки, установки и управления пакетами. Является интерфейсом к **dpkg-deb**.

Действия dpkg

Следующие действия выполняются непосредственно **dpkg**:

–i *pkgfile*, **––install** *pkgfile*

Установить указанный пакет. При использовании совместно с **–R** или **––recursive** *pkgfile* должен быть каталогом.

––unpack *pkgfile*

Распаковать пакет, но не производить настройку. При использовании совместно с **–R** или **––recursive** *pkgfile* должен быть каталогом.

––configure [*packages* | **–a** | **––pending**]

Произвести повторную настройку одного или нескольких распакованных пакетов. При указании ключа **–a** или **––pending** вместо списка пакетов обрабатываются все распакованные, но не настроенные пакеты.

–r, ––remove [*packages* | **–a** | **––pending**]
–P, ––purge [*packages* | **–a** | **––pending**]

Удалить или уничтожить один или несколько установленных пакетов (*packages*). При удалении удаляются все файлы, кроме файлов настройки, перечисленных в *debian/conffiles*; при уничтожении удаляются также и эти файлы. При указании ключа **–a** или **––pending** команда **dpkg** удаляет или уничтожает все нераспакованные пакеты, назначенные (в */var/lib/dpkg/status*) для удаления или уничтожения.

––print-avail *package*

Отобразить информацию по указанному пакету из файла */var/lib/dpkg/available*.

––update-avail *pkgs-file*
––merge-avail *pkgs-file*

Обновить записи о доступных файлах, содержащиеся в файле */var/lib/dpkg/available*. Эта информация используется **dpkg** и **dselect** для создания списка доступных пакетов. Вариант **update** приведет к замене этой информации содержимым *pkgs-file*, распространяемых как *Packages*. Вариант **merge** объединяет существующую информацию и данные из *Packages*.

–A *pkgfile*, **––record-avail** *pkgfile*

Обновить записи о доступных файлах, которые хранятся в файле */var/lib/dpkg/available,* информацией из *pkgfile*. Но основе этой информации **dpkg** и **dselect** создают список доступных пакетов. При использовании совместно с параметром **–R** или **––recursive** *pkgfile* должен быть каталогом.

––forget-old-unavail

Удалить информацию об удаленных и более недоступных пакетах.

––clear-avail

Удалить существующую информацию о том, какие пакеты доступны.

–l, **––list** [*pkg-name-pattern*]

Перечислить пакеты, имена которых соответствуют заданному шаблону. Если шаблон не задан, перечисляются все пакеты, сведения о которых есть в файле */var/lib/dpkg/available*. Шаблон может включать стандартные метасимволы интерпретатора команд и в этом случае, если требуется предотвратить обработку шаблона интерпретатором, заключается в кавычки.

–s *packages,* **––status** *packages*

Отобразить состояние одного или нескольких пакетов, прочитав соответствующие записи в базе данных */var/lib/dpkg/status*.

–C, **––audit**

Найти частично установленные пакеты и предложить варианты приведения их в рабочее состояние.

––get-selections [*pattern*]

Вывести список пакетов и их состояний на стандартное устройство. Если указан шаблон *pattern*, вывести данные о состоянии только тех пакетов, которые соответствуют шаблону.

––set-selections

Установить состояния пакетов, используя файл, читаемый со стандартного ввода.

dpkg

––yet-to-unpack

Найти удаленные пакеты, которые были выбраны для установки.

–L *packages*, **––listfiles** *packages*

Перечислить установленные файлы, принадлежащие указанному пакету или пакетам.

–S *filename-pattern,* **––search** *filename-pattern*

Выполнить поиск файлов, принадлежащих установленным пакетам. Шаблон может включать стандартные метасимволы интерпретатора команд и в этом случае, если требуется предотвратить обработку шаблона интерпретатором, заключается в кавычки .

––print-architecture

Отобразить целевую архитектуру.

––print-gnu-build-architecture

Отобразить GNU-версию целевой архитектуры.

––print-installation-architecture

Отображать архитектуру узла для установки.

––compare-versions *ver1 op ver2*

Выполнить сравнение двух версий, где *op* – логический оператор. Операторы `lt le eq ne ge gt` считают отсутствующую версию более ранней. Операторы `lt-nl le-nl ge-nl gt-nl` считают отсутствующую версию более поздней (при этом `nl` означает «не более поздняя чем»). Существует третий набор операторов (`< << <= = >= >> >`), который введен для обеспечения совместимости синтаксиса с используемым в файле управления. **dpkg** возвращает 0 при положительном результате (условие истинно) и ненулевое значение в ином случае.

––help

Отобразить справку и завершить работу.

––force-help

Отобразить справку о параметрах **––force**-*list* и завершить работу.

–Dh, ––debug=help

Отобразить отладочную справку и завершить работу.

––license

Отобразить информацию о лицензировании **dpkg** и завершить работу. Написание **––licence** также считается верным и приводит к тому же результату.

––version

Отобразить информацию о версии **dpkg** и завершить работу.

Действия dpkg-deb

Следующие действия могут быть указаны **dpkg** и будут переданы на выполнение **dpkg-deb**. См. также описание **dpkg-deb**.

–b *dir* [*archive*], **––build** *dir* [*archive*]

Создать пакет.

–c *archive*, **––contents** *archive*

Отобразить содержимое пакета.

–e, **––control** *archive dir*

Извлечь из пакета информацию об управлении.

–f *archive* [*control-fields*], **––field** *archive* [*control-fields*]

Отобразить управляющее поле или поля пакета.

–I *archive* [*control-files*], **––info** *archive* [*control-files*]

Отобразить информацию о пакете.

––fsys-tarfile *archive*

Отобразить файловую систему tar-файла, содержащегося в архиве.

–x *archive dir,* **––extract** *archive dir*

Извлечь файлы из пакета.

–X *archive dir,* **––vextract** *archive dir*

Извлечь файлы из пакета, отображая их имена.

Параметры

––abort-after=*num*

Произвести аварийное завершение после указанного количества ошибок. Значение по умолчанию равно 50.

dpkg

–B, ––auto-deconfigure

При удалении пакета автоматически «расконфигурировать»[1] любой другой пакет, который зависит от данного.

–D*octal*, **––debug=***octal*

Включить отладку. Восьмеричное значение *octal* определяет требуемый уровень подробности отладочной информации. Для получения возможных значений используйте **–Dh** или **––debug=help**. Значения могут комбинироваться логическим ИЛИ (OR) для получения желаемого результата.

–E, ––skip-same-version

Не устанавливать пакет, если программа этой версии уже существует в системе.

––force-*list*, **––no-force-***list*, **––refuse-***list*

Принудительное выполнение (или запрет на выполнение) операции. Параметры в списке *list* перечисляются через запятую. Команда **––force** отображает предупреждение, но работа продолжается. **––refuse** и **––no-force** приводят к остановке работы с ошибкой. Параметры для **force**/**refuse** следующие:

architecture

Выполнять обработку, даже если пакет предназначен для другой архитектуры.

auto-select

Выбирать пакеты для установки и отменять выбор для удаления пакетов. По умолчанию включен.

bad-path

Некоторые программы отсутствуют в стандартных путях поиска.

configure-any

Настройка любого ненастроенного пакета, от которого зависит данный пакет.

conflicts

Разрешить установку конфликтующих пакетов. Могут возникать проблемы из-за перезаписи файлов.

---

[1] Это означает, что при удалении какого-либо пакета удаляются также настройки всех пакетов, которые зависели от него, сами же пакеты остаются. – *Примеч. науч. ред.*

depends

Отображать предупреждения о неполадках, связанных с зависимостями.

depends-version

Отображать предупреждения о проблемах с версиями при проверке зависимостей, в иных случаях игнорировать.

downgrade

Установить пакет, даже если установлена уже более новая версия. Включен по умолчанию.

hold

Обрабатывать пакеты даже с включенным флагом **hold**.

not-root

Пытаться установить или удалить пакеты, даже если пользователь не обладает правами суперпользователя.

overwrite

Перезаписать файл из одного пакета таким же файлом из другого. Включен по умолчанию.

overwrite-dir

Перезапись каталога из одного пакета файлом из другого пакета.

overwrite-diverted

Перезаписывать файлы с отклонениями нормальными версиями.

remove-essential

Удалять жизненно важные пакеты. Обратите внимание, что это может привести к потере работоспособности системы.

remove-reinstreq

Удалять пакеты, которые содержат ошибки и помечены для переустановки.

–G Не устанавливать пакет, если более новая версия уже установлена. Действует так же, как и **––refuse-downgrade**.

––ignore-depends=*pkglist*

Проблемы, связанные с зависимостями пакетов из списка *pkglist*, генерируют предупреждения.

dpkg	**––largemem** Указание **dpkg** использовать столько памяти, сколько потребуется. **––new** Новый формат пакетов с исполняемыми файлами. Это параметр **dpkg-deb**. **––no-act** Выполнить все операции, но не записывать изменения. Используется для тестирования. Не забудьте указать этот параметр перед действием, в противном случае изменения могут быть записаны. **––nocheck** Игнорировать содержимое управляющего файла при создании пакета. Это параметр **dpkg-deb**. **–O, ––selected-only** Обрабатывать только пакеты, которые отмечены для установки. **––old** Старый формат пакетов с исполняемыми файлами. Это параметр **dpkg-deb**. **–R, ––recursive** Рекурсивная обработка файлов *.deb*, найденных в каталогах, указанных при помощи параметров **–A, ––install, ––unpack** или **––avail**, а также в подкаталогах. **–R, ––root=***dir,* **––admindir=***dir,* **––instdir=***dir* Изменение каталогов по умолчанию. Каталог **admindir** содержит административные файлы с информацией о состоянии пакетов и т. д.; по умолчанию это */var/lib/dpkg*. Каталог **instdir** – каталог, в который устанавливаются пакеты, по умолчанию – /. Изменение корневого каталога на *dir* автоматически изменит **instdir** на *dir*, а **admindir** – на */dir/var/lib/dpkg*. **––smallmem** Предписание **dpkg** экономить память при работе.
dpkg-deb	**dpkg-deb** *action* [*options*] Основная команда создания и управления пакетными архивами Debian. См. также **dpkg**.

dpkg-deb

Прямому вызову **dpkg-deb** часто предпочитают команду **dpkg** с передачей необходимых параметров команде **dpkg-deb**.

Действия

–b *dir* [*archive*], **––build** *dir* [*archive*]

Создать архив *archive* из поддерева файловой системы, начинающегося с каталога *dir*. Этот каталог должен содержать подкаталог *DEBIAN*, в котором находится файл управления и вся остальная информация такого рода. Если архив (*archive*) указан и является именем файла, пакет записывается в этот файл; если же имя архива не указано, пакет записывается в файл *dir.deb*. Если архив уже существует, он перезаписывается. Если указанный архив является именем каталога, **dpkg-deb** выполняет поиск информации, необходимой для генерации имени пакета, в файле управления. (Обратите внимание, что по этой причине невозможно использовать **––no-check** с именем каталога.)

–c *archive*, **––contents** *archive*

Отобразить содержимое файлового дерева архива.

–e, **––control** *archive dir*

Извлечь управляющую информацию из архива в каталог *dir*, создав его при необходимости.

–f *archive* [*control-fields*], **––field** *archive* [*control-fields*]

Извлечение информации из одного или нескольких полей управляющего файла архива. Если поля не заданы, отображается весь управляющий файл.

–h, **––help**

Отобразить информацию по использованию программы и завершить работу.

–I *archive* [*control-files*], **––info** *archive* [*control-files*]

Отображение информации о пакете с исполняемыми файлами. Если управляющие файлы не заданы, отобразить сводку содержимого пакета, в противном случае отобразить управляющие файлы в указанном порядке. Сообщение об ошибке выдается в стандартный поток ошибок, если какая-либо составляющая отсутствует.

––fsys-tarfile *archive*

Извлечь дерево файловой системы из пакета и выдать на стандартный вывод в виде архива **tar**. Может использоваться совместно с **tar** для извлечения из архива отдельных файлов.

dpkg-deb

--license

Отобразить информацию о лицензировании **dpkg-deb** и завершить работу. Написание **--licence** также считается верным и приводит к тому же результату.

--version

Отобразить информацию о версии **dpkg-deb** и завершить работу.

-x *archive dir*, **--extract** *archive dir*
-X *archive dir*, **--vextract** *archive dir*

Извлечение дерева файловой системы из архива в указанный каталог. Если каталог не существует, он будет автоматически создан. Выполнение команды **-x (--extract)** не сопровождается выводом, а команда **-X (--vextract)** перечисляет файлы по мере их извлечения. Для установки пакетов лучше выбрать команду **dpkg**, а не этот параметр.

Параметры

-D Включить отладку.

--new

Создавать архивы нового образца (режим по умолчанию).

--no-check

Не сверяться с управляющим файлом при создании архива. Это позволяет создавать некорректные архивы.

--old

Создавать архивы старого образца.

dpkg-split

dpkg-split [*action*] [*options*]

Разбивает пакет с исполняемыми файлами на части и позволяет собирать части вручную или в автоматическом режиме. В автоматическом режиме работает очередь секций, подлежащих сборке. Команда полезна для переноса пакетов на дискеты.

Действия

--a -o *output part*, **--auto -o** *output part*

Добавить секцию *part* к очереди автоматической сборки, и, если доступны все секции, собрать пакет *output*.

–d [*packages*], **––discard** [*packages*]

Удалить секции из очереди автоматической сборки. Если указаны какие-либо пакеты, удалить только секции, принадлежащие этим пакетам. В противном случае очередь очищается полностью.

–I *parts*, **––info** *parts*

Отобразить информацию об указанной секции или секциях.

–j *parts*, **––join** *parts*

Объединить указанные секции файла пакета. По умолчанию создается пакет с именем вида *package-version.deb*.

–l, ––listq

Отобразить содержимое очереди секций, ожидающих сборки, приведя имя пакета, секции пакета, находящиеся в очереди, и их объем в байтах.

–s *full-package* [*prefix*], **––***full-package* [*prefix*]

Разделить указанный пакет *full-package* на секции, именуемые в формате *prefixNofM.deb*. По умолчанию префикс принимает значение имени пакета без расширения *.deb*.

–h, ––help

Отобразить справку и завершить работу.

––license

Отобразить информацию о лицензировании **dpkg-split** и завершить работу. Написание **––licence** также считается верным и приводит к тому же результату.

––version

Отобразить информацию о версии **dpkg-split** и завершить работу.

Параметры

––depotdir

Указать альтернативный каталог *depotdir*, в котором хранятся секции в ожидании сборки. По умолчанию используется каталог */var/lib/dpkg*.

––msdos

Использовать MS-DOS-совместимые имена секций.

–Q, ––npquiet

Не выводить сообщение об ошибке, если в очереди сборки присутствует секция, не принадлежащая собираемому пакету.

dpkg-split

–o *output*, **––output** *output*

Использовать *output* в качестве имени собираемого пакета.

–S *num*, **––partsize** *num*

Ограничить максимальный размер секции числом *num* (Кбайт). Значение по умолчанию составляет 450 Кбайт.

dselect

dselect [*options*] [*action*]

Экранно-ориентированный интерфейс к **dpkg**. Основной пользовательский интерфейс для установки и управления пакетами. Информация по созданию пакетов приведена в описаниях команд **dpkg** и **dpkg-deb**.

Действия

Если при запуске **dselect** в командной строке не было указано действие, отображается следующее меню:

```
* 0. [A]ccess   Выбор метода установки.
  1. [U]pdate   Обновление списка доступных пакетов, если возможно.
  2. [S]elect   Выбор набора пакетов для установки, обновления или
                удаления.
  3. [I]nstall  Установка и обновление пакетов.
  4. [C]onfig   Настройка ненастроенных пакетов.
  5. [R]emove   Удаление ненужных пакетов.
  6. [Q]uit     Выход.
```

Звездочка (отображаемая здесь в первой строке) указывает на выбранный в данный момент пункт меню. Любой из пунктов меню может быть указан в командной строке в качестве действия (**access**, **update**, **select**, **install**, **config**, **remove**, **quit**), которое будет немедленно выполнено при старте. Например:

```
% dselect access
```

Если выбрать действие **quit**, то **dselect** завершит работу немедленно, не совершая никаких операций. Дополнительным действием, доступным в командной строке, является **menu**. Это действие эквивалентно выполнению команды без параметров.

Параметры

––admindir *dir*

Изменить каталог, в котором содержатся файлы с внутренними данными, на *dir*. По умолчанию это */var/lib/dpkg*.

–D [*file*], **––debug** [*file*] Включить отладку. Вывод записывать в файл, если указано его имя. **––help** Отобразить справку и завершить работу. **––license** Отобразить информацию о лицензировании **dselect** и завершить работу. Написание **––licence** также считается верным и приводит к тому же результату. **––version** Отобразить информацию о версии **dselect** и завершить работу.	**dselect**

Глава 6

Обзор командных интерпретаторов Linux

Командный интерпретатор является программой, которая представляет собой прослойку между пользователем и операционной системой. В качестве такой прослойки интерпретатор должен быть практически незаметен пользователю. Но его также можно применять для простого программирования.

В данной главе представлены три интерпретатора, часто встречающиеся в системах Linux – усовершенствованный интерпретатор Борна (**bash**, или Bourne-Again Shell), интерпретатор C shell (**csh**) и его расширенная версия **tcsh**, а также рассмотрены их основные отличия. Подробнее интерпретаторы описаны в главе 7 «bash: командный интерпретатор Борна» и главе 8 «Интерпретаторы csh и tcsh». (В некоторых дистрибутивах встречается также интерпретатор Корна, **ksh** – еще одна популярная версия интерпретатора Борна с аналогичными возможностями.)

В этой главе представлены следующие темы:

- Назначение командного интерпретатора
- Разновидности командных интерпретаторов
- Общие возможности
- Различия

Назначение командного интерпретатора

Интерпретаторы позволяют:

- Работать в диалоговом режиме
- Настраивать параметры сеанса работы с Linux
- Программировать

Работа в диалоговом режиме

Когда интерпретатор работает в режиме диалога с пользователем, он ожидает ввод команд, обрабатывает их (интерпретируя, в частности, специальные символы шаблонов), а затем выполняет. Командные интерпретаторы в дополнение к командам Linux также предоставляют набор команд, которые известны как *встроенные*.

Настройка параметров сеанса работы с Linux

Командный интерпретатор позволяет контролировать сеанс работы посредством набора переменных, которые определяют, к примеру, расположение домашнего каталога пользователя или почтовой очереди. Некоторые из переменных определяются системой, прочие можно задавать в файлах стартовой конфигурации, учитываемых интерпретатором при входе пользователя в систему. Файлы стартовой конфигурации могут также содержать команды Linux или команды интерпретатора, выполняемые сразу после входа в систему.

Программирование

Последовательность отдельных команд (команд интерпретатора или других команд Linux, доступных в системе), объединенных в одну программу, называется *сценарием командного интерпретатора*. По схожему механизму работают командные файлы в системе MS-DOS. **bash** считается интерпретатором с мощной поддержкой программирования, в то время как, по слухам, программирование в **csh** может оказаться опасным для здоровья.

Разновидности интерпретаторов команд

Существует много различных командных интерпретаторов для Linux. Здесь мы опишем лишь три самых популярных интерпретатора:

- Новый интерпретатор команд Борна (**bash**), основанный на стандартном интерпретаторе Борна, является стандартным для системы Linux.
- C shell (**csh**), который использует синтаксис языка C и обладает многочисленными удобствами для работы.
- Интерпретатор **tcsh** является расширенным вариантом **csh** и заменяет его во многих дистрибутивах Linux.

На подавляющем числе систем доступно более одного интерпретатора, и зачастую для написания сценариев применяется один интерпретатор, а для диалоговой работы – другой.

Когда пользователь входит в систему, запускается командный интерпретатор, определенный для него в файле */etc/passwd*. Последнее поле каждой записи файла представляет собой команду для запуска интерпретатора. Например:

Команда	Запускаемый интерпретатор
/bin/sh	Интерпретатор Борна
/bin/bash	Интерпретатор Борна (**bash**)
/bin/csh	Интерпретатор С (или **tcsh**)
/bin/tcsh	**tcsh**

Для того чтобы перейти к работе с другим интерпретатором, надо ввести соответствующую команду. Например, чтобы сменить **bash** на **tcsh** , наберите:

```
$ exec tcsh
```

Общие возможности

Приведенная ниже таблица является объединением общих черт интерпретаторов **bash**, **csh** и **tcsh**. Поскольку **tcsh** является расширенной версией **csh**, он предоставляет все возможности **csh** плюс ряд дополнений.

Символ/команда	Значение/действие
>	Перенаправление вывода
>>	Добавление вывода к файлу
<	Перенаправление ввода
<<	Документ «здесь» (перенаправление ввода)
|	Передача вывода в конвейер
&	Выполнение задачи в фоновом режиме
;	Разделение команд в строке
*	Любые символы в имени файла
?	Любой символ (только один) в имени файла
!*n*	Счетчик повторения команд *n*
[ ]	Поиск любых символов из заключенных в скобки
( )	Выполнение в отдельной сессии интерпретатора
` `	Подстановка результатов работы команды, данной в кавычках
" "	Частичное экранирование (выполняется подстановка переменных и команд)
\	Экранирование следующего символа

Символ/команда	Значение/действие
`$var`	Подстановка значения переменной *var*
`$$`	Идентификатор процесса
`$0`	Имя команды
`$n`	Аргумент команды с порядковым номером *n* ($0<n\leq 9$)
`$*`	Все аргументы в исходном виде
`#`	Начало комментария
`bg`	Фоновое выполнение задачи
`break`	Выход из цикла
`cd`	Смена текущего каталога
`continue`	Переход к следующей итерации цикла
`echo`	Отображение вывода
`eval`	Вычисление значений аргументов
`exec`	Запустить новый командный интерпретатор или другую программу
`fg`	Обычное выполнение задачи
`jobs`	Перечислить работающие задачи
`kill`	Принудительное завершение работающих задач
`newgrp`	Смена текущей группы
`shift`	Сдвиг позиционных параметров
`stop`	Приостановить выполнение фоновой задачи
`suspend`	Приостановить выполнение задачи
`umask`	Задать или отобразить права доступа, устанавливаемые для создаваемых файлов
`unset`	Удалить переменную или определение функции
`wait`	Дождаться завершения фонового процесса

Различия

В приведенной ниже таблице отображены возможности, различающиеся в этих трех командных интерпретаторах.

Значение/Действие	bash	csh	tcsh
Стандартное приглашение командной строки	$	%	%
Принудительное перенаправление	>|	>!	>!
Принудительное добавление		>>!	>>!
Присваивание значения переменной	*var=val*	set *var=val*	set *var=val*
Установка переменной окружения	export *var=val*	setenv *var val*	setenv *var val*
Количество аргументов	$#	$#argv	$#argv
Код возврата	$?	$status	$?
Выполнить команды из файла (*file*)	. *file*	source *file*	source *file*
Завершение цикла	done	end	end
Завершение оператора **case** или **switch**	esac	endsw	endsw
Обработка в цикле	for/do	foreach	foreach
Пример условного оператора **if**	if [$i -eq 5]	if ($i==5)	if ($i==5)
Завершение условного оператора **if**	fi	endif	endif
Установка ограничения ресурсов	ulimit	limit	limit
Чтение с терминала	read	$<	$<
Запретить изменение значения переменной	readonly		set -r
Поиск файлов ненулевого размера		-s	
Дополнение слова	Tab		Tab
Игнорировать прерывания	trap 2	onintr	onintr
Начало цикла **until**	until/do	until	until
Начало цикла **while**	while/do	while	while

Глава 7

bash: командный интерпретатор Борна

В этой главе представлены следующие темы:

- Обзор возможностей
- Запуск интерпретатора
- Синтаксис
- Переменные
- Арифметические выражения
- Журнал команд
- Встроенные команды
- Управление заданиями

Обзор возможностей

bash – это GNU-реализация стандартного интерпретатора Борна (классического интерпретатора команд Unix-систем), обладающая многими получившими признание возможностями интерпретаторов **csh**, **tcsh** и **ksh** (интерпретатор Корна). В большинстве дистрибутивов Linux доступны как интерпретатор **tcsh**, описанный в следующей главе, так и **ksh**, ставший источником многих возможностей, упомянутых в данной главе. При этом **bash** является стандартным интерпретатором для Linux – он назначается по умолчанию при создании большинства учетных записей пользователей системы.

При входе пользователя в систему **bash** выполняет команды из файла */etc/profile*, затем команды из файла *~/.bash_profile*, *~/.bash_login* или *~/.profile* (каждый из файлов принимается во внимание только в том случае, если предшествующий файл или файлы не существуют). Во многих дистрибути-

вах стандартное поведение интерпретатора изменяется в файле */etc/profile*; иногда эти изменения затрагивают и обычные команды, подобные **ls**.

Кроме того, при любом запуске интерпретатора **bash** читает файл *~/.bashrc*. Многие инструменты системного администрирования создают небольшие файлы *~/.bashrc* автоматически, а многие пользователи – довольно объемные стартовые файлы. В этих файлах может присутствовать любая команда, которая выполнима в интерпретаторе. Ниже приведен пример небольшого файла; подробности происходящего можно понять, обратившись либо к текущей главе, либо к главе 3 «Команды Linux».

```
# Количество хранимых в журнале команд.
HSTSIZE=50
#
# Отображать в приглашении интерпретатора текущий рабочий каталог
# и номер команды в журнале.
PS1='\w: Command \!$ '
#
# Выполнять поиск программ сначала в каталогах пользователя,
# а затем в стандартных путях.
PATH=~/bin:~/scripts:$PATH
#
# Запретить группе и другим пользователям производить запись
# в созданные данным пользователем файлы
umask 022
#
# Обозначать тип файлов цветом.
alias ls='ls --color=yes'
#
# Выделить исполняемые и объектные файлы неприятным желтым цветом,
# чтобы их было проще найти и удалить.
export LS_COLORS="ex=43:*.o=43"
#
# Быстрая проверка программы, состоящей из одного файла.
function gtst () {
    g++ -o $1 $1.C && ./$1
}
#
# Удалить объектные файлы.
alias clean='find ~ -name \*.o -exec rm {} \;'
```

В **bash** доступны следующие возможности:

- Перенаправление ввода/вывода
- Специальные символы (маски) для сокращения имен файлов
- Переменные интерпретатора, позволяющие изменять параметры сеанса
- Мощная поддержка программирования
- Редактирование командной строки (с использованием команд редактирования в стиле **vi** или Emacs)

- Доступ к выполнявшимся ранее командам (журнал команд)
- Целочисленная арифметика
- Арифметические выражения
- Сокращение имен команд (применение псевдонимов)
- Управление заданиями
- Интегрированные возможности для программирования
- Управляющие структуры
- Работа со стеком каталогов (посредством **pushd** и **popd**)
- Расширения фигурной скобки и тильды
- Клавиатурные сокращения

Запуск интерпретатора

Командный интерпретатор **bash** может быть запущен следующим образом:

bash [*options*] [*arguments*]

bash может выполнять команды с терминала (при заданном параметре **–i**), из файла (когда первым аргументом является имя исполняемого файла сценария) или со стандартного ввода (если нет аргументов или задан параметр **–s**).

Параметры

Параметры, указанные ниже с двумя (**––**) дефисами, можно предварять и одним (**–**) дефисом, но рекомендуется использовать два дефиса, поскольку это является стандартом.

–, ––

Считать все последующие строки аргументами, а не параметрами.

––dump-po-strings

Аналогично действию **––dump-strings**, но использует специальный формат «переносимых объектов», более подходящий для сценариев.

––dump-strings

При выполнении с неанглийскими параметрами локализации выдавать все строки, преобразованные **bash**.

–c *str*

Читать команды из строки *str*.

–i Создать диалоговую среду (вывести приглашение командной строки).

––help

Отобразить информацию об установленной версии **bash** и перечень доступных параметров.

−−login

Работать в режиме интерпретатора, вызываемого при регистрации; пытаться обработать */etc/profile* при старте, затем в порядке уменьшения приоритета *~/.bash_profile*, *~/.bash_login* или *~/.profile*.

−−nobraceexpansion

Запретить расширение фигурных скобок.

−−noediting

Запретить редактирование командной строки посредством курсорных и специальных клавиш.

−−noprofile

Не обрабатывать файлы */etc/profile*, *~/.bash_profile*, *~/.bash_login* или *~/.profile* при запуске.

−−norc

Не обрабатывать файл *~/.bashrc* при запуске.

−p Запуск выполняется привилегированным пользователем; не обрабатывать *$HOME/.profile*.

−−posix

Режим работы в стандарте POSIX.

−r Предоставлять пользователю ограниченную среду повышенной безопасности. В частности, запрещается изменять рабочий каталог и применять символ > для перенаправления вывода.

−−rcfile *file*

При запуске использовать *file* вместо файла *~/.bashrc*.

−−restricted

Действует идентично **−r**.

−s Читать команды со стандартного ввода; вывод встроенных команд перенаправляется в файловый дескриптор с номером 1; весь остальной вывод – в файловый дескриптор с номером 2.

−v Отображать строки по мере выполнения (используется для отладки сценариев).

−−verbose

Действует так же, как **−v**.

−−version

Отобразить информацию об установленной версии **bash**.

−x Включить отладку (см. параметр **−x** в описании команды **set**).

−D При выполнении с неанглийскими параметрами локализации выдавать все строки, преобразованные **bash**.

Все прочие параметры запуска **bash** перечислены в описании встроенной команды **set**.

Аргументы

Аргументы присваиваются позиционным переменным **$1**, **$2** и т. д. в соответствии с их порядком. Если первый из аргументов является исполняемым сценарием, производится чтение команд из указанного файла, а значения оставшихся аргументов присваиваются переменным **$1**, **$2** и т. д.

Синтаксис

В этом подразделе описаны многочисленные особенности, специфичные для интерпретатора **bash**. Темы расположены следующим образом:

- Специальные файлы
- Специальные символы в именах файлов
- Редактирование командной строки
- Экранирование
- Формы команд
- Формы перенаправления
- Сопроцессы

Специальные файлы

Файл	Назначение
/etc/profile	Автоматически выполняется при входе в систему
$HOME/.bash_profile	Автоматически выполняется при входе в систему
$HOME/.bashrc	Автоматически выполняется при запуске интерпретатора
$HOME/.bash_logout	Автоматически выполняется при выходе из системы
$HOME/.bash_history	Журнал команд последнего сеанса работы
/etc/passwd	Информация о домашних каталогах пользователей для расширения *~name*

Спецсимволы в именах файлов

Символы	Значение
*	Любая строка, пустая или состоящая из произвольного количества символов
?	Один произвольный символ

Символы	Значение
[*abc...*]	Любой из перечисленных в скобках символов; для задания интервала может использоваться дефис (например, a–z, A–Z, 0–9)
[!*abc...*]	Любой из символов, отсутствующих в списке
{*str1, ...*}	Расширение группы: любая из перечисленных строк
~*name*	Домашний каталог пользователя с регистрационным именем *name*
~+	Текущий рабочий каталог (PWD)
~–	Предыдущий рабочий каталог (OLDPWD, см. также описание встроенной команды **pushd**)
~+*n*	Элемент в стеке каталогов с порядковым номером *n*. Первый элемент имеет номер 0. Отсчет от начала стека
~–*n*	Элемент в стеке каталогов с порядковым номером *n*. Первый элемент имеет номер 0. Отсчет от конца стека

Шаблон может быть представлен последовательностью шаблонов, разделенных символом «|». Если результат поиска для одного из подшаблонов положителен, то положительным считается и результат поиска для всего шаблона в целом. Расширенный синтаксис задания шаблонов схож с расширенным синтаксисом для **egrep** и **awk**.

Примеры

`$ ls new*`	*Вывести список файлов, имена которых начинаются на* new
`$ cat ch?`	*Работа производится с файлами, имена которых имеют вид* ch9, *но не* ch10
`$ vi [D-R]*`	*Редактировать файлы, имена которых начинаются на любую букву от прописной D до R*

Редактирование командной строки

Командная строка может редактироваться подобно тому, как редактируются строки в редакторах Emacs и **vi**. По умолчанию действуют сочетания клавиш в стиле Emacs. Дополнительная информация приведена далее в этой главе в разделе «Режим построчного редактирования».

Известны две разновидности режима редактирования в стиле **vi** – режим вставки и командный режим. По умолчанию принимается режим вставки; переключение режимов производится клавишей <Esc> либо (в командном режиме) командами **a** (добавление) и **i** (вставка).

На некоторых терминалах клавиши <Del> или <Backspace> не удаляют символ перед курсором, как это должно быть. Иногда ситуацию можно исправить выполнением следующих команд (или записью их в файл ~/*.bashrc*):

```
stty erase ^?
stty erase ^H
```

Команда **stty** подробно описана в главе 3 «Команды Linux». Для системы X Window альтернативным решением является применение команды **xmodmap**, которое не может быть здесь приведено, поскольку требует некоторых исследований конкретного пользовательского терминала.

В табл. 7.1–7.14 перечислены различные команды Emacs и **vi**.

Таблица 7.1. Основные команды Emacs-режима

Команда	Описание
Ctrl-B	Откат на один символ (без удаления)
Ctrl-F	Перемещение вперед по тексту на один символ
Del	Удаление символа в предыдущей позиции
Ctrl-D	Удаление символа в следующей позиции

Таблица 7.2. Команды работы со словами Emacs-режима

Команда	Описание
Esc b	Перемещение назад по тексту на одно слово
Esc f	Перемещение вперед по тексту на одно слово
Esc Del	Удаление слова перед курсором
Esc d	Удаление слова после курсора
Ctrl-Y	Возврат (из специального буфера) последнего удаления

Таблица 7.3. Команды работы со строками Emacs-режима

Команда	Описание
Ctrl-A	Перейти в начало строки
Ctrl-E	Перейти в конец строки
Ctrl-K	Удалить все символы от курсора до конца строки

Таблица 7.4. Команды Emacs-режима для навигации по файлу журнала команд

Команда	Описание
Ctrl-P	Переход к предыдущей строке
Ctrl-N	Переход к следующей строке
Ctrl-R	Поиск в обратном направлении
Esc <	Перейти к первой строке файла журнала команд
Esc >	Перейти к последней строке файла журнала команд

Таблица 7.5. Команды дополнений

Команда	Описание
Tab	Попытаться автоматически дополнить текст
Esc ?	Перечислить варианты возможных дополнений
Esc /	Попытаться автоматически дополнить имя файла
Ctrl-X /	Перечислить варианты возможных дополнений имени файла
Esc ~	Попытаться автоматически дополнить имя пользователя
Ctrl-X ~	Перечислить варианты возможных дополнений имени пользователя
Esc $	Попытаться автоматически дополнить имя переменной
Ctrl-X $	Перечислить варианты возможных дополнений имени переменной
Esc @	Попытаться автоматически дополнить имя узла (hostname)
Ctrl-X @	Перечислить варианты возможных дополнений имени узла
Esc !	Попытаться автоматически дополнить имя команды
Ctrl-X !	Перечислить варианты возможных дополнений имени команды
Esc Tab	Попытаться дополнить команду; источником вариантов дополнений является журнал команд

Таблица 7.6. Различные команды Emacs-режима

Команда	Описание
Ctrl-J	Идентично нажатию клавиши <Enter>
Ctrl-L	Очистить экран, поместив текущую строку в его верхней части
Ctrl-M	Идентично нажатию клавиши <Enter>
Ctrl-O	Идентично нажатию клавиши <Enter> с отображением следующей строки из журнала команд
Ctrl-T	Поменять местами символы справа и слева от курсора
Ctrl-U	Удалить символы от начала строки до текущего положения курсора
Ctrl-V	Отобразить символ нажатия клавиши, не интерпретировать его как команду
Ctrl-[	На большинстве клавиатур идентично нажатию клавиши <Esc>
Esc c	Изменить букву, на которой расположен курсор, на прописную
Esc u	Изменить все буквы слова над курсором или после него на прописные
Esc l	Изменить все буквы слова над курсором или после него на строчные
Esc .	Вставить последнее слово предыдущей команды после текущей позиции
Esc _	Идентично нажатию клавиши <Esc>

Таблица 7.7. Команды редактирования в режиме ввода vi

Команда	Описание
Del	Удалить символ перед курсором
Ctrl-W	Удалить предыдущее слово (т. е. до пробела)
Ctrl-V	Отобразить символ нажатия клавиши, не интерпретировать его как команду
Esc	Перейти в командный режим (см. табл. 7.8.)

Таблица 7.8. Основные команды управляющего режима навигации vi

Команда	Описание
h	Перейти влево на один символ
l	Перейти вправо на один символ
b	Перейти влево на одно слово
w	Перейти вправо на одно слово
B	Перейти к началу предыдущего непустого слова
W	Перейти к началу следующего непустого слова
e	Перейти к концу текущего слова
E	Перейти к концу текущего непустого слова
0	Перейти к началу строки
^	Перейти к первому непустому символу в строке
$	Перейти к концу строки

Таблица 7.9. Команды ввода в vi-режиме

Команда	Описание
i	Вставка текста перед текущим символом (вставка)
a	Вставка текста после текущего символа (добавление)
I	Вставка текста в начале строки
A	Вставка текста в конце строки
r	Замена текущего символа текстом
R	Режим замены существующего текста

Таблица 7.10. Некоторые команды удаления vi-режима

Команда	Описание
dh	Удалить предыдущий символ
dl	Удалить следующий символ
db	Удалить предыдущее слово
dw	Удалить следующее слово
dB	Удалить одно предыдущее непустое слово
dW	Удалить одно следующее непустое слово
d$	Удалить все символы до конца строки
d0	Удалить все символы до начала строки

Таблица 7.11. Сокращения команд удаления vi-режима

Команда	Описание
D	Эквивалентно **d$** (удаление до конца строки)
dd	Эквивалентно **0d$** (удаление всей строки)
C	Эквивалентно **c$** (удалить символы до конца строки, перейти в режим ввода)
cc	Эквивалентно **0c$** (удалить всю строку, перейти в режим ввода)
X	Эквивалентно **dl** (удалить следующий символ)
x	Эквивалентно **dh** (удалить предыдущий символ)

Таблица 7.12. Команды навигации по файлу журнала команд в режиме навигации vi

Команда	Описание
k или **–**	Перейти к предыдущей строке
j или **+**	Перейти к следующей строке
G	Перейти к строке, заданной счетчиком повторений
/*string*	Поиск строки *string* (в обратном направлении)
?*string*	Поиск строки *string* (прямой)
n	Повторить последний поиск
N	Повторить последний поиск в противоположном направлении

Таблица 7.13. Команды поиска символов vi-режима

Команда	Описание
f*x*	Перейти вперед к следующему вхождению символа *x*
F*x*	Перейти назад к предыдущему вхождению символа *x*
t*x*	Перейти вперед к следующему вхождению символа *x* и затем на символ назад
T*x*	Перейти назад к предыдущему вхождению символа *x* и затем на символ вперед
;	Повторить последнюю команду поиска символа
,	Повторить последнюю команду поиска символа, направление поиска поменять на противоположное

Таблица 7.14. Различные команды vi-режима

Команда	Описание
~	Инвертировать (сменить) регистр текущего символа (символов)
_	Добавить последнее слово предыдущей команды; перейти в режим ввода
Ctrl-L	Очистить экран и перерисовать текущую строку; применяется при «порче» изображения
#	Добавить символ комментария (#) к строке и строку в журнал команд; полезно таким образом сохранять команды, чтобы не приходилось в дальнейшем повторно их вводить

Экранирование

Экранирование отменяет специальное значение символа и позволяет использовать его буквально. Следующие символы имеют специальное значение для командного интерпретатора **bash**.

Символ	Значение
;	Разделитель команд
&	Фоновое выполнение
( )	Группировка команд (вызов дополнительного интерпретатора)
{ }	Блок команд
|	Конвейер
> < &	Символы перенаправления
* ? [ ] ~ !	Специальные символы, использующиеся для задания масок имен файлов

Символ	Значение
" ' \	Используются для экранирования символов
`	Подстановка результатов работы команд
$	Подстановка значений переменных (или результатов работы команд)
новая строка, пробел, табуляция	Разделители слов
#	Комментарий

Следующие символы могут применяться для экранирования:

Символ	Действие
" "	Все символы между кавычками понимаются буквально, кроме следующих, которые по-прежнему сохраняют специальное значение: $ Производится подстановка значения переменной ` Подстановка результата выполнения команды " Парные двойные кавычки закрывают последовательность
' '	Все символы между кавычками понимаются буквально, кроме символа одинарной кавычки, закрывающего последовательность
\	Символ, следующий за этим, понимается буквально. Используется внутри последовательности символов, заключенных в двойные кавычки, для экранирования символов ", $ и `. Часто применяется для экранирования символа \, пробела или символа новой строки

Примеры

```
$ echo 'Одинарные кавычки "защищают" двойные кавычки'
Одинарные кавычки "защищают" двойные кавычки

$ echo "Ну, разве это не \"особый случай\"?"
Ну, разве это не "особый случай"?

$ echo "У вас `ls | wc -l` файла в каталоге `pwd`"
У вас 43 файла в каталоге /home/bob

$ echo “Значением \$x является $x”
Значением $x является 100
```

Формы команд

Синтаксис	Действие
cmd &	Выполнить команду в фоновом режиме
cmd1 ; *cmd2*	Последовательность команд; выполнить команды, перечисленные в строке

Синтаксис	Действие
(*cmd1* ; *cmd2*)	«Подчиненный интерпретатор»; считать команды *cmd1* и *cmd2* группой команд
cmd1 | *cmd2*	Конвейер; использовать вывод команды *cmd1* как ввод для команды *cmd2*
cmd1 `*cmd2*`	Подстановка команд; результат работы команды *cmd2* передается в качестве аргументов команде *cmd1*
cmd1 $(*cmd2*)	Подстановка команд с допустимой вложенностью
cmd1 && *cmd2*	Логическое И; команда *cmd2* выполняется только в том случае, если успешно выполнена команда *cmd1*
cmd1 || *cmd2*	Логическое ИЛИ; команда *cmd2* выполняется только в том случае, если *cmd1* выполнена неудачно
{ *cmd1* ; *cmd2* }	Выполнять команды в текущем интерпретаторе

Примеры

`$ nroff file &`	*Фоновое форматирование текста*
`$ cd; ls`	*Последовательное выполнение команд*
`$ (date; who; pwd) > logfile`	*Перенаправление всего вывода в файл*
`$ sort file | pr -3 | lp`	*Сортировка файла, разбивка результата на страницы, печать*
`$ vi `grep -l ifdef *.c``	*Редактировать файлы, найденные grep*
`$ egrep '(yes|no)' `cat list```	*Определить список файлов для поиска*
`$ egrep '(yes|no)' $(cat list)`	*bash-эквивалент предыдущей команды*
`$ egrep '(yes|no)' $(<list)`	*Действует так же, как и предыдущая команда, но быстрее выполняется*
`$ grep XX file && lp file`	*Печать файла, содержащего заданный текст*
`$ grep XX file || echo "XX not found"`	*Вывод сообщения об ошибке, если текст XX не найден*

Формы перенаправления

Файловый дескриптор	Имя	Принятое сокращение	Типичное устройство
0	Стандартный поток ввода	stdin	Клавиатура
1	Стандартный поток вывода	stdout	Терминал
2	Стандартный поток ошибок	stderr	Терминал

Источник ввода и цель вывода могут быть переопределены, как показано в табл. 7.15.

Таблица 7.15. Символы перенаправления ввода/вывода

Символ	Действие
> *file*	Перенаправление стандартного потока вывода в файл *file*
< *file*	Получение стандартного потока ввода из файла *file*
cmd1 | *cmd2*	Конвейер. Стандартный поток вывода команды *cmd1* перенаправляется в стандартный поток ввода команды *cmd2*
>>*file*	Перенаправление стандартного потока вывода в файл *file* с добавлением данных к концу файла, если таковой уже существует
>| *file*	Принудительное перенаправление стандартного потока вывода в файл *file*, даже в режиме работы **noclobber**
n>| *file*	Перенаправление вывода файлового дескриптора *n* в файл *file*, даже в режиме работы **noclobber**
<> *file*	Использовать файл *file* и как стандартный поток ввода, и как стандартный поток вывода
<< *text*	Выполнять чтение из стандартного ввода до появления строки, идентичной строке *text* (строка может храниться в переменной окружения). Ввод обычно осуществляется с клавиатуры. Программы, с которыми чаще всего используется этот синтаксис, – **cat**, **echo**, **ex** и **sed**. Если вводимый текст *text* заключается в кавычки, то в нем не подставляются значения переменных, результаты работы команд и т. п.
n>*file*	Перенаправить файловый дескриптор *n* в *file*
n<*file*	Присвоить файлу *file* файловый дескриптор *n*
>&*n*	Дублировать стандартный поток вывода в файловый дескриптор *n*
<&*n*	Дублировать стандартный поток ввода из файлового дескриптора *n*
&>*file*	Перенаправить стандартный поток вывода и стандартный поток ошибок в файл *file*
<&–	Закрыть стандартный поток ввода
>&–	Закрыть стандартный поток вывода
n>&–	Закрыть вывод из файлового дескриптора *n*
n<&–	Закрыть ввод из файлового дескриптора *n*

Примеры

```
$ cat part1 > book
$ cat part2 part3 >> book
$ mail tim < report
$ grep Chapter part* 2> error_file
$ sed 's/^/XX /' << END_ARCHIVE
> This is often how a shell archive is "wrapped",
> bundling text for distribution. You would normally
```

```
> run sed from a shell program, not from the command line.
> END_ARCHIVE
XX This is often how a shell archive is "wrapped",
XX bundling text for distribution. You would normally
XX run sed from a shell program, not from the command line.
```

Для того чтобы перенаправить стандартный поток вывода в стандартный поток ошибок, можно использовать следующую команду:

```
$ echo "Usage error: see administrator" 1>&2
```

Следующая команда записывает результат (список файлов) в файл *filelist*, а сообщения об ошибках (файлы, к которым нет доступа) – в файл *no_access*:

```
$ find / -print > filelist 2>no_access
```

Сопроцессы

Сопроцессы являются уникальной возможностью интерпретатора **bash** и в других интерпретаторах не встречаются.

Синтаксис	Действие
cmd1 | *cmd2* |&	Сопроцессы; выполнять конвейерную последовательность в фоновом режиме. Командный интерпретатор создает двунаправленный конвейер, позволяя перенаправлять одновременно стандартный ввод и стандартный вывод.
read -p *var*	Читать ввод сопроцесса в переменную *var*.
print -p *string*	Передать строку *string* сопроцессу.
cmd <&*p*	Вводом для команды *cmd* являются данные сопроцесса.
cmd >&*p*	Вывод команды *cmd* перенаправляется сопроцессу.

Примеры

```
cat memo                    Отобразить содержимое файла
Sufficient unto the day is
A word to the wise.
ed - memo |&                Запустить сопроцессы
print -p /word/             Послать команду ed сопроцессу
read -p search              Прочитать вывод ed в переменную search
print "$search"             Отобразить строку на стандартный вывод
A word to the wise.
```

Переменные

Переменные предваряются символом $ и могут заключаться в фигурные скобки. Значения переменным можно присваивать с помощью знака равенства (=), пробелы по обе стороны которого не допустимы.

```
$ TMP=temp.file
```

По умолчанию переменные «видны» только в текущем сеансе интерпретатора. Чтобы передать переменную другим программам, выполняющимся в интерпретаторе, необходимо воспользоваться встроенной командой **export**.

Если переменная индексируется числом в квадратных скобках **([])**, она считается массивом. Например:

```
$ DIR_LIST[0]=src
$ DIR_LIST[1]=headers
$ ls ${DIR_LIST[1]}
```

Перечисляется содержимое каталога *headers*. Многие подстановки и команды в этой главе обрабатывают каждый элемент массива отдельно.

В этом разделе представлены:

- Подстановка переменных
- Встроенные переменные интерпретатора

Подстановка переменных

В приводимых ниже подстановках фигурные скобки являются необязательными, кроме тех случаев, когда необходимо отделить имя переменной от следующих символов, которые в ином случае считались бы частью этого имени.

Переменная	Значение
${*var*}	Значение переменной *var*.
$0	Имя программы.
${*n*}	Отдельные аргументы командной строки (позиционные параметры); Значение *n* лежит в интервале [1, 9].
$#	Количество аргументов командной строки.
$*	Все аргументы командной строки.
$@	Аналогично $*, но содержимое разделяется на слова, если переменная заключена в двойные кавычки.
$$	Номер процесса текущего интерпретатора; часто используется как часть имени файла при создании временных файлов с уникальными именами.
$?	Код завершения последней команды (0 обычно означает успешное завершение).

Переменная	Значение
$!	Номер процесса последней фоновой команды.
$-	Текущие параметры выполнения (см. описание встроенной команды **set**). По умолчанию для сценариев имеет значение **hB**, а для диалоговых интерпретаторов – **himBH**.
$_	Изначально имеет значение имени файла, выполненного для запуска текущего интерпретатора, затем заново устанавливается при выполнении каждой команды и имеет значение предыдущей команды, которая была выполнена.

Ниже в табл. 7.16.–7.18. перечислены различные типы операторов, которые могут применяться для работы с переменными в **bash**.

Таблица 7.16. Операторы подстановки

Оператор	Подстановка
${*varname*:-*word*}	Если переменная *varname* существует и ее значение не пустое, вернуть это значение; иначе вернуть *word*.
Назначение:	Возврат значения по умолчанию, если переменная не определена.
Пример:	**${count:–0}** принимает значение 0, если значение переменной **count** не определено.
${*varname*:=*word*}	Если переменная *varname* существует и ее значение не пустое, вернуть его; иначе установить значение переменной в *word* и вернуть значение *word*. Позиционные и специальные параметры не могут получать значения таким образом.
Назначение:	Установка значения переменной по умолчанию, если переменная не определена.
Пример:	**${count:=0}** устанавливает значение переменной **count** в 0, если значение переменной не определено.
${*varname*:?*message*}	Если переменная *varname* существует и ее значение не пустое, вернуть это значение; иначе отобразить имя *varname*, сопроводив его сообщением *message*, а затем прекратить выполнение текущей команды или сценария (только в пакетном режиме работы интерпретатора). Если параметр *message* опущен, выводится стандартное сообщение «parameter null or not set» (значение не существует или не установлено).
Назначение:	Предотвращение ошибок из-за неопределенности переменных.
Пример:	**{$count:?»undefined!»}** выдаст сообщение «count: undefined!» и завершит работу сценария, если переменная **count** не определена.

Оператор	Подстановка
${*varname*:+*word*}	Если переменная *varname* существует и ее значение не пустое, вернуть *word*; иначе вернуть пустое значение.
Назначение:	Проверка существования переменной.
Пример:	**${count:+1}** возвращает 1 (которую можно интерпретировать как **true**), если переменная **count** определена.
${#*varname*}	Вернуть длину переменной *varname*.
Назначение:	Подготовка к замене или извлечению подстрок.
Пример:	Если **${USER}** имеет значение **root**, то **${#USER}** принимает значение 4.

Таблица 7.17. Операторы поиска по шаблону

Оператор	Значение
${*variable*#*pattern*}	Если начало значения переменной соответствует шаблону, удалить соответствующую часть минимально возможной длины и вернуть остаток.
${*variable*##*pattern*}	Если начало значения переменной соответствует шаблону, удалить соответствующую часть максимально возможной длины и вернуть остаток.
${*variable*%*pattern*}	Если конец значения переменной соответствует шаблону, удалить соответствующую часть минимально возможной длины и вернуть остаток.
${*variable*%%*pattern*}	Если конец значения переменной соответствует шаблону, удалить соответствующую часть максимально возможной длины и вернуть остаток.
${*var*/*pat*/*sub*}	Вернуть значение переменной *var*, заменив первое вхождение *pat* на *sub*. Может применяться к переменным $* и $@; в этом случае каждое слово обрабатывается отдельно. Если строка *pat* начинается с символа #, происходит привязка к началу строки *var*; если строка *pat* заканчивается символом %, то привязка к концу строки *var*.
${*var*//*pat*/*sub*}	Вернуть значение *var*, заменив все вхождения *pat* на *sub*.
${*variable*:*n*}	Усечь начало значения переменной и вернуть часть, которая начинается с символа с порядковым номером *n*. Первый символ имеет номер 0.
${*variable*:*n*:*l*}	Вернуть подстроку длины *l*, которая начинается с символа с порядковым номером *n*. Первый символ имеет номер 0.

Таблица 7.18. Вычисление выражений

Оператор	Значение
$((*arithmetic-expression*))	Вернуть результат вычисления выражения. Арифметические операторы описаны в разделе «Арифметические выражения».
Пример:	При выполнении **TODAY='date +% –d' ; echo $((TODAY+7))** номер текущего дня недели сохраняется в **$TODAY**, затем к этому значению прибавляется 7 (номер этого же дня на следующей неделе) и сумма выводится на экран.
[[$*condition*]]	Вернуть значение 1, если условие *condition* истинно, в противном случае – 0. Синтаксис условий приведен в описании встроенной команды **test**.

Встроенные переменные интерпретатора

Встроенные переменные автоматически устанавливаются интерпретатором и, как правило, используются в сценариях. Встроенные переменные могут работать с шаблонами подстановок переменных, которые описаны выше. При установке значений переменных символ $ не применяется, но при их последующем использовании он необходим.

В табл. 7.19.–7.22 описаны наиболее часто встречающиеся встроенные переменные **bash**.

Таблица 7.19. Переменные, определяющие поведение

Переменная	Описание
auto_resume	Разрешить переход фонового процесса в режим приоритетного выполнения при вводе подстроки, содержащейся в командной строке процесса. Возможные значения: **substring** (введенная команда является подстрокой исходной), **exact** (введенная команда в точности совпадает с исходной) либо иное значение (введенная команда совпадает с началом исходной).
BASH_ENV	Файл, содержащий выполняемые при старте команды; используется при вызове **bash** для выполнения сценария.
CDPATH	Список каталогов, в которых выполняется поиск аргумента команды **cd**; элементы разделяются двоеточием.
EDITOR	Путь текстового редактора пользователя.
IFS	Разделитель слов; используется интерпретатором команд для разбора команд на элементы.
IGNOREEOF	Запретить применение одиночной комбинации **CTRL-D** для выхода из интерпретатора; при этом для выхода используется команда **exit**.

Переменная	Описание
PATH	Список каталогов, в которых происходит поиск выполняемых команд. Элементы списка разделяются двоеточием.
PROMPT_COMMAND	Команда, выполняемая **bash** перед выдачей следующего приглашения командной строки.
PS1	Первичное приглашение командной строки. В разделе «Переменные в приглашениях» описаны способы отображения динамически меняющейся информации в приглашениях интерпретатора, в частности, текущего рабочего каталога или номера команды в журнале.
PS2	Вторичное приглашение новой строки для незаконченной команды.
PS3	Приглашение встроенной команды **select**.
PS4	Приглашение, отображаемое при отладке с параметром **-x** (см. раздел «Запуск интерпретатора») и встроенной командой **set**.

Таблица 7.20. Переменные журнала команд

Переменная	Описание
FCEDIT	Путь редактора, используемого совместно с командой **fc**.
HISTCMD	Порядковый номер текущей команды в журнале команд.
HISTCONTROL	Если переменная имеет значение **ignorespace**, то строки, начинающиеся пробелом, не попадают в журнал команд. Если переменная имеет значение **ignoredups**, то в журнал команд не заносятся строки, совпадающие с последней строкой журнала. Значение **ignoreboth** объединяет оба случая.
HISTFILE	Имя файла журнала команд, с которым происходит работа.
HISTFILESIZE	Максимальное количество строк, хранимых в файле журнала команд. По умолчанию равно 500.
HISTSIZE	Максимальное количество команд, хранимых в журнале. По умолчанию равно 500.

Таблица 7.21. Почтовые переменные

Переменная	Описание
MAIL	Имя файла, который проверяется на наличие новых писем.
MAILCHECK	Интервал (в секундах) проверки наличия новой почты (60 секунд по умолчанию).
MAILPATH	Список имен файлов (элементы разделяются двоеточием), которые необходимо проверять на наличие новых писем.

Таблица 7.22. Переменные состояния

Переменная	Описание
BASH	Путь текущего экземпляра интерпретатора
BASH_VERSION	Номер версии текущего интерпретатора
COLUMNS	Количество доступных колонок дисплея
DIRSTACK	Перечень каталогов, с которыми работают команды **pushd** и **popd**
EUID	Эффективный идентификатор владельца процесса, выполняющего данную копию интерпретатора, в виде числа, опознаваемого системой
GROUPS	Список групп, к которым принадлежит пользователь, в виде чисел, опознаваемых системой
HOME	Имя домашнего (начального) каталога пользователя
HOSTNAME	Узел, на котором выполняется интерпретатор
HOSTTYPE	Краткое имя, указывающее тип машины, на которой выполняется интерпретатор; например i486
LINES	Количество доступных строк дисплея
MACHTYPE	Длинное имя, указывающее тип машины, на которой выполняется интерпретатор; например **i486-pc-linux-gnu**
OLDPWD	Предыдущий рабочий каталог
OSTYPE	Краткий идентификатор операционной системы; например «linux-gnu»
PPID	Идентификатор процесса, породившего текущий интерпретатор
PWD	Текущий рабочий каталог
SECONDS	Количество секунд, прошедших с начала работы интерпретатора в данном сеансе
SHELL	Путь текущего интерпретатора
SHLVL	Глубина вложенности работающих интерпретаторов
TERM	Тип терминала, с которым работает пользователь
UID	Реальный идентификатор процесса, вызвавшего текущий интерпретатор, в виде числа, опознаваемого системой

Арифметические выражения

Команда **let** позволяет выполнять целочисленные операции. Интерпретатор **bash** предоставляет способ подстановки целых значений (в качестве аргументов команд или в переменные), а также средства перехода от одной системы счисления к другой.

Выражение	Значение
`$(( expr ))`	Использовать значение арифметического выражения в скобках

Операторы

В **bash** применяются операторы языка программирования C; они представлены в нижеследующем списке в порядке понижения приоритета. Для изменения порядка вычисления выражений используются скобки.

Оператор	Значение
−	Унарный минус
! ~	Логическое отрицание; двоичная инверсия (дополнение единицы)
* / %	Умножение, деление, взятие остатка от деления
+ −	Сложение, вычитание
<< >>	Поразрядный сдвиг влево, поразрядный сдвиг вправо
<= >=	Меньше либо равно, больше либо равно
< >	Меньше, больше
== !=	Проверка на равенство и неравенство (ассоциативность – слева направо)
&	Поразрядное И (AND)
^	Поразрядное исключающее ИЛИ (OR)
|	Поразрядное ИЛИ
&&	Логическое И
||	Логическое ИЛИ
=	Присваивание значения
+= −=	Присваивание после сложения/вычитания
*= /= %=	Присваивание после умножения/деления/взятия остатка
&= ^= |=	Присваивание после выполнения операции поразрядного AND/XOR/OR
<<= >>=	Присваивание после выполнения операции поразрядного сдвига влево/вправо

Примеры

Более подробная информация приведена в описании встроенной команды **let**.

```
let "count=0" "i=i+1"   Присвоить значения переменным i и count
let "num % 2"           Проверка числа на четность
```

Журнал команд

bash позволяет отображать и редактировать ранее введенные команды. По механизму работы эта возможность сходна с журналом интерпретатора C shell. Команды из журнала можно изменять:

- В режиме построчного редактирования
- С помощью команды **fc**

Кроме того, подстановки команд, описанные в главе 8 «Интерпретаторы csh и tcsh», также действительны в **bash**.

Режим построчного редактирования

Режим построчного редактирования позволяет эмулировать многие возможности редакторов **vi** и Emacs. Список команд журнала хранится в файле. После запуска редактора можно с помощью команд перемещаться по строкам файла в поисках нужной строки. Клавиши управления курсором работают на большинстве терминалов как в режиме Emacs, так и в режиме **vi**. Перед применением найденной строки ее можно изменить. В табл. 7.23 приведены примеры распространенных команд редактирования строки. Для выполнения команды необходимо лишь нажать клавишу <Enter>.

По умолчанию в режиме построчного редактирования работают сочетания клавиш Emacs. Чтобы переключиться на **vi**-редактирование, введите команду:

```
$ set -o vi
```

Помните, что **vi** запускается в режиме вставки; для того чтобы ввести команду **vi**, необходимо сначала нажать клавишу <Esc>.

Режим, установленный для редактирования команд **bash**, никак не связан с выбором редактора, который автоматически запускается при выполнении многих команд (в частности, в почтовых клиентах при создании нового письма). Чтобы изменить стандартный редактор, необходимо установить переменную VISUAL или EDITOR в соответствующее значение (полный путь редактора):

```
$ export EDITOR=emacs
```

Таблица 7.23. Распространенные сочетания клавиш, используемые для редактирования

vi	Emacs	Действие
k	Ctrl-P	Получить предыдущую команду
j	Ctrl-N	Получить следующую команду
/*string*	Ctrl-R *string*	Получить предыдущую команду, которая содержит строку *string*
h	Ctrl-B	Переместить курсор на символ влево

vi	Emacs	Действие
l	Ctrl-F	Переместить курсор на символ вправо
b	Esc B	Переместить курсор на слово влево
w	Esc F	Переместить курсор на слово вправо
X	Del	Удалить предыдущий символ
x	Ctrl-D	Удалить один символ
dw	Esc D	Удалить слово после курсора
db	Esc H	Удалить слово перед курсором
xp	Ctrl-T	Поменять местами два символа

Команда fc

Команда **fc –l** перечисляет команды из журнала, а **fc –e** позволяет их редактировать. Более подробное описание **fc** см. в разделе о встроенных командах.

Примеры

`$ history`	*Вывести список команд из журнала*[1]
`$ fc -l 20 30`	*Перечислить команды с 20 по 30*
`$ fc -l -5`	*Перечислить пять последних команд*
`$ fc -l cat`	*Начиная с последней команды из журнала, вывести список команд, начинающихся с* `cat`[2]
`$ fc -ln 5 > doit`	*Сохранить в файле* `doit` *все команды, начиная с пятой (без номеров команд)*[3]
`$ fc -e vi 5 20`	*Редактировать с 5 по 20 команды с помощью* ***vi***
`$ fc -e emacs`	*Редактировать предыдущую команду с помощью Emacs*
`$ !!`	*Повторно выполнить последнюю команду*
`$ !cat`	*Повторно выполнить последнюю команду* `cat`
`$ !cat foo-file`	*Повторно выполнить последнюю команду* `cat`*, добавив файл* `foo-file` *в конец списка аргументов*

Подстановка команд

Синтаксис	Значение
!	Указание интерпретатору на начало подстановки из журнала команд
!!	Выполнить последнюю команду

[1] Обещанные 16 команд можно увидеть, если применить команду **history 16**. Команда **fc –l** выводит список последних 17 команд из журнала. – *Примеч. науч. ред.*

[2] Данная команда делает именно это, а не то, что обещано в книге. – *Примеч. науч. ред.*

[3] Аналогично предыдущему примеру действие команды отлично от обещанного. – *Примеч. науч. ред.*

Синтаксис	Значение
!*N*	Команда номер *N* из журнала
!–*N*	Команда, отстоящая от текущей на *N*
!*string*	Самая недавняя команда, начинающаяся со строки *string*
!?*string***?**	Самая недавняя команда, содержащая строку *string*
!?*string***?%**	Аргумент самой недавней команды, содержащий строку *string*
!$	Последний аргумент предыдущей команды
!#	Текущая команда до этой точки
!!*string*	Выполнить предыдущую команду с добавлением строки *string*
!*N string*	Выполнить команду с номером *N* с добавлением строки *string*
!{*s1***}***s2*	Самая недавняя команда, начинающаяся со строки *s1*; выполнить с добавлением строки *s2*
*^old^new^*	Быстрая замена; в последней команде заменить строку *old* на строку *new* и выполнить измененную команду

Переменные в приглашениях

Перечисленные ниже переменные позволяют выводить информацию о состоянии интерпретатора или системы непосредственно в приглашении командной строки. Необходимо инициализировать переменную PS1 строкой, включающей необходимые ключи-переменные. Например, следующая команда инициализирует PS1 строкой, которая содержит переменную **\w**, обеспечивающую отображение текущего рабочего каталога, и переменную **\!**, которая позволяет отобразить номер текущей команды. Следующая строка отражает изменение в приглашении.

```
$ PS1='\w: Command \!$ '
~/book/linux: Command 504$
```

Некоторые из переменных являются относительно новыми, как **\j** и **\l**, поэтому они могут не поддерживаться более старыми версиями **bash**.

Переменная	Результат подстановки
\a	Звуковой сигнал
\d	Дата в формате «Mon May 8»
\e	Терминальная escape-последовательность (не путать с символом обратного слэша)
\h	Имя узла
\j	Число фоновых заданий (активных или приостановленных)

Переменная	Результат подстановки
\l	Имя текущего терминала
\n	Символ новой строки
\r	Возврат каретки
\s	Текущий интерпретатор
\t	Время в 24-часовом формате; 3:30 p.m. отображается как 15:30:00
\u	Регистрационное имя пользователя
\v	Версия и выпуск **bash**
\w	Текущий рабочий каталог
\H	Аналогично **\h**
\T	Время в 12-часовом формате; 3:30 p.m. отображается как 03:30:00
\V	Версия, выпуск и patch-версия **bash**
\W	Последний компонент (следующий за последним разделителем /) имени текущего рабочего каталога
\\	Символ «обратный слэш»
\!	Порядковый номер текущей команды в журнале
\#	Порядковый номер текущей команды; отсчет начинается с момента запуска интерпретатора, первая команда имеет номер 1
\@	Время в 12-часовом формате; 3:30 p.m. отображается как 03:30 p.m.
\$	Индикатор пользователя; отображается как # для пользователя **root** и как $ – для всех остальных пользователей
\[	Начало последовательности неотображаемых символов; последовательность должна заканчиваться символами **\]**
\]	Окончание последовательности неотображаемых символов, начатой символами **\[**
nnn	ASCII-символ, соответствующий восьмеричному числу *nnn*

Встроенные команды

Примеры, которые можно использовать в командной строке, начинаются с приглашения $. В ином случае приведенный код можно рассматривать как пригодный к применению в сценариях. Для удобства включены некоторые из зарезервированных слов, используемых многострочными командами.

# Игнорировать весь текст, следующий за символом «#» в той же строке. Символ используется в сценариях для записи комментариев и по сути не является командой.	#
#!*shell* Используется в качестве первой строки сценария для вызова указанного командного интерпретатора (возможна передача аргументов). Некоторые из старых не-Linux систем не работают со сценариями, начинающимися с такой строки. Например: `#!/bin/bash`	#!

:
 	
 Пустая команда. Код возврата – 0. Если является первым символом файла, обозначает сценарий **bash**. За символом «:» может следовать переменная интерпретатора; в таком случае комбинация заменяется значением переменной.
 	
 Пример
 	
 Проверить, работает ли конкретный пользователь в системе:
 	
 `if who | grep $1 > /dev/null`
`then :  # ничего не происходит`
`  # если найдено соответствие шаблону`
`else echo “User $1 is not logged in”`
`fi`
	:

. *file* [*arguments*] Идентично **source**.	.
alias [**–p**] [*name*[='*cmd*']] Создать псевдоним для применения в качестве синонима команды *cmd*. Если опущен параметр ='*cmd*', отображается текущее значение для псевдонима *name*; если и параметр *name* опущен или указан параметр **–p**, отображаются все существующие псевдонимы. См. также **unalias**.	**alias**

bg	**bg** [*jobIDs*] Перевести текущее задание (или несколько заданий) в фоновый режим. См. раздел «Управление заданиями» далее в этой главе.
bind	**bind** [*options*] **bind** [*options*] *key*:*function* Отобразить связки или связать клавиши с функциями, такими как движение курсора или редактирование строки. Типичный синтаксис задания клавиш (*keys*): "**C-t**" для <Ctrl>+<T> и "**M-t**" или "**et**" для Esc T (кавычки необходимы для экранирования escape-последовательностей). Имена функций можно просмотреть, указав параметр **–l**. **Параметры** **–f** *filename* Добавить связки из файла *filename*. Формат описаний должен быть идентичен формату, применяемому при задании связок через параметры. **–l** Перечислить все функции Readline (которые являются функциями, связываемыми с клавиатурными сочетаниями). **–m** *keymap* Задать клавиатурную карту для последующих связок. Возможные варианты: **emacs**, **emacs-standard**, **emacs-meta**, **emacs-ctlx**, **vi**, **vi-move**, **vi-command** и **vi-insert**. **–p** Перечислить все функции и связанные с ними сочетания клавиш в формате, который устанавливается для задания связок. **–q** *function* Перечислить связки для функции *function*. **–r** *key* Удалить все связки для сочетания *key*. **–s** Перечислить все макросы и связанные с ними сочетания клавиш в формате, который используется для задания связок. **–u** *function* Удалить все связки для указанной функции.

bind

–v Перечислить все Readline-переменные (настройки, влияющие на редактирование журнала и строк) и их текущие значения в формате, применяемом для установки переменных.

–x *key:command*

Связать сочетание с командой интерпретатора (новая возможность; поддерживается не всеми версиями **bash**).

–P Перечислить все сочетания и функции, с которыми они связаны.

–S Перечислить все макросы и сочетания, с которыми они связаны.

–V Перечислить все Readline-переменные (настройки, влияющие на редактирование журнала и строк) и их текущие значения.

Пример

Привязать сочетание <Ctrl>+<T> к функции **copy-forward-move**, которая копирует часть слова, следующую за курсором, для последующей вставки:

```
$ bind "\C-t":copy-forward-move
```

break

break [*n*]

Выход из цикла **for**, **while** или **until** самой глубокой вложенности или из *n* циклов самой глубокой вложенности. Также выход из списка **select**.

builtin

builtin *command* [*arguments*]

Выполнить встроенную команду (*command*) интерпретатора. Используется при вызове встроенных команд в сценариях с таким же именем.

case

case *string*
in
regex)
commands
;;
. . .
esac

case	Если строка *string* соответствует регулярному выражению *regex*, выполнить следующие команды (*commands*). Продолжать перебор регулярных выражений, пока не будет получен положительный результат сравнения со строкой (в качестве последнего регулярного выражения можно использовать «*» для обработки строк, для которых не найдено соответствующее регулярное выражение).
cd	**cd** [*dir*] Команда без аргументов меняет текущий рабочий каталог на домашний каталог пользователя. В ином случае происходит смена рабочего каталога на каталог *dir*. Если *dir* является относительным путем, но находится не в данном каталоге, происходит поиск по переменной CDPATH.
command	**command** [*options*] *command* [*arguments*] Выполнить команду *command*; не выполнять поиск функций (т. е. команда не будет выполнена, если она не является встроенной или не найдена в путях переменной PATH). Код возврата равен коду возврата выполняемой команды, за исключением тех случаев, когда команда не найдена – тогда код возврата равен 127. **–p** Искать команду в каталогах по умолчанию, игнорируя значение переменной PATH. **––** Все последующие строки считаются аргументами, а не параметрами.
continue	**continue** [*n*] Принудительно перейти к следующей итерации цикла **for**, **while** или **until**, пропустив команды, которые еще не были выполнены в данной итерации. Если задан численный параметр, пропустить *n* итераций цикла.
declare	**declare** [*options*] [*name*[=*value*]] **typeset** [*options*] [*name*[=*value*]] Отобразить или установить переменные. Параметры, предваряемые знаком +, а не –, имеют противоположное значение. **–a** Считать следующие имена именами массивов.

declare

−f Считать следующие имена именами функций.

−i Считать переменную целочисленной и вычислить ее значение.

−p Вывести имена и значения всех переменных и функций интерпретатора; при использовании **−p** дополнительные параметры игнорируются.

−r Не допускать последующего сброса значений переменных.

−x Отметить переменные для последующего экспорта.

−F Отобразить имена всех функций интерпретатора; не совершать никаких действий помимо этого.

dirs

dirs [*options*]

Отобразить каталоги, расположенные в стеке в результате использования **pushd**/**popd**.

Параметры

+*entry*

Отобразить элемент стека с номером *entry* (первый элемент имеет порядковый номер 0).

−*entry*

Отобразить элемент стека с номером *entry* (отсчет от конца списка).

−l Подробный список.

disown

disown [*options*] [*jobIDs*]

Продолжить выполнение задачи, отделив ее от интерпретатора. По умолчанию не включать задачу в список активных, не отображать в командах **jobs** и **fg**. Если указан ключ **−h**, задача опознается, но не завершает свою работу при завершении работы породившей ее копии интерпретатора.

Параметры

−a Выполнить для всех заданий.

−h Не передавать задаче сигнал SIGHUP, получаемый интерпретатором.

echo

echo [*options*] [*string*]

Печать строки на стандартный вывод с завершающим символом новой строки. Если строка (*string*) опущена, печатается

echo

символ новой строки. В интерпретаторе **bash echo** является псевдонимом **print –**. См. также описание команды **echo** в главе 3 «Команды Linux».

–e Включить распознавание escape-последовательностей.

- **\a** Звуковой сигнал
- **\b** Забой
- **\c** Подавление символа новой строки (идентично **–n**)
- **\f** Новая страница
- **\n** Символ новой строки
- **\r** Возврат каретки
- **\t** Горизонтальная табуляция
- **\v** Вертикальная табуляция
- **** Обратный слэш
- *****nnn*
  Символ, ASCII-код которого равен восьмеричному значению *nnn*.
- **\x***nnn*
  Символ, ASCII-код которого равен шестнадцатеричному значению *nnn*.

–n Не добавлять символ новой строки при печати.

–E Отключить распознавание escape-последовательностей.

enable

enable [*options*] [*built-in* **...**]

Разрешить (с параметром **–n** – запретить) применение встроенных команд интерпретатора. Если опущен параметр *built-in* либо указан параметр **–p**, отобразить доступные встроенные команды; если задан параметр **–a**, отобразить состояние всех встроенных команд. Пользователь может отключать встроенные команды интерпретатора, чтобы заменить их собственными реализациями.

Параметры

–a Перечислить все встроенные команды – допустимые и запрещенные к применению.

–n *builtin*

Запретить использование указанной команды.

–p Перечислить допустимые команды.

enable

−s Ограничить перечисление специальными встроенными командами, определяемыми стандартом POSIX.

eval [*command args*...]

eval

Выполнить команду *command*, передав ей аргументы *args*.

exec [*options*] [*command*]

exec

Выполнить команду в рамках текущего процесса, не создавая нового процесса. **exec** также служит для открытия, закрытия и копирования файловых дескрипторов.

Параметры

−a *name*

Вызвать указанную команду с именем *name*.

−c Удалить все переменные среды для процесса при выполнении новой команды.

−l Приравнивать новый процесс ко входу пользователя в систему.

Примеры

`$ trap 'exec 2>&-' 0`	*Закрыть стандартный поток ошибок, когда сценарий интерпретатора завершит работу (сигнал 0)*
`$ exec /bin/tcsh`	*Заменить текущий интерпретатор интерпретатором C shell*
`$ exec < infile`	*Назначить infile стандартным вводом*

exit [*n*]

exit

Принудительно завершить выполнение сценария с кодом *n* (например, **exit 1**). Параметр *n* может принимать значение 0 (успешное завершение) и ненулевые значения (завершение с ошибкой). Если значение *n* не указано, код возврата будет равен коду возврата последней выполненной команды. Команду **exit** можно запускать из командной строки для завершения текущего сеанса работы.

Пример

```
if [ $# -eq 0 ]; then
   echo "Usage: $0 [-c] [-d] file(s)"
   exit 1     # Error status
fi
```

export

export [*options*] [*variables*]
export [*options*] [*name*=[*value*]] . . .

Передать (экспортировать) значение одной или нескольких переменных окружения (*variables*), сделав переменные глобальными (по умолчанию они локальны). Например, в сценарии определяется переменная, которая должна быть экспортирована, чтобы другие программы, выполняемые сценарием, могли использовать ее. Если не заданы переменные, команда **export** перечисляет переменные, уже экспортированные в данном интерпретаторе. Если задано имя (*name*) и значение (*value*), происходит присваивание значения переменной с указанным именем.

Параметры

−− Считать все последующие строки аргументами, а не параметрами.

−f Считать, что имена в списке *variables* являются именами функций.

−n Удалить переменную из области экспортированных.

−p Перечислить переменные, экспортированные в данном интерпретаторе.

fc

fc [*options*] [*first*] [*last*]

fc −e − [*old*=*new*] [*command*]

Отобразить или редактировать команды из журнала команд (параметры **−l** и **−e** являются взаимоисключающими). Команда **fc** предоставляет возможности, аналогичные работе с журналом команд в интерпретаторе C shell и с командой **!**. Параметры *first* и *last* являются номерами или строками, определяющими интервал отображения или редактирования команд. Если опущен параметр *last*, **fc** работает только с одной строкой (номер которой определяется параметром *first*[1]). Если параметр *first* также опущен, происходит редактирование последней команды или перечисление последних 16 команд. Вторая форма использования **fc** позволяет восстановить команду из журнала, заменить строку *old* на строку *new*, а затем выполнить измененную команду. Если строки не заданы, команда просто выполняется повторно.

[1] Но только при редактировании. Если указан параметр **−l**, опущенный параметр *last* приравнивается текущей команде. – *Примеч. науч. ред.*

fc

Если не задана команда, повторно выполняется последняя команда из журнала. Команда (*command*) является строкой либо числом, как и параметр *first*. См. примеры в разделе «Журнал команд».

Параметры

–e [*editor*]

Запустить редактор *editor* для редактирования выбранных команд из журнала. Редактор по умолчанию задается переменной интерпретатора FCEDIT.

–l [*first last*]

Перечислить команды из указанного интервала или 16 последних команд.

–n Подавить нумерацию команд при перечислении (**–l**).

–r Обратить порядок списка команд (**–l**).

–s *pattern=newpattern*

Редактировать команду (команды), заменяя все вхождения *pattern* на *newpattern*. Выполнить измененную команду.

fg

fg [*jobIDs*]

Перевести текущее задание (или задание с указанным идентификатором) в режим приоритетного выполнения. См. далее раздел «Управление заданиями».

for

for *x* [**in** *list*]

do
commands
done

Итеративно присваивать переменной *x* значения из списка (*list*) и выполнять команды (*commands*). Если список опущен, то в этом качестве выступают позиционные параметры сценария ($@).

Примеры

Разбить все файлы в текущем каталоге на страницы; сохранить результат для каждого файла:

```
for $file in *
do
      pr $file > $file.tmp
done
```

for	Найти слова из списка в главах книги (сходно с **fgrep –f**):
	`for item in `cat program_list``
	`do`
	`  echo "Глава проверяется"`
	`  echo "на наличие упоминаний программы $item…"`
	`  grep -c "$item.[co]" chap*`
	`done`

function	**function** *command* **{** **…** **}** Определение функции. Ссылка на аргументы происходит так же, как и на позиционные параметры сценария (**$1**, **$2** и т. д.). Определение функции завершается символом **}**.
getopts	**getopts** *string name* [*args*] Команда позволяет обрабатывать аргументы командной строки (или указанные параметром *args*) и проверять их правильность. **getopts** используется в циклах сценариев интерпретатора и предназначена для обеспечения стандартного синтаксиса доступа к параметрам командной строки. Строка *string* состоит из символов допустимых параметров. Допустимые параметры обрабатываются последовательно и сохраняются в переменной интерпретатора *name*. Если за буквой параметра в списке следует двоеточие, параметр должен иметь по меньшей мере один аргумент.
hash	**hash** [**–r**] [*commands*] Выполнять поиск команд и запоминать, в каком каталоге расположена каждая из них. Таким образом, интерпретатор заранее знает о расположении конкретных программ и не тратит время на поиск в путях, определяемых переменной PATH. Команда без аргументов отображает текущие команды, о которых интерпретатор уже «помнит». Информация состоит из количества *выполнений* (hits) и имен команд (с полными путями).

help [**–s**] [*string*] **help**

Отобразить справку по всем встроенным командам или по командам, перечисленным в строке *string*. Если указан параметр **–s**, отображать только краткий синтаксис, в противном случае отображать и справку.

history [*options*] **history**
history [*lines*]

Отобразить список пронумерованных команд из журнала, отмечая измененные команды символом «*». Включает в результат команды из предыдущих сеансов работы. Можно задать максимальное количество выводимых строк (*lines*).

Параметры

–a [*file*]

bash ведет файл с именем *.bash_history* в домашнем каталоге пользователя, который является записью команд предыдущих сеансов работы. Параметр **–a** добавляет журнал команд текущего сеанса к файлу *.bash_history* или к файлу *file*.

–c Очистить журнал: удалить все введенные команды из списка, который хранится интерпретатором.

–n [*file*]

Добавить к журналу команд те строки из файла *.bash_history* (или *file*), которых в нем еще нет.

–r [*file*]

Использовать в качестве рабочего журнала команд файл *.bash_history* (или указанный).

–s *command*

Добавить команду к рабочему журналу, не выполняя ее.

–w [*file*]

Перезаписать файл *.bash_history* (или *file*) текущим журналом команд.

if *test-cmds* **if**

Условный оператор. Возможные варианты синтаксиса:

```
if test-cmds    if test-cmds    if test-cmds
then            then            then
   cmds1           cmds1           cmds1
```

if

```
fi          else          elif test-cmds
              cmds2       then
            fi               cmds2
                        ...
                        else
                           cmdsn
                        fi
```

Обычно первоначальное условие **if** и любое условие **elif** выполняют одну команду **test** или **[]** (хотя допустима любая последовательность команд). Если условие **if** верно (т. е. последняя из команд *test-cmds* возвращает нуль), выполняются команды *cmds1*; иначе проверяются условия **elif** и **else**.

jobs

jobs [*options*] [*jobIDs*]

Перечислить все запущенные и приостановленные задания либо задания с указанными идентификаторами (*jobIDs*). Это полезно, когда необходимо перед выходом из системы проверить состояние заданий, требующих долгого времени для завершения, либо наличие запущенных заданий. См. также раздел «Управление заданиями» далее в этой главе.

Параметры

–l Отобразить идентификаторы заданий и их идентификаторы групп.

–n Отображать только задания, состояние которых изменилось с момента последней проверки.

–p Отображать только идентификаторы групп заданий.

–r Отображать только активные, работающие задания.

–s Только приостановленные задания.

–x *command* [*arguments*]

Выполнить команду (*command*). Если указаны идентификаторы заданий, заменить их этой командой.

kill

kill [*options*] *IDs*

Принудительное завершение процессов или заданий по идентификатору (*ID*). Для уничтожения процесса необходимо быть его владельцем или привилегированным пользователем. См. также раздел «Управление заданиями» в конце главы.

kill

Параметры

–signal

Номер сигнала или имя сигнала (имена можно выяснить при помощи команды **kill –l**). Если номер сигнала равен 9, то завершение не может быть перехвачено заданием. По умолчанию посылается сигнал TERM.

–– Считать все последующие строки аргументами, а не параметрами.

–l Перечислить имена сигналов.

–s *signal*

Определить сигнал (*signal*). Можно задать имя сигнала.

---

let *expressions*

let

Выполнить арифметические действия, заданные в целочисленных выражениях (*expressions*). Выражения состоят из чисел, операторов и переменных интерпретатора (без предшествующего символа $). Выражения должны браться в кавычки, если содержат пробелы, либо другие специальные символы. Подробнее и с примерами арифметические выражения описаны ранее в этой главе. См. также описание команды **expr** в главе 3.

Примеры

В каждом из следующих примеров значение переменной **i** увеличивается на 1.

```
let i=i+1
let "i = i + 1"
```

---

local [*options*] [*variable*[=*value*]] [*variable2*[=*value*]] ...

local

Команда без аргументов отображает все локальные переменные. В противном случае создает (и инициализирует, если задано значение) локальную переменную. Описание параметров приводится в описании встроенной команды **declare**.

---

logout [*status*]

logout

Завершение работы с интерпретатором с возвратом кода завершения *status* в родительский процесс. Может использоваться только в интерпретаторе, запускаемом при входе в систему. В противном случае используется **exit**.

popd

popd [*options*]

Работа со стеком каталогов. По умолчанию удалить из вершины стека каталог и перейти (**cd**) в него.

Параметры

+*n* Удалить из стека каталог с номером *n* (первый имеет порядковый номер 0), считая от вершины стека.

−*n* Удалить из стека каталог с номером *n*, считая от основания стека (последний элемент имеет порядковый номер 0).

printf

printf *string* [*arguments*]

Функция форматирования строки, аналогичная библиотечной функции **printf** языка C. В форматной строке распознаются стандартные ключи с символом процента, например %**i**. Escape-последовательности, такие как **n**, могут включаться в форматную строку и распознаются автоматически. Если вы намереваетесь использовать их в аргументах, укажите в качестве строки *string* %**b**. Символы в аргументах можно экранировать (с целью использования результатов в качестве ввода других команд), используя в качестве строки *string* %**q**.

Примеры

```
$ printf "Previous command: %i\n" "$(($HISTCMD-1))"
Previous command: 534
$ echo $PAGER
less -E
$ printf "%q\n" "\t$PAGER"
\\tless\ -E
```

Последняя команда может быть использована для записи настройки в файл, из которого она затем может быть прочитана и присвоена другому сценарию интерпретатора.

pushd

pushd *directory*

pushd [*options*]

По умолчанию обменять местами два каталога с вершины стека. Если задано имя каталога, добавить его в стек и перейти в этот каталог.

pushd

Параметры

+*n* Прокрутка элементов стека таким образом, чтобы на вершине оказался элемент с номером *n* (первый элемент имеет порядковый номер 0).

−*n* Прокрутка элементов стека таким образом, чтобы на вершине оказался *n*-ный элемент от основания стека (последний элемент имеет порядковый номер 0).

pwd

pwd [−P]

Отобразить абсолютный путь текущего рабочего каталога. Если установлен встроенный режим **−P**, путь не будет содержать символические ссылки.

read

read [*options*] *variable1* [*variable2*...]

Прочитать одну строку со стандартного потока ввода и присвоить слова строки (определяемые по IFS) соответствующим переменным, а все лишние слова добавить к последней переменной. Если задана только одна переменная, она станет равной всей прочитанной строке. Код возврата – 0, если не встретился символ *EOF*; это различие полезно для прогонки циклов для файлов ввода. Если имена переменных опущены, вся строка читается в переменную окружения REPLY.

Параметры

−a *var*

Прочитать все слова в переменную *var*, которая является массивом.

−d *char*

Прекратить чтение по указанному символу, а не по символу новой строки.

−e Построчное редактирование и журнал команд активны во время ввода.

−n *num*

Читать только *num* символов строки.

−p *string*

Отображать диалоговое приглашение в виде строки *string* перед чтением каждой из строк ввода.

read

–r Режим без преобразования; символ продолжения строки (\) игнорируется.

–s Не отображать символы, вводимые пользователем (возможность полезна при запросе паролей).

–t *seconds*

Для диалогового ввода установить интервал ожидания в указанное количество секунд, по истечении которых возвращать соответствующие переменные неинициализированными.

Примеры

```
$ read first last address
Sarah Caldwell 123 Main Street
$ echo "$last, $first\n$address"
Caldwell, Sarah
123 Main Street
```

В следующих командах производится чтение пароля в переменную **$user_pw** и отображение значения этой переменной, при этом используются относительно новые параметры, которые поддерживаются не всеми версиями **bash**.

```
$ read -sp "Введите пароль (буквы не отображаются при вводе)" user_pw
Введите пароль (буквы не отображаются при вводе)
$ echo $user_pw
Вам это знать не положено!
```

Следующий сценарий читает данные из файла паролей, в котором поля разделяются двоеточием (что и делает этот файл популярным объектом для примеров разбора ввода):

```
IFS=:
cat /etc/passwd |
while
read account pw user group gecos home shell
do
echo "Account name $account has user info $gecos"
done
```

readonly

readonly [*options*] [*variable1 variable2* ...]

Запретить присваивание новых значений перечисленным переменным интерпретатора. Можно читать значения переменных, но изменять их нельзя. В интерпретаторе **bash** для присваивания неизменяемого значения допускается следующая запись: *variable=value*.

readonly

Параметры

–a Считать все последующие аргументы именами массивов.

–f Считать все последующие аргументы именами функций, устанавливать для них режим read-only (только для чтения).

–p Отобразить все переменные, значения которых изменять запрещено (режим по умолчанию).

return [*n*]

return

Размещается в теле функции. Данное ключевое слово обеспечивает выход из функции и возврат кода *n* либо кода завершения последней выполненной команды.

select *name* [**in** *wordlist;*]
do
commands
done

select

Присвоить значение переменной *name* в соответствии с выбором пользователя из списка вариантов *wordlist*. Ввод пользователя сохраняется в переменной REPLY, а выбранное значение в переменной *name*. После этого выполняются команды *commands*, пока не встретится ключевое слово **break** или **return**. Стандартное приглашение может изменяться с помощью переменной интерпретатора PS3.

set [*options*] [*arg1 arg2* ...]

set

Команда без аргументов отображает значения всех переменных, доступных работающему интерпретатору. Режимы могут включаться (–*option*) или выключаться (+*option*). Режимы также могут устанавливаться при запуске интерпретатора посредством **bash**. Аргументы присваиваются соответственно переменным **$1**, **$2** и т. д.

Параметры

– Выключить режимы **–v** и **–x**, а также обработку параметров.

–– Используется в качестве последнего параметра. Отключает обработку параметров, чтобы аргументы, начинающиеся с символа «–», не принимались ошибочно за пара-

set

метры. (Например, если необходимо присвоить **$1** значение –1.) Если за параметром не следуют никакие аргументы, происходит освобождение позиционных параметров.

–a После выполнения этой команды автоматически отмечать переменные для экспортирования после их определения или изменения.

–b Отображать состояние фонового задания по завершении, не дожидаясь следующего приглашения интерпретатора.

–e Завершить работу, если код завершения команды не равен нулю.

–f Не производить расширение специальных символов (*globbing*) в именах файлов (например, *** ? [ ]**).

–h Запоминать расположение команд по мере их определения.

–k Присваивания переменной среды (*var=value*) будут выполняться независимо от того, в каком месте командной строк они фигурируют. В обычных условиях присваивания должны предшествовать командам.

–m Режим наблюдения. Включается управление заданиями; фоновые процессы выполняются в отдельной группе процессов. Режим **–m**, как правило, устанавливается автоматически.

–n Производить чтение команд, но не выполнять их; используется для поиска ошибок. Удобно применять в пакетном режиме работы командных интерпретаторов.

–o [*m*]

Перечислить режимы интерпретатора либо включить режим *m*. Многие режимы могут быть установлены с помощью отдельных параметров. Существующие режимы:

allexport

Действует идентично **–a**.

braceexpand

Действует идентично **–B**.

emacs

Режим по умолчанию. Редактирование командной строки в Emacs-стиле.

errexit

Действует идентично **–e**.

hashall

Действует идентично **–h**.

histexpand

Действует идентично **–H**.

history

Режим по умолчанию. Вести журнал команд.

ignoreeof

Не выполнять завершение по нажатию клавиш **Ctrl-D** (EOF, символ конца файла). Для выхода из интерпретатора в таком случае необходимо использовать **exit**. Эквивалентно установке значения переменной интерпретатора IGNOREEOF=1.

interactive-comments

Считать комментариями все слова, начинающиеся с символа #, и следующие за ними в той же строке.

keyword

Действует идентично **–k**.

monitor

Действует идентично **–m**.

noclobber

Действует идентично **–C**.

noexec

Действует идентично **–n**.

noglob

Действует идентично **–f**.

notify

Действует идентично **–b**.

nounset

Действует идентично **–u**.

onecmd

Действует идентично **–t**.

physical

Действует идентично **–P**.

posix

Соблюдать рамки стандарта POSIX.

set

privileged

Действует идентично **–p**.

verbose

Действует идентично **–v**.

vi Перейти к **vi**-стилю редактирования командной строки.

xtrace

Действует идентично **–x**.

+o [*m*]

Перечислить режимы или выключить указанный режим. Список режимов см. выше.

–p Запуск для привилегированного пользователя; не обрабатывать файл *$HOME/.profile*.

–t Завершение работы после выполнения одной команды.

–u При подстановке выдавать ошибку, если переменная не определена.

–v Отображать каждую командую строку интерпретатора по мере чтения.

–x Отображать выполненные команды и аргументы с префиксом, определяемым значением переменной PS4. Этот режим способствует пошаговой отладке сценариев интерпретатора (режим идентичен **–o xtrace**).

–B Режим по умолчанию. Разрешить расширение фигурных скобок.

–C То же, что и **noclobber**.

–H Режим по умолчанию. Разрешить команды с префиксами **!** и **!!**.

–P Отображать абсолютные пути в выводе команды **pwd**. По умолчанию **bash** включает в путь символические ссылки (если таковые имеются).

Примеры

`set -- "$num" -20 -30`	*Присвоить переменным $1, $2 и $3 значения $num, –20 и –30, соответственно*
`set -vx`	*Читать каждую командную строку, отображать ее, выполнять, отображать повторно (с аргументами)*
`set +x`	*Запретить отслеживание выполнения команд*
`set -o noclobber`	*Запретить перезапись файлов*
`set +o noclobber`	*Разрешить перезапись файлов*

shift [*n*] — **shift**

Смещение позиционных параметров (например, **$2** становится **$1**). Если параметр *n* задан, происходит смещение влево на *n* позиций.

source *file* [*arguments*] — **source**

Последовательно читать строки файла *file*. Файл не обязательно должен быть исполняемым, но должен находиться в одном из каталогов, задаваемых переменной PATH.

suspend [**−f**] — **suspend**

Идентично **Ctrl-Z**. Часто используется для приостановки работы программы **su**.

Параметр

−f Принудительная приостановка, даже если текущий интерпретатор является основным (интерпретатором по умолчанию).

test *condition* — **test**

или

[*condition*]

Вычислить условие (*condition*) и, если оно истинно, вернуть нулевой код завершения. В противном случае вернуть ненулевой код завершения. В альтернативной форме команды используются квадратные скобки вместо слова **test**. Условие строится из приведенных ниже выражений. Условие истинно, если истинно его описание.

Условия на файл

−a *file*

Файл существует.

−b *file*

Файл существует и является специальным файлом блочного устройства.

−c *file*

Файл существует и является специальным файлом символьного устройства.

test

–d *file*

Файл существует и является каталогом.

–e *file*

Файл существует.

–f *file*

Файл существует и является обычным файлом.

–g *file*

Файл существует и у него установлен бит смены идентификатора группы.

–h *file*

Файл существует и является символической ссылкой.

–k *file*

Файл существует и у него установлен бит-«липучка» (sticky-bit) (в настоящее время не используется).

–p *file*

Файл существует и является именованным каналом (FIFO).

–r *file*

Чтение файла разрешено.

–s *file*

Размер файла ненулевой.

–t [*n*]

Открытый файловый дескриптор *n* связан с устройством терминала; по умолчанию *n* равно 1.

–u *file*

Файл существует и у него установлен бит смены идентификатора пользователя.

–w *file*

Запись в файл разрешена.

–x *file*

Файл является исполняемым.

–G *file*

Файл существует и его группа-владелец является эффективной группой-владельцем процесса.

–L *file*

Файл существует и является символической ссылкой.

–N *file*

Файл был изменен со времени последнего доступа к нему.

–O *file*

Файл существует и его владелец является эффективным владельцем процесса.

–S *file*

Файл является сокетом (socket).

f1 **–ef** *f2*

Файлы *f1* и *f2* связаны (ссылаются на один и тот же файл).

f1 **–nt** *f2*

Файл *f1* новее файла *f2*.

f1 **–ot** *f2*

Файл *f1* старше файла *f2*.

Строковые условия

–n *s1*

Строка *s1* имеет ненулевую длину.

–o *s1*

Установлен режим интерпретатора *s1*. Режимы интерпретатора приведены в описании встроенной команды **set**.

–z *s1*

Строка *s1* имеет нулевую длину.

s1 = *s2*

Строки *s1* и *s2* идентичны.

s1 == *s2*

Строки *s1* и *s2* идентичны.

s1 **!=** *s2*

Строки *s1* и *s2* не идентичны.

s1 < *s2*

Строка *s1* меньше строки *s2* по алфавиту (либо по другому активному критерию сортировки). По умолчанию производится посимвольное сравнение в рамках таблицы символов ASCII.

s1 > *s2*

Строка *s1* больше строки *s2* по алфавиту (либо по другому активному критерию сортировки).

test

string

Строка не пуста.

Целочисленные сравнения

n1 **–eq** *n2*

n1 равно *n2*.

n1 **–ge** *n2*

n1 больше либо равно *n2*.

n1 **–gt** *n2*

n1 больше *n2*.

n1 **–le** *n2*

n1 меньше либо равно *n2*.

n1 **–lt** *n2*

n1 меньше *n2*.

n1 **–ne** *n2*

n1 не равно *n2*.

Сочетание условий

! *condition*

Истинно, если условие *condition* ложно.

condition1 **–a** *condition2*

Истинно, если истинны оба условия.

condition1 **–o** *condition2*

Истинно, если истинно хотя бы одно из условий.

Примеры

Каждый из следующих примеров представляет собой первую строку какого-либо оператора, использующего проверку условий:

`while test $# -gt 0`	*Пока есть аргументы...*
`while [ -n "$1" ]`	*Пока первый аргумент не пуст...*
`if [ $count -lt 10 ]`	*Если $count меньше 10...*
`if [ -d RCS ]`	*Если существует каталог RCS...*
`if [ "$answer" != "y" ]`	*Если ответ – не y...*
`if [ ! -r "$1" -o ! -f "$1" ]`	*Если первый аргумент является файлом, запрещенным для чтения, или не является обычным файлом...*

times **times**

Вывести суммарное время исполнения процессов для пользователя и всей системы.

trap [**–l**] [[*commands*] *signals*] **trap**

Выполнить команды *commands* при получении любого из перечисленных сигналов (*signals*). Обычно это сигналы с номерами 0, 1, 2 и 15. Если команд больше одной, они должны быть заключены в кавычки и разделены символом точка с запятой. Если команды являются пустой строкой (например **trap** ""*signals*), сигналы будут игнорироваться интерпретатором. Если команды опущены совсем, восстанавливается обработка перечисленных сигналов, принимаемая по умолчанию. Если опущены и команды и сигналы, то перечисляются текущие установки перехвата сигналов. См. примеры в конце описания этой команды, а также для команды **exec**.

Параметр

–l Перечислить сигналы.

Сигналы

Сигналы перечисляются в парах с событиями, посылающими эти сигналы.

0 Выход из интерпретатора (обычно по завершении сценария).

1 Hang up (обычно по завершении сеанса работы).

2 Прерывание (обычно **Ctrl-C**).

3 Выход.

4 Недопустимая команда.

5 Отслеживание перехвата.

6 Аварийное завершение.

7 Не используется.

8 Исключение при работе с действительными числами.

9 Принудительное завершение.

10 Определяется пользователем.

11 Ссылка на недействительную область памяти.

12 Определяется пользователем.

13 Запись данных в конвейер при отсутствии читающего процесса.

trap

14 Завершение аварийного сигнала по лимиту времени.

15 Программное принудительное завершение (обычно с помощью **kill**).

16 Ошибка стека сопроцессора.

17 Принудительное завершение порожденного процесса.

18 Продолжение (если выполнение было приостановлено).

19 Остановить процесс.

20 Остановка с терминала.

21 Ввод с терминала для фонового задания.

22 У фонового задания есть данные для вывода на терминал.

23 Не используется.

24 Не используется.

25 Не используется.

26 Не используется.

27 Не используется.

28 Не используется.

29 Доступны операции ввода/вывода для канала.

Примеры

```
trap "" 2        Игнорировать сигнал 2
trap 2           Вернуться к прежнему поведению
```

Удалить файл **$tmp**, когда программа интерпретатора прекратит работу либо пользователь закончит сеанс работы, нажмет клавиши <Ctrl>+<C> или принудительно завершит задачу посредством команды **kill**.

```
trap "rm -f $tmp; exit" 0 1 2 15
```

type

type [*options*] *commands*

Отобразить абсолютные пути программ, выполняемых по командам *commands*, и вывести информацию о том, хешировано ли (см. **hash**) их расположение.

–– Считать все последующие строки аргументами, а не параметрами.

–a, ––all

Отображать все экземпляры программы, а не только те, которые будут реально запущены.

type

−p, −path

Отобразить хешированное значение для команды, которое может отличаться от пути, записанного в переменной окружения PATH.

−t, −type

Определить, является ли команда псевдонимом, ключевым словом, функцией, встроенной командой или файлом, и вывести результат.

Пример

```
$ type mv read
mv is /bin/mv
read is a shell built-in
```

typeset

typeset

См. **declare**.

ulimit [*options*] [*n*]

ulimit

Отобразить значение одного или нескольких ограничений на использование ресурсов либо установить значение ограничения в *n*, если таковое задано. Ограничения могут быть жесткими (**−H**) или мягкими (**−S**). По умолчанию **ulimit** устанавливает оба ограничения либо отображает мягкое ограничение. Параметры определяют, с каким ресурсом происходит работа.

Параметры

−− Считать все последующие строки аргументами, а не параметрами.

−a Отобразить все текущие ограничения.

−H Жесткое ограничение ресурсов.

−S Мягкое ограничение ресурсов.

Специальные ограничения

Эти параметры налагают ограничение на использование специальных видов ресурсов:

−c Файлы отображения памяти.

−d Размер сегмента данных процесса.

−f Размер файла, создаваемого в сеансе работы с интерпретатором.

ulimit	**–l** Объем памяти, захватываемой процессом. **–m** Размер набора резидентов. **–n** Количество файловых дескрипторов. Ограничение отсутствует во многих системах. **–p** Размер конвейера, измеряемый в блоках по 512 байт. **–s** Размер стека. **–t** Количество процессорного времени в секундах. **–u** Количество процессов на одного пользователя. **–v** Виртуальная память, используемая интерпретатором.
umask	**umask** [*nnn*] **umask [–p][–S]** Отобразить или установить значение маски прав доступа для создаваемого файла (в восьмеричной системе счисления). Маска определяет, какие права доступа отсутствуют. (Так, результатом выполнения команды **umask 002** будут права доступа **rw–rw–r––**). **Параметры** **–p** Отобразить значение маски в команде **umask**, чтобы пользователь мог прочитать ее и выполнить команду. **–S** Отобразить значение **umask** в символьном виде, а не в виде восьмеричного числа.
unalias	**unalias [–a]** *names* Удалить имена (*names*) из списка псевдонимов. См. также **alias**. **Параметр** **–a** Удалить все псевдонимы.
unset	**unset** [*options*] *names* Удалить определения функций или переменных, имена которых перечислены в списке *names*. **Параметры** **–f** Имя является именем функции. **–v** Имя является именем переменной (режим по умолчанию).

until *test-commands* **do** *commands* **done** Выполнить команды *test-commands* (как правило, **test** или **[ ]**), а если код завершения не равен нулю (т. е. условие ложно), выполнить команды *commands*; повторить.	**until**
wait [*ID*] Приостановить выполнение и дождаться завершения всех фоновых процессов (будет получен нулевой код завершения) либо приостановить до завершения фонового процесса или задания с идентификатором *ID*. Полезно иметь в виду, что переменная интерпретатора $! содержит идентификатор последнего фонового процесса. Если не выполняется работа с заданиями, идентификатор процесса *ID* может быть только численным. См. раздел «Управление заданиями». **Пример** `wait $!` *Дождаться завершения последнего фонового процесса*	**wait**
while *test-commands* **do** *commands* **done** Выполнить команды *test-commands* (обычно **test** или **[ ]**), а если получен нулевой код возврата, выполнить команды *commands*; повторить.	**while**

Управление заданиями

Управление заданиями обеспечивает пользователю возможность переводить приоритетные задания в фоновый режим, фоновые задания – в приоритетный режим или временно приостанавливать выполнение заданий. По умолчанию работа с заданиями активирована. Будучи отключенной, она может быть вновь активирована любой из следующих команд:

```
bash -m -i
set -m
set -o monitor
```

Многим командам управления заданиями требуется идентификатор задания в качестве аргумента. Этот аргумент может быть задан одним из следующих способов:

%*n* Задание с номером *n*.

%*s* Задание, командная строка которого начинается с подстроки *s*.

%?*s* Задание, командная строка которого содержит подстроку *s*.

%% Текущее задание.

%+ Текущее задание (идентично %%).

%− Предыдущее задание.

bash предоставляет следующие команды для управления заданиями. Подробнее эти команды описаны в разделе «Встроенные команды» ранее в этой главе.

bg Перевести задание в фоновый режим.

fg Перевести задание в приоритетный режим.

jobs

Перечислить активные задания.

kill

Принудительно завершить задание.

stop

Приостановить выполнение фонового задания.

stty tostop

Приостанавливать фоновые задания, пытающиеся послать вывод на терминал.

wait

Дождаться завершения фоновых заданий.

Ctrl-Z

Приостановить выполнение приоритетного задания (в зависимости от используемого терминала комбинация клавиш для приостановки задания может быть иной).

Глава 8

Интерпретаторы csh и tcsh

В этой главе описан командный интерпретатор **csh** и его усовершенствованная версия – **tcsh**. В некоторых дистрибутивах Linux **tcsh** используется в качестве C shell, и все описанное в этой главе будет справедливо даже для **csh** в таких системах. Интерпретатор C shell имеет такое название, потому что многие символы и конструкции программирования в этом интерпретаторе напоминают язык C.

Стандартным для Linux интерпретатором является **bash**. Если вы хотите использовать **csh** или **tcsh**, необходимо прежде всего изменить интерпретатор, запускаемый по умолчанию и определяемый в файле паролей. Интерпретатор по умолчанию может быть указан при создании учетной записи для нового пользователя. Если учетная запись уже существует, необходимо воспользоваться командой **chsh**, чтобы произвести замену (описание команды см. в главе 3 «Команды Linux»).

В этой главе представлены следующие темы:

- Обзор возможностей
- Вызов интерпретатора
- Синтаксис
- Переменные
- Выражения
- Журнал команд
- Работа с командной строкой
- Управление заданиями
- Встроенные команды

Обзор свойств

Интерпретатор C shell предоставляет следующий набор возможностей:

- Перенаправление ввода/вывода
- Метасимволы (маски) для сокращения имен файлов
- Переменные интерпретатора, позволяющие настраивать параметры окружения
- Целочисленная арифметика
- Доступ к выполнявшимся ранее командам (журнал команд)
- Сокращение имен команд (использование псевдонимов)
- Встроенный набор команд для создания сценариев интерпретатора
- Управление заданиями

Интерпретатор **tcsh** предоставляет все перечисленные выше возможности, но обладает и рядом дополнительных:

- Редактирование командной строки и команды редактора
- Дополнение слов (по клавише <Tab>)
- Проверка орфографии
- Расширенный журнал команд
- Расширенные возможности навигации в каталогах
- Планирование событий, таких как завершение сеанса работы или блокирование терминала после определенного интервала бездействия и отложенное выполнение команд
- Дополнительные встроенные команды интерпретатора
- Новые переменные интерпретатора, а также новые переменные окружения
- Новые форматирующие последовательности для переменной приглашения, а также два дополнительных приглашения – для циклов и проверки орфографии
- Доступ к переменным «только для чтения»

Вызов интерпретатора

Командный интерпретатор может быть запущен следующим образом:

csh [*options*] [*arguments*]
tcsh [*options*] [*arguments*]

В **csh** и **tcsh** используется синтаксис, напоминающий синтаксис языка C, а команды могут выполняться как с терминала, так и из файла. При отладке сценариев полезны параметры **–n**, **–v** и **–x**.

Параметры

–b Считать последующие параметры аргументами выполняемой команды, а не параметром собственно **csh**.

–c Выполнить указанные в параметре команды.

–d Загрузить стек каталогов из файла ~/*.cshdirs*, даже если интерпретатор вызывается после входа пользователя в систему (**tcsh**).

–e Завершить работу при получении ошибочного кода возврата выполняемой команды.

–f Быстрый запуск; при вызове **csh** не выполняются *.cshrc* и *.tcshrc*.

–i Создать диалоговую среду (вывести приглашение командной строки).

–l Интерпретатор, выполняемый при входе в систему (использование этого параметра исключает применение всех остальных).

–m Загрузить файл ~/*.tcshrc*, даже если пользователь не является его владельцем.

–n Производить разбор команд, но не выполнять их.

–s Выполнять команды, поступающие со стандартного ввода.

–t Завершить работу после выполнения одной команды.

–v Отображать команды перед их выполнением; производить расширение подстановок журнала команд, но не другие расширения (имен файлов, переменных и команд). Идентично установке **verbose**.

–V Идентично **–v**, но также отображать *.cshrc*.

–x Отображать команды перед их выполнением, расширять все подстановки. Идентично установке **echo**.

–X Идентично **–x**, но также отображать *.cshrc*.

Аргументы

Аргументы присваиваются позиционным параметрам **$1**, **$2** и т. д. в соответствии с их порядком. Если первый из аргументов является исполняемым сценарием, производится чтение команд из указанного файла, а значения оставшихся аргументов присваиваются переменным **$1**, **$2** и т. д.

Синтаксис

В этом подразделе описаны многочисленные особенности, специфичные для интерпретаторов **csh** и **tcsh**. Темы расположены следующим образом:

- Специальные файлы
- Специальные символы в именах файлов
- Экранирование

- Формы команд
- Формы перенаправления

Специальные файлы

Имя файла	Описание
~/*.cshrc* или ~/*.tcshrc*	Выполняется при каждом запуске интерпретатора
~/*.login*	Выполняется интерпретатором после *.cshrc* при входе в систему
~/*.cshdirs*	Выполняется интерпретатором после *.login* при входе в систему **(tcsh)**
~/*.logout*	Выполняется при завершении сеанса работы с интерпретатором
/*etc*/*passwd*	Информация о домашних каталогах пользователей для расширения ~*name*

Специальные символы в именах файлов

Символы	Значение
*	Любая строка, пустая или состоящая из произвольного количества символов
?	Один произвольный символ
[*abc*...]	Любой из перечисленных в скобках символов; для задания интервала может использоваться дефис (например, a–z, A–Z, 0–9)
{*abc*, *xxx*,...}	Расширение каждой из перечисленных строк
~	Домашний каталог текущего пользователя
~*name*	Домашний каталог пользователя с регистрационным именем *name*

Примеры

`% ls new*`	*Вывести список файлов, имена которых начинаются на* `new`.
`% cat ch?`	*Соответствует файлам, имена которых имеют вид* `ch9`, *но не* `ch10`
`% vi [D-R]*`	*Редактировать файлы, имена которых начинаются на любую букву от прописной D до R*
`% ls {ch,app}?`	*Расширить список и вывести список файлов* `ch1`, `ch2`, `app1`, `app2`
`% cd ~tom`	*Перейти в домашний каталог пользователя* `tom`

Экранирование

Экранирование отменяет специальное значение символа и позволяет использовать его буквально. Следующие символы имеют специальное значение для командного интерпретатора C shell:

Символы	Описание
;	Разделитель команд
&	Выполнение в фоновом режиме
( )	Группировка команд
|	Конвейер
* ? [] ~	Метасимволы в именах файлов
{ }	Расширение строк (обычно экранирование не требуется)
> < & !	Символы перенаправления
! ^	Подстановка из журнала команд, быстрая подстановка
“ ‘ \	Используются для экранирования символов
`	Подстановка результатов работы команд
$	Подстановка значений переменных
новая строка, пробел, табуляция	Разделители слов

Следующие символы могут использоваться для экранирования:

" " Все символы между двойными кавычками понимаются буквально, кроме следующих, которые сохраняют специальное значение:

- $ Производится подстановка значения переменной.
- ` Подстановка результата выполнения команды.
- " Парные двойные кавычки закрывают последовательность.
- \ Экранирование следующего символа.
- ! Доступ к журналу команд.

newline

Символ новой строки.

‘ ‘ Все символы между одинарными кавычками понимаются буквально, кроме символа журнала команд «!», еще одной одинарной кавычки, закрывающей последовательность, и символа новой строки.

\ Символ, следующий за этим, понимается буквально. Применяется для экранирования символов ", $ и ` в последовательности, заключенной в двойные кавычки. Часто применяется для экранирования собственно символа \, пробелов или символов новой строки. Символ доступа к журналу команд (обычно !) необходимо экранировать всегда.

Примеры

```
% echo ‘Одинарные кавычки “защищают” двойные кавычки’
Одинарные кавычки “защищают” двойные кавычки
```

```
% echo “Ну, разве это не \“особый случай\”?”
Ну, разве это не “особый случай”?

% echo “У Вас `ls | wc -l` файла в каталоге `pwd`”
У Вас 43 файла в каталоге /home/bob

% echo “Значением \$x является $x”
Значением $x является 100
```

Формы команд

Команда	Действие
cmd &	Выполнить команду в фоновом режиме
cmd1 ; *cmd2*	Последовательность команд; выполнить команды, перечисленные в строке
(*cmd1* ; *cmd2*)	«Дополнительный интерпретатор»; считать команды *cmd1* и *cmd2* одной группой команд
cmd1 | *cmd2*	Конвейер; использовать вывод команды *cmd1* в качестве ввода команды *cmd2*
cmd1 `*cmd2*`	Подстановка команд; результат работы команды *cmd2* передается в качестве аргументов команде *cmd1*
cmd1 || *cmd2*	Логическое ИЛИ; команда *cmd2* выполняется только тогда, когда выполнение *cmd1* завершилось неудачей
cmd1 && *cmd2*	Логическое И; команда *cmd2* выполняется только тогда, когда успешно выполнилась команда *cmd1*

Примеры

`% nroff file > output &`	*Фоновое форматирование текста*
`% cd; ls`	*Последовательное выполнение команд*
`% (date; who; pwd) > logfile`	*Перенаправление всего вывода в файл*
`% sort file | pr -3 | lp`	*Сортировка файла, разбивка на страницы, затем печать*
`% vi `grep -l ifdef *.c``	*Редактировать файлы, найденные* `grep`
`% egrep ‘(yes|no)’ `cat list``	*Определить список файлов для поиска*
`% grep XX file && lp file`	*Печать файла, содержащего заданный текст*
`% grep XX file || echo “XX not found”`	*Отображение сообщения об ошибке, если текст* `XX` *не найден*

Формы перенаправления

Файловый дескриптор	Имя	Принятое сокращение	Типичное устройство
0	Стандартный поток ввода	stdin	Клавиатура
1	Стандартный поток вывода	stdout	Терминал
2	Стандартный поток ошибок	stderr	Терминал

Источник ввода и цель вывода могут быть переопределены при помощи команд перенаправления, перечисленных в следующих разделах.

Простое перенаправление

Команда	Действие
cmd > *file*	Перенаправить вывод команды *cmd* в указанный файл *file* (перезаписывается)
cmd >! *file*	То же, но перезапись даже при установленном режиме **noclobber**
cmd >> *file*	Перенаправление вывода в файл *file* (добавление к существующему содержимому)
cmd >>! *file*	То же, даже при установленном режиме **noclobber**
cmd < *file*	Перенаправление ввода команды из файла *file*
cmd << *text*	Чтение со стандартного ввода вплоть до появления строки *text* (строка может быть значением переменной интерпретатора). Ввод обычно происходит на терминале или в процессе выполнения сценария. Как правило, такой синтаксис используют команды **cat**, **echo**, **ex** и **sed**. Если текст *text* заключен в кавычки, не происходит подстановки значений переменных для стандартного ввода, подстановок команд и т. д.

Множественное перенаправление

Команда	Действие
cmd >& *file*	Перенаправить стандартный вывод и поток ошибок в указанный файл
cmd >&! *file*	То же, даже при установленном режиме **noclobber**
cmd >>& *file*	Добавление стандартного вывода и потока ошибок к содержимому указанного файла
cmd >>&! *file*	То же, даже при установленном режиме **noclobber**
cmd1 |& *cmd2*	Перенаправление стандартного вывода и потока ошибок через конвейер
(*cmd* > *f1*) >& *f2*	Перенаправление стандартного вывода в файл *f1*, а стандартного потока ошибок в файл *f2*

Команда	Действие
cmd | **tee** *files*	Перенаправить результат работы команды на стандартный вывод (обычно терминал) и в файлы *files* (см. примеры в описании команды **tee** в главе 3)

Примеры

```
% cat part1 > book                          Скопировать part1 в book
% cat part2 part3 >> book                   Добавить part2 и part3 в тот же файл,
                                            что и part1
% mail tim < report                         Получить текст сообщения из report
% cc calc.c >& error_out                    Сохранить все сообщения, включая
                                            сообщения об ошибках
% cc newcalc.c >&! error_out               Перезаписать старый файл
% grep Unix ch* |& pr                       Передать все сообщения (включая со-
                                            общения об ошибках) через конвейер
% (find / -print > filelist) >& no_access   Отделить сообщения об ошибках от
                                            списка файлов
% sed 's/^/XX /' << "END_ARCHIVE"           Ввести текст до слова END_ARCHIVE
This is often how a shell archive is "wrapped",
bundling text for distribution. You would normally
run sed from a shell program, not from the command line.
"END_ARCHIVE"
```

Переменные

В этом разделе представлены следующие темы:

- Подстановка значений переменных
- Модификаторы переменных
- Предопределенные переменные интерпретатора
- Форматирование переменной приглашения
- Пример файла *.cshrc*
- Переменные окружения

Подстановка значений переменных

В приводимых ниже вариантах подстановок фигурные скобки необязательны, за исключением тех случаев, когда имя переменной необходимо отделить от следующего символа, который иначе считался бы частью имени.

Переменная	Действие
${*var*}	Значение переменной *var*
${*var*[*i*]}	Выбрать слово (или слова) в *i*-й позиции строки *var*. *i* может быть обычным числом, диапазоном в виде m-n, диапазоном в виде –n (отсутствующее m принимает значение 1), диапазоном в виде m- (отсутствую-

Переменная	Действие
	щее n подразумевает все следующие слова) либо символом * (выбор всех слов). *i* также может быть переменной, принимающей одно из этих значений.
${#*var*}	Количество слов в *var*
${#argv}	Количество аргументов
$0	Имя программы
${argv[*n*]}	Отдельный аргумент командной строки (позиционный параметр); $1 \le n \le 9$
${*n*}	Идентично ${argv[*n*]}
${argv[*]}	Все аргументы командной строки
$*	Идентично $argv[*]
$argv[$#argv]	Последний аргумент
${?*var*}	1, если переменная *var* установлена, и 0 – в противном случае
$$	Номер процесса текущего интерпретатора; полезно при необходимости создания временных файлов с уникальными именами
${?*name*}	1, если параметр *name* установлен, 0 – в противном случае
$?0	1, если введенное имя файла существует, 0 – в противном случае (в диалоговом режиме всегда 0)

Примеры

Сортировка аргументов от третьего до последнего и сохранение результата в файле с уникальным для этого процесса именем:

```
sort $argv[3-] > tmp.$$
```

Обработать команды *.cshrc* только в том случае, если интерпретатор работает в диалоговом режиме (т. е. переменная **prompt** должна быть установлена):

```
if ($?prompt) then
   set commands
   alias commands
   etc.
endif
```

Модификаторы переменных

За исключением вариантов **$?*var***, **$$** и **$?0**, подстановка переменных может иметь дополнительный модификатор. При использовании фигурных скобок модификатор записывается внутри них.

:r Вернуть корень переменной (содержимое до последней точки).

`:e` Вернуть расширение переменной.

`:h` Вернуть заголовок переменной (имя каталога в полном имени файла).

`:t` Вернуть хвост переменной (за последним символом /).

`:gr` Вернуть все корни.

`:ge` Вернуть все расширения.

`:gh` Вернуть все заголовки.

`:gt` Вернуть все хвосты.

`:q` Заключить переменную (список слов) в кавычки, сохраняя разделение элементов. Модификатор полезен, если необходимо хранить имена файлов, содержащие специальные символы, которые не следует расширять.

`:x` Заключить шаблон в кавычки, расширяя его в список слов.

Примеры использования модификаторов путей

В следующей таблице приведены примеры использования модификаторов путей для переменной:

```
set aa=(/progs/num.c /book/chap.ps)
```

Часть переменной	Запись	Результат
Обычная переменная	`echo $aa`	*/progs/num.c /book/chap.ps*
Второй корень	`echo $aa[2]:r`	*/book/chap*
Второй заголовок	`echo $aa[2]:h`	*/book*
Второй хвост	`echo $aa[2]:t`	*chap.ps*
Второе расширение	`echo $aa[2]:e`	*ps*
Корень	`echo $aa:r`	*/progs/num /book/chap.ps*
Все корни	`echo $aa:gr`	*/progs/num /book/chap*
Заголовок	`echo $aa:h`	*/progs /book/chap.ps*
Все заголовки	`echo $aa:gh`	*/progs /book*
Хвост	`echo $aa:t`	*num.c /book/chap.ps*
Все хвосты	`echo $aa:gt`	*num.c chap.ps*
Расширение	`echo $aa:e`	*c /book/chap.ps*
Все расширения	`echo $aa:ge`	*c ps*

Примеры использования экранирующих модификаторов

Без экранирования интерпретатор расширяет специальные символы в именах файлов, представляя содержимое текущего каталога:

```
% set a="[a-z]*" A="[A-Z]*"
% echo "$a" "$A"
[a-z]* [A-Z]*

% echo $a $A
at cc m4 Book Doc

%echo $a:x $A
[a-z]* Book Doc

% set d=($a:q $A:q)
% echo $d
at cc m4 Book Doc

% echo $d:q
[a-z]* [A-Z]*

% echo $d[1] +++ $d[2]
at cc m4 +++ Book Doc

% echo $d[1]:q
[a-z]*
```

Предопределенные переменные интерпретатора

Переменные могут устанавливаться двумя способами. Присваиванием значения:

```
set var=value
```

или простым включением:

```
set var
```

Ниже перечислены переменные. Те из них, которые могут содержать значения, отмечены знаком равенства и типом принимаемого значения, за которыми следует описание переменной. (Имейте в виду, что некоторым переменным, например **argv**, **cwd** или **status**, значения никогда не присваиваются явным образом.) Для переменных, которые могут быть включены или выключены, описан только эффект включенного состояния. Интерпретатор **tcsh** автоматически устанавливает (и в некоторых случаях обновляет) значения следующих переменных: **addsuffix**, **argv**, **autologout**, **cwd**, **dirstack**, **echo-style**, **edit**, **gid**, **home**, **loginsh**, **logout**, **oid**, **owd**, **path**, **prompt**, **prompt2**, **prompt3**, **shell**, **shlvl**, **status**, **tcsh**, **term**, **tty**, **uid**, **user** и **version**. Переменные, выделенные курсивом, являются особенностью **tcsh**.

Переменная	Описание
addsuffix	Добавлять символ / к именам каталогов и пробел к именам файлов при дополнении до уникального имени
ampm	Отображать время в 12-часовом формате

Переменная	Описание
argv=(*args*)	Список аргументов, передаваемый текущей команде; по умолчанию имеет значение **( )**
autocorrect	Проверка орфографии перед попыткой дополнения команды
autoexpand	Расширение обращений к журналу команд **(!)** при дополнении
autolist[=**ambiguous**]	Отображать список возможных дополнений в случае неоднозначности. Если задано значение **ambiguous**, дописывать совпадающую часть возможных дополнений
autologout=logout-minutes [*locking-minutes*]	Завершить работу с интерпретатором через *logout-minutes* бездействия. Заблокировать терминал после *locking-minutes* бездействия; разблокировка требует ввода пароля пользователя. Не используется при установленной переменной окружения DISPLAY
backslash_quote	Обратный слэш всегда экранирует символы \, ' и "
cdpath=*dirs*	Список альтернативных каталогов, в которых производится поиск аргументов для команд **cd**, **popd** или **pushd**
color	Включить цвет для **ls –F** или **ls** либо для обеих команд (пустое значение)
command	Если переменная установлена, она содержит команду, переданную интерпретатору с ключом **–c**
complete=**enhance**	При значении **enhance** игнорировать регистр символов при дополнении, считать точку, дефис и символ подчеркивания разделителями слов, не делать различий между символом подчеркивания и дефисом
correct={**cmd**|**complete**|**all**}	При значении **cmd** проверять орфографию команд, при значении **complete** дополнять команды, при значении **all** проверять орфографию во всей командной строке
cwd=*dir*	Полное имя рабочего каталога
dextract	Если переменная установлена, команда **pushd** помещает имя каталога на вершину стека и не выполняет смещение элементов стека
dirsfile=file	Журнал каталогов, читаемый по **dirs –S** и **dirs –L**. По умолчанию это файл *~/.cshdirs*
dirstack	Стек каталогов в формате массива. Элемент **dirstack[0]** всегда эквивалентен **cwd**. Остальные элементы могут меняться при участии пользователя
dsmpbyte=code	Поддержка многобайтного кода; для работы с Kanji. Подробности можно найти в руководстве по **tcsh**

Переменная	Описание
dunique	Хранить в стеке только одну копию имени каждого каталога
echo	Отображать каждую командную строку перед выполнением. Идентично команде **csh –x**
echo_style={**bsd**|**sysv**|**both**|**none**}	Не отображать символ новой строки при заданном параметре **–n (bsd)** | разбирать escape-последовательности **(sysv)** | оба действия | не делать ни того ни другого
edit	Включить редактор командной строки
ellipsis	Используется для настройки переменной **prompt**. Заменять пропущенные каталоги многоточиями **(…)**
fignore=chars	Список суффиксов имен файлов, которые игнорируются при дополнении (см. **filec**)
filec	Если переменная установлена, частично набранное в командной строке имя файла может быть расширено до полного имени нажатием клавиши <Esc>. Если существует более одного варианта дополнения, просмотреть их можно, введя команду EOF. В **tcsh** переменная игнорируется
gid	Идентификатор группы пользователя
group	Имя группы пользователя
histchars=*ab*	Двухсимвольная строка, устанавливающая символы, которые используются для подстановок из журнала команд и быстрых подстановок (по умолчанию это строка **!^**)
histdup={**all**|**prev**}	Запоминать только уникальные события **(all)** либо не запоминать событие в журнале команд, если оно в точности совпадает с предыдущим **(prev)**
histfile=file	Журнал команд, читаемый по **history –S** и **history –L**. По умолчанию это файл ~/*.history*
histlit	Не производить расширения строк из журнала команд
history=*n format*	*n* – количество команд, хранимых в журнале. Формат отображения команд (*format*) доступен только в **tcsh**; описание форматов см. в разделе «Форматирование приглашения»
home=*dir*	Домашний каталог пользователя. Переменная инициализируется значением HOME. Символ «~» является сокращением этого значения
ignoreeof	Игнорировать символ EOF на терминале; это предотвращает непреднамеренное завершение работы интерпретатора

Переменная	Описание
implicitcd	Если имя каталога вводится в качестве команды, сделать этот каталог рабочим. Если переменная имеет значение **verbose**, переход отображается на терминале
inputmode={**insert** | **overwrite**}	Управление режимом работы редактора
listflags=*flags*	Один или более параметров **x**, **a** или **A** для встроенной команды **ls −F**. Второе слово может иметь значение пути к команде **ls**
listjobs=**long**	Когда задание приостанавливается, перечислить все задания (если указано, то в подробном формате)
listlinks	При выполнении команды **ls −F** отображать типы файлов, на которые указывают ссылки
listmax=*num*	Не позволять команде **list-choices** отображать более *num* вариантов выбора перед выдачей запроса
listmaxrows=*num*	Не позволять **list-choices** заполнять более *num* строк перед выдачей запроса
loginsh	Переменная установлена, если интерпретатор является интерпретатором, запускаемым при входе в систему
logout	Тип предстоящего завершения работы (**normal**, **automatic** или **hangup**)
mail=(*n files*)	Список файлов, проверяемых на наличие новых почтовых сообщений каждые *n* секунд (по умолчанию проверка указанных файлов выполняется каждые пять минут)
matchbeep={**never** | **nomatch** | **ambiguous** | **notunique**}	Предписание издавать звуковой сигнал при попытке дополнения, если вариантов дополнения нет (**nomatch**), существует более одного варианта (**ambiguous**), существует более одного варианта и один является точным (**notunique**) либо никогда (**never**)
nobeep	Отключить звуковой сигнал
noclobber	Запретить перенаправление вывода в существующий файл. Это предотвращает непреднамеренное уничтожение содержимого файлов
noglob	Отключить расширение имен файлов; бывает полезно в сценариях интерпретатора
nokanji	Отключить Kanji (если поддерживается)
nonomatch	Интерпретировать буквально специальные символы в именах файлов, если не существует точного соответствия маске. Так, **vi ch*** создаст новый файл **ch*** вместо выдачи сообщения «No match» (соответствие маске не найдено)

Переменная	Описание
nostat=directory-list	Не учитывать каталоги *directory-list* при дополнении имен
notify	Уведомлять о завершении работы задания
owd	Предыдущий рабочий каталог
path=(*dirs***)**	Список путей, в которых производится поиск выполняемых команд. Инициализируется значением переменной PATH. Значение по умолчанию: **./usr/ucb/usr/bin**
printexitvalue	Отображать все ненулевые коды возврата
prompt=*'str'*	Строка диалогового приглашения; по умолчанию – %. Более подробная информация содержится в разделе «Форматирование приглашения» далее в этой главе
prompt2='str'	Строка диалогового приглашения операторов циклов **foreach** и **while** и разорванных строк (с экранированными символами новой строки). См. также раздел «Форматирование приглашения»
prompt3='str'	Строка диалогового приглашения для ввода исправлений при проверке орфографии. См. также раздел «Форматирование приглашения»
promptchars=cc	Использовать два символа, заданные в виде *cc*, вместо последовательности %# переменной интерпретатора **prompt** (соответственно, первый символ для обычных пользователей, а второй – для суперпользователя)
pushdsilent	Не отображать стек каталогов при выполнении **pushd** и **popd**
pushdtohome	При вызове **pushd** без аргументов переходить в домашний каталог пользователя
recexact	Считать дополнение завершенным на первом же точном соответствии
recognize_only_executables	При дополнении команд отображать только исполняемые файлы
rmstar	Запрашивать подтверждение перед выполнением команды **rm ***
rprompt=string	Строка, отображаемая в правой части экрана при отображении приглашения в левой. Формат такой же, как для **prompt**
savedirs	Выполнить команду **dirs –S** при завершении работы

Переменная	Описание
savehist=*max* **[merge]**	Выполнить команду **history −S** при завершении работы. Сохранять не более *max* строк команд. При соответствующем указании объединить (*merge*) эти строки с уже созданным журналом команд и отсортировать по времени
sched=*string*	Формат отображения событий **sched**. Информацию о формате можно найти в разделе «Форматирование приглашения»
shell=*file*	Полный путь к текущему интерпретатору. По умолчанию – */bin/csh*
shlvl	Количество вложенных интерпретаторов
status=*n*	Код завершения последней команды. Встроенные команды возвращают значение 0 (успешное выполнение) или 1 (неудача)
symlinks={**chase**|**ignore**|**expand**}	Определить способ работы с символическими ссылками. Расширять их до реального имени каталога по **cwd** **(chase)**, считать действительными каталогами **(ignore)** или расширять аргументы, напоминающие путевые имена **(expand)**
tcsh	Версия **tcsh**
term	Тип терминала
time='*n* %*c*'	Если выполнение команды занимает более *n* секунд процессорного времени, отобразить пользовательское время, системное время, прошедшее время (для команды) и загрузку процессора (в процентах). Указывайте альтернативные %*c* флаги для отображения другой информации
tperiod	Количество минут между выполнениями псевдонима **periodic**
tty	Имя терминала, если таковой имеется
uid	Идентификатор пользователя
user	Регистрационное имя пользователя
verbose	Отображать команду после выполнения подстановки из журнала команд; идентично выполнению **csh −v**
version	Номер версии интерпретатора и дополнительная информация, включая параметры, установленные на этапе компиляции
visiblebell	Мерцание экрана вместо звукового сигнала

Переменная	Описание
watch=([*n*] *user terminal...*)	Наблюдение за входом в систему и выходом пользователя *user* на терминале *terminal* (*terminal* – имя терминала или **any**. *User* тоже может принимать значение **any**). Проверка каждые *n* минут (по умолчанию равно 10)
who=*string*	Задать информацию, отображаемую командой **watch**
wordchars=*chars*	Список всех не буквенно-цифровых символов, которые могут являться частью слова. По умолчанию это символы ***?_-.[ ]~=**

Форматирование переменной приглашения

Интерпретатор **tcsh** поддерживает набор подстановок, которые определяют вид приглашения (в **csh** доступны только простые строки приглашений и подстановка команд из журнала по символу **!**). Вот список существующих подстановок:

%% Символ %.

%/ Текущий рабочий каталог.

%~ Текущий рабочий каталог в записи с символом ~.

%# Символ # – для суперпользователя, > – для всех остальных пользователей.

%**?** Код завершения предыдущей команды.

%**b** Конец выделения полужирным шрифтом.

%**c**[[**0**]*n*], %**.**[[**0**]*n*]

Последние *n* (по умолчанию равно 1) компонент имени текущего рабочего каталога. Если задан 0, неотображаемые компоненты заменяются на /<**skipped**>.

%**d** День недели (Mon, Tue и т. д.).

%**h**, %**!**, **!**

Номер текущего события.

%**l** Текущий терминал.

%**m** Первый компонент имени узла.

%**n** Имя пользователя.

%**p** Текущее время с точностью до секунды (в 12-часовом формате).

%**s** Конец инверсного выделения (reverse video).

%**t**, %@

Текущее время (в 12-часовом формате).

%**u** Конец подчеркивания.

%**w** Текущий месяц (Jan, Feb и т. д.).

%y Текущий год (99, 00 и т. д.).

%B Начало полужирного выделения.

%C Аналогично **%c**, но с использованием абсолютных имен, а не записи через ~.

%D Число месяца (09, 10 и т. д.).

%M Полное имя узла.

%P Текущее время с точностью до секунды (в 24-часовом формате).

%S Начало инверсного выделения (reverse video).

%T Текущее время (в 24-часовом формате).

%U Начало подчеркивания.

%W Текущий месяц (09, 10 и т. д.).

%Y Текущий год (1999, 2000 и т. д.).

Пример файла .cshrc

```
# Предопределенные переменные

set path=(~  ~/bin  /usr/ucb  /bin  /usr/bin .)
set mail=(/usr/mail/tom)

if ($?prompt)  then              # настройки для диалогового режима
  set echo
  set noclobber ignoreeof

  set cdpath=(/usr/lib  /usr/spool/uucp)
# Теперь можно вводить команду cd macros вместо cd /usr/lib/macros

  set history=100
  set prompt='tom \!% '          # содержит номер команды
  set time=3

# Мои переменные

  set man1="/usr/man/man1"       # теперь можно использовать cd $man1, ls $man1
  set a="[a-z]*"                 # а также vi $a
  set A="[A-Z]*"                 # или grep string $A

# Псевдонимы

  alias c "clear; dirs"          # кавычки экранируют ; или |
  alias h "history|more"
  alias j jobs -l
  alias ls ls -sFC               # переопределяем команду ls
  alias del 'mv \!* ~/tmp_dir'   # безопасная альтернатива rm
endif
```

Переменные окружения

В интерпретаторе C shell существует набор *переменных окружения*, которые отличаются от переменных интерпретатора и не являются, по сути, его составляющей. Переменные интерпретатора имеют смысл только в текущем интерпретаторе, а переменные окружения автоматически экспортируются из интерпретатора и видимы глобально. К примеру, переменные C shell доступны только определенному сценарию, в котором они определяются, тогда как переменные окружения могут использоваться любым сценарием интерпретатора, почтовыми службами или редакторами, с которыми работает пользователь.

Переменные окружения инициализируются следующим образом:

```
setenv VAR value
```

По принятым соглашениям все буквы имен переменных окружения прописные. Можно создавать собственные переменные окружения либо использовать предопределенные существующие, которые перечислены ниже.

Каждой из этих переменных окружения соответствует переменная интерпретатора C shell. Если меняется значение одной из этих переменных, значение второй изменяется автоматически (курсивом отмечены переменные, действительные только для **tcsh**):

GROUP

Имя группы пользователя; то же, что и **group**.

HOME

Домашний каталог; то же, что и **home**.

PATH

Пути поиска команд; то же, что и **path**.

SHLVL

Уровень вложенности интерпретаторов; то же, что и **shlvl**.

TERM

Тип терминала; то же, что и **term**.

USER

Регистрационное имя пользователя; то же, что и **user**.

Вот некоторые из прочих переменных окружения, не имеющих соответствий среди переменных интерпретатора (курсивом выделены переменные, действительные только для **tcsh**):

COLUMNS

Количество колонок текста на терминале.

DISPLAY

Переменная, указывающая на экран пользователя в системе X Window. Если она установлена, интерпретатор отменяет действие **autologout**.

EDITOR

Полное имя редактора по умолчанию. См. также **VISUAL**.

HOST

Имя машины.

HOSTTYPE

Тип машины. Переменная устарела и будет в будущем изъята из употребления.

HPATH

Список каталогов, в которых производится поиск документации. Элементы разделяются двоеточием.

LANG

Предпочтительный язык. Используется для организации поддержки языков, отличных от английского.

LC_CTYPE

Параметры локализации, влияющие на обработку символов (локаль). Применяется для организации поддержки языков, отличных от английского.

LINES

Количество строк текста на терминале.

LOGNAME

Еще одно имя переменной USER.

MACHTYPE

Тип машины.

MAIL

Файл почтового ящика. Используется почтовыми программами. Переменная не идентична переменной C shell **mail**, по которой производится только проверка наличия новых сообщений.

NOREBIND

Отображаемые символы не могут переопределяться. Предназначена для организации поддержки языков, отличных от английского.

OSTYPE

Операционная система.

PWD

Текущий рабочий каталог; значение копируется из **cwd**.

REMOTEHOST

Имя машины (удаленного узла).

SHELL

Переменная по умолчанию не определена; инициализируется значением **shell**, после чего обе переменные идентичны.

TERMCAP

Файл, содержащий последовательности позиционирования курсора для разных типов терминалов пользователей. По умолчанию это файл */etc/termcap*.

VENDOR

Название производителя системы.

VISUAL

Полное имя экранного редактора по умолчанию. См. также **EDITOR**.

Выражения

Выражения используются в операторах **@**, **if** и **while** для выполнения арифметических и строковых сравнений, проверки свойств файлов и т. д. Команды **exit** и **set** также могут включать выражения, как и встроенная команда **filetest**. Выражения формируются из переменных и констант в сочетании с операторами, которые схожи с операторами языка программирования C. Приоритеты операторов идентичны приоритетам в языке C. Их легко запомнить следующим образом:

1. */ %
2. + −

Все прочие выражения необходимо заключать в круглые скобки. Скобки необходимы, если выражение содержит операторы <, >, & или |.

Операторы

Существуют операторы нескольких типов:

Операторы присваивания

Оператор	Описание
`=`	Присвоить значение
`+= -=`	Присвоить после сложения/вычитания
`*= /= %=`	Присвоить после умножения/деления/взятия остатка
`&= ^= |=`	Присвоить после выполнения поразрядных операций AND/XOR/OR
`++`	Инкремент
`--`	Декремент

Арифметические операторы

Оператор	Описание
`* / %`	Умножение, целочисленное деление, взятие остатка
`+ -`	Сложение, вычитание

Поразрядные и логические операторы

Оператор	Описание
`~`	Двоичная инверсия (дополнение до единицы)
`!`	Логическое отрицание
`<< >>`	Поразрядный сдвиг влево и вправо
`&`	Поразрядное AND (И)
`^`	Поразрядное исключающее OR (ИЛИ) – XOR
`|`	Поразрядное OR (ИЛИ)
`&&`	Логическое AND (И)
`||`	Логическое OR (ИЛИ)
`{ command }`	1, если команда завершилась успешно, 0 – в противном случае. Обратите внимание, что происходит инвертирование обычного кода завершения команды. Более практичным в некоторых случаях является использование переменной **$status**

Операторы отношений

Оператор	Описание
`== !=`	Равенство; неравенство
`<= >=`	Меньше либо равно; больше либо равно
`< >`	Строго меньше; строго больше

Операторы опроса файлов

Для строки *file* выполняется подстановка команд и расширение файловых имен перед проверкой. В **tcsh** условия можно комбинировать (например, **–ef**). Доступны следующие операторы:

Оператор	Описание
`-d file`	Файл является каталогом
`-e file`	Файл существует
`-f file`	Файл является обычным
`-o file`	Пользователь является владельцем этого файла

Оператор	Описание
`-r` *file*	У пользователя есть права на чтение файла
`-w` *file*	У пользователя есть права на запись в файл
`-x` *file*	У пользователя есть права на выполнение файла
`-z` *file*	Нулевой размер файла
`!`	Отрицание любого из вышеперечисленных операторов

Далее приведены некоторые операторы, специфичные для **tcsh**:

Оператор	Описание
`-b` *file*	Файл является специальным файлом блочного устройства
`-c` *file*	Файл является специальным файлом символьного устройства
`-g` *file*	Для файла установлен бит set-group-ID
`-k` *file*	Установлен бит «клейкости» файла (sticky bit)
`-l` *file*	Файл является символической ссылкой
`-L` *file*	Выполнять все последующие операторы для символической ссылки, а не для файла, на который она указывает
`-p` *file*	Файл является именованным каналом (FIFO)
`-s` *file*	Размер файла ненулевой
`-S` *file*	Файл является сокетом (socket)
`-t` *file*	Файл является открытым файловым дескриптором терминального устройства
`-u` *file*	Установлен бит set-user-ID для файла
`-X` *file*	Файл является исполняемым и расположен в одном из каталогов, заданных переменной окружения PATH, либо является встроенной командой интерпретатора

И наконец, в **tcsh** существуют следующие операторы, которые позволяют получать информацию иного рода:

Оператор	Описание
`-A[:]` *file*	Время последнего доступа к файлу в секундах, прошедших с начала эпохи (1 января 1970 г.). При указании двоеточия результат отображается в стандартном формате времени
`-C[:]` *file*	Время последнего изменения индексного дескриптора (inode-блока). При указании двоеточия результат отображается в стандартном формате времени
`-D` *file*	Номер устройства

Оператор	Описание
-F *file*	Составной идентификатор файла в виде *устройство:индексный дескриптор* (*device*:*inode*)
-G[:] *file*	Численный идентификатор группы файла. При указании двоеточия отображается имя группы, если оно доступно, в противном случае – численный идентификатор
-I *file*	Номер inode-блока
-L *file*	Имя файла, на который указывает символическая ссылка *file*
-M[:] *file*	Время последнего изменения файла. При указании двоеточия результат отображается в стандартном формате времени
-N *file*	Количество жестких ссылок для файла
-P[:] *file*	Права доступа для файла в восьмеричной системе, без первого нуля. При указании двоеточия нуль включается в отображение
-Pmode[:] *file*	Эквивалентен оператору –P *file* AND *mode* (логическая конъюнкция). При указании двоеточия ведущий нуль включается в отображение
-U[:] *file*	Численный идентификатор владельца файла. При указании двоеточия отображается имя пользователя, если оно доступно, в противном случае – численный идентификатор
-Z *file*	Размер файла в байтах

Примеры

В следующих примерах отображено использование операторов с командой @ в предположении, что n = 4:

Выражение	Значение $x
`@ x = ($n > 10 || $n < 5)`	1
`@ x = ($n >=0 && $n < 3)`	0
`@ x = ($n << 2)`	16
`@ x = ($n >> 2)`	1
`@ x = $n % 2`	0
`@ x = $n % 3`	1

Далее следуют примеры первых строк операторов **if** и **while**:

Выражение	Значение
`while ($#argv != 0)`	Пока не исчерпаны аргументы. . .
`if ($today[1] == “Fri”)`	Если первое слово – «Fri». . .

Выражение	Значение
`if (-f $argv[1])`	Если первым аргументом является имя обычного файла. . .
`if (! -d $tmpdir)`	Если `tmpdir` не является каталогом. . .

Журнал команд

Ранее выполнявшиеся команды сохраняются в специальном журнале. В интерпретаторе C shell существует возможность доступа к этому журналу с целью проверки, повторения команд либо выполнения измененных вариантов. Встроенная команда **history** отображает содержимое журнала; предопределенные переменные **histchars** и **history** влияют на работу механизма журнала команд. Существует четыре варианта использования журнала:

- Повторное выполнение одной из предыдущих команд
- Подстановка команд
- Подстановка аргументов (определенных параметров команды)
- Использование модификаторов для извлечения или замены частей команды или слова

В следующих подразделах описаны инструменты **csh**, применяемые для редактирования и повторного выполнения команд. При работе с **tcsh** доступна любая из этих возможностей. Кроме того, клавиши управления курсором обеспечивают навигацию в командной строке, а команды редактирования, описанные далее в разделе «Редактирование командной строки», – изменение команд. Клавиши управления курсором в **tcsh** следующие:

Клавиша	Действие
<Вверх>	Предыдущая команда
<Вниз>	Следующая команда
<Влево>	Переместиться влево по командной строке
<Вправо>	Переместиться вправо по командной строке

Подстановка команд

Команда	Описание
`!`	Указание интерпретатору на начало подстановки из журнала команд
`!!`	Выполнить последнюю команду
`!`*N*	Команда номер *N* из журнала
`!-`*N*	Команда, отстоящая назад от текущей на *N*
`!`*string*	Самая недавняя команда, начинающаяся со строки *string*

Команда	Описание
!?*string*?	Самая недавняя команда, содержащая строку *string*
!?*string*?%	Аргумент самой недавней команды, содержащий строку *string*
!$	Последний аргумент предыдущей команды
!!*string*	Выполнить предыдущую команду с добавлением строки *string*
!*N string*	Выполнить команду с номером *N* с добавлением строки *string*
!{*s1*}*s2*	Самая недавняя команда, начинающаяся со строки *s1*; выполнить с добавлением строки *s2*
^*old*^*new*^	Быстрая подстановка; в последней команде заменить строку *old* на строку *new* и выполнить измененную команду

Примеры подстановки команд

Рассматривается следующая команда:

```
%3 vi cprogs/01.c ch002 ch03
```

Номер события	Введенная команда	Выполняемая команда
4	^00^0	vi cprogs/01.c ch02 ch03
5	nroff !*	nroff cprogs/01.c ch02 ch03
6	nroff !$	nroff ch03
7	!vi	vi cprogs/01.c ch02 ch03
8	!6	nroff ch03
9	!?01	vi cprogs/01.c ch02 ch03
10	!{nr}.new	nroff ch03.new
11	!!|lp	nroff ch03.new | lp
12	more !?pr?%	more cprogs/01.c

Подстановка слов

Двоеточие может предварять любой из спецификаторов слов:

Спецификатор	Описание
:0	Имя команды
:*n*	Аргумент с номером *n*
^	Первый аргумент
$	Последний аргумент

Спецификатор	Описание
`:n-m`	Аргументы из диапазона от *n* до *m*
`-m`	Слова с 0 по *m*; идентично `:0-m`
`:n-`	Слова с *n* до предпоследнего
`:n*`	Аргументы от *n* до последнего; идентично *n*-$
`*`	Все аргументы; идентично ^–$ или 1–$
`#`	Текущая командная строка до текущей позиции; практически бесполезный спецификатор

Примеры подстановки слов

Рассматривается команда

```
%13 cat ch01 ch02 ch03 biblio back
```

Номер события	Введенная команда	Выполняемая команда
14	`ls !13^`	`ls ch01`
15	`sort !13:*`	`sort ch01 ch02 ch03 biblio back`
16	`more !cat:3*`	`more ch03 biblio back`
17	`!cat:0-3`	`cat ch01 ch02 ch03`
18	`vi !-5:4`	`vi biblio`

Модификаторы команд журнала

Подстановки команд и слов могут модифицироваться следующим образом:

Отображение, замена и экранирование

Модификатор	Описание
:p	Отобразить команду, но не выполнять ее
:s/*old*/*new*	Заменить первое вхождение подстроки *old* на *new*
:gs/*old*/*new*	Заменить все вхождения подстроки *old* на *new*
:&	Повторить предыдущую замену (команда **:s** или ^) первого вхождения
:g&	Повторить предыдущую замену для всех вхождений
:q	Заключить список слов в кавычки
:x	Заключить в кавычки отдельные слова

Усечение

Модификатор	Описание
:r	Извлечение первого доступного корня путевого имени (до последнего символа точки)
:gr	Извлечение всех корней
:e	Извлечение первого доступного расширения имени файла (символы за последней точкой)
:ge	Извлечение всех расширений
:h	Извлечение первого доступного заголовка (до последнего символа /)
:gh	Извлечение всех заголовков из имен файлов
:t	Извлечение первого доступного хвоста путевого имени (символы за последним слэшем)
:gt	Извлечение всех хвостов
:u	Перевод в верхний регистр первой строчной буквы (только в **tcsh**)
:l	Перевод в нижний регистр первой прописной буквы (только в **tcsh**)
:a	Применить к слову модификаторы, следующие за **a**, столько раз, сколько возможно. Если **a** используется совместно с модификатором **g**, то применяется ко всем словам (только в **tcsh**)

Примеры использования модификаторов команд журнала

В продолжение вышеприведенных примеров команда номер 17 выглядит так:

```
%17 cat ch01 ch02 ch03
```

Номер события	Введенная команда	Выполняемая команда
19	`!17:s/ch/CH`	`cat CH01 ch02 ch03`
20	`!17g&`	`cat CH01 CH02 CH03`
21	`!more:p`	`more cprogs/01.c` *(только отобразить)*
22	`cd !$:h`	`cd cprogs`
23	`vi !mo:$:t`	`vi 01.c`
24	`grep stdio !$`	`grep stdio 01.c`
25	`^stdio^include stdio^:q`	`grep "include stdio" 01.c`
26	`nroff !21:t:p`	`nroff 01.c` *(это ли нам нужно?)*
27	`!!`	`nroff 01.c` *(выполнить)*

Специальные псевдонимы в tcsh

В **tcsh** могут быть установлены специальные псевдонимы, которые изначально не определены. Если специальный псевдоним установлен, он вступает в действие по определенному событию. Ниже приводится список специальных псевдонимов:

beepcmd

При выдаче звукового сигнала.

cwdcmd

При изменении текущего рабочего каталога.

periodic

Каждые несколько минут. Точное значение устанавливается переменной интерпретатора *tperiod*.

precmd

Перед отображением нового приглашения командной строки.

shell *shell*

Если назначенный для выполнения интерпретатор не указан в сценарии явно, использовать интерпретатор *shell*. Строка *shell* должна являться абсолютным именем интерпретатора.

Работа с командной строкой

Как в **csh**, так и в **tcsh** доступна определенная функциональность для работы с командной строкой. Оба интерпретатора поддерживают дополнение команд, а **tcsh** позволяет редактировать командную строку.

Дополнение

Как в **tcsh**, так и в **csh** существует возможность дополнения набираемых слов. **tcsh** автоматически дополняет слова и команды по клавише <Tab>, а **csh** делает это при нажатии клавиши <Esc>, причем только когда установлена переменная *filec*. Если дополнение является неоднозначным (т. е. набираемой строке соответствуют несколько дополнений), интерпретатор дополняет слово, насколько это возможно, а затем издает звуковой сигнал, уведомляющий, что слово не окончено. Список возможных дополнений можно получить, нажав клавиши <Ctrl>+<D>. **tcsh** также уведомляет пользователя об окончании дополнения путем добавления пробелов к полным именам файлов и символа / к полным именам каталогов.

Как **csh**, так и **tcsh** опознают запись ~ как обозначение домашнего каталога. Интерпретаторы считают слова на новой строке и последующие символы |, &, ;, || и && командами и соответственно изменяют пути поиска. Дополнения могут использоваться для незаконченных слов; при этом только буквы слева от приглашения проверяются на наличие для них дополнений.

Сопутствующие переменные интерпретатора

- **autolist**
- **fignore**
- **listmax**
- **listmaxrows**

Сопутствующие команды редактирования командной строки

- **complete-word-back**
- **complete-word-format**
- **expand-glob**
- **list-glob**

Сопутствующие встроенные команды

- **comlpete**
- **uncomplete**

Редактирование командной строки в tcsh

tcsh позволяет перемещать курсор по командной строке и редактировать ее по собственному усмотрению. Существует два основных режима редактирования командной строки, которые берут начало от двух популярных текстовых редакторов – Emacs и **vi**. По умолчанию включен режим Emacs; переключаться между режимами можно следующим образом:

`bindkey -e`	*Комбинации Emacs*
`bindkey -v`	*Комбинации vi*

Главное различие между клавиатурными сочетаниями Emacs и **vi** состоит в том, что сочетания Emacs не зависят от режима, т. е. работают всегда однозначно. **vi** имеет два режима работы – командный и вставки; между этими режимами приходится переключаться. В каждом из режимов команды имеют свой смысл. Кроме того:

- Редактирование Emacs проще; редактирование **vi** более гибкое.
- Редактирование Emacs позволяет возвращать удаленный текст и выделять текст; в режиме редактирования **vi** это невозможно.
- Существенно различаются возможности поиска в журнале команд.

Режим Emacs

В табл. 8.1–8.3 приводятся различные команды, доступные в режиме редактирования Emacs.

Таблица 8.1. Команды позиционирования курсора (режим Emacs)

Команда	Описание
Ctrl-B	Откат на один символ
Ctrl-F	Перемещение вперед по тексту на один символ
Esc b	Перемещение назад по тексту на одно слово
Esc f	Перемещение вперед по тексту на одно слово
Ctrl-A	Перейти в начало строки
Ctrl-E	Перейти в конец строки

Таблица 8.2. Команды удаления текста (режим Emacs)

Команда	Описание
DEL или **Ctrl-H**	Удаление символа слева от курсора
Ctrl-D	Удаление символа под курсором
Esc d	Удаление слова
Esc DEL или **Esc Ctrl-H**	Удаление слова перед курсором
Ctrl-K	Удалить все символы от курсора до конца строки
Ctrl-U	Удалить всю строку

Таблица 8.3. Управление командами (режим Emacs)

Команда	Описание
Ctrl-P	Предыдущая команда
Ctrl-N	Следующая команда
<Вверх>	Предыдущая команда
<Вниз>	Следующая команда
cmd-fragment **Esc p**	Поиск в журнале по символам *cmd-fragment*, которые должны быть в начале команды
cmd-fragment **Esc n**	Аналогично **Esc p**, но поиск выполняется в направлении конца журнала
Esc *num*	Повторить следующую команду *num* раз
Ctrl-Y	Вставить последний удаленный текст (из специального буфера)

Режим vi

В режиме **vi** существует два подрежима – вставки и команд. Режимом по умолчанию является вставка. Переключение между режимами происходит по клавише <Esc>; из командного режима в режим вставки можно перейти по команде **a** (добавление) или **i** (вставка).

В табл. 8.4–8.10 содержатся описания команд редактирования, доступных в режиме **vi**.

Таблица 8.4. Команды, доступные в режиме вставки и командном режиме (vi)

Команда	Описание
Ctrl-P	Предыдущая команда
Ctrl-N	Следующая команда
<Вверх>	Предыдущая команда
<Вниз>	Следующая команда
Esc	Переключение режима

Таблица 8.5. Команды редактирования (режим вставки vi)

Команда	Описание
Ctrl-B	Перемещение назад по тексту на один символ
Ctrl-F	Перемещение вперед по тексту на один символ
Ctrl-A	Перемещение курсора в начало строки
Ctrl-E	Перемещение курсора в конец строки
DEL или **Ctrl-H**	Удалить символ слева от курсора
Ctrl-W	Удалить слово перед курсором
Ctrl-U	Удалить символы от начала строки до курсора
Ctrl-K	Удалить символы от курсора и до конца строки

Таблица 8.6. Команды позиционирования курсора (командный режим vi)

Команда	Описание
h или **Ctrl-H**	Перемещение назад по тексту на один символ
l или **пробел**	Перемещение вперед по тексту на один символ
w	Перемещение курсора на слово вперед (вправо)
b	Перемещение курсора на слово назад (влево)
e	Перемещение курсора к концу следующего слова
W, B, E	Аналогично командам **w**, **b**, **e**, но пробелы считаются разделителями слов
^ или **Ctrl-A**	Перемещение курсора в начало строки (к первому отображаемому символу)
0	Перемещение курсора в начало строки
$ или **Ctrl-E**	Перемещение курсора в конец строки

Таблица 8.7. Команды вставки текста (командный режим vi)

Команда	Описание
a	Добавление текста после курсора, пока не будет нажата клавиша **Esc**
i	Вставка нового текста перед курсором, пока не будет нажата клавиша **Esc**
A	Добавление нового текста в конец строки, пока не будет нажата клавиша **Esc**
I	Вставка нового текста в начало строки, пока не будет нажата клавиша **Esc**

Таблица 8.8. Команды удаления текста (командный режим vi)

Команда	Описание
x	Удалить символ под курсором
X или **DEL**	Удалить символ слева от курсора
d*m*	Удалить символы от курсора до конечного адреса команды поиска символов *m*
D	То же, что и **d$**
Ctrl-W	Удалить предыдущее слово
Ctrl-U	Удалить символы от начала строки до курсора
Ctrl-K	Удалить символы от курсора до конца строки

Таблица 8.9. Команды замены текста (командный режим vi)

Команда	Описание
c*m*	Заменить символы от курсора до конечного адреса команды поиска символов *m*, пока не будет нажата клавиша **Esc**
C	То же, что и **c$**
r*c*	Заменить символ под курсором на символ *c*
R	Заменять символы, пока не будет нажата клавиша **Esc**
s	Заменять символ под курсором набираемыми символами, пока не будет нажата клавиша **Esc**

Таблица 8.10. Команды поиска символов (командный режим vi)

Команда	Описание
f*c*	Переместить курсор к следующему вхождению символа *c* в строку
F*c*	Переместить курсор к предыдущему вхождению символа *c* в строку
t*c*	Переместить курсор к символу перед следующим вхождением символа *c* в строку

Команда	Описание
T*c*	Переместить курсор к символу после следующего вхождения символа *c* в строку
;	Повторить последнюю команду **f** или **F**
,	Повторить последнюю команду **f** или **F** в противоположном направлении

Управление заданиями

Работая с заданиями, пользователь может переводить приоритетные задания в фоновый режим, фоновые задания в приоритетный режим или временно приостанавливать выполнение заданий. Ниже представлены команды интерпретатора C shell, предназначенные для работы с заданиями.

bg Перевести задание в фоновый режим.

fg Перевести задание в приоритетный режим.

jobs

Перечислить активные задания.

kill Принудительно завершить задание.

notify

Послать уведомление о завершении фоновой задачи.

stop

Приостановить выполнение фонового задания. Дождаться завершения фоновых заданий.

Ctrl-Z

Приостановить выполнение приоритетного задания.

Многим командам работы с заданиями в качестве аргумента требуется идентификатор задания *jobID*. Этот аргумент может быть задан одним из следующих способов:

%*n* Задание с номером *n*.

%*s* Задание, командная строка которого начинается с подстроки *s*.

%?*s* Задание, командная строка которого содержит подстроку *s*.

%% Текущее задание.

% Текущее задание.

%+ Текущее задание.

%− Предыдущее задание.

Более подробную информацию о командах можно найти в разделе «Встроенные команды csh и tcsh» (далее в этой главе).

Встроенные команды csh и tcsh

@ [*variable* [*n*] =*expression*] **@**

Присвоить вычисленное значение выражения *expression* переменной *variable* либо *n*-ому элементу переменной, если указан аргумент *n*. Если имя переменной и значение не указаны, отобразить значения переменных окружения (идентично работе команды **set**). Операторы выражений и примеры приводятся в разделе «Выражения» ранее в этой главе. Также существуют две специальных формы этой команды:

@ *variable*++

Увеличить значение переменной на единицу.

@ *variable*––

Уменьшить значение переменной на единицу.

# #

Игнорировать часть строки, следующую за этим символом. Используется в сценариях интерпретатора в качестве символа, начинающего комментарий, и, по существу, не является командой.

#!*shell* **#!**

Размещается в первой строке сценария для обозначения интерпретатора, в котором должен выполняться данный сценарий. Интерпретатору могут передаваться дополнительные аргументы. Поддерживается не во всех интерпретаторах. Пример:

```
#!/bin/csh -f
```

: **:**

Пустая команда. Возвращает код завершения 0. Двоеточие часто является первым символом в сценариях интерпретаторов Борна и Корна, для того чтобы предотвратить ошибочное появление на этом месте символа #.

---

alias [*name* [*command*]] **alias**

Создать псевдоним (*name*) для команды *command*. Если аргумент *command* опущен, отобразить текущее назначение псев-

alias

донима *name*; команда без аргументов отображает текущие назначения всех псевдонимов. Псевдонимы могут определяться в командной строке, но, как правило, они описываются в файле *.cshrc* и доступны сразу после начала работы с интерпретатором (см. пример файла *.cshrc* ранее в этой главе). Определения псевдонимов могут содержать ссылки на аргументы командной строки, подобно тому, как происходит работа с журналом команд. Используйте **\!*** для ссылки на все аргументы командной строки, **\!^** – для ссылки на первый аргумент, **\!\!:2** – для ссылки на второй аргумент и т. д., а **\!$** – для ссылки на последний. Псевдоним (*name*) может совпадать с именем любой команды Unix; однако исходная команда будет доступна только при экранировании имени: *name*. См. также описание команды **unalias** и раздел «Специальные псевдонимы в tcsh».

Примеры

Установить размеры окон программы **xterm** в системе X Window:

```
alias R 'set noglob; eval `resize`; unset noglob'
```

Отобразить псевдонимы, содержащие строку **ls**:

```
alias | grep ls
```

Выполнить **nroff** с указанными в командной строке аргументами:

```
alias ms 'nroff -ms \!*'
```

Скопировать файл, имя которого указано в качестве первого аргумента:

```
alias back 'cp \!^  \!^.old'
```

Использовать обычную команду **ls**, а не ее псевдоним:

```
% \ls
```

alloc

alloc

Отображение данных о распределении памяти.

bg

bg [*jobIDs*]

Перевести текущее задание (или указанные задания) в фоновый режим.

bg

Пример

Чтобы перевести программу, выполняющуюся в течение длительного времени, в фоновый режим, можно начать со следующего:

```
4% nroff -ms report Ctrl-Z
```

а затем выполнить одну из команд:

```
5% bg
5% bg %        Текущее задание
5% bg %1       Задание с номером 1
5% bg %nr      Задание с командой nroff (поиск)
5% % &
```

bindkey

bindkey [*options*] [*key*] [*command*]

Только для **tcsh**. Отобразить все сочетания клавиш или назначить сочетание клавиш для команды.

Параметры

–a Перечислить стандартные и альтернативные сочетания клавиш.

–b *key*

key является либо управляющим символом (в ^- или C-записи, например, ^B или C–B), либо метасимволом (M–B), либо функциональной клавишей (F–*string*), либо расширенной префиксной клавишей (например, X–B).

–c *command*

Команда *command* является командой интерпретатора, а не редактора.

–d *key*

Восстановить исходную привязку сочетания клавиш.

–e Привязка к стандартным комбинациям Emacs.

–k *key*

Клавиша является курсорной (**<Влево>**, **<Вправо>**, **<Вверх>** или **<Вниз>**).

–l Перечислить и описать все команды редактора.

–r *key*

Полностью освободить клавиатурную комбинацию *key*.

bindkey	**–s** Интерпретировать команду как обычную строку, получаемую со стандартного ввода. **–u** Отобразить справку по использованию команды. **–v** Привязка к стандартным клавиатурным комбинациям **vi**.
break	**break** Продолжить выполнение с команды, следующей за ключевым словом **end**, принадлежащим ближайшему охватывающему циклу **while** или **foreach**.
breaksw	**breaksw** Аналог **break** для конструкции **switch**; выполнение продолжается с команды, следующей за **endsw**.
built-ins	**built-ins** Только для **tcsh**. Перечислить все встроенные команды интерпретатора.
bye	**bye** Только для **tcsh**. Синоним **logout**.
case	**case** *pattern* **:** Идентификация шаблона *pattern* в конструкции **switch**.
cd	**cd** [*dir*] Сменить рабочий каталог на *dir*; по умолчанию – на домашний каталог пользователя. Если *dir* является относительным путем, но не находится в текущем каталоге, происходит поиск по содержимому переменной **cdpath**. См. пример файла *.cshrc* ранее в этой главе. **Параметры команды для tcsh** **–** Перейти в предыдущий каталог. **–l** Явно расширять символ ~. **–n** Переносить имена ранее конца строки; подразумевается использование **–p**.

cd

−p Отобразить стек каталогов.

−v Отображать элементы стека по одному на строке; подразумевается использование **−p**.

chdir

chdir [*dir*]

Синоним **cd**. Наличие синонима полезно, если **cd** переопределяется.

complete

complete [*string* [*word*/*pattern*/*list*[:*select*]/[*suffix*]]]

Только для **tcsh**. Перечислить все дополнения либо все возможные дополнения указанной строки *string* (которая может являться шаблоном). Доступны следующие параметры:

Параметры для слова (word)

c Дополнять только текущее слово без обращения к шаблону *pattern*.

C Дополнять только текущее слово, обращаясь к шаблону *pattern*.

n Дополнить предыдущее слово.

N Дополнить слово перед предыдущим.

p Считать *pattern* диапазоном чисел. Выполнять дополнение в этом диапазоне.

Параметры для списков

Различные списки (*lists*) строк могут являться источниками возможных дополнений. Некоторые из параметров списков:

(*string*)

Элементы списка *string*.

$*variable*

Значение переменной *variable*.

`*command*`

Вывод команды *command*.

a Псевдонимы.

b Сочетания клавиш.

c Команды.

C Внешние (невстроенные) команды.

d Каталоги.

complete	**D** Каталоги, имена которых начинаются со строки *string*. **e** Переменные окружения. **f** Файлы. **F** Файлы, имена которых начинаются со строки *string*. **g** Группы. **j** Задания. **l** Ограничения. **n** Ничего. **s** Переменные интерпретатора. **S** Сигналы. **t** Текстовые файлы. **T** Текстовые файлы, имена которых начинаются со строки *string*. **u** Пользователи. **v** Любые переменные. **x** Аналогично **n**, но отображается *select* в качестве пояснения с командой редактора **list-choices**. **X** Дополнения. **select** *select* представляет собой расширяемый шаблон. Дополнения ограничиваются словами, которые соответствуют этому шаблону. Ко всем дополнениям добавляется указанный суффикс (*suffix*).
continue	**continue** Продолжить выполнение со следующей итерации охватывающего цикла **while** или **foreach**.
default	**default :** Отметка варианта по умолчанию для конструкции **switch** (является, как правило, последней в конструкции).
dirs	**dirs** [*options*] Отобразить стек каталогов, начиная с текущего каталога. См. также **popd** или **pushd**. Все параметры, кроме **–l**, **–n** и **–v**, являются расширениями **tcsh**.

dirs

Параметры

–c Обнулить стек каталогов.

–l Расширять символ домашнего каталога (~) до действительного пути.

–n Перенос вывода (wrap output) для длинных строк.

–v Отображать каждый каталог на отдельной строке.

–L *file*

Загрузить стек из файла *file*, который ранее был создан командой **dirs –S** *file*.

–S *file*

Отобразить в файл *file* последовательность команд **pushd** и **popd**, которая воспроизводит стек.

---

echo [**–n**] *string*

echo

Послать строку *string* на стандартный вывод; если указан параметр **–n**, не добавлять символ новой строки. В отличие от Unix-версии (*/bin/echo*) и версии команды интерпретатора Борна, **echo** интерпретатора **tcsh** не поддерживает экранирование символов. См. также описание команды **echo** в главе 3 «Команды Linux» и главе 7 «bash: командный интерпретатор Борна».

---

echotc [*options*] [*arguments*]

echotc

Только для **tcsh**. Отобразить характеристики терминала или переместить курсор в зависимости от аргумента.

Параметры

–s Вернуть пустую строку, вместо сообщения об ошибке, если характеристика не существует.

–v Подробные сообщения.

Аргументы

baud

Отобразить текущую скорость работы (в бодах).

cols

Отобразить текущую колонку.

cm *column row*

Переместить курсор в указанную позицию экрана.

echotc	**home** Переместить курсор в положение Home (в первую позицию текущего экрана). **lines** Отобразить количество строк экрана. **meta** Поддерживает ли терминал клавишу Meta? (Как правило, это ALT). **tabs** Поддерживает ли терминал табуляцию?
else	**else** Зарезервированное слово, используемое в операторе **if ... endif**.
end	**end** Зарезервированное слово, которым заканчиваются операторы **while** и **foreach**.
endif	**endif** Зарезервированное слово, которым заканчивается оператор **if**.
endsw	**endsw** Зарезервированное слово, которым заканчивается оператор **switch**.
eval	**eval** *args* Как правило, **eval** используется в сценариях интерпретатора, а аргументы являются строкой кода, которая может содержать переменные интерпретатора. **eval** производит подстановку значений переменных, а затем выполняет полученную команду. Такое «двойное сканирование» незаменимо, если переменные интерпретатора содержат символы перенаправления ввода/вывода, псевдонимы или другие переменные интерпретатора. Например, перенаправление в обычных условиях происходит прежде подстановки значения переменных, поэтому переменная, содержащая перенаправления, должна быть предварительно расширена с помощью **eval**; в противном случае символы перенаправления останутся незамеченными.

eval

Примеры

Следующая строка может присутствовать в файле **.login** в целях установки характеристик терминала:

```
set noglob eval `tset -s xterm` unset noglob
```

Следующие команды показывают действие **eval**:

```
% set b='$a'
% set a=hello
% echo $b            Однократное прочтение команды
$a
% eval echo $b       Двукратное прочтение команды
hello
```

Еще один пример использования **eval** см. в описании команды **alias**.

---

exec *command*

exec

Выполнить команду вместо текущего интерпретатора. Происходит завершение работы интерпретатора, а не порождение процесса внутри него.

---

exit [(*expr*)]

exit

Завершить выполнение сценария интерпретатора с кодом завершения, равным результату вычисления выражения *expr*. Нулевой код завершения означает успешное выполнение; ненулевой – ошибку. Если выражение не задано, код завершения имеет значение переменной **status**. Команда **exit** может быть выполнена в командной строке для завершения текущего сеанса работы с интерпретатором.

---

fg [*jobIDs*]

fg

Перевести текущее задание (или задание с указанным идентификатором) в режим приоритетного выполнения. Идентификатор задания может иметь вид %*job-number* (номер задания).

Пример

Если вы приостановили сеанс редактирования в **vi** (нажав клавиши <Ctrl>+<Z>), продолжить редактирование можно одной из следующих команд:

fg

```
% %
% fg
% fg %
% fg %vi        По имени задачи (поиск)
```

filetest

filetest –*op files*

Только для **tcsh**. Применить оператор проверки (*op*) к перечисленным файлам. Выдать результаты в виде списка. В разделе «Операторы опроса файлов» перечислены допустимые операторы.

foreach

foreach *name* (*wordlist*)
commands
end

Присваивать переменной *name* последовательно значения из списка *wordlist* и выполнять команды *commands* на каждой итерации. **foreach** можно использовать в качестве многострочной команды, набираемой в приглашении интерпретатора C shell (см. первый пример ниже) либо в сценарии (второй пример).

Примеры

Переименовать все файлы, имена которых начинаются с заглавной буквы:

```
% foreach i ([A-Z]*)
? mv $i $i.new
? end
```

Проверить, является ли параметром каждый из аргументов командной строки:

```
foreach arg ($argv)
   # начинается ли с символа - ?
   if ("$arg" =~ -*) then
      echo "Аргумент является параметром"
   else
      echo "Аргумент является именем файла"
   endif
end
```

glob

glob *wordlist*

Выполнить подстановку переменных, команд журнала и расширений имен файлов в список слов *wordlist*. Символы

экранирования «\» не опознаются при расширении, слова разделяются пустыми символами. **glob** обычно используется в сценариях интерпретатора для жесткого присваивания значения, которое не меняется на протяжении сценария.

glob

goto *string*

Безусловный переход к строке, которая начинается с метки *string* и двоеточия, и продолжение выполнения сценария с этой строки. *string* может быть переменной или маской имени файла, но сама метка в тексте программы должна быть буквальным, нерасширяемым значением. Метка не должна находиться внутри тела цикла **while** или **foreach**.

goto

hashstat

Отобразить статистику по эффективности использования хеш-таблицы при поиске команд по переменной **path**.

hashstat

history [*options*]

Отобразить события из журнала команд. (Работа с ним обсуждалась ранее в этой главе в разделе «Журнал команд».)

history

Параметры

–c Только для **tcsh**. Очистить журнал команд.

–h Отобразить список событий без нумерации.

–r Отобразить в обратном порядке; самая старая команда будет последней в списке.

n Отобразить последние *n* команд из журнала, не учитывая значение переменной интерпретатора **history**.

–L *file*

Только для **tcsh**. Загрузить последовательность команд **pushd** и **popd** из файла *file* и воссоздать сохраненный стек каталогов.

–M *file*

Только для **tcsh**. Объединить текущий стек каталогов и стек, сохраненный в указанном файле. Сохранить объединение в файле *file* как последовательность команд **pushd** и **popd**.

history

–S *file*

Только для **tcsh**. Отобразить последовательность команд **pushd** и **popd**, воссоздающую стек, в файл *file*.

Пример

Сохранить и выполнить пять последних команд:

```
history -h 5 > do_it
source do_it
```

hup

hup [*command*]

Только для **tcsh**. Выполнить команду, но принудительно завершить задание при посылке сигнала hangup при завершении работы интерпретатора. Команда без параметра может использоваться внутри сценария, который должен быть принудительно завершен при выходе из интерпретатора.

if

if

Начало условного оператора. Простейший формат оператора:

```
if (expr) cmd
```

Применяются еще три формата, которые приведены ниже:

```
if (expr) then    if (expr) then    if (expr) then
   cmds              cmds1             cmds1
endif             else              else if (expr) then
                     cmds2             cmds2
                  endif             else
                                       cmds3
                                    endif
```

В первой форме оператора команда *cmd* выполняется, если выражение *expr* истинно, иначе ничего не происходит (кроме перенаправления, что является ошибкой). Во всех прочих вариантах выполняется одна или более команд. Если выражение истинно, выполняются команды, расположенные непосредственно после **then**; если *expr* ложно, происходит выполнение команд после **else** (либо переход к **else if** и следующей проверке). Дополнительные примеры см. ранее в этой главе (в разделе «Выражения»), а также в описаниях **while** и **shift**.

Пример

Предпринять действия по умолчанию, если сценарий выполняется без аргументов:

```
if ($#argv == 0) then
   echo "No filename given.  Sending to Report"
   set outfile = Report
else
   set outfile = $argv[1]
endif
```

if

jobs [–l]

jobs

Перечислить все работающие или остановленные задания; включить в перечисление идентификаторы процессов, если указан параметр **–l**. Например, можно проверить, завершился ли уже долгий процесс компиляции или форматирования текста. Также полезно выполнять эту команду перед выходом из интерпретатора.

kill [*options*] *ID*

kill

Принудительно завершить процесс или задание с указанным идентификатором. Для принудительного завершения необходимо быть владельцем процесса или привилегированным пользователем. Эта встроенная команда аналогична */bin/kill*, описанной в главе 3 «Команды Linux», но также позволяет использовать символьные имена задач. «Упрямые» процессы могут быть завершены при помощи сигнала 9.

Параметры

–l Перечислить имена сигналов (параметр исключает все прочие действия).

–signal

Номер сигнала или его имя без префикса SIG. Команда **kill –l** позволяет получить список доступных имен сигналов. Список меняется в зависимости от архитектуры; для PC-системы он может выглядеть так:

```
% kill -l
HUP INT QUIT ILL TRAP ABRT BUS FPE KILL USR1 SEGV USR2
PIPE ALRM TERM STKFLT CHLD CONT STOP TSTP TTIN TTOU URG
XCPU XFSZ VTALRM PROF WINCH POLL PWR UNUSED
```

Сигналы и их номера определены в файле */usr/include/asm/signals.h*, который можно просмотреть, чтобы определить, какие сигналы доступны в системе.

Примеры

Предположим, пользователь выполнил команду:

kill

```
44% nroff -ms report &
```

Принудительно завершить ее выполнение можно любым из следующих способов:

```
45% kill 19536          Идентификатор процесса
45% kill %              Текущее задание
45% kill %1             Задание с номером 1
45% kill %nr            Начало строки команды
45% kill %?report       Поиск по подстроке
```

limit

limit [–h] [*resource* [*limit*]]

Отобразить ограничения, установленные на использование ресурсов текущим и порожденными процессами, либо установить ограничение *limit*. Если ограничение не задано, отображается текущее значение для данного ресурса (*resource*). Если имя ресурса также опущено, отображается сводка по всем типам ресурсов. По умолчанию отображаются или устанавливаются значения мягких ограничений; если указан параметр **–h**, устанавливаются жесткие ограничения. Жесткие ограничения означают конкретное пороговое значение, которое не может быть превышено. Только привилегированный пользователь может устанавливать жесткие ограничения. См. также **unlimit**.

Параметр

–h Жесткие ограничения.

Ресурсы

cputime

Максимальное количество используемого процессорного времени в секундах; может быть сокращено до **cpu**.

filesize

Максимальный размер произвольного файла.

datasize

Максимальный размер сегмента данных (включая стек).

stacksize

Максимальный размер стека.

coredumpsize

Максимальный размер файла образа оперативной памяти.

Ограничения Ограничение – это число, за которым может следовать символ (определитель единиц). Для ресурса **cputime**: *n***h** (*n* часов) *n***m** (*n* минут) *mm:ss* (минуты и секунды) Для прочих ресурсов: *n***k** (*n* Кбайт, по умолчанию) *n***m** (*n* Мбайт)	**limit**
log Только для **tcsh**. Список пользователей, за которыми ведется наблюдение, определяется переменной **watch**. Отобразить список пользователей, работающих с системой в настоящий момент. Если в качестве параметра указан символ «–», сбросить значения переменных окружения, как если бы пользователь только что зарегистрировался в системе (в новой группе).	**log**
login [*user* | **–p**] Для пользователя *user* заменить интерпретатор по умолчанию на */bin/login*. Параметр **–p** указывается для сохранения параметров окружения.	**login**
logout Завершить сеанс работы с интерпретатором.	**logout**
ls–F [*options*] [*files*] Только для **tcsh**. Быстрая альтернатива **ls –F**. При запуске с любыми дополнительными параметрами выполняется **ls**.	**ls–F**
newgrp [–] [*group*] Только для **tcsh**. Изменить идентификатор группы пользователя на указанный либо, если идентификатор не задан, на идентификатор исходной группы. Если в качестве параметра указан символ «–», сбросить значения переменных окружения, как если бы пользователь только что зарегистрировался в системе (в новой группе).	**newgrp**

nice	**nice** [±*n*] *command* Изменить приоритет выполнения указанной команды либо, если имя команды опущено, изменить приоритет выполнения текущего интерпретатора (см. также описание команды **nice** в главе 3 «Команды Linux»). Диапазон приоритетов: от −20 до 20, по умолчанию принимается значение 4. Наивысший приоритет (самое быстрое выполнение) придает значение −20; значение 20 соответствует самому низкому приоритету выполнения. Отрицательные значения приоритета может назначать только привилегированный пользователь. +*n* Прибавить число *n* к значению приоритета (понизить приоритет задачи). −*n* Вычесть *n* из значения приоритета (повысить приоритет задачи). Может использоваться только привилегированным пользователем.
nohup	**nohup** [*command*] Не посылать сигналы hangup. Не завершать принудительно выполнение указанной команды после завершения работы (разрыва соединения или выхода из интерпретатора). Использование команды без параметров внутри сценария интерпретатора предотвращает принудительное завершение сценария (см. также описание **nohup** в главе 3 «Команды Linux»).
notify	**notify** [*jobID*] Немедленно уведомить пользователя о том, что фоновая задача завершилась (это предотвращает ситуации, в которых пользователь, занятый, к примеру, редактированием файла, не может узнать о завершении задачи). Если не задан идентификатор задачи *jobID*, уведомление происходит при завершении текущей фоновой задачи.
onintr	**onintr** *label* **onintr** − **onintr** «On interrupt» (по прерыванию). В сценариях интерпретатора служит для обработки сигналов прерываний (аналогично командам **bash trap 2** и **trap "" 2**). Первый вариант команды похож на оператор **goto** *label*. При получении сигнала преры-

onintr

вания (например, **Ctrl-C**) сценарий переходит на метку *label*. Вторая форма позволяет сценарию игнорировать прерывания. Это полезно в начале сценария и при наличии сегментов команд, которые должны выполняться беспрепятственно (например, при переносе файлов). Третья форма восстанавливает обработку прерываний, которая была отключена командой **onintr –**.

Пример

```
onintr cleanup          Перейти на «cleanup» по прерыванию
.
.                       Команды сценария
.
cleanup:                Метка для прерываний
  onintr -              Игнорировать остальные прерывания
  rm -f $tmpfiles       Удалить все созданные файлы
  exit 2                Завершение с кодом ошибки
```

popd

popd [*options*]

Удалить текущую запись из стека каталогов или *n*-ный элемент. Номер текущей записи – 0, при отображении это самый первый элемент. См. также **dirs** и **pushd**.

Параметры

+*n* Указание *n*-го элемента стека.

–l Расширять символ ~.

–n Переносить длинные строки.

–p Отменить действие переменной интерпретатора **pushdsilent**, предотвращающей отображение содержимого результирующего стека.

–v Отображать один каталог в строке.

printenv

printenv [*variable*]

Отобразить имена и значения всех переменных окружения либо только указанной переменной.

pushd

pushd *name*
pushd [*options*]
pushd

Первый вариант команды меняет рабочий каталог на каталог *name* и добавляет его в стек каталогов. Вторая форма про-

pushd

производит прокрутку *n*-го каталога к вершине стека и делает его рабочим (нумерация начинается с 0). Команда без аргументов меняет местами два каталога на вершине стека и заменяет текущий рабочий каталог. Параметры +*n*, **−l**, **−n** и **−v** имеют такой же смысл, как и в **popd**. См. также **dirs** и **popd**.

Примеры

```
% dirs
/home/bob /usr
% pushd /etc              Добавить /etc в стек каталогов
/etc /home/bob /usr
% pushd +2                Переход к третьему каталогу
/usr /etc /home/bob
% pushd                   Обменять местами два первых каталога
/etc /usr /home/bob
% popd                    Удалить текущую позицию; перейти к
                          следующей
/usr /home/bob
```

rehash

rehash

Создать заново хеш-таблицу для переменной PATH. Используется, когда в текущем сеансе добавлена новая команда. Это позволяет быстрее находить и выполнять команду. Путь к добавляемой команде (программе) следует добавить в переменнуюPATH, прежде чем выполнять **rehash**. См. также **unhash**.

repeat

repeat *n command*

Выполнить команду *n* раз.

Примеры

Распечатать три копии файла **memo**:

```
% repeat 3 pr memo | lp
```

Прочесть 10 строк с терминала и сохранить в файле **item_list**:

```
% repeat 10 line > item_list
```

Добавить 50 экземпляров файла шаблонов к отчету **report**:

```
% repeat 50 cat template >> report
```

sched [*options*]
sched *time command*

sched

Только для **tcsh**. Команда без параметров отображает все запланированные события. Второй вариант команды добавляет событие в расписание.

Время (*time*) должно быть задано в формате *hh*:*mm* (например, 13:00).

Параметры

+*hh*:*mm*
Выполнить команду через *hh*:*mm* от текущего момента.

−*n* Удалить *n*-ную запись из расписания.

---

set *variable* = *value*
set [*option*] *variable* [*n*] = *value*
set

set

Присвоить значение *value* переменной *variable*. Если задано несколько значений, сделать переменную указателем на список слов. Если задан индекс *n*, присвоить значение *value* слову переменной с номером *n* (переменная уже должна являться списком по меньшей мере из *n* слов). Команда без аргументов отображает имена и значения существующих переменных. См. также раздел «Встроенные переменные интерпретатора» в этой главе.

Параметр

−**r** Только для **tcsh**. Перечислить переменные, ограничившись доступными только для чтения, или установить для указанных переменных режим «только для чтения».

Примеры

`% set list=(yes no maybe)`	*Присваивание списка слов*
`% set list[3]=maybe`	*Присваивание значения элементу из существующего списка слов*
`% set quote="Make my day"`	*Присваивание значения переменной*
`% set x=5 y=10 history=100`	*Несколько присваиваний*
`% set blank`	*Присваивание пустого значения переменной* `blank`

---

setenv [*name* [*value*]]

setenv

Присвоить значение *value* переменной окружения с именем *name*. По принятому соглашению все символы имени перемен-

setenv	ной должны быть прописными. Значение *value* может иметь вид отдельного слова или строки, заключенной в кавычки. Если значение не задано, переменная получает пустое значение. Команда без аргументов отображает имена и значения всех переменных окружения. Переменная PATH может не устанавливаться командой **setenv**, поскольку ее значение экспортируется из переменной интерпретатора **path**.
settc	**settc** *capability value* Только для **tcsh**. Установить значение (*value*) характеристики (*capability*) терминала.
setty	**setty** [*options*] [+ | − *mode*] Только для **tcsh**. Запретить изменение интерпретатором определенных режимов терминала. По умолчанию влияет на режимы выполнения. **Параметры** +*mode* Указание параметра без аргумента перечисляет все режимы указанного типа, которые установлены. Иначе включается указанный режим. −*mode* Указание параметра без аргумента перечисляет все режимы указанного типа, которые отключены. Иначе включить указанный режим. **−a** Перечислить все режимы определенного типа. **−d** Работать с режимами редактирования. **−q** Работать с режимами цитирования (используются при буквальном вводе символов). **−x** Работать с режимами выполнения (по умолчанию; устанавливается при выполнении примеров).
shift	**shift** [*variable*] Если задана переменная, сдвинуть слова из списка (т. е. элемент *name*[**2**] занимает место *name*[**1**]). Команда без аргументов сдвигает позиционные параметры (параметры командной строки), т. е. параметр **$2** замещает **$1**. Команда **shift** обычно используется в циклах **while**. См. дополнительный пример в описании **while**.

shift

Пример

```
while ($#argv)          Пока не кончились аргументы
    if (-f $argv[1])
       wc -l $argv[1]
    else
       echo "$argv[1] не является обычным файлом"
    endif
    shift               Берем следующий аргумент
end
```

source

source [–h] *script* **[***args***]**

Прочитать и выполнить команды из сценария интерпретатора C shell. Если указан параметр **–h**, команды из сценария добавляются в журнал команд, но не выполняются.

Пример

```
source ~/.cshrc
```

stop

stop [*jobIDs***]**

Приостановка текущих фоновых заданий или заданий, для которых указаны идентификаторы; команда является дополнением к **Ctrl-Z** (<Ctrl>+<Z>) или **suspend**.

suspend

suspend

Приостановить выполнение приоритетного задания; идентично **Ctrl-Z**. Часто используется для приостановки действия команды **su**.

switch

switch

Выбор последовательности команд в зависимости от значения управляющей переменной. Когда необходимо обработать более трех вариантов, оператор **switch** удобнее конструкции **if-then-else**. Если переменная *string* соответствует шаблону *pattern1*, выполняется первый набор команд; если шаблону *pattern2* – второй набор команд и т. д. Если соответствие шаблону не найдено, выполняются команды из секции **default**. Параметр *string* может задаваться с помощью подстановки команд, подстановки переменных или расширения имен файлов. Шаблоны могут содержать метасимволы *, ? и [ ]. Для выхода из конструкции используется оператор **breaksw**. Если этот оператор опускается (что бывает довольно редко),

switch

switch продолжает выполнение следующего набора команд, пока выполнение не достигнет оператора **breaksw** или **endsw**. Ниже приводится общий синтаксис конструкции **switch** параллельно с конкретным примером, в котором обрабатывается первый аргумент командной строки:

```
switch (string)      switch ($argv[1])
  case pattern1:     case -[nN]:
      commands       nroff $file | lp
      breaksw        breaksw
  case pattern2:     case -[Pp]:
      commands       pr $file | lp
      breaksw        breaksw
  case pattern3:     case -[Mm]:
      commands       more $file
      breaksw        breaksw
    .                case -[Ss]:
    .                sort $file
    .                breaksw
  default:           default
      commands       echo "Error-no such option"
                     exit 1
      breaksw        breaksw
endsw                endsw
```

telltc

telltc

Только для **tcsh**. Отобразить все характеристики терминала и их значения.

time

time [*command*]

Выполнить указанную команду и вычислить время выполнения. Команда без аргумента может использоваться в сценарии для замера времени его выполнения.

umask

umask [*nnn*]

Отобразить маску прав доступа для вновь создаваемых файлов или установить восьмеричное значение маски *nnn*. Маска определяет, какие компоненты прав доступа отсутствуют у создаваемых файлов.

unalias

unalias *name*

Удалить *name* из списка псевдонимов. Подробнее см. **alias**.

uncomplete *pattern* Только для **tcsh**. Удалить дополнения (заданные командой **complete**), имена которых соответствуют шаблону *pattern*.	**uncomplete**
unhash Удалить внутреннюю хеш-таблицу. Интерпретатор C shell перестанет использовать хешированные значения, что приведет к дополнительным затратам времени на поиск файлов по путям **path**. См. также **rehash**.	**unhash**
unlimit [–h] [*resource***]** Снять ограничение на использование указанного ресурса. Если ресурс не задан, снять ограничения для всех ресурсов. См. также **limit**. Параметр **–h** является предписанием снять жесткие ограничения. Эта команда может выполняться только привилегированным пользователем.	**unlimit**
unset *variables* Удалить указанные переменные. Имена переменных могут задаваться масками с использованием специальных символов для имен файлов. Переменные, доступные только для чтения, не изменяются этой командой. См. **set**.	**unset**
unsetenv *variable* Удалить переменную окружения. Расширение маски не производится. См. **setenv**.	**unsetenv**
wait Остановить выполнение до тех пор, пока не завершатся все порожденные процессы либо не будет получен сигнал прерывания.	**wait**
watchlog Только для **tcsh**. То же, что и **log**. Команда должна быть добавлена во время компиляции интерпретатора; см. значение переменной **version**.	**watchlog**

where	**where** *command* Только для **tcsh**. Отобразить все псевдонимы, встроенные команды и программы с именем *command*.
which	**which** *command* Только для **tcsh**. Отображение номера версии команды, которая будет выполнена. Идентично исполняемой программе **which**, но работает быстрее и учитывает встроенные команды **tcsh**.

while	**while** (*expression*)
	  commands
	end
	
	До тех пор пока истинно выражение *expression* (т. е. его значение не равно нулю), выполнять команды (*commands*) из тела цикла (между **while** и **end**). Для выхода из цикла и перехода к следующей итерации могут использоваться операторы **break** и **continue**. См. также пример в описании команды **shift**.
	
	Пример
	
	`set user = (alice bob carol ted)`
	`while ($argv[1] != $user[1])` *Перебирать пользователей в поисках совпадения*
	`  shift user` *Если пользователь не найден...*
	`  if ($#user == 0) then`
	`  echo "$argv[1] нет в списке пользователей"`
	`  exit 1`
	`  endif`
	`end`

Глава 9

Шаблоны и поиск

В Linux многие инструменты, работающие с текстом, позволяют выполнять поиск, а в некоторых случаях и замену, по текстовым шаблонам, а не буквальным строкам. Среди таких инструментов – редакторы **ed**, **ex**, **vi** и **sed**; язык сценариев **awk;** команды **grep** и **egrep**. Текстовые шаблоны (называемые также *регулярными выражениями*) содержат обычные символы и символы специальные (называемые также *метасимволами*).

Поддержка регулярных выражений в языке Perl настолько мощна, что таблиц этой главы на нее не хватит. Подробное рассмотрение данной темы можно найти в книгах издательства O'Reilly «Perl in a Nutshell», «Perl 5 Pocket Reference» (Perl 5: карманный справочник) или «Programming Perl» (Программирование на Perl). В редакторе Emacs также доступна работа с регулярными выражениями, схожими с описываемыми в этой главе.

Редакторы **ed** и **ex** в настоящее время практически не используются сами по себе в качестве диалоговых редакторов. Но **ed** применяется как пакетный обработчик, вызываемый из сценариев интерпретатора, а команды **ex** часто вызываются в редакторе **vi** посредством команды «:» (двоеточие). В данной главе при упоминании свойств **vi** имеются в виду регулярные выражения, доступные как в **vi**, так и в редакторе **ex**, на котором **vi** построен.

Инструменты **sed** и **awk** широко используются в сценариях интерпретатора и в качестве фильтров, обрабатывающих текст.

В данной главе представлены следующие темы:

- Имена и шаблоны файлов
- Список метасимволов, доступных в каждой из программ
- Описание метасимволов
- Примеры

Подробное руководство по шаблонам и поиску можно найти в книге Джеффри Фридла (Jeffrey E.F. Friedl) «Mastering Regular Expressions», вышедшей в серии Nutshell Handbook.

Имена файлов в сравнении с шаблонами

Метасимволы, применяемые при поиске по шаблону, отличаются от специальных символов расширения имен файлов. Специальные символы команды, вводимой в командной строке, сначала обрабатываются интерпретатором и лишь затем программой; как следствие неэкранированные метасимволы используются интерпретатором для расширения имен файлов. К примеру, команда

```
$ grep [A-Z]* chap[12]
```

может быть понята интерпретатором как:

```
$ grep Array.c Bug.c Comp.c chap1 chap2
```

и **grep** будет производить поиск строки «Array.c» в файлах *Bug.c*, *Comp.c*, *chap1* и *chap2*. Чтобы обойти расширение спецсимволов интерпретатором, необходимо экранировать их кавычками:

```
$ grep "[A-Z]*" chap[12]
```

В большинстве случаев достаточно двойных кавычек, но самым безопасным вариантом являются кавычки одинарные.

Отметьте также, что символы ***** и **?** имеют немного разный смысл для поиска по шаблону и расширения имен файлов.

Метасимволы в программах Linux

Некоторые специальные символы могут использоваться не во всех программах. Метасимволы, доступные в определенной программе, отмечены в следующей таблице точкой (•). Примечания даны после таблицы, а полное описание всех символов можно найти в следующем подразделе.[1]

Символ	ed	vi	sed	awk	grep	egrep	Значение
.	•	•	•	•	•	•	Любой символ (в **gawk** включает символ новой строки)
*	•	•	•	•	•	•	Любое количество вхождений предшествующего символа (даже нулевое)

[1] В некоторых из указанных программ (в зависимости от версии) поддерживаемые регулярные выражения могут несколько отличаться от приведенных здесь. Если поведение указанного регулярного выражения отличается от ожидаемого, следует обратиться к документации. – *Примеч. науч. ред.*

Символ	ed	vi	sed	awk	grep	egrep	Значение
^	•	•	•	•	•	•	Привязка к началу строки
$	•	•	•	•	•	•	Привязка к концу строки
\	•	•	•	•	•	•	Экранирует следующий символ
[ ]	•	•	•	•	•	•	Один из перечисленных символов или символ из диапазона
\(\)	•	•	•				Сохранить найденный текст для последующего использования
n	•	•	•				Повторное использование текста, найденного и сохраненного *n*-ным вхождением элемента \(\)
{ }				•			Диапазон символов
\{ \}	•	•	•		•		Диапазон символов
\< \>		•					Привязка к началу или к концу слова
+				•	•	•	Одно или более вхождений предшествующего символа
?				•	•	•	Отсутствие или одно вхождение предшествующего символа
|				•		•	Варианты шаблона
( )				•		•	Группировка выражений для поиска

В некоторых дистрибутивах Linux **grep** является ссылкой на **egrep**, поэтому команда **grep** на самом деле вызывает **egrep**.

При выполнении операции подстановки (поиска и замены) с помощью редакторов **ed**, **vi** и **sed** метасимволы приведенной таблицы имеют смысл только в шаблонах поиска, но не в строке замены.

При работе с **awk** следует помнить, что символы **{ }**, упомянутые в стандарте POSIX, поддерживаются **gawk** только при запуске с параметром **–Wre-interval**.

Следующие дополнительные метасимволы в редакторах **ed**, **vi** и **sed** доступны только для применения в шаблонах замены:

Символ	ex	sed	ed	Значение
\	•	•	•	Экранировать следующий символ
n	•	•	•	Вставить текст, найденный *n*-ным вхождением \(\)
&	•	•		Вставить предыдущий шаблон поиска

Символ	ex	sed	ed	Значение
~	•			Вставить предыдущий шаблон замены
\e	•			Отменить предыдущий метасимвол \u или \l
\E	•			Отменить предыдущий метасимвол \U или \L
\l	•			Сменить регистр следующего символа на нижний
\L	•			Сменить на нижний регистр следующих символов вплоть до \E
\u	•			Сменить регистр следующего символа на верхний
\U	•			Сменить на верхний регистр следующих символов вплоть до \E

Метасимволы

Следующие символы имеют специальное значение только в шаблонах поиска:

Символ	Значение
.	Соответствует любому *одиночному* символу, кроме символа новой строки (newline)
*	Соответствует любому количеству вхождений (или ни одного) непосредственно предшествующего символа. Предшествующий символ также может быть регулярным выражением, например, поскольку «.» (точка) означает любой символ, выражение **.*** означает *найти любое количество любых символов* (кроме символов конца строки)
^	Найти соответствие следующему регулярному выражению в начале строки
$	Найти соответствие следующему регулярному выражению в конце строки
[ ]	Найти любой из перечисленных символов. Диапазон символов можно задавать с помощью символа - (дефис). Диакритический символ ^, будучи первым символом среди перечисленных, меняет смысл выражения на противоположный – выполняется поиск символа, не входящего в указанный список. Дефис или закрывающая скобка (]) в качестве первого элемента списка интерпретируются как обычные символы. Все прочие специальные символы также считаются элементами списка
[^]	Найти любой символ, не относящийся к перечисленным
\{*n*, *m*\}	Найти определенное количество вхождений непосредственно предшествующего символа. Предшествующий символ может быть регулярным выражением. По выражению \{*n*\} выполняется поиск ровно *n* вхождений символа, по выражению \{*n*, \} – поиск по крайней мере *n* вхождений символа, выражение \{*n*, *m*\} задает диапазон для количества вхождений от *n* до *m*
{*n*, *m*}	Аналогично \{*n*, *m*\}. Метасимвол доступен по умолчанию в **grep**, а также в **gawk** при использовании ключа **–Wre-interval**

Символ	Значение
\	Интерпретировать буквально непосредственно следующий символ, не используя его специальное значение
\(\)	Сохранить найденное соответствие шаблону между символами \(и \) в специальной переменной. В одной строке можно сохранять до девяти соответствий. В шаблонах подстановки эти записи можно воспроизводить специальными escape-последовательностями от \1 до \9
n	Вставить текст, найденный *n*-ным вхождением \(\)
( )	В **egrep** и **gawk** сохранять найденный текст, заключенный в символы \(и \) в специальной переменной. Для каждой строки шаблонов может быть задано до девяти таких соответствий. Они могут быть воспроизведены в шаблоне подстановки при помощи escape-последовательностей от \1 до \9
\< \>	Найти соответствие в начале (\<) или в конце слова (\>)
+	Найти одно или более вхождений предшествующего регулярного выражения
?	Найти одно либо ни одного вхождения предшествующего регулярного выражения
|	Найти соответствие предшествующему или следующему регулярному выражению
( )	Группировка регулярных выражений

Многие инструменты в Linux поддерживают и списки символов стандарта POSIX, что позволяет выполнять поиск не-ASCII символов из языков, отличных от английского. Такие списки задаются диапазонами символов, заключаемыми в квадратные скобки **[ ]**. Так, **[[:lower:]]** является эквивалентом **[a-z]**.

В следующей таблице перечислены списки символов POSIX:

Запись	Содержание
[:alnum:]	Буквенно-цифровые символы
[:alpha:]	Буквы любого регистра
[:blank:]	Отображаемые разделители: пробелы и символы табуляции, но не управляющие символы
[:cntrl:]	Управляющие символы, например от ^A до ^Z
[:digit:]	Десятичные цифры
[:graph:]	Отображаемые символы, кроме пробелов и символов табуляции
[:lower:]	Буквы нижнего регистра
[:print:]	Отображаемые символы, включая пробелы и символы табуляции, но не управляющие символы
[:punct:]	Символы пунктуации, подмножество отображаемых символов

Запись	Содержание
[:space:]	Пробелы, символы табуляции и некоторые управляющие символов
[:upper:]	Буквы верхнего регистра
[:xdigit:]	Шестнадцатеричные цифры

Следующие символы имеют специальное значение в шаблонах подстановки:

Символ	Значение
`\`	Экранировать следующий символ
`\`*n*	Вставить текст, сохраненный *n*-ным шаблоном \(и \). Здесь *n* – число от 1 до 9, порядок нумерации найденных соответствий – слева направо
`&`	Использовать шаблон поиска в качестве части шаблона замены
`~`	Использовать предыдущий шаблон замены в текущем
`\e`	Отменить действие шаблона замены `\l` или `\u`
`\E`	Отменить действие шаблона замены `\L` или `\U`
`\l`	Сменить регистр первого символа шаблона замены на нижний
`\L`	Сменить на нижний регистр следующих символов шаблона замены
`\u`	Сменить регистр первого символа шаблона замены на верхний
`\U`	Сменить на верхний регистр следующих символов шаблона замены

Примеры поиска

При работе с **grep** или **egrep** регулярные выражения заключаются в кавычки. (Если шаблон содержит символ $, необходимо использовать одинарные кавычки, например **'$200'**, либо экранировать символ $: **"\$200"**.) При работе с **ed**, **ex**, **sed** и **awk** регулярные выражения обычно заключаются в пару символов / (хотя работает любой ограничитель). Приведем несколько примеров шаблонов:

Шаблон	Результат поиска
bag	Строка *bag*
^bag	Слово «bag» в начале строки
bag$	Слово «bag» в конце строки
^bag$	Слово «bag» как единственное слово в строке
[Bb]ag	Слово «bag» или «Bag»
b[aeiou]g	Вторая буква является гласной
b[^aeiou]g	Вторая буква является согласной (либо буквой верхнего регистра, либо символом)

Шаблон	Результат поиска
b.g	Вторая буква является любым символом, кроме символа новой строки
^. . .$	Любая строка, содержащая ровно три буквы
^\.	Строка, которая начинается с точки
^\.[a-z][a-z]	То же, за точкой следуют две строчные буквы (так выглядят, например, запросы troff)
^\.[a-z]\{2\}	То же, что и в предыдущем случае, но только для **grep** или **sed**
^[^.]	Любая строка, которая не начинается с точки
bugs*	Слова «bug», «bugs», «bugss» и т. д.
"word"	Слово word в кавычках
"*word"*	Слово word в кавычках или без
[A-Z][A-Z]*	Одна или больше прописных букв
[A-Z]+	То же, но только для **egrep** или **awk**
[A-Z].*	Прописная буква, за которой следует произвольное количество (включая нулевое) любых символов
[A-Z]*	Произвольное количество (включая нулевое) прописных букв
[a-zA-Z]	Любая буква
[0-9A-Za-z]+	Любая буквенно-цифровая последовательность

Шаблон egrep или awk	Результат поиска
[567]	Одно из чисел 5, 6 или 7
five|six|seven	Одно из слов *five*, *six* или *seven*
80[23]?86	*8086*, *80286* или *80386*
Compan(y|ies)	Слово *company* или *companies*

Шаблон vi	Результат поиска
\<the	Слова вроде *theater* или *the*
the\>	Слова вроде *breathe* или *the*
\<the\>	Слово *the*

Шаблон sed или grep	Результат поиска
0\{5,\}	Пять или более нулей подряд
[0–9]\{3\}–[0–9]\{2\}–[0–9]\{4\}	Номер социального страхования (*nnn-nn-nnnn*)

Примеры поиска и замены

В следующих примерах используются метасимволы, доступные в **sed** или **vi**. Обратите внимание, что команды **vi** начинаются с двоеточия. Пробел обозначен символом □. Табуляция обозначается как *tab*.

Команда	Результат
s/.*/(&)/	Воспроизвести строку с добавлением скобок
s/.*/mv & &.old/	Преобразовать список слов (по одному слову в строке) в набор команд **mv**
/^$/d	Удалить пустые строки
:g/^$/d	То же, но в редакторе **vi**
/^[□*tab*]*$/d	Удалить пустые строки и строки, содержащие пробелы или символы табуляции
:g/^[□*tab*]*$/d	То же, но в редакторе **vi**
s/□□*/□/g	Сократить непрерывную последовательность пробелов (более одного подряд) до одного пробела
:%s/□□*/□/g	То же, но в редакторе **vi**
:s/[0–9]/Item &:/	Преобразовать число в item-метку (в текущей строке)
:s	Повторить подстановку при первом соответствии
:&	То же
:sg	То же, но для всех вхождений в строке
:&g	То же
:%&g	Глобальная замена вхождений
:.,$s/Fortran/\U&/g	Преобразовать регистр символов слова в верхний во всех вхождениях от текущей строки и до конца файла
:%s/.*/\L&/	Преобразовать регистр всех символов файла в нижний
:s/\<./\u&/g	Преобразовать в прописную все первые буквы слов текущей строки (полезно для оформления названий)
:%s/yes/No/g	Произвести глобальную замену слова **yes** на **No**
:%s/Yes/~/g	Заменить все другие слова словом **No** (используется предыдущая подстановка)

В заключение приведем некоторые примеры для перестановки слов с помощью **sed**. Простая перестановка двух слов может выглядеть так:

```
s/die or do/do or die/
```
Перестановка слов

Настоящим мастерством является использование буферов для замены. Например, так:

```
s/\([Dd]ie\) or \([Dd]o\)/\2 or \1/
```
Перестановка с помощью буферов

Глава 10

Редактор Emacs

В этой главе рассмотрены следующие темы:

- Введение
- Типичные проблемы
- Конспект команд Emacs по группам
- Конспект команд Emacs по клавишам
- Конспект команд Emacs по именам

Введение

Несмотря на то что Emacs не является частью Linux, этот текстовый редактор присутствует во многих Unix-системах в качестве распространенной альтернативы редактору **vi**. Существует несколько вариантов Emacs. Здесь рассматривается GNU Emacs, предоставляемый фондом FSF (Free Software Foundation) (Кембридж, Массачусетс). За более подробной информацией обращайтесь к выпущенной издательством O'Reilly книге «Learning GNU Emacs» (Изучаем GNU Emacs) Дебры Кэмерон (Debra Cameron), Билла Розенблатта (Bill Rosenblatt) и Эрика Рэймонда (Eric Raymond).

Для того чтобы начать работу с Emacs, наберите:

emacs [*file*]

Типичные проблемы

Очень часто пользователи сталкиваются с тем, что на их терминале клавиша <Del> или <Backspace> не удаляет символ перед курсором, как это должно было бы происходить. Вместо этого появляется строка подсказки. Данная

проблема вызвана несовместимостью терминала. Достаточно надежным способом исправления этой ситуации является создание файла *.emacs* в домашнем каталоге пользователя (или редактирование уже существующего) и дополнение его следующими строками:

```
(keyboard translate ?\C-h ?\C-?)
(keyboard translate ?\C-\\ ?\C-h)
```

Теперь удаление клавишами <Del> или <Backspace> должно работать, а вызвать справку можно, нажав клавиши <Ctrl>+<\> (сочетание выбрано произвольно).

Еще одна проблема, которая может возникнуть при работе с удаленного терминала, – комбинация <Ctrl>+<s> (Ctrl-s) может вызвать зависание терминала. Причиной этого является старый протокол установления связи между терминалом и системой. Терминал перезапускается нажатием клавиш Ctrl-q, но это не устранит невозможность ввода команд, содержащих комбинацию Ctrl-s. Единственным решением (если не рассматривать применение более современного протокола соединения) может быть либо создание новой комбинации клавиш для замены Ctrl-s, либо ввод команды посредством последовательности <Meta>+<x> *имя_команды*.

Примечания к таблицам

В командах Emacs активно используются клавиши <Ctrl> и <Meta>. На большинстве современных терминалов существует клавиша <Alt>, заменяющая клавишу <Meta>. В этой главе запись <Ctrl>+ (или C-) означает, что <Ctrl> нажимается одновременно с последующим символом, тогда как запись <Meta>+ (или M-) означает аналогичное нажатие клавиши <Meta> или <Alt>. Вместо <Meta> можно использовать клавишу <Esc>: нажать ее, отпустить, затем ввести символ. Этот вариант выручает, если текущее окно по какой-либо причине не в состоянии перехватить нажатие клавиши <Alt> (такое иногда случается).

В таблицах с командами первая колонка всегда описывает комбинацию клавиш, а последняя – происходящее действие. Если присутствует средняя колонка, в ней находятся имена команд. Команды можно выполнять, нажав клавиши <Meta>+<x>, а затем введя имя команды. Этим вариантом приходится пользоваться, если комбинация клавиш для команды отсутствует (обозначена как «нет»). Если вы точно не помните имя, нажмите пробел или клавишу <Enter> в процессе набора команды, и Emacs перечислит возможные дополнения команды до конечного вида.

Emacs является полнофункциональным редактором и по этой причине содержит сотни команд. Некоторые команды нам пришлось опустить, чтобы сохранить дух краткого справочника. Нажатие комбинации <Ctrl>+<h> (C-h) позволяет получить справку о наборе команд; нажав <Meta>+<x> (M-x), а затем <пробел> или <Tab>, можно вывести имена команд. Также можно

ввести <Ctrl>+<h>, а затем <b> для получения перечня комбинаций клавиш[1] и связанных с ними команд.

Режимы

Одной из особенностей, которые делают Emacs популярным, являются его режимы редактирования. Режим представляет собой окружение на основе редактора Emacs, разработанное специально для редактирования конкретного вида файлов. Режим может включать такие возможности, как назначение особых сочетаний клавиш для контекстных действий, автоматическое форматирование текста в соответствии с соглашениями, принятыми для данного типа документов, и т. д. Существуют режимы для различных языков программирования (например, C и Perl), для работы с текстами (например SGML или обычными текстовыми файлами) и многие другие. Одним из наиболее полезных режимов является Dired (Directory Editor, редактор каталогов), предоставляющий команды для работы с каталогами. Более подробно режимы описаны в книге «Learning GNU Emacs», упомянутой в начале этой главы, а также в системе документации Emacs Info (<Ctrl>+<h> <i>).

Самые необходимые команды

Приведем список наиболее важных команд для начинающих знакомство с Emacs:

Комбинация	Действие
C-h	Система встроенной справки
C-x C-s	Сохранение файла
C-x C-c	Завершение работы с Emacs
C-x u	Отмена последнего редактирования (команда может повторяться)
C-g	Прекратить набор текущей команды
C-p	К предыдущей строке
C-n	К следующей строке
C-f	К следующему символу
C-b	К предыдущему символу
C-v	К следующей экранной странице текста
M-v	К предыдущей экранной странице текста
C-s	Поиск строки в прямом направлении

[1] Создание собственных сочетаний клавиш описано в книге издательства O'Reilly «Learning Gnu Emacs» (Изучаем GNU Emacs).

Комбинация	Действие
C-r	Поиск строки в обратном направлении
C-d	Удаление текущего символа
Del	Удаление символа перед курсором
Backspace	Удаление символа перед курсором

Конспект команд по группам

В таблицах перечислены клавиатурные комбинации, имена команд и выполняемые действия. Напоминаем, что «C-» – это клавиша <Ctrl>, а «M-» – <Meta>.

Команды работы с файлами

Комбинация	Команда	Действие
C-x C-f	find-file	Поиск и чтение файла
C-x C-v	find-alternate-file	Чтение другого файла; файл заменяет прочитанный по C-x C-f
C-x i	insert-file	Вставить файл в текущей позиции курсора
C-x C-s	save-buffer	Сохранить файл
C-x C-w	write-file	Записать содержимое буфера в файл
C-x C-c	save-buffers-kill-emacs	Завершить работу с Emacs
C-z	suspend-emacs	Приостановить выполнение Emacs (возврат по команде **exit** или **fg**)

Команды навигации

Для того чтобы облегчить запоминание, в столбце **Действие** некоторые слова выделены курсивом.

Комбинация	Команда	Действие
C-f	forward-char	Переход *к следующему* символу (вправо)
C-b	backward-char	Переход *к предыдущему* символу (влево)
C-p	previous-line	Переход *к предыдущей* строке (вверх)
C-n	next-line	Переход *к следующей* строке (вниз)
M-f	forward-word	Переход *к следующему* слову (вперед)
M-b	backward-word	Переход *к предыдущему* слову (назад)

Комбинация	Команда	Действие
C-a	beginning-of-line	Переход к *началу* строки
C-e	end-of-line	Переход *к концу* строки
M-a	backward-sentence	Переход к предыдущему предложению
M-e	forward-sentence	Переход к следующему предложению
M-{	backward-paragraph	Переход к предыдущему абзацу
M-}	forward-paragraph	Переход к следующему абзацу
C-v	scroll-up	Переход к следующей экранной странице
M-v	scroll-down	Переход к предыдущей экранной странице
C-x [	backward-page	Переход к предыдущей странице текста
C-x]	forward-page	Переход к следующей странице текста
M->	end-of-buffer	Переход к концу текста
M-<	beginning-of-buffer	Переход к началу текста
(нет)	goto-line	Перейти к строке *n* файла
(нет)	goto-char	Перейти к символу *n* файла
C-l	recenter	Обновить окно с текстом так, чтобы текущая строка с курсором оказалась в центре окна
M-*n*	digit-argument	Повторить следующую команду *n* раз
C-u *n*	universal-argument	Повторить следующую команду *n* раз

Команды удаления

Комбинация	Команда	Действие
Del	backward-delete-char	Удаление символа перед курсором
C-d	delete-char	Удаление символа под курсором
M-Del	backward-kill-word	Удаление предыдущего слова
M-d	kill-word	Удаление слова под курсором
C-k	kill-line	Удаление от курсора и до конца строки
M-k	kill-sentence	Удаление предложения, на котором находится курсор
C-x Del	backward-kill-sentence	Удаление предыдущего предложения
C-y	yank	Возврат удаленного текста
C-w	kill-region	Удаление выделенной области (см. следующую таблицу)

Комбинация	Команда	Действие
(нет)	backward-kill-paragraph	Удаление предыдущего абзаца
(нет)	kill-paragraph	Удаление текста от курсора и до конца абзаца

Абзацы и области текста

Комбинация	Команда	Действие
C-@	set-mark-command	Отметить начало (или конец) области
C-пробел	set-mark-command	Отметить начало (или конец) области
C-x C-p	mark-page	Выделить страницу
C-x C-x	exchange-point-and-mark	Обменять местами курсор и отметку
C-x h	mark-whole-buffer	Выделить буфер
M-q	fill-paragraph	Переформатировать абзац
(нет)	fill-region	Переформатировать отдельные абзацы выделенной области
M-h	mark-paragraph	Выделить абзац
M-{	backward-paragraph	Перемещение к предыдущему абзацу
M-}	forward-paragraph	Перемещение к следующему абзацу
(нет)	backward-kill-paragraph	Удаление предыдущего абзаца
(нет)	kill-paragraph	Удаление текста от курсора и до конца абзаца

Прерывание и откат команд

Комбинация	Команда	Действие
C-g	keyboard-quit	Прекратить набор команды
C-x u	advertised-undo	Многоуровневая отмена последнего редактирования
(нет)	revert-buffer	Восстановить редактируемый текст из сохраненного или автоматически сохраненного файла

Команды перестановки

Комбинация	Команда	Действие
C-t	transpose-chars	Перестановка (обмен местами) пары букв
M-t	transpose-words	Перестановка пары слов
C-x C-t	transpose-lines	Перестановка пары строк

Комбинация	Команда	Действие
(нет)	transpose-sentences	Перестановка пары предложений
(нет)	transpose-paragraphs	Перестановка пары абзацев

Команды преобразования регистров

Комбинация	Команда	Действие
M-c	capitalize-word	Сделать прописной первую букву слова
M-u	upcase-word	Сделать прописными все буквы слова
M-l	downcase-word	Сделать строчными все буквы слова
M- – M-c	negative-argument; capitalize-word	Сделать прописной первую букву предыдущего слова
M- – M-u	negative-argument; upcase-word	Сделать прописными все буквы предыдущего слова
M- – M-l	negative-argument; downcase-word	Сделать строчными все буквы предыдущего слова
(нет)	capitalize-region	Сделать прописными первые буквы слов выделенной области
C-x C-u	upcase-region	Сделать прописными все буквы в выделенной области
C-x C-l	downcase-region	Сделать строчными все буквы в выделенной области

Команды пошагового поиска

Комбинация	Команда	Действие
C-s	isearch-forward	Начать или повторить прямой пошаговый поиск
C-r	isearch-backward	Начать или повторить обратный пошаговый поиск
Enter	(нет)	Завершить успешный поиск
C-g	keyboard-quit	Отменить пошаговый поиск; вернуться к начальному положению в тексте
Del	(нет)	Удалить неверный символ в строке поиска
M-C-r	isearch-backward-regexp	Обратный пошаговый поиск по регулярному выражению
M-C-s	isearch-forward-regexp	Прямой пошаговый поиск по регулярному выражению

Команды сокращений слов

Комбинация	Команда	Действие
(нет)	abbrev-mode	Переключение режима ввода сокращенных слов
C-x a –	inverse-add-global-abbrev	Добавить предыдущее слово в качестве независящего от режима глобального сокращения
C-x a i l	inverse-add-mode-abbrev	Добавить предыдущее слово в качестве зависящего от режима сокращения
(нет)	unexpand-abbrev	Отменить последнее сокращение слова
(нет)	write-abbrev-file	Записать файл сокращений слов
(нет)	edit-abbrevs	Редактировать сокращения слов
(нет)	list-abbrevs	Перечислить сокращения
(нет)	kill-all-abbrevs	Удалить все сокращения этого сеанса работы

Команды работы с буфером

Комбинация	Команда	Действие
C-x b	switch-to-buffer	Переход к указанному буферу
C-x C-b	list-buffers	Перечисление списка буферов
C-x k	kill-buffer	Удалить указанный буфер
(нет)	kill-some-buffers	Подтверждение удаления каждого из буферов
(нет)	rename-buffer	Изменить имя буфера на указанное
C-x s	save-some-buffers	Подтверждение сохранения каждого из измененных буферов

Команды работы с окнами

Комбинация	Команда	Действие
C-x 2	split-window-horizontally	Разделить текущее окно по горизонтали на пару окон
C-x 3	split-window-vertically	Разделить текущее окно по вертикали на пару окон
C-x >	scroll-right	Прокрутка окна вправо
C-x <	scroll-left	Прокрутка окна влево
C-x o	other-window	Переход в соседнее окно

Комбинация	Команда	Действие
C-x 0	delete-window	Удалить текущее окно
C-x 1	delete-other-windows	Удалить все окна кроме текущего
(нет)	delete-windows-on	Удалить все окна определенного буфера
C-x ^	enlarge-window	Увеличить высоту окна
(нет)	shrink-window	Уменьшить высоту окна
C-x }	enlarge-window-horizontally	Увеличить ширину окна
C-x {	shrink-window-horizontally	Уменьшить ширину окна
M-C-v	scroll-other-window	Прокрутка соседнего окна
C-x 4 f	find-file-other-window	Поиск файла в соседнем окне
C-x 4 b	switch-to-buffer-other-window	Выбрать буфер в соседнем окне
C-x 5 f	find-file-other-frame	Поиск файла в новом фрейме
C-x 5 b	switch-to-buffer-other-frame	Выбрать буфер в другом фрейме
(нет)	compare-windows	Сравнить два буфера, отобразить различия

Специальные символы режима командного интерпретатора

Комбинация	Команда	Действие
C-c C-c	interrupt-shell-subjob	Принудительное завершение текущего задания
C-c C-d	shell-send-eof	Символ конца файла
C-c C-u	kill-shell-input	Удалить текущую строку
C-c C-w	backward-kill-word	Удалить предыдущее слово
C-c C-z	stop-shell-subjob	Приостановить текущее задание

Команды отступа

Комбинация	Команда	Действие
C-x .	set-fill-prefix	Предварить каждую строку абзаца символами от начала строки до положения курсора в строке; это можно отменить, выполнив команду с курсором в первом столбце
(нет)	indented-text-mode	Основной режим: каждый табулятор увеличивает отступ для следующей строки
(нет)	text-mode	Прекратить ввод текста в режиме отступов, вернуться в обычный текстовый режим

Комбинация	Команда	Действие
M-C-\	indent-region	Отформатировать выделенную область, выровняв отступы по первой строке
M-m	back-to-indentation	Переход (курсора) к первому символу в строке
M-^	delete-indentation	Объединяет текущую строку с предыдущей
M-C-o	split-line	Разорвать строку на курсоре; отступ для продолжения строки определяется положением курсора
(нет)	fill-individual-paragraphs	Переформатировать абзацы с отступами, сохранив отступы

Команды центрирования

Комбинация	Команда	Действие
(нет)	center-line	Центрировать текущую строку
(нет)	center-paragraph	Центрировать текущий абзац
(нет)	center-region	Центрировать выделенную область

Макрокоманды

Комбинация	Команда	Действие
C-x (	start-kbd-macro	Начать определение макрокоманды
C-x)	end-kbd-macro	Закончить определение макрокоманды
C-x e	call-last-kbd-macro	Выполнить последнюю из определенных макрокоманд
M-*n* C-x e	digit-argument и call-last-kbd-macro	Выполнить последнюю из определенных макрокоманд *n* раз
C-u C-x (	start-kbd-macro	Выполнить последнюю из определенных макрокоманд и добавить в нее нажатия клавиш
(нет)	name-last-kbd-macro	Дать имя последней определенной макрокоманде (перед сохранением)
(нет)	insert-last-keyboard-macro	Добавить определенную макрокоманду в файл
(нет)	load-file	Прочитать ранее созданный файл макрокоманд
(нет)	*macroname*	Выполнить ранее сохраненную клавиатурную макрокоманду

Комбинация	Команда	Действие
C-x q	kbd-macro-query	Вставить запрос в определение макрокоманды
C-u C-x q	(нет)	Вставить рекурсивное редактирование в определение макрокоманды
M-C-c	exit-recursive-edit	Прекратить рекурсивное редактирование

Команды для получения помощи

Комбинация	Команда	Действие
C-h a	command-apropos	Какие команды работают с этим понятием?
(нет)	apropos	Какие команды, функции и переменные работают с этим понятием?
C-h c	describe-key-briefly	Какие команды выполняются по этой комбинации клавиш?
C-h b	describe-bindings	Какие комбинации клавиш действительны в этом буфере ?
C-h k	describe-key	Какие команды выполняются по этой комбинации клавиш и что она делает?
C-h l	view-lossage	Перечислить последние 100 введенных символов
C-h w	where-is	Какая комбинация клавиш соответствует этой команде?
C-h f	describe-function	Что делает эта функция?
C-h v	describe-variable	Каков смысл этой переменной и каково ее значение?
C-h m	describe-mode	В каком режиме находится текущий буфер?
C-h s	describe-syntax	Какова синтаксическая таблица этого буфера?

Справочные команды

Комбинация	Команда	Действие
C-h t	help-with-tutorial	Вызвать руководство по Emacs
C-h i	info	Чтение информационных файлов
C-h n	view-emacs-news	Новости обновлений Emacs
C-h C-c	describe-copying	Лицензия GPL для Emacs
C-h C-d	describe-distribution	Информация о том, как заказать Emacs в FSF
C-h C-w	describe-no-warranty	Условия применения Emacs (отказ от гарантий)

Конспект команд Emacs по клавишам

Ниже представлены команды Emacs в двух алфавитных списках. В таблицах перечислены клавиатурные комбинации, имена команд и выполняемые действия. Напоминаем, что «C-» – это клавиша <Ctrl>, а «M-» – <Meta>.

Комбинации с клавишей <Ctrl>

Комбинация	Команда	Действие
C-@	set-mark-command	Отметить начало (или конец) области
C-пробел	set-mark-command	Отметить начало (или конец) области
C-]	abort-recursive-edit	Прекратить рекурсивное редактирование и замену по запросу
C-a	beginning-of-line	Переход к началу строки
C-b	backward-char	Переход к *предыдущему* символу (влево)
C-c C-c	interrupt-shell-subjob	Принудительное завершение текущего задания
C-c C-d	shell-send-eof	Символ конца файла
C-c C-u	kill-shell-input	Удалить текущую строку
C-c C-w	backward-kill-word	Удалить предыдущее слово
C-c C-z	stop-shell-subjob	Приостановить текущее задание
C-d	delete-char	Удалить символ под курсором
C-e	end-of-line	Переход к *концу* строки
C-f	forward-char	Переход к *следующему* символу (вправо)
C-g	keyboard-quit	Прекратить набор команды
C-h	help-command	Вызвать систему встроенной справки
C-h a	command-apropos	Какие команды работают с этим понятием?
C-h b	describe-bindings	Какие комбинации клавиш действительны в этом буфере ?
C-h c	describe-key-briefly	Какие команды выполняются по этой комбинации клавиш?
C-h C-c	describe-copying	Лицензия GPL для Emacs
C-h C-d	describe-distribution	Информация о том, как заказать Emacs в FSF
C-h C-w	describe-no-warranty	Условия использования Emacs (гарантии и их отсутствие)
C-h f	describe-function	Что делает эта функция?
C-h i	info	Чтение информационных файлов

Комбинация	Команда	Действие
C-h k	describe-key	Какие команды выполняются по этой комбинации клавиш и что она делает?
C-h l	view-lossage	Перечислить последние 100 введенных символов
C-h m	describe-mode	В каком режиме находится текущий буфер?
C-h n	view-emacs-news	Новости обновлений Emacs
C-h s	describe-syntax	Какова синтаксическая таблица этого буфера?
C-h t	help-with-tutorial	Открывает учебник Emacs
C-h v	describe-variable	Каков смысл этой переменной и каково ее значение?
C-h w	where-is	Какая комбинация клавиш соответствует этой команде?
C-k	kill-line	Удаление от курсора и до конца строки
C-l	recenter	Обновить окно с текстом так, чтобы текущая строка с курсором оказалась в центре окна
C-n	next-line	Переход к следующей строке (вниз)
C-p	previous-line	Переход к предыдущей строке (вверх)
C-q	quoted-insert	Вставить следующий набранный символ. Полезно для набора управляющих символов
C-r	isearch-backward	Начать или повторить обычный обратный поиск
C-r	(нет)	Рекурсивное редактирование (во время замены по запросу)
C-s	isearch-forward	Начать или повторить прямой обычный поиск
C-t	transpose-chars	Перестановка (обмен местами) пары букв
C-u *n*	universal-argument	Повторить следующую команду *n* раз
C-u C-x (	start-kbd-macro	Выполнить последнюю из определенных макрокоманд и добавить в нее нажатия клавиш
C-u C-x q	(нет)	Вставить рекурсивное редактирование в определение макрокоманды
C-v	scroll-up	Переход к следующей экранной странице
C-w	kill-region	Удаление выделенной области
C-x (	start-kbd-macro	Начать определение макрокоманды
C-x)	end-kbd-macro	Закончить определение макрокоманды
C-x [	backward-page	Переход к предыдущей странице текста

Комбинация	Команда	Действие
C-x]	forward-page	Переход к следующей странице текста
C-x ^	enlarge-window	Увеличить высоту окна
C-x {	shrink-window-horizontally	Уменьшить ширину окна
C-x }	enlarge-window-horizontally	Увеличить ширину окна
C-x <	scroll-left	Прокрутка окна влево
C-x >	scroll-right	Прокрутка окна вправо
C-x .	set-fill-prefix	Предварить каждую строку абзаца символами от начала строки до положения курсора в строке; это можно отменить, выполнив команду с курсором в первом столбце
C-x 0	delete-window	Удалить текущее окно
C-x 1	delete-other-windows	Удалить все окна кроме текущего
C-x 2	split-window-horizontally	Разделить текущее окно по горизонтали
C-x 3	split-window-vertically	Разделить текущее окно по вертикали
C-x 4 b	switch-to-buffer-other-window	Выбрать буфер в соседнем окне
C-x 4 f	find-file-other-window	Поиск файла в соседнем окне
C-x 5 b	switch-to-buffer-other-frame	Поиск файла в новом фрейме
C-x 5 f	find-file-other-frame	Выбрать буфер в другом фрейме
C-x a –	inverse-add-global-abbrev	Добавить предыдущее слово в качестве независящего от режима глобального сокращения
C-x a i l	inverse-add-mode-abbrev	Добавить предыдущее слово в качестве зависящего от режима сокращения
C-x b	switch-to-buffer	Переход к указанному буферу
C-x C-b	list-buffers	Перечисление списка буферов
C-x C-c	save-buffers-kill-emacs	Завершить работу с Emacs
C-x C-f	find-file	Поиск и чтение файла
C-x C-l	downcase-region	Сделать строчными все буквы в выделенной области
C-x C-p	mark-page	Выделить всю страницу
C-x C-q	(нет)	Триггер режима «только для чтения» текущего буфера

Комбинация	Команда	Действие
C-x C-s	save-buffer	Сохранить файл (при блокировании терминала нажать клавиши <Ctrl>+<q> для перезапуска)
C-x C-t	transpose-lines	Перестановка пары строк
C-x C-u	upcase-region	Сделать прописными все буквы в выделенной области
C-x C-v	find-alternate-file	Чтение другого файла; файл заменяет прочитанный по C-x C-f
C-x C-w	write-file	Записать содержимое буфера в файл
C-x C-x	exchange-point-and-mark	Обменять местами курсор и отметку
C-x Del	backward-kill-sentence	Удаление предыдущего предложения
C-x e	call-last-kbd-macro	Выполнить последнюю из определенных макрокоманд
C-x h	mark-whole-buffer	Выделить буфер
C-x i	insert-file	Вставить файл в текущей позиции курсора
C-x k	kill-buffer	Удалить указанный буфер
C-x o	other-window	Переход в соседнее окно
C-x q	kbd-macro-query	Вставить запрос в определение макрокоманды
C-x s	save-some-buffers	Подтверждение сохранения каждого из измененных буферов
C-x u	advertised-undo	Многоуровневая отмена последнего редактирования
C-y	yank	Возврат удаленного текста
C-z	suspend-emacs	Приостановить выполнение Emacs (возврат по команде **exit** или **fg**)

Комбинации с клавишей <Meta>

Комбинация	Команда	Действие
M-– M-c	negative-argument; capitalize-word	Сделать прописной первую букву предыдущего слова
M-– M-l>	negative-argument; downcase-word	Сделать строчными все буквы предыдущего слова
M-– M-u>	negative-argument; upcase-word	Сделать прописными все буквы предыдущего слова
M-$	spell-word	Проверить орфографию следующего слова

Комбинация	Команда	Действие
M-<	beginning-of-buffer	Переход к началу текста
M->	end-of-buffer	Переход к концу текста
M-{	backward-paragraph	Переход к предыдущему абзацу
M-}	forward-paragraph	Переход к следующему абзацу
M-^	delete-indentation	Объединить текущую строку с предыдущей
M-*n*	digit-argument	Повторить следующую команду *n* раз
M-*n* C-x e	digit-argument; call-last-kbd-macro	Выполнить последнюю из определенных макрокоманд *n* раз
M-a	backward-sentence	Переход к предыдущему предложению
M-b	backward-word	Переход к предыдущему слову (назад)
M-c	capitalize-word	Сделать прописной первую букву слова
M-C-\	indent-region	Отформатировать выделенную область, выровняв отступы по первой строке
M-C-c	exit-recursive-edit	Прекратить рекурсивное редактирование
M-C-o	split-line	Разорвать строку на курсоре; отступ для хвоста строки определяется положением курсора
M-C-r	isearch-backward-regexp	Обратный пошаговый поиск по регулярному выражению
M-C-s	isearch-forward-regexp	Прямой пошаговый поиск по регулярному выражению
M-C-v	scroll-other-window	Прокрутка соседнего окна
M-d	kill-word	Удаление слова над курсором
M-Del	backward-kill-word	Удаление предыдущего слова
M-e	forward-sentence	Переход к следующему предложению
M-f	forward-word	Переход к следующему слову (вперед)
(нет)	fill-region	Переформатировать отдельные абзацы выделенной области
M-h	mark-paragraph	Выделить абзац
M-k	kill-sentence	Удаление предложения, на котором находится курсор
M-l	downcase-word	Сделать строчными все буквы слова
M-m	back-to-indentation	Переход (курсора) к первому символу в строке
M-q	fill-paragraph	Переформатировать абзац

Комбинация	Команда	Действие
M-s	center-line	Центрировать текущую строку
M-t	transpose-words	Перестановка пары слов
M-u	upcase-word	Сделать прописными все буквы слова
M-v	scroll-down	Переход к предыдущей экранной странице
M-x	(нет)	Выполнить команду, набрав ее имя

Конспект команд Emacs по именам

Ниже приведены команды Emacs в алфавитном порядке. Чтобы набрать имя команды, используйте комбинацию <Meta>+<x>. В таблице перечислены клавиатурные комбинации, имена и описания действий команд. Как и ранее, «C-» – это клавиша <Ctrl>, а «M-» – <Meta>.

Комбинация	Команда	Действие
(нет)	*macroname*	Выполнить ранее сохраненную клавиатурную макрокоманду
(нет)	abbrev-mode	Переключение режима ввода сокращенных слов
C-]	abort-recursive-edit	Завершить рекурсивное редактирование
C-x u	advertised-undo	Многоуровневая отмена последнего редактирования
(нет)	apropos	Какие команды, функции и переменные работают с этим понятием?
M-m	back-to-indentation	Переход (курсора) к первому символу в строке
C-b	backward-char	Переход к предыдущему символу (влево)
Del	backward-delete-char	Удаление символа перед курсором
(нет)	backward-kill-paragraph	Удаление предыдущего абзаца
C-x Del	backward-kill-sentence	Удаление предыдущего предложения
C-c C-w	backward-kill-word	Стирание предыдущего слова
M-Del	backward-kill-word	Удаление предыдущего слова
C-x [	backward-page	Переход к предыдущей странице
M-{	backward-paragraph	Переход к предыдущему абзацу
M-a	backward-sentence	Переход к предыдущему предложению
M-b	backward-word	Переход к предыдущему слову (назад)
M-<	beginning-of-buffer	Переход к началу файла

Комбинация	Команда	Действие
C-a	beginning-of-line	Переход к началу строки
C-x e	call-last-kbd-macro	Выполнить последнюю из определенных макрокоманд
(нет)	capitalize-region	Сделать прописными первые буквы слов выделенной области
M-c	capitalize-word	Сделать прописной первую букву слова
(нет)	center-line	Центрировать текущую строку
(нет)	center-paragraph	Центрировать текущий абзац
(нет)	center-region	Центрировать выделенную область
C-h a	command-apropos	Какие команды работают с этим понятием?
(нет)	compare-windows	Сравнить два буфера, отобразить различия
C-d	delete-char	Удаление символа перед курсором
M-^	delete-indentation	Объединить текущую строку с предыдущей
C-x 1	delete-other-windows	Удалить все окна кроме текущего
C-x 0	delete-window	Удалить текущее окно
(нет)	delete-windows-on	Удалить все окна определенного буфера
C-h b	describe-bindings	Какие комбинации клавиш действительны в этом буфере ?
C-h C-c	describe-copying	Лицензия GPL для Emacs
C-h C-d	describe-distribution	Информация о том, как заказать Emacs в FSF
C-h f	describe-function	Что делает эта функция?
C-h k	describe-key	Какие команды выполняются по этой комбинации клавиш и что она делает?
C-h c	describe-key-briefly	Какие команды выполняются по этой комбинации клавиш?
C-h m	describe-mode	В каком режиме находится текущий буфер?
C-h C-w	describe-no-warranty	Условия применения Emacs (гарантии и их отсутствие)
C-h s	describe-syntax	Какова синтаксическая таблица этого буфера?
C-h v	describe-variable	Каков смысл этой переменной и каково ее значение?
M-*n*	digit-argument	Повторить следующую команду *n* раз
C-x C-l	downcase-region	Сделать строчными все буквы в выделенной области

Комбинация	Команда	Действие
M-l	downcase-word	Сделать строчными все буквы слова
(нет)	edit-abbrevs	Редактировать сокращения слов
C-x)	end-kbd-macro	Закончить определение макрокоманды
M->	end-of-buffer	Переход к концу файла
C-e	end-of-line	Переход к концу строки
C-x ^	enlarge-window	Увеличить высоту окна
C-x }	enlarge-window-horizontally	Увеличить ширину окна
C-x C-x	exchange-point-and-mark	Обменять местами курсор и отметку
M-C-c	exit-recursive-edit	Прекратить рекурсивное редактирование
(нет)	fill-individual-paragraphs	Переформатировать абзацы с отступами, сохранив отступы
M-q	fill-paragraph	Переформатировать абзац
(нет)	fill-region	Переформатировать отдельные абзацы выделенной области
C-x C-v	find-alternate-file	Чтение другого файла; файл заменяет прочитанный по C-x C-f
C-x C-f	find-file	Поиск и чтение файла
C-x 5 f	find-file-other-frame	Поиск файла в соседнем фрейме
C-x 4 f	find-file-other-window	Поиск файла в соседнем окне
C-f	forward-char	Переход к следующему символу (вправо)
C-x]	forward-page	Переход к следующей странице текста
M-}	forward-paragraph	Переход к следующему абзацу
M-e	forward-sentence	Переход к следующему предложению
M-f	forward-word	Переход к следующему слову
(нет)	goto-char	Перейти к символу *n* файла
(нет)	goto-line	Перейти к строке *n* файла
C-h	help-command	Вызвать систему встроенной справки
C-h t	help-with-tutorial	Вызвать руководство по Emacs
M-C-\	indent-region	Отформатировать выделенную область, выровняв отступы по первой строке
(нет)	indented-text-mode	Основной режим: каждый табулятор увеличивает отступ для следующей строки

Комбинация	Команда	Действие
C-h i	info	Чтение информационных файлов
C-x i	insert-file	Вставить файл в текущей позиции курсора
(нет)	insert-last-keyboard-macro	Добавить определенную макрокоманду в файл
C-c C-c	interrupt-shell-subjob	Принудительное завершение текущего задания
C-x a –	inverse-add-global-abbrev	Добавить предыдущее слово в качестве зависящего от режима сокращения
C-x a i l	inverse-add-mode-abbrev	Отменить последнее сокращение слова
C-r	isearch-backward	Начать обратный пошаговый поиск
M-C-r	isearch-backward-regexp	То же, но производить поиск по регулярному выражению
C-s	isearch-forward	Начать прямой пошаговый поиск
M-C-s	isearch-forward-regexp	То же, но производить поиск по регулярному выражению
C-x-q	kbd-macro-query	Вставить запрос в определение макрокоманды
C-g	keyboard-quit	Прекратить набор команды
(нет)	kill-all-abbrevs	Удалить все сокращения этого сеанса работы
C-x k	kill-buffer	Удалить указанный буфер
C-k	kill-line	Удалить от курсора и до конца строки
(нет)	kill-paragraph	Удалить предыдущий абзац
C-w	kill-region	Удалить выделенную область
M-k	kill-sentence	Удалить предложение, на котором находится курсор
C-c C-u	kill-shell-input	Удалить текущую строку
(нет)	kill-some-buffers	Подтверждение удаления каждого из буферов
M-d	kill-word	Удалить слово под курсором
(нет)	list-abbrevs	Перечислить сокращения
C-x C-b	list-buffers	Перечислить буферы
(нет)	load-file	Прочитать ранее созданный файл макрокоманд
C-x C-p	mark-page	Выделить всю страницу автоматически
M-h	mark-paragraph	Выделить абзац

Комбинация	Команда	Действие
C-x h	mark-whole-buffer	Выделить буфер
(нет)	name-last-kbd-macro	Дать имя последней определенной макрокоманде (перед сохранением)
M-– M-c	negative-argument; capitalize-word	Сделать прописной первую букву предыдущего слова
M-– M-l	negative-argument; downcase-word	Сделать строчными все буквы предыдущего слова
M-– M-u	negative-argument; upcase-word	Сделать заглавными все буквы предыдущего слова
C-n	next-line	Переход к следующей строке
C-x o	other-window	Переход в соседнее окно
C-p	previous-line	Переход к предыдущей строке
(нет)	query-replace-regexp	Замена по регулярному выражению с запросом подтверждений
C-q	quoted-insert	Вставить в буфер следующий набранный символ. Удобный способ вставки управляющих символов
C-l	recenter	Обновить окно с текстом так, чтобы текущая строка с курсором оказалась в центре окна
(нет)	rename-buffer	Изменить имя буфера на указанное
(нет)	replace-regexp	Безусловная замена по регулярному выражению
(нет)	re-search-backward	Простой обратный поиск по регулярному выражению
(нет)	re-search-forward	Простой прямой поиск по регулярному выражению
(нет)	revert-buffer	Восстановить редактируемый текст из сохраненного или автоматически сохраненного файла
C-x C-s	save-buffer	Сохранить файл
C-x C-c	save-buffers-kill-emacs	Завершить работу с Emacs
C-x s	save-some-buffers	Подтверждение сохранения каждого из измененных буферов
M-v	scroll-down	Переход к предыдущей экранной странице
C-x <	scroll-left	Прокрутка окна влево
M-C-v	scroll-other-window	Прокрутка соседнего окна

Комбинация	Команда	Действие
C-x >	scroll-right	Прокрутка окна вправо
C-v	scroll-up	Переход к следующей экранной странице
C-x .	set-fill-prefix	Предварить каждую строку абзаца символами от начала строки до положения курсора в строке; это можно отменить, выполнив команду с курсором в первом столбце
C-@ или C-пробел	set-mark-command	Отметить начало (или конец) области
C-c C-d	shell-send-eof	Символ конца файла
(нет)	shrink-window	Уменьшить высоту окна
C-x {	shrink-window-horizontally	Уменьшить ширину окна
(нет)	spell-buffer	Произвести проверку орфографии текущего буфера
(нет)	spell-region	Проверка орфографии выделенной области текста
(нет)	spell-string	Проверка орфографии строки, введенной в минибуфер
M-$	spell-word	Проверка орфографии следующего слова
M-C-o	split-line	Разорвать строку на курсоре; отступ для продолжения строки определяется положением курсора
C-x 3	split-window-horizontally	Разделить текущее окно по горизонтали на пару окон
C-x 2	split-window-vertically	Разделить текущее окно по вертикали на пару окон
C-x (	start-kbd-macro	Начать определение макрокоманды
C-c C-z	stop-shell-subjob	Приостановить текущее задание
C-z	suspend-emacs	Приостановить выполнение Emacs (возврат по команде **fg**)
C-x b	switch-to-buffer	Переход к указанному буферу
C-x 5 b	switch-to-buffer-other-frame	Выбрать буфер в соседнем фрейме
C-x 4 b	switch-to-buffer-other-window	Выбрать буфер в соседнем окне
(нет)	text-mode	Прекратить ввод текста в режиме отступов, вернуться в обычный текстовый режим

Комбинация	Команда	Действие
C-t	transpose-chars	Перестановка (обмен местами) пары букв
C-x C-t	transpose-lines	Перестановка пары строк
(нет)	transpose-paragraphs	Перестановка пары абзацев
(нет)	transpose-sentences	Перестановка пары предложений
M-t	transpose-words	Перестановка пары слов
(нет)	unexpand-abbrev	Отменить последнее сокращение слова
C-u *n*	universal-argument	Повторить следующую команду *n* раз
C-x C-u	upcase-region	Сделать прописными все буквы в выделенной области
M-u	upcase-word	Сделать прописными все буквы слова
C-h n	view-emacs-news	Новости обновлений Emacs
C-h l	view-lossage	Перечислить последние 100 введенных символов
C-h w	where-is	Какая комбинация клавиш соответствует этой команде?
(нет)	write-abbrev-file	Записать файл сокращений слов
C-x C-w	write-file	Записать содержимое буфера в файл
C-y	yank	Возврат удаленного текста

Глава 11

Редактор vi

Редактор **vi** – классический экранный редактор Unix-систем. Существует большое количество его улучшенных версий, таких как **nvi**, **vim**, **vile** и **elvis**. На системах Linux команда **vi** обычно является ссылкой на одну из этих программ.

В основе **vi** лежит более старый редактор **ex**. Для того чтобы получить в редакторе **vi** доступ к мощным возможностям редактирования **ex**, следует нажать «:», набрать команду **ex** и нажать <Enter>. Команды **ex** могут также помещаться в стартовый файл *~/.exrc*, который загружается редактором **vi** в начале рабочего сеанса. Поскольку команды **ex** являются важной частью **vi**, они также описаны в этой главе. В системах Linux **ex** иногда называется **hex**.

В этой главе описывается стандартный **vi**, а также расширения **nvi**. Рассмотрены следующие темы:

- Обзор операций **vi**
- Параметры командной строки **vi**
- Параметры командной строки **ex**
- Команды навигации
- Команды редактирования
- Сохранение и завершение работы
- Доступ к нескольким файлам
- Взаимодействие с интерпретатором
- Макрокоманды
- Различные команды
- Алфавитный перечень команд, действующих в командном режиме

- Синтаксис команд **ex**
- Алфавитный перечень команд **ex**
- Настройка **vi**

Более подробную информацию можно найти в книге Линды Лэмб (Linda Lamb) и Арнольда Роббинса (Arnold Robbins) «Learning the vi Editor» (Изучаем редактор vi) издательства O'Reilly.

Обзор операций vi

В этом разделе рассмотрены:

- Синтаксис командной строки
- Режимы работы **vi**
- Синтаксис команд **vi**
- Команды строки состояния

Командный режим

После открытия файла **vi** находится в командном режиме. Возможны следующие действия:

- Переход в режим вставки
- Ввод команд редактирования
- Перемещение курсора к нужной позиции в файле
- Ввод команд **ex**
- Запуск командного интерпретатора Linux
- Сохранение или завершение работы с данной версией файла

Режим вставки

В режиме вставки можно добавлять новый текст в файл. Чтобы из режима вставки вернуться в командный режим, необходимо нажать клавишу <Esc> или сочетание <Ctrl>+<[>. Режим вставки активизируется следующими командами:

`a` Добавить текст после курсора.

`A` Добавить текст в конце строки.

`c` Начать операцию замены (за которой должна следовать команда перемещения).

`C` Замена до конца строки.

`i` Вставка текста перед курсором.

`I` Вставка текста в начале строки.

`o` Начать новую строку после текущей.

`O` Начать новую строку перед текущей.

`R` Запись текста поверх существующего.

`s` Заменить символ.

`S` Заменить всю строку.

Синтаксис команд vi

Команды **vi** имеют следующий общий вид:

[*n*] *operator* [*m*] *object*

Основные операторы редактирования:

`c` Внести изменения.

`d` Удалить текст.

`y` Возврат (или копирование) текста.

Если объектом операции является текущая строка, то оператор совпадает с объектом: **cc**, **dd**, **yy**. В ином случае операторы редактирования действуют на объект, заданный командами перемещения курсора или командами поиска по шаблону. Параметры *n* и *m* – числа, определяющие количество повторений команды или количество объектов, для которых выполняются операции. Если заданы оба числа, они перемножаются.

Объект может представлять одну из следующих совокупностей символов:

word

Слово. Все символы до следующего пробела или знака препинания. Если первая буква объекта прописная (т. е. Word), то все символы до следующего пробела.

sentence

Предложение. Все символы до точки, восклицательного или вопросительного знаков, за которыми следуют два пробела.

paragraph

Абзац. Все символы до следующей пустой строки или макроса абзаца **nroff/troff** (определяемого параметром **para=**).

section

Все символы до заголовка **nroff/troff** следующего раздела (определяемого параметром **sect=**).

Примеры

`2cw` Изменить два следующих слова.

`d}` Удалить все до следующего абзаца.

`d^` Удалить все до начала текущей строки.

5yy Скопировать пять следующих строк во временный буфер (для последующей вставки).

y]] Скопировать все до начала следующего раздела во временный буфер (для последующей вставки).

Команды строки состояния

Большинство команд не отображаются на терминале во время ввода. Однако команды, перечисленные ниже, во время ввода отображаются в строке состояния в нижней части экрана:

/ Прямой поиск по шаблону.

? Обратный поиск по шаблону.

: Выполнить команду **ex**.

! Выполнить команду интерпретатора. Вводом команды является выделенный текст. После выполнения команды текст заменяется ее выводом.

Ввод команд в строке состояния должен завершаться нажатием клавиши <Enter>. Кроме того, в строке состояния отображаются сообщения об ошибках и результат работы команды **Ctrl-G** (<Ctrl>+<G>).

Параметры командной строки vi

Три самых распространенных способа начала сеанса работы с **vi**:

```
vi file
vi + n file
vi +/ pattern file
```

Файл *file* можно открыть для редактирования, для редактирования с автоматическим переходом к строке с номером *n* либо с переходом к строке, соответствующей шаблону *pattern*. Если имя файла не задано, **vi** открывает для редактирования пустой буфер. Доступны следующие параметры командной строки **vi**:

+[*num*]

Перейти к редактированию строки *num* либо последней строки файла, если номер не указан.

+/*pattern*

Перейти к редактированию первой строки, соответствующей шаблону *pattern*. Не работает при установленном в файле *.exrc* параметре **nowrapscan**.

−c *command*

Выполнить указанную команду **vi** после запуска. Допустим только один параметр **−c**. Команды **ex** могут выполняться, если перед ними указать

символ «:». Также поддерживается более старая форма использования этого параметра: +*command*.

–e Работать как в **ex** (построчное редактирование, а не полноэкранный режим).

–l LISP-режим, предназначенный для выполнения LISP-программ (поддерживается не во всех версиях редактора).

–r [*file*]

Восстановить файл и продолжить его редактирование после аварийного завершения сеанса редактирования или сбоя в системе. Если имя файла не указано, перечисляются доступные для восстановления файлы.

–t *tag*

Редактировать файл, содержащий указанный тег, и переместить курсор к месту его определения (см. также описание **ctags** в главе 3 «Команды Linux»).

–v Работа в полноэкранном режиме (по умолчанию).

–w *rows*

Установить размер окна в *rows* строк. Параметр полезен для редактирования при медленных удаленных соединениях.

–x Запрашивать ключ, который будет использоваться для шифрования и расшифровки файла посредством **crypt** (поддерживается не во всех версиях.)

–C Аналогично **–x**, но считать, что файл уже зашифрован (поддерживается не во всех версиях).

–L Перечислить файлы, сохраненные в момент аварийного завершения сеанса редактирования или системного сбоя (поддерживается не во всех версиях).

–R Открывать файлы в режиме «только для чтения».

Параметры командной строки ex

Несмотря на то что многим команды **ex** знакомы прежде всего по их использованию в редакторе **vi**, **ex** существует в качестве самостоятельной программы и может быть вызван из интерпретатора (например, для редактирования файлов из сценариев). В **ex** можно ввести команду **vi** или **visual** для перехода к работе с **vi**. Аналогично в **vi** команда **Q** позволяет перейти к работе с **ex**.

Если **ex** используется в качестве самостоятельного редактора, доступны следующие параметры командной строки:

+[*num*]

Перейти к редактированию строки *num* либо последней строки файла, если номер не указан.

+/*pattern*

Перейти к редактированию первой строки, соответствующей шаблону *pattern*. Не работает при установленном в файле *.exrc* параметре **nowrapscan**.

–c *command*

Выполнить указанную команду **ex** после запуска. Допустим только один параметр **–c**. Также поддерживается более старая форма использования этого параметра: +*command*.

–e Работать в режиме строчного редактора, а не в полноэкранном режиме. Это параметр по умолчанию.

–l LISP-режим, предназначенный для выполнения LISP-программ (поддерживается не во всех версиях редактора).

–r [*file*]

Восстановить файл и продолжить его редактирование после аварийного завершения сеанса редактирования или сбоя в системе. Если имя файла не указано, перечисляются доступные для восстановления файлы.

–s Не отображать приглашения. Параметр полезен при выполнении **ex** из сценария. Такое поведение может быть установлено при помощи более старого параметра «–».

–t *tag*

Редактировать файл, содержащий указанный тег, переместить курсор к месту его определения (см. также описание **ctags** в главе 3 «Команды Linux»).

–v Работа в полноэкранном режиме (идентично вызову **vi**).

–w *rows*

Установить размер окна в *rows* строк. Параметр полезен для редактирования при медленных удаленных соединениях.

–x Запрашивать ключ, который будет использоваться для шифрования и расшифровки файла посредством команды **crypt** (поддерживается не во всех версиях).

–C Аналогично **–x**, но в данном случае считается, что файл уже зашифрован (поддерживается не во всех версиях).

–L Перечислить файлы, сохраненные в момент аварийного завершения сеанса редактирования или системного сбоя (поддерживается не во всех версиях).

–R Открывать файлы в режиме «только для чтения»; сохранять изменения запрещено.

Завершить работу с **ex** можно несколькими способами:

`:x` Завершить работу (с сохранением изменений).

`:q!` Завершение работы без сохранения изменений.

`:vi` Перейти к работе с редактором **vi**.

Команды навигации

Число, предваряющее команду, задает количество повторений. Команды навигации также являются объектами для изменения, удаления и операций копирования в буфер.

Символ

Команда	Действие
h, j, k, l	Влево, вниз, вверх, вправо (←, ↓, ↑, →)
Spacebar	Вправо
Backspace	Влево
Ctrl-H	Влево

Текст

Команда	Действие
w, b	К следующему слову, к предыдущему слову (знаки препинания считаются отдельными словами)
W, B	К следующему слову, к предыдущему слову (знаки препинания не считаются разделителями, только пробелы)
e	К концу слова (знаки препинания считаются концом слова)
E	К концу слова (разделителями считаются только пробелы)
), (	К началу следующего, текущего предложения
}, {	К началу следующего, текущего абзаца
]], [[	К началу следующего, текущего раздела
Ctrl-D	К предыдущему табулятору
Ctrl-T	К следующему табулятору
Ctrl-W	К предыдущему слову

Строки

Команда	Действие
0, $	Первый, последний символ текущей строки
^, _	Первый видимый символ текущей строки
+, -	Первый символ следующей, предыдущей строки
Enter	Первый видимый символ следующей строки

Команда	Действие
n|	Колонка *n* текущей строки
H	Первая строка экранной страницы
M	Строка, расположенная в центре экранной страницы
L	Последняя строка экранной страницы
*n*H	*n*-ная строка, считая от первой сверху
*n*L	*n*-ная строка, считая от последней (вверх)
Ctrl-J	К следующей строке
Ctrl-M	К первому видимому символу в следующей строке

Экранные страницы

Команда	Действие
Ctrl-F, Ctrl-B	Прокрутка вперед, назад на одну экранную страницу
Ctrl-D, Ctrl-U	Прокрутка вперед, назад на половину экранной страницы
Ctrl-E, Ctrl-Y	Прокрутка на одну строку вперед либо назад
z Enter	Обновить экран так, чтобы строка с курсором стала первой на экранной странице
z .	Обновить экран так, чтобы строка с курсором оказалась в центре экранной страницы
z -	Обновить экран так, чтобы строка с курсором оказалась последней на экранной странице
Ctrl-L, Ctrl-R	Обновить экран (без прокрутки)

Поиск

Команда	Действие
/*pattern*	Прямой поиск по шаблону *pattern*
/	Повторить прямой поиск
/*pattern*/+*n*	Перейти к строке *n* после строки, соответствующей шаблону *pattern*
?*pattern*	Обратный поиск по шаблону *pattern*
?	Повторить обратный поиск
?*pattern*?-*n*	Переход к строке, отстоящей на *n* строк к началу файла от строки, соответствующей шаблону *pattern*
n	Повторить последний поиск

Команда	Действие
N	Повторить поиск в противоположном направлении
%	Найти парную скобку для текущей обычной, фигурной или квадратной скобки
f*x*	Перейти вперед к символу *x* в текущей строке
F*x*	Перейти назад к символу *x* в текущей строке
t*x*	Перейти вперед к символу текущей строки, предшествующему символу *x*
T*x*	Перейти назад к символу текущей строки, следующему за символом *x*
,	Изменить направление поиска последней команды **f**, **F**, **t** или **T**
;	Повторить последний символьный поиск (**f**, **F**, **t** или **T**)

Нумерация строк

Команда	Действие
Ctrl-G	Отобразить номер текущей строки и имя файла
*n*G	Перейти к строке с номером *n*
G	Перейти к последней строке файла
:*n*	Перейти к строке с номером *n*

Отметка положения

Команда	Действие
m*x*	Пометить текущую позицию в файле символом *x*
`*x*	(обратная кавычка) Переход к отметке *x*
'*x*	(апостроф) Перейти к началу строки, содержащей отметку *x*
``	(обратные кавычки) Вернуться к предыдущей отметке (или к позиции в тексте, предшествовашей поиску)
''	(апострофы) То же, но возврат к началу строки

Команды редактирования

Помните, что основными операторами редактирования являются **c**, **d** и **y**.

Вставка нового текста

Команда	Действие
a	Добавить текст после курсора
A	Добавить текст в конце строки

Команда	Действие
`i`	Вставить текст перед курсором
`I`	Вставить текст в начале строки
`o`	Начать новую строку после текущей
`O`	Начать новую строку перед текущей
`Esc`	Завершить работу в режиме вставки
`Tab`	Вставить табуляцию
`Backspace`	Удаление предыдущего символа (в режиме вставки)
`Ctrl-I`	Вставить табуляцию
`Ctrl-U`	Удалить текущую строку
`Ctrl-V`	Вставить следующий символ буквально
`Ctrl-[`	Завершить работу в режиме вставки

Некоторые из управляющих последовательностей, приведенных в последней таблице, устанавливаются с помощью **stty**. Настройки вашего терминала могут отличаться.

Изменение и удаление текста

Следующая таблица не является исчерпывающей, но иллюстрирует наиболее часто используемые операции.

Команда	Действие
`cw`	Изменить слово
`cc`	Изменить строку
`c$`	Изменить текст от курсора до конца строки
`C`	Идентично **c$**
`dd`	Удалить текущую строку
`d$`	Удалить остаток строки
`D`	Идентично **d$**
n`dd`	Удалить *n* строк
`dw`	Удалить слово
`d}`	Удалить все символы до начала следующего абзаца
`d^`	Удалить текст от курсора до начала строки
`d/`*pattern*	Удалить все символы от курсора до первого соответствия шаблону *pattern*
`dn`	Удалить все символы от курсора до следующего соответствия шаблону

Команда	Действие
df*a*	Удалить все символы от курсора до символа *a* в текущей строке (включительно)
dt*a*	Удалить все символы от курсора до символа *a* в текущей строке (исключая сам символ)
dL	Удалить все символы от курсора до последней строки экранной страницы
dG	Удалить все символы от курсора и до конца файла
p	Вставить последний удаленный объект после курсора
P	Вставить последний удаленный объект перед курсором
r*x*	Заменить текущий символ символом *x*
R*text*	Заменить текст *text*, начиная с положения курсора
s	Заменить символ
*n*s	Заменить *n* символов
S	Заменить всю строку
u	Откат последнего изменения текста
U	Восстановить текущую строку
x	Удалить текущий символ
X	Удалить символ перед курсором
*n*X	Удалить *n* символов перед курсором
.	Повторить последнее изменение
~	Сменить регистр символов
&	Повторить последнюю операцию замены
Y	Скопировать текущую строку во временный буфер
YY	Идентично **Y**
"*x*YY	Скопировать текущую строку в буфер *x*
Ye	Скопировать текст до конца слова во временный буфер
Yw	Идентично **ye**
y$	Скопировать текст до конца строки во временный буфер
"*x*dd	Удалить текущую строку в буфер *x*
"*X*dd	Удалить текущую строку и добавить ее к буферу *x*
"*x*p	Вставить содержимое буфера *x*
J	Объединить предыдущую строку с текущей
:j!	Идентично **J**

Сохранение и завершение работы

Сохранение файла – это запись последних изменений и обновление времени изменения файла.

Команда	Действие
`ZZ`	Завершить работу с **vi**, сохранив файл, если он был изменен в ходе сеанса работы
`:x`	Идентично **ZZ**
`:wq`	Сохранить файл и закончить его редактирование
`:w`	Сохранить файл
`:w` *file*	Сохранить копию в файл *file*
`:`*n1*`,`*n2*`w` *file*	Записать строки с *n1* по *n2* в файл *file*
`:`*n1*`,`*n2*`w >>` *file*	Добавить строки с *n1* по *n2* в существующий файл *file*
`:w!`	Принудительно записать файл
`:w!` *file*	Перезаписать файл содержимым текущего буфера редактирования
`:w %.`*new*	Записать текущий буфер с именем *file* под именем *file.new*
`:q`	Завершить редактирование файла
`:q!`	Завершить редактирование файла (без сохранения изменений)
`Q`	Завершить работу с **vi** и вызвать **ex**
`:vi`	Вернуться к работе с **vi** после ввода команды **Q**
`%`	Имя текущего файла
`#`	Альтернативное имя файла

Доступ к нескольким файлам

Команда	Действие
`:e` *file*	Начать редактирование файла *file*, текущий файл становится альтернативным
`:e!`	Загрузить последнюю сохраненную версию текущего файла
`:e+` *file*	Начать редактирование в конце файла *file*
`:e+` *n file*	Открыть файл *file* и перейти к строке *n*
`:e#`	Открыть альтернативный файл в предыдущей позиции
`:ta` *tag*	Редактировать файл в позиции *tag*
`:n`	Редактировать следующий файл

Команда	Действие
`:n!`	Принудительное редактирование следующего файла в текущем буфере (не сохраняя изменения в текущем файле)
`:n` *files*	Задать новый список файлов для редактирования (*files*)
`:args`	Отобразить список файлов для редактирования
`:rew`	Перейти к началу списка файлов для редактирования

Взаимодействие с интерпретатором

Команда	Действие
`:r` *file*	Вставить содержимое файла *file* после курсора
`:r !`*command*	Вставить вывод команды *command* после текущей строки
`:`*n*`r !`*command*	То же, но вставить после строки *n* (строка 0 является первой строкой файла)
`:!`*command*	Выполнить команду *command* и вернуться в **vi**
`!`*object command*	Послать содержимое объекта *object* в качестве ввода команде *command*; заменить объект *object* выводом команды
`:`*n1*`,`*n2*`!` *command*	Послать строки с *n1* по *n2* на ввод команды *command*; заменить строки выводом команды
n`!!`*command*	Послать *n* строк на ввод команды *command*; заменить их выводом команды
`!!`	Повторить последнюю команду интерпретатора
`!!`*command*	Заменить текущую строку выводом команды *command*
`:sh`	Вызвать субинтерпретатор; вернуться к редактированию по символу *EOF*
`Ctrl-Z`	Приостановить работу редактора, продолжить с помощью команды **fg**
`:so` *file*	Прочитать и выполнить команды **ex** из файла *file*

Макрокоманды

Команда	Действие
`:ab` *in out*	Использовать *in* в качестве сокращения *out*
`:unab` *in*	Удалить сокращение *in*
`:ab`	Перечислить сокращения
`:map` *c sequence*	Связать символ *c* с последовательностью команд *sequence*
`:unmap` *c*	Отменить связку для символа *c*

Команда	Действие
`:map`	Перечислить связанные символы
`:map!` *c sequence*	Связать символ *c* с режимом ввода *sequence*
`:unmap!` *c*	Отменить связку ввода для символа *c* (возможно, потребуется экранировать символ комбинацией **Ctrl-V**)
`:map!`	Перечислить символы, связанные с режимами ввода

Следующие символы не задействованы в командном режиме и могут быть связаны с командами пользователя:

Буквы:

g K q V v

Управляющие символы:

^K ^O ^T ^W ^X

Символы:

_ * \ =

Символ **=** используется в Lisp-режиме **vi**. Различные версии **vi** могут использовать перечисленные «свободные» символы, поэтому рекомендуется проверять их перед переопределением.

Различные команды

Команда	Действие
`<`	Сдвинуть строку влево в положение, указанное следующей командой навигации
`>`	Сдвинуть строку вправо в положение, указанное следующей командой навигации
`<<`	Сдвинуть строку влево на один шаг (шаг по умолчанию равен 8 колонкам)
`>>`	Сдвинуть строку вправо на один шаг (шаг по умолчанию равен 8 колонкам)
`>}`	Сместиться вправо до конца абзаца
`<%`	Сместиться влево до нахождения парного символа скобки, фигурной или квадратной скобки и т. д. (Курсор изначально должен находиться на символе, для которого существует парный.)
`^[`	Прервать выполнение команды или завершить работу в режиме вставки
`^]`	Выполнить поиск тега для текста под курсором
`^\`	Перейти в режим построчного редактирования **ex**
`^^`	(<Ctrl>+<^>) Вернуться к предыдущему файлу

Алфавитный перечень команд, действующих в командном режиме

Для краткости <Ctrl>+ (или C-) заменено в таблице на символ «^».

Команда	Действие
a	Добавление текста после курсора
A	Добавление текста в конец строки
^A	Поиск следующего вхождения слова под курсором
b	Переход к началу слова в текущей строке
B	Переход к предыдущему слову, считая знак препинания отдельным символом
^B	Прокрутка на одну экранную страницу назад
c	Заменить текст до позиции, в которую переводит курсор следующая команда навигации
C	Замена до конца текущей строки
^C	Окончание работы в режиме вставки; принудительное завершение длительной операции
d	Удалить текст до позиции, в которую переводит курсор следующая команда навигации
D	Удаление до конца текущей строки
^D	Прокрутка вперед на половину экранной страницы; в режиме вставки сократить отступ до размера **shiftwidth**, если установлен режим **autoindent**
e	Переход к концу слова
E	Переход к концу слова, считая знаки препинания частью слова
^E	Прокрутка вперед на одну строку
f	Прямой поиск введенного символа в текущей строке
F	Обратный поиск введенного символа в текущей строке
^F	Прокрутка вперед на одну экранную страницу
g	Не используется
G	Перейти к указанной строке или к концу файла
^G	Отобразить информацию о файле в строке состояния
h	Курсорная клавиша <Стрелка влево>
H	Перемещение курсора в позицию Home (в первую позицию на экране)
^H	Клавиша <Стрелка влево>; в режиме вставки клавиша <Backspace>

Команда	Действие
i	Вставка текста перед курсором
I	Вставка текста перед первым отображаемым символом строки
^I	В командном режиме не используется; в режиме вставки идентично <Tab>
j	Клавиша <Стрелка вниз>
J	Объединение двух строк
^J	Клавиша <Стрелка вниз>; в режиме вставки переход к следующей строке
k	Клавиша <Стрелка вверх>
K	Не используется
^K	Не используется
l	Клавиша <Стрелка вправо>
L	Перемещение курсора в последнюю позицию текущего окна текста
^L	Обновление экрана
m	Отметка текущей позиции строчным символом (a-z)
M	Перемещение курсора в центр текущего окна текста
^M	Перемещение в начало следующей строки
n	Повторить последнюю команду поиска
N	Повторить последнюю команду поиска в противоположном направлении
^N	Курсорная клавиша <Стрелка вниз>
o	Начать новую строку после текущей
O	Начать новую строку перед текущей
^O	Не используется
p	Вставить вырезанный или скопированный текст после или под курсором
P	Вставить вырезанный или скопированный текст перед или над курсором
^P	Клавиша <Стрелка вверх>
q	Не используется
Q	Завершить работу с **vi** и вызвать **ex**
^Q	Не используется. (На некоторых терминалах – возобновление обработки потока данных.)
r	Заменить текущий символ следующим набранным символом
R	Замена символов
^R	Обновление экрана

Команда	Действие
s	Заменить текущий символ вводимыми символами
S	Заменить всю строку
^S	Не используется. (На некоторых терминалах прекращает обработку потока данных.)
t	Найти в текущей строке (в прямом направлении) символ, совпадающий с введенным, и поместить перед ним курсор
T	Найти в текущей строке (в обратном направлении) символ, совпадающий с введенным, и поместить курсор после него
^T	В командном режиме не используется; в режиме вставки – переход к следующему табулятору
u	Отмена последнего изменения
U	Восстановить текущую строку, отменив изменения
^U	Прокрутка назад на половину экранной страницы
v	Не используется
V	Не используется
^V	В командном режиме не используется; в режиме вставки – буквальная вставка следующего символа
w	Переход к началу следующего слова
W	Переход к началу следующего слова, считая знаки препинания отдельными словами
^W	В командном режиме не используется; в режиме вставки – переход к началу слова
x	Удалить текущий символ
X	Удалить символ перед курсором
^X	Не используется
y	Оператор вставки или копирования текста (во временный буфер) до позиции, в которую переместит курсор следующая команда навигации
Y	Создать копию текущей строки
^Y	Прокрутка назад на одну строку
z	Перепозиционировать окно с текстом. За командой **z** должны следовать: <Enter> – строка, содержащая курсор, оказывается первой строкой экранной страницы, <.> – строка, содержащая курсор, оказывается в центре экранной страницы или <–> – строка, содержащая курсор, оказывается последней строкой экранной страницы
ZZ	Сохранение изменений и завершение работы с редактором

Синтаксис команд ex

Команда **ex** внутри редактора **vi** выглядит следующим образом:

:[*address*] *command* [*options*]

Предваряющее двоеточие говорит о том, что это команда **ex**. Во время набора команда отображается в строке состояния. Ввод команды завершается нажатием клавиши <Enter>. Адрес (*address*) – это номер строки или диапазон строк, для которых выполняется команда (*command*). Параметры (*options*) и адреса (*addresses*) описаны далее в этом разделе. Команды **ex** описаны в алфавитном перечне.

Параметры

! Предписание использовать измененную форму команды, а не стандартную.

count

Повторить команду *count* раз. В отличие от команд **vi**, команды **ex** не могут быть предваряемы числом, поскольку это число воспринимается как адрес строки. Так, **d3** удаляет три строки, начиная с текущей, а **3d** удаляет третью строку.

file Имя файла, с которым работает команда. Символ % означает текущий файл, а # – предыдущий.

Адреса

Если адрес не задан, команда применяется к текущей строке текста. Диапазон адресов строк задается следующим образом:

x, *y*

где *x* и *y* – адреса первой и последней строк диапазона (причем строка *x* должна предшествовать строке *y*). *x* и *y* могут быть числами или символами. Использование точки с запятой (;) вместо запятой (,) является предписанием установить адрес текущей строки в *x*, прежде чем интерпретировать *y*. Команда **1**,$ описывает весь диапазон строк файла (как и символ %).

Символы адресации

Символ	Значение
1, $	Все строки файла
%	Все строки; то же, что и **1**,$
x, *y*	Строки с *x* по *y*
x; *y*	Строки с *x* по *y* с установкой текущей строки в адрес *x*

Символ	Значение
0	Начало файла
.	Текущая строка
n	Строка с порядковым номером *n*
$	Последняя строка файла
x-*n*	*n* строк, предшествующих строке *x*
x+*n*	*n* строк, следующих за строкой *x*
-[*n*]	Предыдущая строка или строка, отстоящая назад от текущей на *n* строк
+[*n*]	Следующая строка или строка, отстоящая от текущей на *n* строк вперед
'*x*	Строка, отмеченная *x*
''	Предыдущая отметка
/*pattern*/	Вперед до строки, соответствующей шаблону *pattern*
?*pattern*?	Назад до строки, соответствующей шаблону *pattern*

Более подробно работа с шаблонами описана в главе 9 «Шаблоны и поиск».

Алфавитный перечень команд ex

Команды **ex** могут заменяться на их уникальные сокращения. В следующей таблице полные имена команд приведены в индексном столбце, а максимально сокращенные – в описании синтаксиса команды. В примерах предполагается, что команды вызываются в **vi**, поэтому они предваряются двоеточием.

abbrev

ab [*string text*]

Определить замену строки *string* на строку *text* при наборе. Если не заданы строка (*string*) и текст (*text*), перечислить все существующие сокращения.

Примеры

Примечание: Для печати символа ^M необходимо нажать клавиши <CTRL>+<V>, а затем <Enter>.

```
:ab ora O'Reilly & Associates, Inc.
:ab id Name:^MRank:^MPhone:
```

[*address*] **a**[**!**] *text* . Добавить текст (*text*) по заданному адресу (*address*) или по текущему адресу, если этот параметр опущен. Символ **!** является переключателем автоматического отступа (**autoindent**). Так, если автоматический отступ был включен, использование **!** отключит его на время работы команды. Ввод завершается при получении строки, содержащей единственный символ – точку.	**append**
ar Перечислить аргументы файловых имен (список файлов для редактирования). Имя текущего файла заключается в квадратные скобки ([]).	**args**
cd *dir* **chdir** *dir* Сменить текущий каталог внутри редактора.	**cd**
[*address*] **c**[**!**] *text* . Заменить указанные строки текстом *text*. Символ **!** является переключателем **autoindent** на время работы команды. Ввод завершается при получении строки, содержащей единственный символ – точку.	**change**
[*address*] **co** *destination* Копировать строки, определяемые адресом *address* в указанный адрес *destination*. Команда **t** работает идентично **copy**. **Пример** `:1,10 co 50` *Скопировать первые 10 строк после строки 50*	**copy**

delete

[*address*] **d** [*buffer*]

Удалить строки, определяемые адресом *address*. Если указан буфер (*buffer*), записать или добавить удаленные строки к буферу.

Примеры

`:/Part I/,/Part II/-1d`	*Удалить до строки над «Part II»*
`:/main/+d`	*Удалить строку после «main»*
`:.,$d`	*Удалить все строки после текущей*

edit

e[**!**] [+*n*] [*file*]

Начать редактирование файла *file*. Символ **!** запрещает сохранение изменений текущего файла. Если файл не задан, открыть для редактирования еще один экземпляр текущего файла. Если задан параметр +*n*, начать редактирование со строки *n*.

Примеры

`:e file`	
`:e#`	*Вернуться к редактированию предыдущего файла*
`:e!`	*Отменить изменения, сделанные с момента последнего сохранения файла*

exusage

exu [*command*]

Отобразить краткую справку с описанием указанной команды либо список доступных команд, если имя команды опущено.

file

f [*filename*]

Изменить имя текущего файла на *filename*. При этом файл получает статус «не редактированного файла». Если не задано новое имя файла, отобразить текущее состояние файла.

Пример

`:f %.new`

global

[*address*] **g**[**!**]/*pattern*/[*commands*]

Применить команды к строкам, соответствующим шаблону *pattern*, либо к диапазону строк, если таковой определен. Если не заданы команды (*commands*), перечислить все строки, соответствующие шаблону или диапазону. Символ **!** является отрицанием – команды выполняются для всех строк, не соответствующих шаблону. См. команду **v**.

Примеры **global**

`:g/Unix/p`	*Отобразить все строки, содержащие «Unix»*
`:g/Name:/s/tom/Tom`	*Во всех строках, содержащих «Name:», заменить «tom» на «Tom»*

h **help**

Отобразить краткую справку. Информация по конкретным командам может быть получена при помощи **exusage** или **viusage**.

address **i[!]** **insert**
text
.

Вставить текст *text* по заданному адресу *address* или по текущему адресу, если этот параметр опущен. Символ **!** является переключателем автоматического отступа **(autoindent)** на время ввода текста. Ввод завершается при получении строки, содержащей единственный символ – точку.

[*address*] **j[!]** [*count*] **join**

Объединить текст по перечисленным адресам в одну строку, добавляя по два пробела после каждой точки (.), по одному – после любого символа, кроме). Символ **!** предотвращает выравнивание пробелов.

Пример

`:1,5j!`	*Объединить первые пять строк, сохраняя пробелы*

[*address*] **k** *char* **k**

Пометить указанный адрес символом *char*, чтобы позже вернуться к строке по команде *'char*.

[*address*] **l** [*count*] **list**

Отобразить указанные строки, преобразуя символы табуляции в **^I**, а концы строк – в символ **$**. **l** является временной версией команды **:set list**.

map[!] [*char commands*] **map**

Определить макрос с именем *char*, представляющий собой последовательность команд (*commands*). Параметр *char*, как

map

правило, бывает одиночным символом или последовательностью вида #*n*, представляющей одну из функциональных клавиш. С помощью символа **!** можно создать макрос для режима ввода. Команда без аргументов перечисляет существующие макросы.

Примеры

```
:map K dwwP          Обменять местами пару слов
:map q :w^M:n^M      Сохранить текущий файл, открыть следующий
:map! + ^[bi(^[ea)   Заключить предыдущее слово в скобки
```

mark

[*address*] **ma** *char*

Пометить указанную строку строчным символом *char*. Позже можно вернуться к строке по команде *'char*. Идентично **k**.

mkexrc

mk[!] *file*

Создать файл *.exrc*, содержащий команду **set** для каждого из параметров **ex**, устанавливающую этот параметр в значение по умолчанию.

move

[*address*] **m** *destination*

Переместить указанные строки (*address*) по новому адресу *destination*.

Пример

```
:.,/Note/m /END/        Поместить блок текста за строкой,
                        содержащей «END»
```

next

n[!] [[+*command*] *filelist*]

Редактировать следующий файл из списка аргументов. Используйте команду **args** для отображения списка. Если задан список файлов *filelist*, заменить им текущий список и начать редактирование с первого файла. Если задана команда *command* (не содержащая пробелов), выполнить эту команду после редактирования первого файла. Восклицательный знак (**!**) является указанием не сохранять никакие изменения, сделанные в текущем файле.

Пример

```
:n chap*     Редактировать все файлы «chapter»
```

[*address*] **nu** [*count*] Отобразить все строки, указанные диапазоном адресов (*address*), предваряя каждую ее порядковым номером в файле. Используйте символ # в качестве альтернативного сокращения **number**. Параметр *count* представляет собой количество строк, которое необходимо отобразить, начиная с адреса *address*.	**number**
[*address*] **o** [/*pattern*/] Перейти в открытый режим редактирования **vi** указанных строк (по адресу *address* либо по шаблону *pattern*). Переключение открытого режима осуществляется нажатием клавиши <Q>. Открытый режим допускает применение обычных команд **vi**, но лишь при работе с одной строкой единовременно. Режим может быть полезен для медленных удаленных соединений.	**open**
pre Сохранить текущий буфер, как при аварийном отказе системы.	**preserve**
prev[**!**] Редактировать предыдущий файл из списка, заданного в командной строке.	**previous**
[*address*] **p** [*count*] [*address*] **P** [*count*] Вывести строки, заданные адресами (*address*). Параметр *count* определяет количество выводимых строк, начиная со строки *address*. Добавление восклицательного знака является указанием не сохранять изменения, сделанные в текущем файле. **Пример** `:100;+5p` *Отобразить 100-ую строку и 5 следующих строк*	**print**
[*address*] **pu** [*char*] Восстановить строки, которые были удалены или извлечены из буфера *char*, и поместить их после строки с адресом *address*. Если параметр *char* опущен, восстановить последний удаленный или извлеченный текст.	**put**

quit	**q[!]** Завершить текущий сеанс работы. Укажите символ **!**, чтобы запретить сохранение изменений, сделанных после последнего сохранения. Сеанс редактирования, в котором не изменялись файлы, определенные в списке аргументов, можно завершить вводом **q!** или **qq**.
read	[*address*] **r** *file* Скопировать текст из файла *file*, вставив его после строки с адресом *address*. Если имя файла не задано, использовать текущий файл. **Пример** `:0r $HOME/data` *Вставка файла в начало текущего*
read	[*address*] **r !***command* Вставить вывод команды Linux *command* в текст после строки с адресом *address*. **Пример** `:$r !cal` *Вставка календаря в конец файла*
recover	**rec** [*file*] Восстановить файл *file* из зоны системного сохранения.
rewind	**rew[!]** Обнулить указатель на список аргументов и начать редактирование первого файла из списка. Если присутствует символ **!**, производится обнуление указателя и начинается редактирование первого файла с потерей всех несохраненных изменений предыдущего.
script	**sc[!]** [*file*] Создать новый интерпретатор в буфере, который может быть сохранен, с необязательным указанием имени файла для сохранения. Команда может применяться только из **vi**.

se *parameter1 parameter2* . . . — **set**

Установить значения *parameter1* и *parameter2*. Команда без параметров выводит те из них, значения которых отличаются от принятых по умолчанию. Для булевых переменных, имеющих состояния включено/выключено, значения могут устанавливаться как *parameter* или **no***parameter* (как в первом примере). Другим параметрам можно присваивать значения, используя синтаксис *parameter=value*. Ключевое слово **all** перечисляет параметры и их текущие значения.

Примеры

```
:set nows wm=10
:set all
```

sh — **shell**

Создать экземпляр интерпретатора. Продолжить редактирование по завершении работы с ним.

so *file* — **source**

Прочитать и выполнить команды **ex** из файла *file*.

Пример

```
:so $HOME/.exrc
```

st — **stop**

Приостановить сеанс редактирования. Идентично **Ctrl-Z**. Команда **fg** позволяет продолжить работу с редактором.

[*address*] **s** [/*pattern*/*replacement*/] [*options*] [*count*] — **substitute**

Заменить в указанных строках (*address*) каждое соответствие шаблону *pattern* на *replacement*. Если опущены шаблон и подстановка, повторить последнюю замену. Параметр *count* определяет количество строк, в которых должна производиться замена, начиная со строки *address*. Если команде замены предшествует команда **global** (**g**) или **v**, шаблон может быть опущен; в таком случае используемый шаблон определяется этими командами. Большее количество примеров приводится в разделе «Примеры поиска и замены» главы 9.

substitute

Параметры

c Перед каждой заменой запрашивать подтверждение.

g Заменить все вхождения соответствий шаблону во всех строках.

p Отобразить последнюю строку, подвергшуюся операции замены.

Примеры

`:1,10s/yes/no/g` *Произвести замену в первых 10 строках*
`:%s/[Hh]ello/Hi/gc` *Подтвердить глобальную замену*
`:s/Fortran/\U&/ 3` *Перевести в верхний регистр все слова «Fortran» в следующих трех строках*
`:g/^[0-9][0-9]*/s//Line &:/` *Перед каждой строкой, которая начинается с одной или более цифр, добавить префикс «Line» и двоеточие*

suspend

su

Приостановить сеанс редактирования. Идентично **Ctrl-Z**. Команда **fg** позволяет продолжить работу с редактором.

t

[*address*] **t** *destination*

Скопировать указанные строки (*address*) в указанный конечный адрес *destination*. Команды **t** и **copy** являются синонимами.

Пример

`:%t$` *Скопировать файл и добавить его в конец текста*

tag

[*address*] **ta**[**!**] *tag*

Переключить сеанс редактирования на файл, содержащий тег *tag*.

Пример

Выполнить команду **ctags**, а затем переключиться на файл, содержащий функцию *myfunction*:

```
:!ctags *.c
:tag myfunction
```

tagn[!] Найти следующее вхождение текущего тега.	**tagnext**
tagp[!] Вернуться к последнему вхождению предыдущего тега.	**tagpop**
tagpr[!] Вернуться к предыдущему вхождению текущего тега.	**tagprev**
tagt[!] Вернуться к работе с первым из тегов.	**tagtop**
una *word* Удалить слово *word* из списка сокращений.	**unabbreviate**
u Обратить изменения, внесенные последней командой редактирования.	**undo**
unm[!] *char* Удалить символ *char* из списка макросов. Используйте символ **!** для удаления макросов режима ввода.	**unmap**
[*address*] **v**/*pattern*/[*commands*] Применить команды *commands* ко всем строкам, не содержащим соответствия шаблону *pattern*. Если команды не заданы, отобразить все соответствующие строки. Команда **v** эквивалентна **g!**. См. **global**. **Пример** `:v/#include/d` *Удалить все строки, кроме содержащих «#include»*	**v**
ve Отобразить номер версии редактора.	**version**

vi	**vi** [+*n*] *file* Начать редактирование указанного файла. Может быть указан номер строки, с которой необходимо начать редактирование. Может использоваться только в **vi**.
visual	[*address*] **vi** [*type*] [*count*] Перейти в режим визуального редактирования (**vi**) для строки, заданной адресом *address*. Возврат по команде **Q**. Тип *type* может быть символом –, ^ или . (см. команду **z**). Параметр *count* определяет начальный размер окна.
viusage	**viu** [*key*] Отобразить краткую справку по использованию команды *key* либо перечень команд, если имя не указано.
wq	**wq**[**!**] Записать файл и завершить редактирование одной командой. Восклицательный знак является указанием редактору принудительно перезаписать текущее содержимое файла содержимым буфера.
write	[*address*] **w**[**!**] [[>>] *file*] Записать указанные строки (*address*) в файл *file* или записать все содержимое буфера, если параметр *address* опущен. Если опущено и имя файла, содержимое буфера сохраняется в текущий файл. Символы >> добавляют указанные строки или содержимое буфера в конец существующего файла *file*. Символ **!** (флаг) является разрешением на перезапись любого существующего файла.
write	[*address*]**w !***command* Записать указанные строки в *command*. **Примеры** `:1,10w name_list` *Записать первые десять строк в name_list* `:50w >> name_list` *Добавить пятидесятую строку*

x **xit**

Сохранение файла, если он был изменен с момента последней записи в файл, затем завершение работы.

[*address*] **ya** [*char*] [*count*] **yank**

Поместить указанные строки (*address*) в буфер с именем *char*. Если имя буфера не задано, строки помещаются в обычный буфер. Параметр *count* определяет количество строк для копирования, начиная со строки *address*.

Пример

```
:101,200 ya a
```

[*address*] **z** [*type*] [*count*] **z**

Отобразить окно с текстом, содержащее строки, начиная с *address*. Параметр *count* определяет количество отображаемых строк.

Тип

- + Поместить указанную строку в верхней части (в начале) окна (тип по умолчанию).
- – Поместить указанную строку в нижней части (в конце) окна.
- . Поместить указанную строку в центре окна.
- ^ Прокрутка назад на одну экранную страницу.
- = Поместить указанную строку в центре окна и сделать ее текущей.

[*address*] **!***command* **!**

Выполнить команду Linux в интерпретаторе. Если задан адрес, передать указанные строки в качестве ввода команды и заменить их полученным выводом.

Примеры

`:!ls`	*Отобразить содержимое текущего каталога*
`:11,20!sort -f`	*Отсортировать строки с 11 по 20*

[*address*] = =

Отобразить номер следующей строки из диапазона *address*. Если адрес не задан, отобразить номер последней строки.

<>	[*address*] < [*count*] [*address*] > [*count*] Сдвинуть *count* строк по указанному адресу влево (<) или вправо (>). При сдвиге влево удаляются только пробелы и символы табуляции.
address	*address* Отобразить строку, имеющую указанный адрес.
Enter	*Enter* Отобразить следующую строку файла.
&	& [*options*] [*count*] Повторить последнюю команду замены **(s)**. Параметр *count* определяет количество строк, в которых будет производиться замена, начиная со строки *address*. **Примеры** `:s/Overdue/Paid` *Однократная замена в текущей строке* `:g/Status/&` *Повторить замену для всех строк, содержащих слово «Status»*
~	[*address*] ~ [*count*] Заменить предыдущее регулярное выражение предыдущим шаблоном подстановки из команды **substitute (s)**.
^D	**^D** Просмотр файла.
^Z	**^Z** Приостановка сеанса редактирования. Возврат по команде **fg**.

Настройка vi

В этом разделе рассматриваются:

- Команда **:set**
- Параметры команды **:set**
- Пример файла *~/.exrc*

Команда :set

Команда **:set** позволяет задавать параметры, которые изменяют характеристики окружения. Параметры могут быть заданы в файле *~/.exrc* или во время сеанса работы с **vi**.

В командах внутри файла *.exrc* следует опускать двоеточие.

Команда	Действие
`:set` *x*	Включить параметр *x*
`:set no`*x*	Отключить параметр *x*
`:set` *x*=*val*	Присвоить значение *val* параметру *x*
`:set`	Перечислить измененные параметры
`:set all`	Отобразить все параметры
`:set` *x*?	Отобразить значение параметра *x*

Параметры команды :set

В следующей таблице приведены параметры, доступные команде **:set**. В первой колонке, помимо самого параметра, приводится его возможное сокращение, если таковое существует. Знак равенства используется для обозначения параметра, который может принимать значения. Во второй колонке приводится значение параметра по умолчанию, а в третьей колонке описано действие параметра.

Параметр	Умолчание	Описание
autoindent (ai)	**noai**	В режиме вставки выравнивать отступы следующей строки по предыдущей или последующей. Используется с параметром **shiftwidth**.
autoprint (ap)	**ap**	Обновление изображения после каждой команды редактора. (При глобальных заменах обновление происходит после завершения замены.)
autowrite (aw)	**noaw**	Автоматически сохранять файл, если он был изменен, перед открытием другого файла командой **:n** или перед выполнением команды Linux с помощью **:!**.

Параметр	Умолчание	Описание
beautify (bf)	**nobf**	Игнорировать все управляющие символы во время ввода (кроме табуляции, символа новой строки или новой страницы).
directory= (dir)	**/tmp**	Имя каталога, в котором редактор **ex** хранит файлы буферов. (Запись в каталог должна быть разрешена.)
edcompatible	**noed-compatible**	Использовать команды замены в стиле **ed**.
errorbells (eb)	**errorbells**	Выдавать звуковой сигнал, если произошла ошибка.
exrc (ex)	**noexrc**	Разрешить исполнение файла *.exrc*, который находится не в домашнем каталоге пользователя.
hardtabs= (ht)	**8**	Задать размер терминальной табуляции.
ignorecase (ic)	**noic**	Поиск не чувствителен к регистру символов.
lisp	**nolisp**	Отступы в формате Lisp. Изменять символы ( ), { }, [[и]] на имеющие смысл для Lisp.
list	**nolist**	Отображать символы табуляции как ^I; отмечать концы строк символом $ (использовать **list**, чтобы отличать символы конца строк от табуляций или пробелов).
magic	**magic**	Метасимволы **.** (точка), ***** (звездочка) и **[ ]** (квадратные скобки) имеют специальное значение в шаблонах.
mesg	**mesg**	Разрешить отображение системных сообщений на терминале во время работы с **vi**.
number (nu)	**nonu**	Отображать номера строк по левому краю экрана во время сеанса редактирования.
open	**open**	Разрешает переход в открытый режим **vi** из **ex**.
optimize (opt)	**noopt**	Удалять символы возврата каретки в конце строк при отображении нескольких строк. Это повышает скорость вывода данных на устаревших терминалах, если первыми символами строк являются пробелы или символы табуляции.
paragraphs= (para)	**IPLPPPQPLI pplpipbp**	Определить разделители абзацев для навигации с помощью { или }. Пары символов значения являются именами макрокоманд **nroff/troff**, начинающих абзацы.
prompt	**prompt**	Отображать приглашение **ex (:)** по **vi**-команде **Q**

Параметр	Умолчание	Описание
readonly (ro)	**noro**	Сохранение файла будет работать только при добавлении к команде символа **!** (работает с **w**, **ZZ** или **autowrite**).
redraw (re)	**noredraw**	Терминал обновляет изображение, когда происходит изменение (иными словами, в режиме вставки происходит замена символов или удаление строки в пределах текущего окна). Значение по умолчанию зависит от скорости соединения и типа терминала. Значение **noredraw** полезно на устаревших терминалах: удаленные строки отмечаются символом @, вставленный текст записывается поверх старого. Реально изменения отображаются по нажатию клавиши <Esc>.
remap	**remap**	Разрешить задание вложенных связок.
report=	**5**	Отображать соответствующее сообщение при удалении по крайней мере такого количества строк (по умолчанию равно 5). Например, команда **6dd** отображает сообщение «6 lines deleted» (удалено 6 строк).
scroll=	**<1/2 window>**	Количество строк прокрутки.
sections= (sect)	**SHNHH HU**	Определить разделители разделов для навигации с помощью [[и]]. Пары символов значения являются именами макрокоманд **nroff/troff**, начинающих разделы.
shell= (sh)	**/bin/sh**	Полное имя (с путем) интерпретатора, которое используется для выполнения команд Linux (**:!**) и команды **:sh**. По умолчанию используется значение переменной SHELL.
shiftwidth= (sw)	**8**	Определить количество пробелов для команд отступа (**^T**, **^D**, >> и <<).
showmatch (sm)	**nosm**	В **vi** при нажатии клавиши <)> или <}> курсор на короткое время переходит на парный символ в пределах текущей экранной страницы (если парный символ находится за пределами страницы, генерируется звуковой сигнал). Удобная возможность для программистов .
showmode	**noshowmode**	В режиме вставки отображать сообщение в строке состояния, указывающее на конкретный тип применяемой вставки. Например, «Open Mode» (открытый режим) или «Append Mode» (режим добавления).

Параметр	Умолчание	Описание
slowopen (slow)		Не обновлять изображение в процессе вставки. Значение по умолчанию зависит от скорости соединения и типа терминала.
tabstop= (ts)	**8**	Определить количество пробелов, содержащихся в табуляции (для текущего сеанса). Печатающее устройство будет по-прежнему использовать системную установку табулятора в 8 пробелов.
taglength= (tl)	**0**	Определить количество значимых символов в теге (tag). По умолчанию параметр равен 0, т. е. значимы все символы.
tags=	**tags/usr/lib/tags**	Определить полные пути файлов, содержащих теги (см. команду **ctags** в главе 3). По умолчанию происходит поиск файлов **tags** (в текущем каталоге) и */usr/lib/tags*.
term=		Задание типа терминала.
terse	**noterse**	Краткие сообщения об ошибках.
timeout (to)	**timeout**	Клавиатурные связки прекращают ожидание через 1 секунду.
ttytype=		Задание типа терминала. По умолчанию принимает значение переменной окружения TERM.
warn	**warn**	Отображать сообщение «No write since last change» (последние изменения не сохранены).
window= (w)		Отображать определенное количество строк файла на экране. Значение по умолчанию зависит от скорости соединения и типа терминала.
wrapmargin= (wm)	**0**	Определить правое поле страницы. Если значение больше нуля, автоматически разрывать в этом месте строку посредством вставки символа возврата каретки.
wrapscan (ws)	**ws**	Кольцевой поиск по файлу (не прекращается по достижении конца или начала файла).
writeany (wa)	**nowa**	Разрешить сохранение в любой файл.

Пример файла ~/.exrc

```
set nowrapscan wrapmargin=7
set sections=SeAhBhChDh nomesg
map q :w^M:n^M
map v dwElp
ab ORA O'Reilly & Associates, Inc.
```

Глава 12

Редактор sed

Темы, рассмотренные в этой главе:

- Общий обзор **sed**
- Синтаксис командной строки
- Синтаксис команд **sed**
- Перечень команд **sed** по группам
- Алфавитный перечень команд **sed**

Более подробную информацию можно найти во втором издании книги Дэйла Дауэрти (Dale Dougherty) и Арнольда Роббинса (Arnold Robbins) «sed & awk» издательства O'Reilly.

Общий обзор sed

Редактор **sed** является пакетным или поточно-ориентированным (stream-oriented) редактором. Редактор интерпретирует сценарий и выполняет указанные в нем действия. Он поточно-ориентированный, поскольку, подобно многим программам Unix, читает информацию со стандартного ввода и выдает ее, измененную, на стандартный вывод. Например, программа **sort** – поточно-ориентированная, а **vi** – нет. Ввод **sed**, как правило, поступает из файла, но возможен и ввод с клавиатуры. Вывод по умолчанию поступает на терминал, но может быть перенаправлен в файл.

Обычно **sed** используется для:

- Автоматического редактирования одного или нескольких файлов
- Автоматизации повторяющихся изменений многочисленных файлов
- Создания программ преобразования

sed действует следующим образом:

- Каждая строка ввода копируется в пространство шаблонов
- Все команды редактирования из сценария **sed** применяются последовательно к каждой строке ввода
- Команды редактирования **sed** применяются ко всем строкам, если на производимые изменения не наложены адресные ограничения
- Если команда изменяет ввод, все последующие команды применяются к измененной строке, а не к исходной
- Исходный обрабатываемый файл не изменяется, поскольку **sed** обрабатывает копии исходных строк. Обработанные копии посылаются на стандартный вывод (либо в файл при перенаправлении)

Синтаксис командной строки

Существует два варианта запуска **sed**:

sed [*options*] '*command*' *file*(*s*)
sed [*options*] **–f** *scriptfile file*(*s*)

В первом варианте возможно задание команды редактирования **sed** (заключенной в одинарные кавычки) в командной строке. Во втором варианте задается файл сценария *scriptfile*, содержащий команды **sed**. Если не заданы обрабатываемые файлы, происходит чтение со стандартного ввода.

Доступны следующие параметры командной строки:

–e *cmd*

Следующий аргумент является инструкцией редактирования; параметр необходим только при задании более чем одной инструкции.

–f *scriptfile*

Аргумент является файлом, содержащим команды редактирования.

–n Подавить вывод по умолчанию; **sed** отображает строки только по команде **p** или при установленном ключе **p** команды **s**.

–V Отобразить номер версии **sed**.

––quiet

Идентично **–n**.

––expression=*cmd*

Идентично **–e**.

––file=*file*

Идентично **–f**.

––help

Отобразить краткую справку, включающую информацию по параметрам командной строки.

−−silent

Идентично **−n**.

−−version

Идентично **−V**.

Синтаксис команд sed

Общий вид команд **sed**:

[*address*[,*address*]][**!**]*command*[*arguments*]

Команды состоят из *адресов* и *инструкций редактирования* (*commands*). Инструкции являются отдельными буквами или символами; они описаны далее в этой главе в алфавитном порядке и по группам. Аргументы (*arguments*) могут содержать метки для инструкций **b** и **t** или имена файлов для инструкций **r** и **w**, а также ключи подстановок для инструкции **s**. Адреса (*addresses*) описаны ниже.

Адресация шаблонов

В команде **sed** адрес может отсутствовать или могут быть заданы один или пара адресов. Адрес может являться номером строки, символом $ (адрес последней строки) или регулярным выражением, заключенным в символы «/» (/*шаблон*/). Регулярные выражения описаны в главе 9 «Шаблоны и поиск». Кроме того, в **sed** можно использовать **\n** для нахождения символа новой строки в пространстве шаблонов (появившегося в результате выполнения команды **N**), но не в конце пространства шаблонов.

Адрес	Команда применяется к
Отсутствует	Каждой из исходных строк
Один адрес	Каждой строке с таким адресом. Некоторые инструкции могут использоваться только с одним адресом: `a`, `i`, `r`, `q` и `=`
Пара адресов, разделенных запятыми	Диапазону строк, соответствующему адресам
За адресом следует символ «!»	Ко всем строкам, адрес которых не совпадает с указанным

Примеры

`s/xx/yy/g`	*Заменить все строки (все вхождения)*
`/BSD/d`	*Удалить строки, содержащие BSD*
`/^BEGIN/,/^END/p`	*Отобразить строки от BEGIN до END включительно*
`/SAVE/!d`	*Удалить все строки, которые не содержат SAVE*
`/BEGIN/,/END/!s/xx/yy/g`	*Произвести замену во всех строках диапазона между BEGIN и END*

Фигурные скобки ({}) используются для задания вложенных адресов или для применения нескольких команд к одному адресу:

[/*address*/[,/*address*/]]{
command1
command2
}

Открывающая фигурная скобка должна быть последним символом в строке, а закрывающая – единственным символом в строке. Пробелы после скобок не допустимы.

Перечень команд sed по группам

В следующих таблицах команды **sed** сгруппированы по их назначению, дано их краткое описание. Полное описание, включающее формат использования и примеры, приводится ниже в алфавитном перечне команд.

Простое редактирование

Команда	Действие
a\	Добавление текста после строки
c\	Замена текста (обычно области текста)
i\	Вставка текста перед строкой
d	Удаление строк
s	Замена
y	Преобразование символов (аналогично действию команды tr, описанной в главе 3 «Команды Linux»)

Информация о строках

Команда	Действие
=	Отобразить порядковый номер строки
l	Отображать управляющие символы в кодах ASCII
p	Отобразить строку

Обработка ввода/вывода

Команда	Действие
n	Пропустить текущую строку и перейти к следующей
r	Послать на ввод **sed** содержимое другого файла
w	Записать исходные строки в другой файл
q	Завершить работу сценария **sed** (конец вывода)

Копирование и вставка

Команда	Действие
h	Создать дубликат пространства шаблонов в специальном буфере; предыдущее содержимое буфера уничтожается
H	Добавить дубликат пространства шаблонов к содержимому специального буфера
g	Перезаписать пространство шаблонов содержимым буфера
G	Добавить к пространству шаблонов содержимое буфера
x	Обменять содержимое пространства шаблонов и специального буфера

Команды ветвления

Команда	Действие
b	Перейти к метке *label* или к концу сценария
t	То же, что и **b**, но переход совершается только после замены
:*label*	Метка, на которую совершается переход по командам **b** или **t**

Обработка многострочного ввода

Команда	Действие
N	Прочитать следующую строку из ввода (с добавлением символа новой строки)
D	Удалить все до символа новой строки
P	Отобразить символы до символа новой строки

Алфавитный перечень команд sed

#	# Начало комментария в сценарии **sed**. Допустимо использование только в качестве первого символа первой строки. (Некоторые версии **sed**, включая реализацию GNU, допускают использование комментариев в любом месте сценария, но предпочтительнее этим не пользоваться.) Если первой строкой сценария является **#n**, это эквивалентно заданию параметра **−n**.
:	:*label* Метка строки сценария для передачи управления инструкциям **b** или **t**. Метка *label* может содержать до семи символов.
=	[/*pattern*/]= Выдать в стандартный вывод номера всех строк, соответствующих шаблону *pattern*.

a	[*address*]a\
	text
	
	Добавление после каждой из указанных строк текста *text*. Если текст содержит более одной строки, символы новой строки должны быть экранированы при помощи символа \. Текст завершается первым неэкранированным символом новой строки. Текст не доступен в пространстве шаблонов, поэтому следующие команды не могут к нему применяться. Результат работы этой команды выводится на стандартный вывод по исчерпании списка инструкций редактирования, независимо от того, что происходило с копией данной исходной строки.
	
	Пример
	
	`$a\`
	`This goes after the last line in the file\`
	`(marked by $). This text is escaped at the\`
	`end of each line, except for the last one.`

[*address1*[,*address2*]]b[*label*] **b**

Передать управление строке в сценарии, которая содержит метку *label*. То есть следующей командой, применяемой к текущей строке, будет команда после метки *label*. Если метка не указана, происходит переход к концу сценария, и к данной строке не применяются больше никакие команды.

Пример

Игнорировать строки между **.TS** и **.TE**; продолжить выполнение сценария после **.TE**:

```
/^\.TS/,/^\.TE/b
```

[*address1*[,*address2*]]c\
text **c**

Заменить указанные строки текстом *text*. Если задан диапазон строк, все они заменяются единственной копией текста. Все символы новой строки в тексте, кроме последнего, должны экранироваться символом \. Содержимое пространства шаблонов удаляется, и поэтому нет смысла применять еще какие-либо команды к этим строкам.

Пример

Заменить первые 100 строк файла.

```
1,100c\
\
<First 100 names to be supplied>
```

[*address1*[,*address2*]]d **d**

Удалить указанные строки из пространства шаблонов. Удаленные строки не выводятся. Из ввода читается следующая строка для обработки, начиная с первой команды сценария.

Пример

Удалить все пустые строки:

```
/^$/d
```

[*address1*[,*address2*]]D **D**

Удалить первую часть (до символа новой строки) многострочной копии, созданной командой **N**, и продолжить редактиро-

D

вание с первой команды в сценарии. Если эта команда опустошает пространство шаблонов, из ввода читается новая строка, как если бы выполнялась команда **d**.

Пример

Удалить множественные пустые строки, оставив только одну:

```
/^$/{
N
/^\n$/D
}
```

g

[*address1*[,*address2*]]g

Передать содержимое специального буфера (см. команды **h** и **H**) обратно в пространство шаблонов, перезаписав существующее содержимое. Ниже приводится пример простого копирования строк.

Пример

Этот сценарий собирает все строки, содержащие слово *Item:*, и заменяет ими специальный маркер, расположенный ниже в тексте.

```
/Item:/H
/<Replace this line with the item list>/g
```

G

[*address1*[,*address2*]]G

То же, что и **g**, только специальный буфер добавляется к пространству шаблонов, не перезаписывая содержимое. В примере показан простой способ вырезания и вставки строк.

Пример

В этом сценарии собираются все строки, содержащие слово *Item:*. Строки копируются после специального маркера, расположенного далее в файле. Исходные строки со словом *Item:* удаляются.

```
/Item:/{
H
d
}
/Summary of items:/G
```

h

[*address1*[,*address2*]]h

Скопировать содержимое пространства копий в специальный временный буфер. Предыдущее содержимое специального буфера уничтожается. Команду **h** можно использовать для сохранения строки перед редактированием.

Пример

```
# Редактировать строку; отобразить изменения;
# отобразить исходный вариант
/Linux/{
h
s/.*Linux \(.\) .*/\1:/
p
x
}
```

Пример ввода:

```
This describes the Linux ls command.
This describes the Linux cp command.
```

Пример вывода:

```
ls:
This describes the Linux ls command.
cp:
This describes the Linux cp command.
```

H

[*address1*[,*address2*]]H

Добавить содержимое пространства шаблонов (предваряемое символом новой строки) к содержимому специального буфера. Символ новой строки добавляется, даже если буфер пуст. Команда **H** обеспечивает пошаговое копирование. См. примеры для команд **g** и **G**.

i

[*address1*]i\
text

Вставить текст *text* перед каждой строкой, соответствующей указанному адресу. (См. описание параметра *text* в информации по команде **a**).

Пример

```
/Item 1/i\
The five items are listed below:
```

l

[*address1*[,*address2*]]l

Отобразить содержимое пространства шаблонов, заменяя непечатаемые символы соответствующими выражениями в кодах ASCII. Длинные строки переносятся.

n

[*address1*[,*address2*]]n

Прочитать следующую строку ввода в пространство шаблонов. Текущая строка отправляется на стандартный вывод, а следующая становится текущей. Управление передается команде, следующей за **n**, а не в начало сценария.

Пример

В макросе **ms** заголовок раздела располагается строкой ниже макроса **.NH**. Чтобы отобразить все заголовки разделов, можно использовать такой сценарий:

```
/^\.NH/{
n
p
}
```

N

[*address1*[,*address2*]]N

Добавить следующую строку ввода к пространству шаблонов; две строки разделяются вложенным символом новой строки. (Эта команда предназначена для разрешения проблемы параллельного поиска по шаблонам в двух строках.) Используя **\n** для нахождения вложенного символа новой строки, можно производить параллельный поиск в строках. См. пример для команды **D**.

Примеры

Аналогично предыдущему примеру, но вместе со строкой заголовка печатать и строку **.NH**.

```
/^\.NH/{
N
p
}
```

Объединение двух строк (символ новой строки заменяется пробелом):

```
/^\.NH/{
N
```

`s/\n/ /`
`p`
`}`
	N

[*address1*[,*address2*]]`p` Отобразить указанные строки. Если не задан параметр командной строки **–n**, эта команда приведет к дублированию строк при выводе. Обычно она используется перед командами, которые изменяют порядок работы с данными **(d, N, b)** и могут привести к тому, что строка вовсе не будет отображена. См. примеры для **h**, **n** и **N**.	**p**
[*address1*[,*address2*]]`P` Отобразить первую часть (до вложенного символа новой строки) многострочной копии, созданной с помощью команды **N**. То же, что и **p**, если команда **N** не применялась к строке.	**P**

[*address*]`q`
 	
Завершить работу по достижении строки *address*. Указанная строка предварительно посылается на вывод (если не включено подавление потока вывода по умолчанию), как и любой текст, добавленный к этой строке предшествующими командами **a** и **r**.
 	
Примеры
 	
Удалить все после указанной строки:
 	
`/Garbled text follows:/q`
 	
Отобразить только 50 первых строк файла:
 	
`50q`
	q

[*address*]`r` *file* Добавить содержимое файла к содержимому пространства шаблонов. Между инструкцией **r** и именем файла должен находиться только один пробел. **Пример** `/The list of items follows:/r item_file`	**r**

s

[*address1*[,*address2*]]s/*pattern*/*replacement*/[*flags*]

Заменить соответствие шаблону *pattern* на *replacement* во всех указанных строках. Если используются шаблонные адреса, шаблон «//» представляет собой последний заданный адрес. Могут быть указаны следующие ключи:

n Заменить *n*-ное вхождение соответствия шаблону /*pattern*/ в каждой из строк диапазона. *n* может быть любым числом от 1 до 512. По умолчанию $n = 1$.

g Заменить все вхождения соответствий шаблону /*pattern*/ в каждой из строк диапазона, а не только первое из вхождений.

p Отобразить строку, если произведена успешная замена. Если произведено несколько таких замен, будет отображено соответствующее количество копий строки.

w *file*

Записать строку в файл *file*, если была произведена успешная замена.

Примеры

Вот несколько примеров коротких сценариев с комментариями:

```
# Замена третьей и четвертой кавычек символами ( и ):
/function/{
s/"/(/3
s/"/)/4
}

# Удалить все кавычки в указанной строке:
/Title/s/"//g

# Удалить первое двоеточие или все кавычки; отобразить результат:
s/://p
s/"//gp

# Заменить первое “if”, но не заменять “ifdef”
/ifdef/!s/if    if/
```

t

[*address1*[,*address2*]]t[*label*]

Проверить, были ли произведены какие-либо замены в указанных строках, и если результат положительный, перейти к строке сценария, содержащей метку *:label* (см. команды **b** и **:**). Если не задана метка *label*, выполняется переход к концу сценария. Команда **t** подобна оператору выбора **case** языка

программирования C или сценариев интерпретатора. Выполняется поочередная проверка всех случаев (*cases*), и если результат положительный, работа оператора прекращается. **t**

Пример

Предположим, требуется заполнить пустые поля базы данных. Имеются следующие записи:

```
ID: 1  Name: greg  Rate: 45
ID: 2  Name: dale
ID: 3
```

А вот что необходимо получить:

```
ID: 1  Name: greg  Rate: 45  Phone: ??
ID: 2  Name: dale  Rate: ??  Phone: ??
ID: 3  Name: ????  Rate: ??  Phone: ??
```

Для этого следует выяснить количество уже заполненных полей. Посмотрите на сценарий, с помощью которого решается эта задача (поля разделены табуляцией):

```
/ID/{
s/ID: .* Name: .* Rate .*/&   Phone: ??/p
t
s/ID: .* Name: .*/&    Rate: ??   Phone: ??/p
t
s/ID: .*/&   Name: ??    Rate: ??   Phone: ??/p
}
```

[*address1*[,*address2*]]w *file* **w**

Добавить содержимое пространства шаблонов в файл *file*. Это действие производится при выполнении команды, а не при выводе строк результата. Инструкцию и имя файла должен разделять только один пробел. Команда создает файл, если он еще не существует; а если файл существует, то его содержание будет перезаписываться при каждом выполнении сценария. Множественные команды записи, выводом для которых служит один и тот же файл, добавляют данные в конец файла.

Пример

```
# Сохранить блоки tbl и eqn в файле:
/^\.TS/,/^\.TE/w troff_stuff
/^\.EQ/,/^\.EN/w troff_stuff
```

x	[*address1*[,*address2*]]x Обмен содержимого пространства шаблонов и специального буфера. См. пример к описанию команды **h**.

y	[*address1*[,*address2*]]y /*abc*/*xyz*/
	
	Преобразование символов. Вхождения символов *a*, *b* и *c* заменяются, соответственно, на *x*, *y* и *z*.
	
	Пример
	
	`# Изменить “item 1, 2, 3…” на “Item A, B, C …”`
	`/^item [1-9]/y/123456789/ABCDEFGHI/`

Глава 13

Язык сценариев gawk

Язык сценариев **gawk** является GNU-версией **awk** – мощного инструмента для обработки текстовых файлов при помощи шаблонов. Файлы могут состоять из записей фиксированной или переменной длины. Границы записей обозначаются разделителем (по умолчанию это символ новой строки). **gawk** может использоваться как из командной строки, так и в **gawk**-сценариях. Обычно эту программу можно вызвать, введя команду **awk** или **gawk** в командной строке интерпретатора.

В этой главе рассмотрены следующие темы:

- Обзор возможностей
- Синтаксис командной строки
- Шаблоны и процедуры
- Системные переменные **gawk**
- Операторы
- Работа с переменными и массивами
- Перечень команд **awk** по группам
- Алфавитный перечень команд

Более подробную информацию по этим темам можно найти во втором издании книги Дэйла Дауэрти (Dale Dougherty) и Арнольда Роббинса (Arnold Robbins) «sed & awk» издательства O'Reilly.

Обзор возможностей

Язык сценариев **gawk** позволяет:

- С легкостью обрабатывать текстовый файл, как если бы он состоял из полей и записей текстовой базы данных
- Использовать переменные для внесения изменений в базу данных
- Выполнять команды интерпретатора из сценария
- Выполнять арифметические и строковые операции
- Использовать конструкции программирования, такие как циклы и условные операторы
- Определять пользовательские функции
- Обрабатывать вывод команд интерпретатора
- Обрабатывать аргументы командной строки постепенно
- Отображать оформленные отчеты

Синтаксис командной строки

gawk можно вызвать двумя способами:

gawk [*options*] *'script' var=value file(s)*
gawk [*options*] **−f** *scriptfile var=value file(s)*

Вы можете задать все команды сценария прямо в командной строке или сохранить сценарий в файле *scriptfile* и указать его при помощи параметра **−f**. Допустимы множественные параметры **−f**. В случае указания нескольких файлов их тексты объединяются. Такая возможность полезна для включения библиотек.

gawk работает с указанными исходными файлами (*files*). Если имена файлов опущены (или задан параметр «−»), выполняется чтение со стандартного ввода.

В командной строке переменным могут присваиваться значения. Значение (*value*) может быть строкой, переменной интерпретатора ($*name*) или командной подстановкой (`*cmd*`), но значение переменной становится доступным только после чтения первой строки ввода (т. е. после оператора **BEGIN**).

Чтобы вывести, к примеру, первые три поля файла паролей (поля разделяются двоеточием), можно использовать параметр **−F** для задания разделителя:

```
gawk -F: '{print $1; print $2; print $3}' /etc/password
```

Ниже в разделе «Шаблоны и процедуры» приводятся многочисленные примеры.

Параметры

Все параметры доступны как в традиционном однобуквенном варианте POSIX, так и в GNU-стиле (длинные имена). Вот некоторые из них:

–– Все последующие строки являются командами или именами файлов, а не параметрами.

–f *scriptfile*, **––file=***scriptfile*

Читать команды **gawk** из файла *scriptfile*.

–v *var=value*, **––assign=***var=value*

Присвоить переменной *var* значение *value*. Параметр позволяет инициализировать переменные до начала работы сценария.

–F*c*, **––field-separator=***c*

Считать символ *c* разделителем полей. Действие параметра идентично инициализации переменной **FS**. Параметр *c* может быть регулярным выражением. Каждая строка или запись разделяется на поля пробелами и символами табуляции или другими символами, которые могут определяться пользователем. Поля записи адресуются переменным **$1**, **$2**, ..., $*n*. Переменная **$0** содержит всю запись.

–W *option*

Параметр **–W** специфичен для **gawk** и отсутствует в **awk**. Альтернативный способ задания параметра: **––***option* (например, **––compat**). Аргумент параметра (*option*) может быть одним из следующих:

compat

Идентично **traditional**.

copyleft

Отобразить информацию по лицензии General Public License и завершить работу.

copyright

Отобразить информацию по вопросам обладания правами и завершить работу.

help

Отобразить справку по синтаксису и перечень параметров, а затем завершить работу.

lint

Выдавать предупреждение при использовании команд, которые могут оказаться непереносимыми на другие версии **awk**, либо команд, которые **gawk** считает неясными.

lint-old

Аналогично **lint**, но выполняется проверка соответствия более старым версиям **awk**.

posix

Команды точно соответствуют стандарту POSIX. Кроме того, игнорируются escape-последовательности вида *x*, операторы ** и **=.

re-interval

Разрешить использование диапазонов вида {*n*,*m*} в регулярных выражениях.

source=*script*

Считать последовательность *script* сценарием **gawk**. Имеет сходство с аргументом '*script*', но **source** дает возможность комбинировать команды из файлов (задаваемых параметром **–f** командной строки) с командами, задаваемыми в командной строке.

traditional

Работа в полном соответствии с традиционным (не-GNU) **awk**.

usage

Идентично **help**.

version

Отобразить информацию о номере версии и завершить работу.

Шаблоны и процедуры

Сценарий **gawk** состоит из шаблонов и процедур:

pattern {*procedure*}

Оба элемента являются необязательными. Если отсутствует шаблон (*pattern*), процедура (*procedure*) применяется ко всем строкам. Если отсутствует процедура, отображается строка, соответствующая шаблону. По умолчанию каждая строка ввода считается записью, но можно переопределить разделитель записей через переменную **RS**.

Шаблоны

Шаблонами считаются конструкции вида:

/regular expression/
relational expression
pattern-matching expression
pattern,pattern
BEGIN
END

Существуют некоторые правила, применяемые к шаблонам:

- Выражение может состоять из строк, заключенных в кавычки, чисел, операторов, функций, определенных переменных или любых предопределенных переменных, описанных далее в разделе «Системные переменные gawk».

- Регулярные выражения используют расширенный набор метасимволов и описаны в главе 9 «Шаблоны и поиск».
- Кроме того, символы ^ и $ могут использоваться для привязки к началу или концу поля (а не записи в целом), соответственно.
- В выражениях отношения (*relational expressions*) используются операторы отношений, описанные далее в разделе «Операторы». Сравнивать можно и строки и числа. Например, выражение **$2 > $1** выбирает строки, в которых второе поле длиннее первого.
- В шаблонах поиска (*pattern-matching expressions*) используются операторы ~ (совпадение с шаблоном) и **!~** (несовпадение с шаблоном). См. раздел «Операторы» далее в этой главе.
- Шаблон **BEGIN** позволяет определить процедуры, которые выполняются до обработки первой строки ввода (как правило, эта возможность используется для установки значений переменных).
- Шаблон **END** позволяет определить процедуры, которые выполняются после чтения последней записи ввода.
- Если в сценарии присутствуют несколько шаблонов **BEGIN** и **END**, то соответствующие им действия выполняются в порядке следования шаблонов.
- Шаблон вида *pattern,pattern* задает диапазон строк. В этом случае **BEGIN** или **END** нельзя задавать в качестве шаблонов.

Шаблоны, за исключением **BEGIN** и **END**, можно комбинировать с помощью логических операторов || (ИЛИ), && (И) и **!** (НЕ).

В дополнение к регулярным выражениям GNU-**awk** поддерживает и списки символов стандарта POSIX, что позволяет производить поиск по не-ASCII символам языков, отличных от английского. Такие списки задаются диапазонами символов, заключаемыми в квадратные скобки **[ ]**. Например, **[[:lower:]]** является эквивалентом **[a-z]** в английском языке. Таблица для списков символов POSIX приводится в главе 9.

Процедуры

Процедура состоит из одной или более команд, функций, присваиваний переменных, разделяемых точкой с запятой или символом новой строки. Процедура заключается в фигурные скобки. Команды делятся на четыре группы:

- Инициализация переменных и массивов
- Команды отображения
- Встроенные функции
- Команды управления обработкой

Простые примеры комбинаций шаблон–процедура

1. Отобразить первое поле каждой строки (шаблон не задан):

   ```
   { print $1 }
   ```

2. Отобразить все строки, содержащие «Linux»:

   ```
   /Linux/
   ```

3. Отобразить первое поле каждой строки, содержащей «Linux»:

   ```
   /Linux/ { print $1 }
   ```

4. Отобразить записи, содержащие более двух полей:

   ```
   NF > 2
   ```

5. Считать разделителями записей пустую строку, т. е. объединять идущие подряд строки в одну запись:

   ```
   BEGIN { FS = "\n"; RS = "" }
   ```

6. Отобразить поля 2 и 3 в обратном порядке, но только для строк, первое поле которых является строкой «URGENT»:

   ```
   $1 ~ /URGENT/ { print $3, $2 }
   ```

7. Подсчитать и отобразить количество вхождений строки «ERR»:

   ```
   /ERR/ { ++x } END { print x }
   ```

8. Сложить числа второй колонки таблицы и отобразить сумму:

   ```
   { total += $2 }; END { print "column total is", total }
   ```

9. Отобразить строки, длина которых меньше 20 символов:

   ```
   length() < 20
   ```

10. Отобразить все строки, начинающиеся с подстроки «Name:» и содержащие ровно семь полей:

    ```
    NF == 7 && /^Name:/
    ```

11. Изменить порядок следования полей на противоположный:

    ```
    { for (i = NF; i >= 1; i--) print $i }
    ```

Системные переменные gawk

Переменная	Описание
$*n*	Поле текущей записи, имеющее порядковый номер *n*; поля разделяются символом **FS**
$0	Вся запись
ARGC	Количество аргументов командной строки

Переменная	Описание
ARGIND	Порядковый номер текущего файла в командной строке (начиная с 0)
ARGV	Массив, содержащий аргументы командной строки
CONVFMT	Формат преобразования для чисел (по умолчанию %**.6g**)
ENVIRON	Ассоциативный массив переменных окружения
ERRNO	Описание последней системной ошибки
FIELDWIDTHS	Список длин полей (разделенный пробелами)
FILENAME	Имя текущего файла
FNR	Аналогично **NR**, но относительно текущего файла
FS	Разделитель полей (любой пробельный символ по умолчанию; пустая строка разделяется на отдельные символы)
IGNORECASE	Если значение переменной – истина (true), то поиск не чувствителен к регистру
NF	Количество полей в текущей записи
NR	Номер текущей записи
OFMT	Формат отображения чисел (по умолчанию %**.6g**)
OFS	Разделитель полей при отображении (пустой по умолчанию)
ORS	Разделитель записей при отображении (новая строка по умолчанию)
RLENGTH	Длина строки, найденной функцией **match**
RS	Разделитель записей (по умолчанию – символ новой строки)
RSTART	Адрес первого символа соответствия шаблону в строке, найденной функцией **match**
SUBSEP	Символ-разделитель индексов массивов (по умолчанию \034)

Операторы

В приводимой далее таблице перечислены операторы **gawk** в порядке возрастания приоритета:

Символы	Значение
`= += -= *= /= %= ^= **=`	Присваивание
`?:`	Условный оператор C
`||`	Логическое ИЛИ
`&&`	Логическое И

Символы	Значение
~ !~	Соответствие регулярному выражению и отрицание соответствия
< <= > >= != ==	Операторы отношений
(пустой)	Конкатенация
+ -	Сложение, вычитание
* / %	Умножение, деление, остаток от деления
+ - !	Унарный плюс и минус, логическое отрицание
^ **	Возведение в степень
++ --	Инкремент и декремент переменной; префиксная или постфиксная запись
$	Ссылка на поле
in	Принадлежность массиву (см. описание команды **for**)

Работа с переменными и массивами

Переменным можно присваивать значения при помощи символа «=»:

```
FS = ","
```

В правой части присваивания может находиться выражение, содержащее операторы +, –, / и % (остаток от деления).

Массивы создаются при помощи функции **split** (см. раздел «Алфавитный перечень команд») или непосредственно при инициализации. Элементы массивов адресуются индексами (*array*[1]) или именами. Например, чтобы подсчитать количество соответствий шаблону *pattern*, можно использовать следующий сценарий:

```
/pattern/ { array["/pattern/"]++ }
END { print array["/pattern/"] }
```

В **gawk** переменные необязательно объявлять перед их использованием и нет необходимости задавать размерность массива заранее; память выделяется при первом обращении к индексу массива или переменной. Все переменные хранятся в строковом формате, но могут использоваться и как строки и как числа. Тип переменной в **gawk** определяется по контексту сценария, но пользователь может задать тип принудительно. Для того чтобы переменная трактовалась как строка, добавьте к ней пустую строку:

```
var ""
```

Для того чтобы переменная трактовалась как число, добавьте к ней нуль:

```
var + 0
```

Перечень команд awk по группам

Команды **gawk** можно классифицировать следующим образом:

Арифметические функции	Строковые функции	Операторы управления	Обработка ввода/вывода	Функции времени	Прочие
atan2	gensub	break	close	strftime	delete
cos	gsub	continue	fflush	systime	function
exp	index	do/while	getline		system
int	length	exit	next		
log	match	for	nextfile		
rand	split	if	print		
sin	sub	return	printf		
sqrt	substr		sprintf		
srand	tolower		while		
	toupper				

Алфавитный перечень команд

В следующий перечень включены все доступные в Linux операторы и функции **gawk**.

atan2 (y, x) Вернуть арктангенс y/x в радианах.	**atan2**
break Прекратить выполнение цикла **while** или **for**.	**break**
close (*filename-expr*) **close** (*command-expr*) Закрыть канал или файл, из которого выполняется чтение по команде **getline**; в качестве аргумента принимается то же выражение, что использовалось для открытия файла или канала.	**close**
continue Перейти к следующей итерации цикла **while** или **for**, не дожидаясь завершения текущей.	**continue**

cos	**cos** (*x*) Вернуть косинус *x;* угол задается в радианах.
delete	**delete** *array*[*element*] **delete** *array* Удалить элемент (*element*) массива *array*. Если не указан индекс, удаляются все элементы массива.
do	**do** *body* **while** (*expr*) Оператор цикла. Операторы тела цикла (*body*) выполняются до тех пор, пока выражение *expr* истинно.
exit	**exit** Выход из сценария. Новые данные не читаются, и более не выполняются никакие операторы, кроме процедур шаблона **END**.
exp	**exp** (*arg*) Вернуть e в степени *arg* (операция, обратная **log**).
fflush	**fflush** (*filename)* Запись результата в указанный файл; по умолчанию – на стандартный вывод.
for	**for** (*i=lower; i<=upper; i++*) *command* Пока значение переменной *i* находится в диапазоне между *lower* и *upper*, выполнять команду *command*. Последовательность команд должна заключаться в фигурные скобки. Можно использовать <= и любой другой оператор отношения. Для увеличения и уменьшения переменной цикла можно использовать ++ или −−.

for (*item* **in** *array*)
command

Для каждого элемента (*item*) ассоциативного массива *array* выполнить команду *command*. Последовательность команд заключается в фигурные скобки. Каждый из элементов массива может адресоваться как *array*[*item*]. Элементы массивов в **gawk** хранятся таким образом, чтобы доступ к каждому из них занимал примерно равное время. Порядок может оказаться неприемлемым, поэтому для получения результатов в отсортированном виде необходимо через конвейер передать вывод команде **sort**.

for

---

function *name*(*parameter-list*) {
statements
}

Определить пользовательскую функцию *name*, состоящую из **gawk**-операторов (*statements*) и работающую со списком указанных параметров.

function

---

gensub (*r*, *s*, *n*, *t*)

Подставить *s* вместо *n*-ного соответствия регулярному выражению *r* в строке *t*. Строка *t* не изменяется, но **gensub** возвращает ее измененную копию в качестве результата. Если *n* принимает значение «g» или «G», изменить все соответствия. Если параметр *t* не задан, в качестве строки используется **$0**.

gensub

---

getline [*var*] [<*file*]
command | **getline** [*var*]

В первом случае данные читаются из файла (*file*) или из следующего по списку аргументов файла, а во втором – из вывода команды *command*. И в том и в другом случае **getline** за один раз считывает одну строку. Прочитанная строка записывается в переменную **$0** и разбивается на поля, устанавливая переменные **NF**, **NR** и **FNR**. Если задана переменная *var*, результат чтения присваивается этой переменной, а значения **$0** и **NF** не изменяются. Таким образом, если строка прочитана в переменную, текущая строка не изменяется. **getline**, по существу, является функцией и возвращает значения: 1 – при успешном чтении записи, 0 – по достижении символа *EOF* и –1, если чтение не было выполнено по какой-либо другой причине.

getline

gsub	**gsub** (*r, s, t*) Подставить *s* вместо всех соответствий регулярному выражению *r* в строке *t*. Вернуть количество подстановок. Если параметр *t* не задан, в качестве строки используется **$0**.
if	**if** (*condition*) *command1* [**else** *command2*] Если условие (*condition*) истинно, выполнить команду *command1*, иначе выполнить команду *command2*. Условие может быть выражением, включающим любые из операторов отношения (<, <=, ==, !=, >=, >), а также оператором соответствия шаблону ~. Последовательность команд должна заключаться в фигурные скобки. **Пример** В следующей строке сценария определяется, начинается ли первое слово каждой строки с буквы A или a. `if ($1 ~ /[Aa]*/)` *...Начинается с A или a*
index	**index** (*substr*,*str*) Вернуть позицию начала подстроки *substr* в строке *str*. Если подстрока не содержится в строке, возвращается значение 0.
int	**int** (*arg*) Вернуть целую часть аргумента *arg*.
length	**length** (*arg*) Вернуть длину аргумента *arg*. Если аргумент опущен, в качестве строки используется значение переменной **$0**.
log	**log** (*arg*) Вернуть натуральный логарифм аргумента (обратно **exp**).
match	**match** (*s, r*) Вернуть позицию начала первого соответствия регулярному выражению *r* в строке *s* либо 0, если соответствие не найдено.

Устанавливаются значения переменных **RSTART** и **RLENGTH**.	**match**
next Прочитать следующую строку данных ввода и начать новый цикл операторов шаблон/процедура.	**next**
nextfile Перейти к обработке следующего файла из перечисленных в командной строке **gawk** и начать новый цикл операторов шаблон/процедура.	**nextfile**
print [*args*] [*destination*] Отобразить аргументы (*args*). Последовательности символов должны заключаться в кавычки. Печать происходит в порядке перечисления аргументов. При печати разделенные запятыми аргументы разделяются символом, определяемым переменной **OFS**. Если аргументы разделены пробелами, при печати они выводятся слитно. Параметр *destination* является перенаправлением в стиле интерпретатора либо конвейерным выражением (например, это может быть >*file*), перенаправляющим стандартный вывод.	**print**
printf [*format* [, *expressions*]] Печать с форматированием. Выражения или переменные оформляются в соответствии с указаниями параметра *format*. Количество выражений (*expressions*) должно соответствовать количеству аргументов в параметре *format*. Параметр *format* следует соглашениям языка С для функции **printf**. Вот некоторые из наиболее часто используемых элементов формата: %**s** Строка. %**d** Десятичное число. %*n*.*m***f** Число с плавающей точкой; *n* – общее количество цифр числа, *m* – количество цифр после десятичной запятой. %[–]*nc* *n* определяет минимальную длину поля для типа формата *c*, параметр «–» производит выравнивание отобра-	**printf**

printf	жаемого значения по левому краю поля. По умолчанию выполняется выравнивание по правому краю.
	Может быть задана ширина полей выражения. К примеру, формат **%3.2f** ограничивает число с плавающей точкой тремя цифрами, из которых две находятся после запятой.
	Аргумент *format* также может содержать escape-последовательности, из которых чаще всего используются **\n** (новая строка) и **\t** (символ табуляции). Пробелы и буквальный текст могут содержаться в формате, если он полностью заключен в кавычки. Если необходимо напечатать несколько выражений, следует задать соответствующее количество элементов формата.
	Пример
	Использование сценария:
	`{printf ("Сумма чисел в строке %s равна %d.\n", NR, $1+$2)}`
	со следующей строкой ввода:
	`5  5`
	приведет к такому результату:
	`Сумма чисел в строке 1 равна 10.`

rand	**rand ()** Сгенерировать случайное число из интервала от 0 до 1. Функция возвращает при каждом выполнении сценария и последовательных вызовах одинаковые результаты, если только генератор случайных чисел не был инициализирован с помощью функции **srand**.
return	**return** [*expr*] Применяется в пользовательских функциях для возврата в основной сценарий; возвращается значение *expr*.
sin	**sin** (*x*) Вернуть синус *x*; угол задается в радианах.
split	**split** (*string*, *array*[,*sep*]) Записать поля строки *string* в элементы массива *array*[1], ..., *array*[*n*]. Поля строки считаются разделенными символом *sep*.

Если разделитель не задан, используется значение переменной **FS**. Если разделитель *sep* является пустой строкой, полем строки считается каждый символ. Функция возвращает количество созданных элементов массива.	**split**
sprintf [*format* [, *expression*(*s*)]] Вернуть значение одного или нескольких выражений *expression*(*s*), используя указанный формат *format* (см. **printf**). Данные форматируются, но не отображаются.	**sprintf**
sqrt (*arg*) Вернуть квадратный корень аргумента.	**sqrt**
srand (*expr*) Использовать выражение *expr* для инициализации генератора случайных чисел. По умолчанию используется текущее время.	**srand**
strftime ([*format* [, *timestamp*]]) Вернуть строку, отображающую время (*timestamp*) в соответствии с форматом. Параметр *timestamp* задается как количество секунд, прошедших с полуночи 1 января 1970 года. Строка *format* схожа со строками формата **sprintf** (пример приведен в описании **systime**). Если параметр *timestamp* не указан, используется значение текущего времени. По умолчанию параметр *format* равен соответствующему аргументу команды **date**.	**strftime**
sub (*r*, *s*, *t*) Подставить *s* вместо первого соответствия регулярному выражению *r* в строке *t*. Если соответствие не найдено, возвращается 0, иначе – 1. Если опущен параметр *t*, в качестве строки используется переменная **$0**.	**sub**
substr (*string*, *m*[, *n*]) Вернуть подстроку строки *string*, начиная с символа *m*. *n* определяет длину возвращаемой подстроки. Если *n* не указано, возвращаются все символы до конца строки.	**substr**

system

system (*command*)

Выполнить указанную команду в интерпретаторе и вернуть код завершения. Как правило, код завершения определяет результат работы программы: обычно 1 означает успешное завершение, 0 – простое завершение, а –1 – непредвиденный сбой. Вывод команды *command* не доступен для обработки в сценарии **gawk**.

systime

systime ()

Вернуть количество секунд, прошедших с полуночи первого января 1970 года по UTC.

Пример

Записать время начала и конца обработки данных программой:

```
BEGIN {
        now = systime()
        mesg = strftime("Started at %m/%d/%Y  %H:%M:%S", now)
        print mesg
}
process data . . .
END {
       now = systime()
       mesg = strftime("Ended at %m/%d/%Y  %H:%M:%S", now)
       print mesg
}
```

tolower

tolower (*str*)

Перевести все прописные символы строки *str* в нижний регистр и вернуть измененную строку.

toupper

toupper (*str*)

Перевести все строчные символы строки *str* в верхний регистр и вернуть измененную строку.

while

while (*condition*)
command

Выполнять команду *command*, пока истинно условие *condition* (допустимые условия см. в описании оператора **if**). Последовательность команд должна заключаться в фигурные скобки.

Глава 14

CVS и RCS

CVS, а также более старая система, RCS, обеспечивают *контроль версий* (или *контроль обновлений*) – метод сохранения информации о развитии проекта, позволяющий в любой момент получать более старые версии файлов, отслеживать изменения, а главное – координировать усилия команды разработчиков.

Основные понятия

Система контроля версий RCS (Revision Control System) работает в пределах одного каталога. Для поддержки крупных проектов, в которых задействована целая иерархия каталогов, созданы две новые концепции CVS – *репозиторий* (*repository*) и *песочница* (*sandbox*).

Репозиторий (называемый также *архивом*) является централизованным хранилищем данных, которое управляется системой контроля версий и администратором репозитория. Репозиторий содержит файлы проекта, а также информацию, необходимую для воссоздания произвольной версии любого файла. Администратор создает и управляет данными в репозитории, используя процедуры и команды, описанные далее в разделе «Справочник администратора CVS».

Песочница (также называемая *рабочим каталогом*) содержит копии файлов из репозитория. Непосредственно разработка происходит в рабочих каталогах, и любое количество рабочих каталогов может быть создано из единственного репозитория. Рабочие каталоги не зависимы друг от друга и могут содержать файлы, датированные разными стадиями разработки одного и того же проекта. Пользователи могут создавать и контролировать свои рабочие каталоги, используя процедуры и команды, описанные далее в этой главе в разделе «Справочник пользователя CVS».

В типичной ситуации взаимодействия с системой контроля версий разработчик извлекает из репозитория наиболее свежие файлы, вносит изменения, проверяет результат, а затем, когда результат оказывается удовлетворительным, помещает измененные файлы в репозиторий.

Блокировка и слияние

В некоторых системах, включая и RCS, используется *механизм блокировки* (*locking model*), призванный координировать работу нескольких разработчиков путем последовательного внесения изменений в файл. Прежде чем внести изменения в файл, разработчик должен не только получить копию файла, но еще запросить и получить блокировку файла у системы. Блокировка нужна для предотвращения (на самом деле для предупреждения) случаев одновременной работы нескольких разработчиков с одним и тем же файлом. После помещения изменений в репозиторий разработчик снимает блокировку с файла, позволяя, таким образом, другим разработчикам получить к нему доступ.

Данная модель блокировки является пессимистичной: предполагается, что *следует* избегать конфликтов. Конфликты предотвращаются посредством сериализации изменений файла. Но довольно обременительно блокировать файлы для редактирования при поиске ошибок. Зачастую разработчики работают в обход механизма блокировки, что является прямой дорогой к неприятностям.

В отличие от RCS и SCCS, в CVS используется *механизм слияния* (*merging model*), позволяющий всем и каждому иметь доступ к файлам в любое время, способствуя параллельной разработке. Механизм слияния является оптимистичным: подразумевается, что конфликтные ситуации возникают нечасто, а когда возникают, то их, *как правило*, нетрудно разрешить.

В CVS доступна работа с механизмом блокировки посредством параметров **–L** и **–l** команды **admin**. Помимо этого в CVS существуют специальные команды (**edit** и **watch**) для поддержки дополнительной координации разработки. В CVS используется внутренняя блокировка, позволяющая предотвратить повреждения при одновременном доступе к репозиторию нескольких человек, но этот механизм не имеет ничего общего с механизмом пользовательской блокировки, описанным выше.

Конфликты и слияние

В ситуации, когда два разработчика одновременно пытаются поместить в репозиторий обновленные версии одного и того же файла, CVS автоматически откладывает операцию для одного из них. Второй разработчик выполняет в этом случае команду **cvs update**, которая вносит изменения, сделанные первым разработчиком, в локальную версию файла. Во многих случаях изменения затрагивают различные области файла, поэтому слияние происходит успешно. Однако если оба разработчика внесли изменения в одну и ту же область файла, второй из них будет вынужден разрешить противоречия перед

помещением файла в репозиторий. Разрешение противоречий означает изучение области (или областей) файла, в которой возникли проблемы, и выбор среди нескольких версий изменений или внесение изменений, разрешающих конфликт.

CVS может определить только чисто текстовый конфликт, но собственно разрешение противоречий связано с сохранением логической непротиворечивости проекта в целом. Поэтому разрешение противоречий иногда связано с редактированием других файлов, помимо файла, непосредственно содержащего конфликт версий.

Например, если один из разработчиков добавляет параметр в определение функции, может понадобиться изменить все вызовы этой функции для передачи дополнительного параметра. Это логический конфликт, поэтому его нахождение и разрешение является задачей разработчиков (которые могут взять себе в помощь такие инструменты, как компилятор и отладчик); CVS на эту проблему не обратит внимания.

В любой ситуации слияния, независимо от того, возникли конфликты или нет, разработчик, помещающий файлы вторым, может по желанию произвести дополнительную проверку окончательной версии проекта, поскольку она изменилась со времени исходного обновления версии. Если все в порядке, версию можно поместить в репозиторий.

Идентификаторы версий

CVS отслеживает версии файлов по номерам обновлений, которые могут использоваться для извлечения определенной версии файла из репозитория. Кроме того, можно создавать символьные имена (symbolic tags), или метки, для того чтобы обращаться к группе файлов (либо проекту в целом) по единственному идентификатору даже тогда, когда номера версий файлов различны (в большинстве случаев они различны). Эта возможность часто используется для отслеживания выпускаемых версий и других важных промежуточных остановок в проекте.

Например, символьное имя `hello-1_0` может быть связано с версией 1.3 файла *hello.c* и версией 1.1 файла сборки *Makefile* (символьные имена создаются командами **tag** и **rtag**).

Ветвление

Простейшей формой разработки является *линейная* разработка, в которой существует последовательность версий файла, каждая из которых прямо наследует предыдущую. Во многих проектах вполне достаточно строго линейного процесса разработки, но в крупных проектах (крупных по количеству файлов, количеству разработчиков и/или размеру сообщества пользователей) зачастую требуется более сложное управление и дополнительные возможности. Иногда желательно произвести пробную разработку, не прерывая при этом основную линию развития проекта. В других случаях ошибки в текущей выпущенной версии должны быть исправлены, в то время как

работа над следующей версией идет своим чередом. И в том и в другом случаях решением является создание ветви (*развилки*) в соответствующей точке развития проекта. Если в какой-то момент потребуется интегрировать некоторые или все изменения ветви в основную линию разработки, они могут быть слиты (*объединены*).

Ветвление выполняется командой **tag –b**, а объединение – командой **update –j**.

Инструмент CVS

В данном разделе представлены общие сведения о CVS.

Формат команд CVS

Команды CVS имеют следующий вид:

```
cvs global_options command command_options
```

Например, приведем простую последовательность команд, в которой отображены оба вида параметров в контексте создания репозитория, импортирования существующих файлов и выполнения нескольких распространенных операций над ними:

```
user@localhost$ cvs -d /usr/local/cvsrep init
user@localhost$ cd ~/work/hello
user@localhost$ cvs -d /usr/local/cvsrep import -m 'Import' hello vendor start
user@localhost$ cd ..
user@localhost$ mv hello hello.bak
user@localhost$ cvs -d /usr/local/cvsrep checkout hello
user@localhost$ cd hello
user@localhost$ vi hello
user@localhost$ cvs commit -m 'Fixed a typo'
user@localhost$ cvs tag hello-1_0
user@localhost$ cvs remove -f Makefile
user@localhost$ cvs commit -m 'Removed old Makefile'
user@localhost$ cvs upd -r hello-1_0
user@localhost$ cvs upd -A
```

Некоторые глобальные параметры являются общими для команд администратора и пользователей, а некоторые являются специфичными для каждой из этих категорий. Общие глобальные параметры описаны в следующем разделе, а специфичные для пользователей и администратора – в разделах «Справочник пользователя CVS» и «Справочник администратора CVS», соответственно.

Общие глобальные параметры

В табл. 14.1 приведены глобальные параметры, которые могут использоваться как в командах пользователей, так и в командах администратора.

Таблица 14.1. Общие глобальные параметры

Параметр	Описание
−b *bindir*	Расположение внешних программ RCS. Данный параметр вышел из употребления в CVS версий выше 1.9.18
−T *tempdir*	Абсолютный путь для создания временных файлов. Перезаписывает значение $TMPDIR
−v **−version**	Вывести информацию о версии и авторских правах

Возможные проблемы

Данный раздел посвящен некоторым аспектам CVS, которые иногда могут вызвать путаницу.

Работа с файлами в CVS

Хотя каталоги поддерживаются, версии для них не отслеживаются, как это делается для обычных файлов. Это особенно важно на ранних стадиях проекта, когда структура еще может быть нечеткой. Кроме того, если проект подвергается заметным изменениям, структура, скорее всего, изменится. Смотрите далее раздел «Тонкая работа с репозиторием».

Работа с текстом в CVS

Не существует эквивалента программы **diff** для двоичных файлов, хотя поддержки CVS таких файлов обычно бывает достаточно. Используйте команду **admin −kb**, чтобы указать CVS на двоичный файл.

Работа со строками в CVS

Перемещение сегмента кода из одной области файла в другую рассматривается как удаление (из первоначальной области) и не связанное с этим удалением добавление (в новой области).

CVS не в курсе синтаксиса

Если речь идет о CVS, то небольшие изменения в форматировании эквивалентны масштабному изменению логики в том же диапазоне строк.

Анахронизмы RCS

CVS изначально являлся надстройкой RCS, но в настоящее время вся функциональность RCS является внутренней для CVS. Останки RCS можно наблюдать в имени переменной окружения $RCSBIN и описании параметра **−b**, которые теперь вышли из употребления.

Справочник администратора CVS

В данном разделе подробно рассматриваются создание и настройка репозиториев, а также выполнение различных задач администрирования CVS. На

одном компьютере могут выполняться несколько копий сервера CVS, каждая из которых может обслуживать несколько репозиториев.

Создание репозитория

Выберите каталог, который будет содержать файлы репозитория (в приводимых ниже примерах используется каталог */usr/local/cvsrep*). Используйте команду **init** для инициализации репозитория. При этом либо предварительно установите переменную окружения $CVSROOT:

```
user@localhost$ export CVSROOT=/usr/local/cvsrep
user@localhost$ cvs init
```

либо воспользуйтесь параметром **–d**, чтобы указать местоположение репозитория:

```
user@localhost$ cvs -d /usr/local/cvsrep init
```

Информацию по импортированию существующего кода смотрите в разделе «Справочник пользователя CVS», особенно в описании команд **import** и **add**.

Настройка сервера паролей

Для того чтобы предоставить доступ к репозиторию с других компьютеров, необходимо настроить `pserver`, выполнив следующие действия с привилегиями суперпользователя:

- Убедитесь, что в файле */etc/services* есть запись, похожая на следующую:

  ```
  cvspserver 2401/tcp
  ```

- Если вы не используете `tcpwrappers`, поместите в файл */etc/inetd.conf* строку следующего вида:

  ```
  cvspserver stream tcp nowait root /usr/bin/cvs cvs
  --allow-root=/usr/local/cvsroot pserver
  ```

- Либо, если вы *используете* `tcpwrappers`, воспользуйтесь такой строкой:

  ```
  cvspserver stream tcp nowait root /usr/sbin/tcpd /usr/bin/cvs
  --allow-root=/usr/local/cvsroot pserver
  ```

- После этого необходимо перезапустить **inetd** (либо послать соответствующий сигнал, который заставит сервер перечитать файл *inetd.conf*).

Вопросы безопасности

Ниже перечислены вопросы, касающиеся безопасности, которые должны приниматься во внимание при работе с CVS:

- Содержимое файлов будет передаваться в открытом виде по сети при использовании сервера паролей `pserver` и **rsh**. При использовании `pserver` пароли также передаются в открытом виде.

- При использовании локального репозитория (т. е. когда CVS не работает по модели клиент-сервер) разработчики должны иметь право записи в репозиторий, и это означает, что они могут изменять его неблагоприятным образом.
- Перед сменой пользовательского идентификатора сервер CVS краткое время работает с привилегиями суперпользователя.
- Владельцу необходимо предотвращать прочтение файла ~/*.cvspass* другими пользователями, чтобы не допустить распространения пароля.
- Пользователь, имеющий право вносить изменения в файлы модуля *CVSROOT*, может выполнять произвольные программы.
- Некоторые из параметров команды **admin** являются очень опасными, поэтому рекомендуется ограничить ее использование. Это может быть сделано посредством создания группы пользователей `cvsadmin`. Если эта группа существует, только пользователи, входящие в нее, имеют право выполнять команду **admin** (кроме варианта **admin –k***kflag*, доступного всем пользователям).

Структура репозитория

Репозиторий CVS реализован в виде обычного каталога со специфическим содержимым, которое и рассматривается в текущем разделе.

Каталог CVSROOT

Каталог *CVSROOT* содержит административные файлы репозитория; другие каталоги репозитория содержат модули. Административные файлы помимо строк с существенной информацией могут содержать (игнорируемые) пустые строки и строки комментариев. Строки комментариев начинаются с символа «#».

Некоторые административные файлы содержат маски файлов для отбора имен файлов и каталогов. Эти маски являются регулярными выражениями, аналогичными тем, что используются в редакторе GNU Emacs. В табл. 14.2 приведены наиболее часто используемые конструкции.

Таблица 14.2. Метасимволы в именах файлов

Конструкция	Описание
^	Привязка к началу строки
$	Привязка к концу строки
.	Любой символ
*	Предыдущая конструкция может повторяться ноль или более раз

CVS выполняет важные подстановки в содержимом административных файлов до интерпретации результата. Во-первых, типичный синтаксис интерпретатора, используемый для ссылки на домашний каталог, – «~/», заме-

няется полным путем к домашнему каталогу пользователя, выполняющего CVS; а *~user* – полным путем к домашнему каталогу пользователя *user*.

Помимо этого CVS предоставляет механизм, схожий по возможностям с действием механизма подстановок для переменных командного интерпретатора. Конструкции, такие как ${*variable*}, заменяются значением указанной переменной. Имена переменных начинаются с букв и могут состоять из букв, цифр и символов подчеркивания. Фигурные скобки могут быть опущены, если символ, непосредственно следующий за ссылкой на переменную, не является допустимым в имени переменной. Несмотря на то что эта конструкция выглядит точно так же, как в интерпретаторе, доступны не все переменные окружения. В табл. 14.3 перечислены встроенные переменные.

Таблица 14.3. Переменные административных файлов

Переменная	Описание
CVSEDITOR EDITOR VISUAL	Редактор, используемый CVS для создания записей журнала изменений
CVSROOT	Путь к используемому репозиторию
USER	Имя пользователя (на сервере, если используется удаленный репозиторий), который использует CVS
=*var*	Значение определенной пользователем переменной *var*. Значения этих переменных можно получить с помощью глобального параметра **–s**

Для того чтобы изменить административные файлы, извлеките модуль *CVSROOT* из репозитория, отредактируйте нужные файлы и поместите их обратно в репозиторий. Изменения должны помещаться в репозиторий, иначе поведение CVS никак не изменится.

В табл. 14.4 перечислены административные файлы и назначение каждого из них.

Таблица 14.4. Файлы модуля CVSROOT

Файл	Описание
checkoutlist	Дополнительные файлы, поддерживаемые в *CVSROOT*
commitinfo	Спецификации управления помещением файлов в репозиторий
config	Настройки, влияющие на поведение CVS
cvsignore	Маски файловых имен, игнорируемых CVS
cvswrappers	Спецификации фильтров **checkout** и **commit**
editinfo	Спецификации редакторов журнала (вышли из употребления)
history	Журнал информации команды **history**
loginfo	Настройка программ уведомления для **commit**

Таблица 14.4 (продолжение)

Файл	Описание
modules	Определения модулей
notify	Спецификации обработки уведомлений
passwd	Перечень всех пользователей и их CVS-паролей
rcsinfo	Шаблон для записей журнала
readers	Перечень пользователей, имеющих доступ только для чтения
taginfo	Спецификации обработки символьных имен (меток)
users	Альтернативные почтовые адреса пользователей для *notify*
verifymsg	Определение программы проверки корректности записи для журнала изменений
writers	Перечень пользователей, имеющих доступ в режиме чтение/запись

Поскольку файл *editinfo* вышел из употребления, рекомендуется использовать переменную окружения $EDITOR (или параметр **–e**) для указания редактора и файл *verifymsg* – для определения программы проверки.

Каждая строка файла *taginfo* содержит маску файловых имен и командную строку, выполняемую при отметке файлов с соответствующими именами.

Файл checkoutlist

При помещении измененных файлов модуля *CVSROOT* в репозиторий выводится следующее сообщение:

```
cvs commit: Rebuilding administrative file database
```

чтобы уведомить пользователя о том, что извлеченная копия в репозитории была только что обновлена и отражает теперь все внесенные изменения. Как и каталог любого другого модуля в репозитории, *CVSROOT* содержит файлы RCS (*,*v*), в которых хранятся изменения файлов. Однако для того чтобы воспользоваться этими файлами, CVS необходимо иметь копии их последних версий. Поэтому при выводе вышеприведенного сообщения происходит извлечение последних версий административных файлов.

Если вы добавили к модулю *CVSROOT* файлы (например, сценарии, вызываемые спецификациями файла *loginfo*), необходимо перечислить эти файлы в файле *checkoutlist*. Таким образом, эти файлы будут обрабатываться CVS точно так же, как и стандартные файлы модуля *CVSROOT*.

Каждая строка в этом файле состоит из имени файла и необязательного сообщения об ошибке, которое отображается в том случае, когда файл невозможно извлечь из репозитория.

Файл commitinfo

При выполнении команды **commit** CVS обращается к этому файлу, чтобы определить, необходимо ли производить какие-либо предварительные проверки файла. Каждая строка файла *commitinfo* содержит маску каталога, за которой следует имя программы, вызываемой для проверки файлов при помещении их в соответствующие каталоги.

Помимо обычных масок, существуют два специальных шаблона:

`ALL`

Если данный шаблон присутствует в файле, тогда все без исключения файлы передаются для проверки указанной программе. Затем CVS ищет шаблон, соответствующий имени каждого конкретного файла, и вызывает дополнительные проверяющие программы, если они определены.

`DEFAULT`

Если в файле присутствует данный шаблон, все файлы, не соответствующие ни одной из перечисленных масок, передаются для проверки указанной программе. Автоматическое соответствие всех файлов ключевому слову `ALL`, если таковое найдено, не считается за соответствие, когда речь идет об определении необходимости передачи файла на проверку программе, отмеченной как `DEFAULT`.

CVS создает командную строку для вызова программы проверки путем добавления полного пути к каталогу внутри репозитория и списка файлов, которые помещаются в репозиторий (это означает, что в файле *commitinfo* можно при необходимости указать несколько первых параметров проверяющей программы). Если проверяющая программа завершается с ненулевым кодом возврата, помещение файлов в репозиторий не происходит.

Программы, выполняемые по такому механизму, выполняются на сервере при использовании удаленного репозитория: Вот пример файла *commitinfo*:

```
ALL $CVSROOT/CVSROOT/commit-ALL.pl
DEFAULT $CVSROOT/CVSROOT/commit-DEFAULT.pl
CVSROOT$ $CVSROOT/CVSROOT/commit-CVSROOT.pl
```

В данном примере предполагается, что вы создадите файлы сценариев в модуле *CVSROOT* и добавите их в файл *checkoutlist*.

Файл config

Настройки репозитория определяются в административном файле *config*.

`LockDir=`*`dir`*

Указание CVS помещать файлы блокировки в указанный альтернативный каталог вместо репозитория, позволяя, таким образом, пользователям, не имеющим права записи в репозиторий (но имеющим права записи в указанный каталог *`dir`*), читать файлы из репозитория.

В версии 1.10 не поддерживаются альтернативные каталоги для файлов блокировки, и указание этого параметра вызывает сообщение об ошибке.

Более старые версии CVS (1.9 и старше) также не поддерживают эту возможность, но без сообщений об ошибках. Не смешивайте версии, поддерживающие альтернативные каталоги для файлов блокировки с теми, которые не поддерживают их, поскольку при этом исчезает смысл использования файлов блокировки как таковых.

`RCSBIN=dir`

Параметр вышел из употребления (использовался в версиях с 1.9.12 по 1.9.18). Он использовался для указания CVS каталога, в котором нужно искать программы RCS. Поскольку вся функциональность RCS теперь реализуется непосредственно CVS, параметр ни на что не влияет.

`SystemAuth=value`

CVS пытается сначала идентифицировать пользователей с помощью файла *CVSROOT/passwd*; и если это не удается, а данный параметр имеет значение `yes`, CVS пытается идентифицировать пользователей посредством системной базы данных пользователей. Этот параметр используется с применением сервера паролей. Значение по умолчанию – `yes`.

`TopLevelAdmin=value`

Если значение параметра равно `yes`, при извлечении файлов (при запуске **checkout**) создается дополнительный каталог *CVS* в каталоге верхнего уровня. Это позволяет программе-клиенту определять путь к репозиторию (см. раздел «Путь к репозиторию»). Значение по умолчанию – `no`.

Данный параметр полезен в том случае, если происходит извлечение нескольких модулей в один рабочий каталог. При установленном параметре `TopLevelAdmin` вам не придется указывать путь к репозиторию после извлечения первого модуля, поскольку эта информация будет извлекаться из каталога верхнего уровня *CVS*, создаваемого при первом извлечении.

Файл cvsignore

Административный файл *cvsignore* содержит список масок файлов, которые следует игнорировать. Аналогичным влиянием обладают файлы *.cvsignore*, которые могут существовать в рабочих каталогах CVS и домашних каталогах пользователей. В отличие от прочих административных файлов, для *cvsignore* используется синтаксис масок в **sh**-стиле, а не в стиле регулярных выражений GNU Emacs. В одной строке может быть задано несколько масок, которые разделяются пробелами (и, следовательно, сами маски не должны содержать пробелы).

В табл. 14.5 отображены наиболее часто используемые метасимволы масок в **sh**-стиле.

Таблица 14.5. Маски имен файлов для cvsignore

Конструкция	Описание
?	Произвольный символ
*	Любая последовательность (ноль или более) символов

В продолжение несоответствия стандарту прочих административных файлов, в файле *cvsignore* запрещается использовать комментарии.

Файл cvswrappers

В то время как файл *cvsignore* позволяет CVS игнорировать определенные файлы, файл *cvswrappers* позволяет определить стандартные параметры CVS для команд, которые работают с файлами. Строки в файле *cvswrappers* состоят из масок в **sh**-стиле, за которыми следует параметр **–k** (режим подстановки ключевых слов) и/или параметр **–m** (метод обновления). Доступные значения параметра **–k** приведены в табл. 14.19. Доступными значениями параметра **–m** являются `COPY` и `MERGE`.

Если указано **–m COPY**, CVS не будет пытаться производить слияние файлов. Вместо этого пользователю будут представлены конфликтующие версии файлов, и он сможет выбрать одну из них или разрешить конфликт вручную.

Например, для того чтобы указать CVS, что все файлы с расширением *.exe* являются двоичными, добавьте в файл строку:

```
*.exe -k b
```

Файл history

Если данный файл существует, CVS пополняет его записями о работе с репозиторием. Эта информация отображается по команде **cvs history**. Данный файл не предназначен для прямой записи или чтения никакими программами, кроме CVS.

Для репозитория, созданного командой **cvs init**, файл *history* создается автоматически.

Файл loginfo

Действие административного файла *loginfo* во многом сходно с действием файла *commitinfo*; этот файл может содержать специальные маски (ключевые слова) `ALL` и `DEFAULT`. Этот файл позволяет производить определенные операции с записями журнала изменений и связанной информацией.

Программы, вызываемые в процессе обработки *loginfo*, получают запись для журнала со стандартного ввода. В табл. 14.6 перечислены три кода, с помощью которых можно передавать вызываемым программам дополнительную информацию через аргументы командной строки.

Таблица 14.6. Специальные переменные loginfo

Переменная	Описание
`s`	Имя файла
`V`	Номер версии до помещения в репозиторий
`v`	Номер версии после помещения в репозиторий

Если переменная, предваряемая символом процента (%), передается в командной строке, CVS производит подстановку необходимой информации в виде списка, элементы которого разделяются пробелами, причем каждый элемент содержит информацию об одном из файлов и предваряется путем к репозиторию (как и в случае с файлом *commitinfo*). В одной командной строке не может присутствовать более одного символа процента, поэтому если необходима информация из нескольких переменных, имена переменных следует поместить в фигурные скобки: `%{…}`. В этом случае каждая специфичная для файла запись имеет одно поле на каждую переменную, а значения переменных разделяются запятыми. К примеру, код `%{sVv}` дает следующий результат:

```
/usr/local/cvsrep/hello Makefile,1.1,1.2 hello.c,1.8,1.9
```

Также может оказаться полезным посылать почтовые уведомления всякий раз, когда происходит помещение файла в репозиторий. Разработчики могут отслеживать поступающие уведомления, чтобы определить, когда необходимо извлечь самый свежий код в свой рабочий каталог. Например, предположим, что разработчик занимается подготовительной работой в своем рабочем каталоге в ожидании стабилизации и помещения в репозиторий новой библиотеки другого разработчика. Как только новая библиотека добавлена и помещена в репозиторий, отправляется почтовое уведомление, и ожидающий разработчик видит, что код готов к использованию. Поэтому он выполняет **cvs upd –d** в нужном каталоге, чтобы извлечь новый библиотечный код, и начинает интегрировать его со своей разработкой.

Такой тип уведомлений достаточно просто настраивается. Достаточно добавить следующую строку в файл *CVSROOT/loginfo*:

```
DEFAULT mail -s %s developers@company.com
```

Адрес, как правило, является адресом списка рассылки, на который подписаны все заинтересованные стороны (разработчики или те, кто непосредственно в проекте не участвуют). Если существует необходимость отправлять уведомление на несколько почтовых адресов, можно создать сценарий, который будет этим заниматься, и вызывать такой сценарий из файла *loginfo*. В качестве альтернативы можно использовать программу *log.pl*, которая входит в дистрибутив CVS (помещенная в */usr/local/src/cvs-1.10.8/contrib/log.pl*, в предположении распаковки дистрибутива в каталог */usr/local/src*). Инструкции по использованию доступны в виде комментариев внутри файла.

Файл modules

Каталоги верхнего уровня в репозитории называются *модулями*. Помимо этих физических модулей, CVS предоставляет механизм создания логических модулей при помощи административного файла *modules*. Существует три типа логических модулей:

Alias (Псевдоним)

Псевдоним модулей определяется строками вида:

```
module_name -a alias_module ...
```

Имя псевдонима модуля может использоваться в командах CVS таким же образом, как используются обычные имена модулей, следующие за параметром **–a**.

Regular (Обычный)

Обычные модули определяются строками вида:

```
module_name [options] directory file ...
```

Извлечение модуля *module_name* приводит к копированию указанных файлов из извлекаемого каталога в каталог с именем *module_name*. Промежуточные каталоги (если таковые присутствуют) не отражаются в структуре рабочего каталога.

Ampersand (Иерархический)

Иерархические модули определяются строками вида:

```
module_name [options] &other_module ...
```

Результатом извлечения такого модуля является каталог *module_name*, содержащий копии модулей *other_module*.

В табл. 14.7 перечислены параметры, которые могут использоваться при определении модулей.

Таблица 14.7. Параметры модулей

Параметр	Описание
–d *name*	Переопределяет имя рабочего каталога, заданное по умолчанию для модуля
–c *prog*	Выполнить программу *prog* при экспортировании файлов из модуля; имя модуля передается программе *prog* в качестве единственного аргумента
–i *prog*	Выполнить программу *prog* при помещении файлов модуля в репозиторий; имя каталога репозитория, в котором хранятся помещаемые файлы, передается программе *prog* в качестве единственного аргумента
–o *prog*	Выполнить программу *prog* при извлечении файлов из модуля; имя модуля передается программе *prog* в качестве единственного аргумента
–s *status*	Присвоить модулю дескриптор состояния
–t *prog*	Выполнить программу *prog* при присвоении файлам модуля символьных имен (меток) посредством **rtag**; программе *prog* передается имя модуля и метка
–u *prog*	Выполнить программу *prog* при обновлении файлов в каталоге верхнего уровня модуля; программе *prog* передается полный путь к модулю внутри репозитория в качестве единственного аргумента

Псевдонимы модулей предоставляют альтернативные имена для других модулей либо сокращения для наборов модулей или подкаталогов модулей. Определения псевдонимов модулей действуют подобно макроподстановкам в том смысле, что результатом их использования становится выполнение команд для всего списка упомянутых модулей и каталогов. Псевдонимы модулей не группируют определяемые модули под псевдонимом (для этого используются иерархические модули). Например, определение:

```
h -a hello
```

делает имя *h* синонимом модуля *hello*. А определение:

```
project -a library client server
```

позволяет извлечь все три модуля проекта в виде одной единицы. Если элементу в определении псевдонима модуля предшествует символ «!», указанный каталог исключается из модуля.

Обычные модули позволяют создавать модули из подмножеств других модулей. Например, определение:

```
header library library.h
```

создает модуль, который содержит только заголовочный файл из модуля *library*.

Иерархические модули являются истинно логическими модулями. Здесь не существует каталогов верхнего уровня в репозитории, но они могут быть извлечены в рабочие каталоги, и каталоги появятся под собственными именами. Модули, перечисленные в определении, расположены иерархически ниже этого каталога. Например:

```
project &library &client &server
```

Это почти то же, что и определение псевдонима модуля, которое приводилось чуть выше, за исключением того, что модули извлекаются внутрь подкаталога с именем *project*.

В файле *modules* длинные строки определений могут занимать несколько строк, при этом все строки, кроме последней, должны завершаться символом обратного слэша (\).

Файл notify

Файл *notify* используется совместно с командой **watch**. Когда должно быть послано уведомление, происходит чтение данного файла с целью определения способа уведомления.

Каждая из строк файла *notify* содержит маску файлов и командную строку. Механизм уведомлений CVS использует заданную командную строку для отправки уведомлений, касающихся файлов, соответствующих указанному шаблону имен.

Существует единственная специальная переменная, `%s`, которая может входить в определение выполняемой команды. При выполнении команды переменная заменяется именем пользователя, которому посылается уведомление. Если существует административный файл *users*, в нем ищутся имена пользователей, и окончательные значения используются для подстановки в `%s`. Это позволяет отправлять почтовые уведомления пользователям не только на локальной машине. Содержание уведомления передается уведомляющей программе через стандартный ввод.

Вот типичный способ использования возможностей файла *notify*:

```
ALL mail %s -s "CVS notification"
```

По существу, эта запись присутствует в стандартном файле *notify*, который создается при инициализации репозитория посредством **cvs init** (хотя изначально строка закомментирована).

Файл passwd

Если вы используете доступ к репозиторию через путь сервера паролей `pserver` (см. раздел «Путь к репозиторию»), CVS может обеспечить собственную идентификацию пользователей, отделенную от системной базы данных пользователей. Эта информация хранится в административном файле *CVSROOT/passwd*.

Данный механизм предоставляет возможность анонимного доступа к CVS через Интернет. Создав запись для публичного пользователя (как правило, *anoncvs* или *anonymous*), можно использовать `pserver` для обеспечения доступа многих людей, использующих эту публичную учетную запись. Если вы не намереваетесь создавать системного пользователя с тем же именем, что у публичного, или если такой пользователь существует, но с другой целью, можно использовать псевдоним пользователя для отображения имени в другое:

```
anonymous:TY7QWpLw8bvus:cvsnoname
```

Не забудьте создать в системе пользователя `cvsnoname`. Можно использовать */bin/false* в качестве интерпретатора по умолчанию для этого пользователя и корневой каталог репозитория в качестве его домашнего каталога.

Для того чтобы ограничить публичного пользователя доступом только для чтения, добавьте его в административный файл *CVSROOT/readers*.

Кроме того, собственная база данных пользователей CVS полезна и в тех случаях, когда нет необходимости в предоставлении анонимного доступа к CVS. Имеется возможность ограничить доступ к CVS-подмножеству пользователей системы, обеспечить удаленный доступ тем пользователям, которые не могут работать с системой напрямую, а также предотвращать посылку в открытом виде системных паролей (см. раздел «Вопросы безопасности»).

Не существует команды **cvs passwd** для работы с CVS-специфичными паролями (определенными в файле репозитория *CVSROOT/passwd*). Работа с пользователями и паролями CVS является задачей, решаемой вручную.

Файл rcsinfo

CVS производит чтение данного файла при помещении файлов в репозиторий (**commit**) или импортировании (**import**), чтобы определить шаблон для редактора, в котором создаются записи для журнала изменений. Каждая запись файла состоит из маски файловых имен и имени файла, используемого в качестве шаблона для каталогов модуля, имена которых соответствуют маске.

В этом файле допустимо использование ключевых слов `ALL` и `DEFAULT`.

Файл readers

Если файл *readers* существует, то перечисленные в нем пользователи имеют ограниченный доступ к репозиторию – только для чтения.

Файл taginfo

CVS производит чтение файла каждый раз, когда выполняются команды **tag** и **rtag**. Записи в файле состоят из масок имен файлов и спецификаций программ. В *taginfo* разрешено использование ключевого слова `ALL`.

Файл *taginfo* обрабатывается в контексте тега, выполняемой операции, имени каталога модуля (относительно корня репозитория), а также имен и версий файлов для всех файлов, затрагиваемых операцией. Существуют следующие операции: `add` (для **tag**), `del` (для **tag –d**) и `mov` (для **tag –F**).

Если программа, вызванная *taginfo*, завершается с ненулевым кодом возврата, происходит прерывание выполнения команды **tag** или **rtag**, вызвавшей выполнение программы.

Файл users

Если данный файл существует, происходит чтение информации из него в процессе обработки содержимого административного файла *notify*. Записи в файле *users* состоят из двух полей, разделяемых двоеточием и расположенных на одной строке. Первое поле содержит имя пользователя, а второе поле является значением (как правило, почтовым адресом пользователя на другой машине). Например:

```
john:john@somecompany.com
jane:jane@anothercompany.com
```

Файл verifymsg

CVS производит чтение данного файла, чтобы определить, необходимо ли производить проверку записей журнала изменений. Если программа завершается с ненулевым кодом возврата, операция помещения в репозиторий прерывается. Файл *verifymsg* обрабатывается в контексте полного пути к файлу, содержащему записи журнала и подлежащему обработке.

Из ключевых слов `ALL` и `DEFAULT` в данном файле может использоваться только `DEFAULT`. Если найдены соответствия нескольким шаблонам, используется первое из них.

Файл writers

Если файл существует, перечисленные в нем пользователи имеют права чтения и записи при работе с репозиторием (за исключением случаев, когда отдельные пользователи упомянуты в файле *readers*; в этом случае они будут иметь доступ только для чтения).

Тонкая работа с репозиторием

Поскольку репозиторий является обычным каталогом, хотя и с особым содержимым, в этот каталог можно перейти с помощью команды **cd**, изучить его содержимое и/или внести изменения в файлы или каталоги. Для каждого добавленного в репозиторий файла существует файл с таким же именем и суффиксом «*,v*» в соответствующем каталоге репозитория. Это файлы RCS (имеется в виду формат, а не программа), в которых хранятся многочисленные версии файлов.

Поскольку действия, описываемые в этом разделе, включают внесение изменений напрямую в репозиторий вместо стандартной работы через команды CVS, следует проявлять крайнюю осторожность и иметь свежие резервные копии репозитория при выполнении описанных операций.

Изменение структуры проекта

Изменение структуры проекта, включающее перемещение файлов или каталогов внутри репозитория (с возможным изменением имен), позволяет сохранить историю изменения файлов. Стандартный способ переименования файла при использовании CVS состоит в следующем: переименовать файл в рабочем каталоге, выполнить **cvs remove** для предыдущего имени файла, а затем выполнить **cvs add** для нового имени файла. В результате происходит отрыв файла под новым именем от старой истории версий, поэтому иногда более предпочтительно произвести переименование напрямую в репозитории. Следует отметить, что опасно производить подобную операцию, если существует вероятность того, что разработчики имеют извлеченные копии файлов в рабочих каталогах, поскольку информация в рабочих каталогах будет относиться к файлу, который больше не существует в репозитории.

Групповое импортирование

При импортировании целого проекта все файлы проекта добавляются в репозиторий. Но если некоторые из этих файлов не должны добавляться в репозиторий, то их можно удалить. Обычно это делается командой **cvs remove**, но при этом копии удаленных файлов навечно останутся в репозитории и будут храниться в каталоге *.Attic*. Чтобы избежать этого, можно удалить файлы прямо из репозитория непосредственно перед началом взаимодействия с репозиторием в частных рабочих каталогах.

Импортирование

При наличии существующей базы кода у вас может возникнуть желание импортировать ее в CVS способом, позволяющим сохранить большую часть информации об истории. В этом разделе приведены инструкции по импортированию в CVS проектов, которые представляют собой моментальные копии кода, или проектов других систем управления версиями. Все способы, кроме прямого импортирования моментальной копии кода, основаны на преобразовании в файлы RCS с последующей заменой файлов RCS в соответствующих точках репозитория CVS.

Импортирование моментальных копий кода

Если вы поддерживали историю проекта путем периодического создания моментальных копий кода, можно импортировать первую из таких копий, пометить ее датой или номером версии, а затем произвести последовательное наложение обновлений файлов из более поздних архивов. Каждый набор может затем помещаться в репозиторий и получать идентификатор с целью создания репозитория с сохранением предшествовавшей импортированию истории проекта.

Для примера сначала распакуем дистрибутивы (предполагается, что они будут распакованы в каталоги, имена которых также содержат номера версий):

```
user@localhost$ tar xvzf foo-1.0.tar.gz
user@localhost$ tar xvzf foo-1.1.tar.gz
user@localhost$ tar xvzf foo-2.0.tar.gz
```

Затем сделаем копию первой версии, импортируем ее в репозиторий CVS, извлечем ее с целью создания *песочницы* (поскольку само по себе импортирование не копирует исходные тексты напрямую в рабочий каталог) и выполним **cvs tag**, чтобы присвоить версии символьное имя (метку), отражающую версию проекта:

```
user@localhost$ mkdir foo
user@localhost$ cp -R -p foo-1.0/* foo
user@localhost$ cd foo
user@localhost$ cvs import -m 'Imported version 1.0' foo vendor start
user@localhost$ cd ..
user@localhost$ mv foo foo.bak
user@localhost$ cvs checkout foo
user@localhost$ cd foo
user@localhost$ cvs tag foo-1_0
user@localhost$ cd ..
```

Теперь запишем изменения между версиями 1.0 и 1.1 в рабочий каталог, поместим изменения в репозиторий, а также создадим метку:

```
user@localhost$ diff -Naur foo-1.0 foo-1.1 | (cd foo; patch -Np1)
user@localhost$ cd foo
user@localhost$ cvs commit -m 'Imported version 1.1'
user@localhost$ cvs tag foo-1_1
```

```
user@localhost$ cd ..
```

Затем запишем изменения между версиями 1.1 и 2.0 в рабочий каталог, поместим изменения в репозиторий, а также создадим метку:

```
user@localhost$ diff -Naur foo-1.1 foo-2.0 | (cd foo; patch -Np1)
user@localhost$ cd foo
user@localhost$ cvs commit -m 'Imported version 2.0'
user@localhost$ cvs tag foo-2_0
```

Теперь можно воспользоваться командой **log** для просмотра истории файлов, более старых версий файлов, и продолжить разработку с контролем версий.

Импортирование из RCS

Если вы переносите проект из RCS в CVS, следование приведенным ниже инструкциям поможет вам создать рабочий репозиторий CVS. Описанные операции включают внесение изменений непосредственно в репозиторий, поэтому должны выполняться с должной осторожностью.

Прежде чем начать, убедитесь, что никакие импортируемые в CVS файлы не заблокированы RCS. Создайте новый репозиторий CVS и модуль (либо новый модуль в существующем репозитории). Затем создайте в репозитории CVS набор каталогов, который отражает структуру каталогов импортируемого проекта. И, наконец, скопируйте все файлы версий (они имеют расширение *,v*) проекта (которые могут быть в подкаталогах *RCS*) в соответствующие каталоги в репозитории (без подкаталогов *RCS*).

Например, сначала скопируем каталог, который управляется RCS, создадим пустой каталог для новой структуры CVS, импортируем каталог, а затем извлечем файлы в рабочий каталог:

```
user@localhost$ mv foo foo-rcs
user@localhost$ mkdir foo
user@localhost$ cd foo
user@localhost$ cvs import -m 'New empty project' foo vendor start
user@localhost$ cd ..
user@localhost$ mv foo foo.bak
user@localhost$ cvs checkout foo
```

Затем создадим каталоги и добавим их в репозиторий, чтобы воспроизвести структуру проекта RCS:

```
user@localhost$ cd foo
user@localhost$ mkdir dir
user@localhost$ cvs add dir
user@localhost$ cd ..
```

Теперь скопируем файлы версий (*,v*) проекта RCS в *репозиторий* проекта CVS:

```
user@localhost$ cp -p foo-rcs/*,v $CVSROOT/foo
user@localhost$ cp -p foo-rcs/dir/*,v $CVSROOT/foo/dir
```

И, наконец, выполним в рабочем каталоге команду **cvs update**, чтобы получить наиболее свежие версии всех файлов:

```
user@localhost$ cd foo
user@localhost$ cvs upd
```

Импортирование из SCCS

Для того чтобы импортировать проект из SCCS, воспользуйтесь сценарием *sccs2rcs*, который расположен в каталоге *contrib* дистрибутива CVS. Сценарий преобразует файлы в формат RCS, после чего останется выполнить описанную выше процедуру импортирования RCS. Чтобы воспользоваться этим способом, необходимо иметь установленные системы CVS и SCCS. Дополнительная информация находится в комментариях к сценарию.

Импортирование из PVCS

Для того чтобы импортировать проект из PVCS, воспользуйтесь сценарием *pvcs_to_rcs*, который расположен в каталоге *contrib* дистрибутива CVS. Сценарий преобразует файлы в формат RCS, после чего останется только выполнить описанную выше процедуру импортирования RCS. Чтобы воспользоваться этим способом, необходимо иметь установленные системы CVS и PVCS. Дополнительная информация находится в комментариях к сценарию.

Использование промежуточного разделяемого рабочего каталога

Иногда случается так, что проект оказывается нежелательным образом привязанным к окружению разработки, особенно если изначально нет никаких требований к возможной смене окружения. Проект, который развивался без контроля версий, мог вообще разрабатываться в среде, которая будет являться основной для применения результатов. Несмотря на то что такая практика не рекомендуется, ситуации, подобные вышеописанным, имеют место в реальных проектах. CVS может успешно использоваться для исправления таких ситуаций, с самого начала позволяя координировать возможный впоследствии перенос проекта между средами.

Стандартным режимом работы для CVS является использование многочисленных, независимых рабочих каталогов, связанных с одним централизованным, разделяемым хранилищем данных. Код, выполняемый в таком окружении, обязательно (по крайней мере, частично) является независимым от среды. Таким образом, использование CVS с самого начала разработки проекта обеспечивает определенную гибкость.

Однако если проект уже в самом разгаре, может быть использовано промежуточное решение. Например, область разработки может быть преобразована в единый, разделяемый рабочий каталог посредством импортирования кода в CVS и последующего извлечения его из репозитория:

```
user@localhost$ cd /usr/local/bar
user@localhost$ cvs import bar vendor start
```

```
user@localhost$ cd ..
user@localhost$ mv bar bar.bak
user@localhost$ cvs checkout bar
```

Высока вероятность того, что такой подход окажется слишком агрессивным и в репозиторий попадет большее число файлов, чем необходимо. В таком случае можно вернуться и удалить из репозитория файлы, которым там не место, либо выполнять команду **cvs remove** по мере обнаружения ненужных файлов.

Кроме того, в рабочем каталоге почти наверняка окажутся двоичные файлы, которые были импортированы как текстовые. Если вы видите двоичный файл, который должен храниться в репозитории, необходимо выполнить команду **cvs admin –kb** *file*, а затем сделать новую копию файла из резервных копий проекта. После этого команда **cvs commit** *file* поместит исправленный файл обратно в репозиторий.

Систему контроля версий лучше вводить в действие в самом начале проекта, до того как станет необходимой оптимизация гибкости, поскольку нежелательные изменения могут быть легко найдены и обращены.

Путь к репозиторию (см. далее раздел «Путь к репозиторию») указывается в параметре **–d** или с помощью переменной окружения $CVSROOT. Путь хранится в файлах *CVS/root* рабочих каталогов. При использовании сервера паролей **(pserver)**, запоминается идентификатор пользователя (ID), извлекающего файлы в рабочий каталог. Если с определенным рабочим каталогом работают несколько человек, им придется делить учетную запись для доступа к CVS.

Это можно реализовать путем создания записи для нейтрального пользователя с паролем, известным всем, кто имеет доступ к репозиторию CVS. Тогда каждый сможет выполнить команду **cvs login** с одним и тем же пользовательским идентификатором и паролем для того, чтобы получить доступ к репозиторию. Когда вы закончите работу с разделяемым рабочим каталогом, такой не совсем удобный способ станет ненужным. Однако в процессе использования разделяемого рабочего каталога необходимо позаботиться о том, чтобы разработчики указывали свои действительные пользовательские идентификаторы в записях журнала изменений, поскольку в противном случае все изменения будут приписаны «коллективному» пользователю.

Глобальные параметры сервера

Сервер имеет всего один глобальный параметр: **–allow-root=***rootdir*. Этот параметр используется для указания серверу CVS принимать и обрабатывать запросы для указанного репозитория.

Команды администратора

В табл. 14.8 перечислены команды CVS, доступные администратору для управления репозиториями.

Таблица 14.8. Команды администратора

Команда	Описание
admin **adm** **rcs**	Выполнение задач администрирования
init	Создание нового репозитория
server	Работа в серверном режиме

Команда admin

```
admin
  [ -b[rev] ]
  [ -cstring ]
  [ -kkflag ]
  [ -l[rev] ]
  [ -L ]
  [ -mrev:msg ]
  [ -nname[:[rev]] ]
  [ -Nname[:[rev]] ]
  [ -orange ]
  [ -q ]
  [ -sstate[:rev]
  [ -t[file] ]
  [ -t-string ]
  [ -u[rev] ]
  [ -U ]
  [ files ...]
```

Команда **admin** используется для выполнения задач администрирования. Если существует группа пользователей *cvsadmin*, только пользователи, входящие в эту группу, могут выполнять **admin** с параметрами, отличными от **–k**. Дополнительные параметры, которые могут быть использованы с командой **admin**, перечислены в табл. 14.9.

Таблица 14.9. Параметры команды admin

Параметр	Описание
–b[*rev***]**	Установить ветвь по умолчанию
–c*string*	Вышел из употребления. Установить строку начала комментария
–k*kflag*	Установить значение по умолчанию для режима подстановки ключевых слов
–l[*rev***]**	Заблокировать указанную версию
–L	Включить жесткую блокировку
–m*rev:msg*	Изменить запись в журнале версий для указанной версии

Таблица 14.9 (продолжение)

Параметр	Описание
−n*name*[**:**[*rev*]]	Связать указанную ветвь или версию с меткой *name*
−N*name*[**:**[*rev*]]	Аналогично **−n**, с тем исключением, что если метка *name* уже используется, она меняет владельца
−o*range*	Полное удаление версий
−q	Не выводить диагностические сообщения
−s*state*[**:***rev*]	Изменить состояние версии
−t[*file*]	Поместить описание RCS-файла
−t-*string*	Изменить описание RCS-файла на указанный текст *string*
−u[*rev*]	Разблокировать указанную версию
−U	Отключить жесткую блокировку

Если версия, указанная для параметра **−l**, является ветвью, используется самая последняя версия из этой ветви. Если версия не указана, используется последняя версия ветви по умолчанию.

Если имя, указанное для параметра **−n**, уже используется, выдается сообщение об ошибке. Для перемещения метки можно воспользоваться параметром **−N** (изменится номер версии, связанной с меткой); однако рекомендуется использовать для этих целей команды **cvs tag** и **cvs rtag**.

Использование параметра **−o** крайне опасно, поскольку приводит к полному удалению информации из репозитория. При его использовании следует проявлять особую осторожность. В табл. 14.10 описаны различные способы задания диапазонов версий. Удаляемая версия не должна иметь ветвлений и не должна быть заблокирована. Также следует иметь в виду возможные взаимодействия этой команды с метками.

Если имя файла *file* не указано для параметра **−t**, CVS производит чтение со стандартного ввода, пока не будет достигнут конец файла или прочитан символ точки на отдельной строке.

Для параметра **−u** версия, над которой производится операция, определяется точно так же, как для **−l**.

Таблица 14.10. Форматы диапазонов

Формат	Описание
rev1::*rev2*	Удалить версии между *rev1* и *rev2*, сохранив минимум информации, необходимый для прямого перехода от *rev1* к *rev2*. Две указанные версии сохраняются
::*rev*	Аналогично *rev1*::*rev2*, с тем исключением, что первая версия является точкой ветвления

Таблица 14.10 (продолжение)

Формат	Описание
rev::	Аналогично *rev1::rev2*, с тем исключением, что вторая версия является концом ветви, и она удаляется
rev	Удалить указанную версию
rev1: *rev2*	Аналогично *rev1::rev2*, с тем исключением, что обе указанные версии также удаляются
:*rev*	Аналогично*::rev2*, с тем исключением, что указанная версия также удаляется
rev:	Аналогично *rev1::*, с тем исключением, что указанная версия также удаляется

Параметры, приведенные в табл. 14.11, присутствуют в CVS по историческим причинам и не должны использоваться (их использование может повредить репозиторий).

Таблица 14.11. Вышедшие из употребления параметры администрирования

Параметр	Описание
–a*logins*	Добавить имена пользователей в список доступа файла RCS
–A*oldfile*	Добавить список доступа указанного файла *oldfile* к списку доступа файла RCS
–e[*logins***]**	Удалить имена указанных пользователей из списка доступа файла RCS или удалить все имена, если список не задан
–i	Создать и инициализировать новый файл RCS. Не используйте этот параметр. Вместо него используйте команду **add** для добавления файлов в репозиторий CVS
–I	Диалоговый режим. Этот параметр не работает с моделью клиент-сервер в CVS и, скорее всего, будет изъят в следующей версии CVS
–V*n*	Вышел из употребления. Этот параметр использовался для указания того, что файлы RCS, используемые CVS, должны создаваться с расчетом на совместимость с определенной версией RCS
–x*suffixes*	Этот параметр использовался для определения суффиксов имен RCS-файлов, но в CVS всегда использовался только *,v* в качестве суффикса RCS-файла

Команда init

```
init
```

Инициализирует репозиторий. Используйте глобальный параметр **–d**, для того чтобы указать каталог для репозитория в тех случаях, когда переменная $CVSROOT не установлена надлежащим образом.

Вновь созданный репозиторий будет содержать только один модуль – *CVSROOT*. После инициализации репозитория можно использовать другие команды CVS для добавления файлов в него или извлечения модуля *CVSROOT* с целью внесения изменений в административные файлы.

Команда pserver

```
pserver
```

Действует как сервер, обеспечивая доступ к репозиторию, указанному перед командой при помощи параметра **–allow-root**. Данная команда используется в файле *inetd.conf*, а не в командной строке. Другим глобальным параметром, часто используемым с данной командой, является **–T** (табл. 14.1).

Справочник пользователя CVS

В этом разделе подробно рассматриваются получение доступа к репозиторию, структура рабочих каталогов и использование команд CVS.

Путь к репозиторию

На сегодняшний день CVS поддерживает пять методов доступа клиентов к репозиторию: локальный, внешний, через сервер паролей, сервер GSS-API (Generic Security Services API), а также сервер Kerberos 4 (большинство пользователей Kerberos предпочтет использовать GSS-API). В табл. 14.12 описаны различные варианты указания пути к репозиторию и соответствующие им методы доступа.

Таблица 14.12. Типы и методы доступа к репозиторию

Метод	Формат пути	Описание
Локальный	*path*	Если каталог репозитория расположен на той же машине, с которой пользователь получает доступ (либо может считаться таковой, как в случаях с монтированием файловых систем с помощью NFS или Samba), путь к репозиторию является обычным путевым именем каталога репозитория, например */usr/local/cvsrep*.
Внешний	`:ext:`*user*`@`*host*`:`*path*	Доступ к внешним репозиториям можно получить с помощью клиента удаленного интерпретатора **rsh** (по умолчанию) или **ssh**. Переменная окружения $CVS_RSH используется для указания используемого клиента удаленного интерпретатора.
Сервер паролей	`:pserver:`*user*`@`*host*`:`*path*	При доступе через сервер паролей необходимо идентифицировать пользователя, прежде чем он получит доступ к репозиторию. Публичные CVS-сервера обычно настраиваются именно таким образом, поэтому они могут предоставлять анонимный

Таблица 14.12 (продолжение)

Метод	Формат пути	Описание
		доступ к CVS. Более подробно об анонимном доступе к CVS можно прочитать в разделе «Файл passwd» ранее в этой главе.
Сервер GSS-API	`:gserver:`	Такой формат пути используется для доступа к серверам с механизмом идентификации Kerberos 5 или другими механизмами идентификации, поддерживаемыми GSS-API.
Сервер Kerberos	`:kserver:`	Этот формат пути используется для доступа к серверам с механизмом идентификации Kerberos 4.

Настройка CVS

На поведение CVS, помимо аргументов командной строки, могут влиять два класса настроек: *переменные окружения* (табл. 14.13) и *специальные файлы* (табл. 14.14).

Таблица 14.13. Переменные окружения

Переменная	Описание
$COMSPEC	Командный интерпретатор на системах OS/2, если это не *cmd.exe*.
$CVS_CLIENT_LOG	Отладочный файл на стороне клиента для соединений типа клиент-сервер. $CVS_CLIENT_LOG – основа имени файла для файлов *$CVS_CLIENT_LOG.in* и *$CVS_CLIENT_LOG.out*, которые создаются в текущем рабочем каталоге во время выполнения команды.
$CVS_CLIENT_PORT	Номер порта для путей формата `:kserver:`. $CVS_CLIENT_PORT можно не устанавливать, если **kserver** принимает запросы через порт 1999 (порт по умолчанию).
$CVS_IGNORE_REMOTE_ROOT	В соответствии с журналом изменений CVS, *ChangeLog*, данная переменная была удалена из CVS, начиная с версии 1.10.3.
$CVS_PASSFILE	Файл паролей для путей формата `:pserver:`. Эта переменная должна быть установлена до выполнения команды **cvs login**. По умолчанию это *$HOME/.cvspass*.
$CVS_RCMD_PORT	Порт для подключения к демону **rcmd** сервера для не-Unix систем.
$CVS_RSH	Программа для использования с путями формата `:ext:` в том случае, если это не **rsh**.

Таблица 14.13 (продолжение)

Переменная	Описание
$CVS_SERVER	Программа на удаленном сервере для путей формата `:ext:` в том случае, если это не **cvs**.
$CVS_SERVER_SLEEP	Задержка выполнения на сервере (в секундах), позволяющая прикрепить к процессу отладчик.
$CVSEDITOR	Редактор, используемый для создания записей журнала изменений; переменная имеет больший приоритет, чем $EDITOR.
$CVSIGNORE	Список шаблонов файловых имен (разделенных пробелами), которые следует игнорировать при работе (см. также *cvsignore* в табл. 14.4 и *.cvsignore* в табл. 14.14).
$CVSREAD	Определяет режим только для чтения (если переменная установлена) или чтения/записи (если не установлена) для **checkout** и **update**.
$CVSROOT	Путь к репозиторию по умолчанию.
$CVSUMASK	Используется для определения прав доступа для файлов репозитория (локальных).
$CVSWRAPPERS	Перечень масок файловых имен для функции *cvswrappers*. См. также ранее раздел «Структура репозитория».
$EDITOR	Определяет редактор, используемый для создания записей журнала изменений; см. примечание в описании переменной $CVSEDITOR ранее в этой таблице.
$HOME	На системах Unix используется для поиска файла *.cvsrc*.
$HOMEDRIVE	На системах Windows NT используется для поиска файла *.cvsrc*.
$HOMEPATH	На системах Windows NT используется для поиска файла *.cvsrc*.
$PATH	Используется для поиска выполняемых программ.
$RCSBIN	Используется для поиска выполняемых программ RCS. Переменная вышла из употребления.
$TEMP $TMP $TMPDIR	Пути для создания временных файлов. $TMPDIR используется сервером. На системах Unix */tmp* (и TMP на системах Windows NT) не могут быть переопределены для некоторых функций CVS по той причине, что они основаны на вызове системной функции `tmpnam()`.

Несмотря на схожесть имен, переменная окружения $CVSROOT и каталог *CVSROOT* репозитория никак не связаны.

Строка «RSH» в имени переменной окружения $CVS_RSH не является указанием на конкретную программу **(rsh)**, но является указанием на то, что переменная используется CVS для вызова программы удаленного интерпретатора (remote shell), а сама программа может иметь имя, отличное от **rsh**, например **ssh**.

Поскольку существует только один способ указать (посредством переменной $CVS_RSH) программу, используемую в качестве удаленного интерпретатора, и эта настройка является глобальной, пользователям, работающим с различными репозиториями, имеет смысл обращать внимание на то, какой именно репозиторий они используют. Если для разных репозиториев потребуются различные программы доступа, необходимо будет изменять значение переменной между сеансами работы. Этот аспект метода доступа к репозиторию никак не отмечается в файле CVS/*Root* рабочего каталога (см. раздел «Каталоги CVS» далее в этой главе). Например, если доступ к некоторым репозиториям осуществляется посредством **rsh**, а к некоторым – посредством **ssh**, можно создать два следующих псевдонима для **cvs** (синтаксис **bash**):

```
user@localhost$ alias cvs="export CVS_RSH=ssh; cvs"
user@localhost$ alias cvr="export CVS_RSH=rsh; cvs"
```

В табл. 14.14 перечислены файлы, используемые клиентом командной строки CVS для соединения с сервером и настройки параметров клиента. Эти файлы расположены в домашнем каталоге пользователя.

Таблица 14.14. Файлы настройки клиента

Файл	Описание
~/.cvsignore	Маски файловых имен, которые следует игнорировать
~/.cvspass	Пароли, кэшированные командой **cvs login**
~/.cvsrc	Командные параметры по умолчанию
~/.cvswrappers	Специфичные для пользователя фильтры для **checkout** и **commit**

Файл *~/.cvspass* является в действительности рабочим файлом, а не файлом настройки. Он используется программой-клиентом **cvs** для хранения пользовательского пароля доступа к репозиторию между выполнением команд **cvs login** и **cvs logoff**.

Некоторые из распространенных параметров *.cvsrc*:

update –dP

При выполнении команды **cvs update** создавать новые каталоги и удалять пустые.

diff –c

Вывод в формате контекстного **diff**.

Создание рабочего каталога

Для того чтобы воспользоваться CVS, необходимо создать рабочий каталог или иметь уже созданный для вас. В этом разделе описан процесс создания рабочего каталога в предположении, что в репозитории уже существует модуль, с которым пользователь намеревается работать. Информацию по импортированию нового модуля в репозиторий см. далее в этой главе в описании пользовательской команды **import**.

1. Определите путь к репозиторию. Если вам требуется помощь в определении полного пути к репозиторию, обратитесь к администратору репозитория.
2. Если этот репозиторий в дальнейшем будет являться основным, установите значение переменной $CVSROOT; в противном случае воспользуйтесь параметром **–d** при выполнении команд CVS, которые не могут определить нужное значение пути, исходя из содержимого файлов рабочего каталога.
3. Выберите извлекаемый модуль.
4. Выберите рабочий каталог и перейдите в его родительский каталог (**cd**).
5. Если для доступа к репозиторию требуется идентификация, выполните команду **cvs login**.
6. Выполните команду **cvs checkout** *module*.

Например:

```
export CVSROOT=/usr/local/cvsroot
cd ~/work
cvs checkout hello
```

Структура рабочего каталога

В данном разделе описаны файлы и каталоги, которые могут встречаться в рабочем каталоге.

Файлы .cvsignore

Рабочие каталоги могут содержать файлы *.cvsignore*, определяющие маски файловых имен, которые существуют в рабочем каталоге, но не должны помещаться в CVS. Это обычно используется для того, чтобы CVS игнорировала производные файлы.

Файлы .cvswrappers

Рабочие каталоги могут содержать файлы *.cvswrappers*, которые обеспечивают специфичную для каталога информацию по обработке файлов с функциональностью, аналогичной той, что предоставляет административный файл *cvswrappers* (см. раздел «Файл cvswrappers» ранее в этой главе).

Каталоги CVS

Каждый каталог в рабочем каталоге содержит подкаталог *CVS*. Файлы в этом каталоге (табл. 14.15) содержат метаданные, используемые CVS для определения расположения репозитория и отслеживания копирования версий файлов в рабочий каталог.

Таблица 14.15. Файлы в каталогах CVS

Файл	Описание
Base *Baserev* *Baserev.tmp*	В каталоге *Base* хранятся копии файлов, создаваемые в процессе работы с командой **edit**. Файл *Baserev* содержит номера версий файлов в *Base*. Файл *Baserev.tmp* используется при обновлении файла *Baserev*
Checkin.prog *Update.prog*	Программы, указанные в файле модулей для параметров **–i** и **–u**, соответственно (если они были указаны)
Entries	Номера версии и информация о времени для файлов в момент их копирования из репозитория в процессе извлечения или обновления
Entries.Backup *Entries.Log* *Entries.Static*	Временные и промежуточные файлы, используемые CVS
Notify *Notify.tmp*	Временные файлы, используемые CVS при работе с уведомлениями для таких команд, как **edit** и **unedit**
Repository	Имя, под которым каталог известен в репозитории
Root	Действующий путь к репозиторию на момент создания рабочего каталога (посредством **cvs checkout**)
Tag	Информация о «липких» метках и датах для файлов в каталоге
Template	Используется для сохранения содержимого административного файла *rcsinfo* для удаленных репозиториев

Поскольку в каждом рабочем каталоге есть только один файл *CVS/Root*, один рабочий каталог соответствует только одному репозиторию. Невозможно извлечь в один рабочий каталог какие-либо файлы из одного репозитория и какие-либо – из другого.

Глобальные параметры клиента

В табл. 14.16 перечислены глобальные параметры, которые контролируют работу программы-клиента CVS.

Таблица 14.16. Глобальные параметры клиента

Параметр	Описание
–a	Идентификация (только для **gserver**)

Таблица 14.16 (продолжение)

Параметр	Описание
–d root	Указать расположение репозитория. Имеет более высокий приоритет, чем значение $CVSROOT
–e editor	Определяет редактор сообщений. Имеет более высокий приоритет, чем значения $CVSEDITOR и $EDITOR
–f	Не читать файл ~/*.cvsrc*. Параметр полезен, когда *.cvsrc* содержит параметры, которые нежелательно использовать в конкретной команде
–H [*command*] **–help** [*command*]	Вывести справку. Если команда не указана, выводится общая справка по CVS, включая перечень других параметров справки
–l	Не заносить команду в журнал
–n	Не изменять никакие файлы. Полезно в том случае, если требуется узнать, на какие файлы повлияет выполняемая команда
–q	Тихий режим
–Q	Очень тихий режим. Выводить сообщения только в случае возникновения серьезных проблем
–r	Создавать новые рабочие файлы с правами доступа только для чтения
–s *variable=value*	Установить значение пользовательской переменной. Пользовательские переменные могут использоваться в содержимом административных файлов
–t	Отслеживание выполнения. Полезно при отладке проблем соединения с удаленным репозиторием, а также в сочетании с **–n** для определения действия незнакомой команды
–w	Создавать новые рабочие файлы с правами доступа для чтения и записи. Имеет более высокий приоритет, чем значение $CVSREAD. Файлы создаются для чтения/записи, если не установлена переменная $CVSREAD или не указан параметр **–r**
–x	Шифрование (новая возможность версии 1.10)
–z gzip_level	Установить уровень сжатия. Полезно использовать при медленных соединениях между клиентом и сервером

Общие параметры клиента

В табл. 14.17 и 14.18 описаны параметры, которые являются общими для многих команд CVS. Табл. 14.17 содержит общие параметры с описанием их действия, а в табл. 14.18 перечислены параметры, которые могут быть использованы с пользовательскими командами. В следующих разделах будут подробно описаны параметры, которые либо не упомянуты здесь, либо функционируют иначе.

Таблица 14.17. Общие параметры

Параметр	Описание
−D *date*	Использовать самую последнюю версию, датированную не позднее *date*
−f	Для команд, которые работают с меткой (при помощи **−r**) или датами (при помощи **−D**), включать файлы, не отмеченные указанной меткой или не соответствующие указанной дате. Включаются самые последние версии таких файлов
−k *kflag*	Определить, как будет производиться подстановка ключевых слов. Пробел между **−k** и флагом *kflag* не обязателен. Режимы подстановки ключевых слов см. в табл. 14.19
−l	Не производить рекурсивный обход подкаталогов
−n	Не выполнять программы для модулей
−R	Производить рекурсивный обход подкаталогов (режим по умолчанию)
−r rev	Использовать номер или метку конкретной версии

В табл. 14.18 показано, какие общие параметры доступны для каждой пользовательской команды.

Таблица 14.18. Применимость общих параметров клиента

Команда	−D	−f	−k	−l	−n	−R	−r
add			•				
annotate	•	•		•		•	•
checkout	•	•	•	•	•	•	•
commit				•	•	•	•
diff	•		•	•		•	•
edit				•		•	
editors				•		•	
export	•	•	•	•	•	•	•
help							
history	•						•
import			•				
log				•		•	
login							

Таблица 14.18 (продолжение)

Команда	–D	–f	–k	–l	–n	–R	–r
logout							
rdiff	•	•		•		•	•
release							
remove				•		•	
rtag	•	•		•		•	•
status				•		•	
tag				•		•	
unedit				•		•	
update	•	•	•	•		•	•
watch				•		•	
watchers				•		•	

Форматы дат

CVS позволяет использовать самые различные форматы задания дат, среди которых:

Стандарт ISO

Предпочтительно задание дат в виде `YYYY-MM-DD HH:MM`, который будет выглядеть как `2000-05-17` или `2000-05-17 22:00`. Технические детали использования формата определяются стандартом ISO 8601.

Стандарт Email

`17 May 2000`. Технические детали использования формата определяются стандартами RFC 822 и RFC 1123.

Относительный

`10 days ago`, `4 years ago` (10 дней назад, 4 года назад).

Общий

`month/day/year`. Этот формат может вызывать путаницу, поскольку в различных странах используется различный порядок следования первых двух полей (дата `1/2/2000` является двусмысленной).

Другие

Другие форматы также допустимы, в частности `YYYY/MM/DD` и форматы без указания года (год принимает значение текущего).

Подстановка ключевых слов

В табл. 14.19 перечислены режимы подстановки ключевых слов, доступные для использования с параметром **–k**. В CVS подстановка ключевых слов используется с целью обновления информации о версии в файлах при их извлечении или обновлении из репозитория.

Таблица 14.19. Режимы подстановки ключевых слов

Режим	Описание
b	Двоичный режим. Работать с файлом так же, как и при режиме o, но не преобразовывать при этом символы новой строки.
k	Режим только ключевых слов. Все ключевые слова сокращаются до их имен. Используйте этот режим для сравнения двух версий файла без учета разницы, вызванной подстановкой значений ключевых слов.
kv	Режим ключевое слово–значение. Происходит подстановка ключевого слова и соответствующего значения. Это режим по умолчанию.
kvl	Режим ключевое слово–значение–блок-владелец (loker). Этот режим аналогичен режиму kv, за исключением того, что если версия является заблокированной, всегда добавляется значение идентификатора пользователя, которому принадлежит блокировка. Блокировку можно получить при помощи команды **cvs admin –l**.
o	Режим «старого содержимого». Не генерировать новые значения, а использовать те значения ключевых слов, которые есть в репозитории.
v	Режим только значений. Происходит подстановка для каждого ключевого слова; значение заменяет все поле ключевого слова, включая даже ограничительные символы $. Этот режим уничтожает собственно поле с ключевым словом, поэтому его следует использовать осторожно.

Поля подстановки ключевых слов имеют вид *$Keyword...$*. Доступны следующие ключевые слова:

Author

Идентификатор пользователя, поместившего версию в репозиторий.

Date

Дата и время (в стандартном формате UTC) помещения версии в репозиторий.

Header

Полный путь RCS-файла в репозитории, номер версии, дата помещения в репозиторий, время, идентификатор пользователя, состояние файла и идентификатор владельца блокировки, если файл заблокирован.

Id

Краткая форма ключевого слова Header – из имени RCS-файла исключается путь и остается только само имя файла.

`Name`

Метка, которая использовалась для извлечения файла, или пустое значение, если метка не применялась для извлечения.

`Locker`

Идентификатор пользователя-владельца блокировки файла или пустое значение, если файл не заблокирован.

`Log`

Имя файла RCS. В дополнение к подстановке ключевого слова непосредственно в поле, каждое помещение файла в репозиторий приводит к добавлению в файл строк, которые следуют непосредственно за строкой, содержащей данное ключевое слово. Первая такая строка содержит номер версии, дату и время помещения версии в репозиторий и идентификатор пользователя. Последующие строки соответствуют содержимому записи из журнала изменений. Результатом, в конце концов, является набор записей журнала изменений для файла, расположенных в обратном хронологическом порядке. Каждая из дополнительных строк предваряется тем же символом, что предваряет поле ключевого слова в строке. Это позволяет задавать форматирование журнала в виде комментариев для большинства используемых языков. Например:

```
#
# foo.pl
#
# $Log: foo.pl,v $
# Revision 1.2  2000/06/09 22:10:23  me
# Fixed the new bug introduced when the last one was fixed.
#
# Revision 1.1  2000/06/09 18:07:51  me
# Fixed the last remaining bug in the system.
#
```

Убедитесь, что вы не используете ключевые слова в теле записей журнала изменений, поскольку для них также будет производиться подстановка.

`RCSfile`

Имя RCS-файла (без пути к нему).

`Revision`

Номер версии файла.

`Source`

Полный путь к RCS-файлу.

`State`

Состояние файла, присвоенное командой **cvs admin –s** (если это не сделать явно, состояние примет значение по умолчанию – `Exp`).

Пользовательские команды

Программа-клиент CVS предоставляет пользователю команды, приведенные в табл. 14.20.

Таблица 14.20. Пользовательские команды

Команда	Описание
ad **add** **new**	Указать CVS, что файлы/каталоги должны быть добавлены к репозиторию
ann **annotate**	Вывести содержимое основной версии файла с номером версии, именем пользователя и датой последнего изменения для каждой строки
checkout **co** **get**	Создать рабочий каталог для модуля
ci **com** **commit**	Поместить изменения рабочего каталога в репозиторий
di **dif** **diff**	Определить различия между версиями файла
edit	Подготовка к редактированию файлов. Используется для улучшения координации усилий разработчиков
editors	Вывести список пользователей, работающих с файлами. Используется для улучшения координации усилий разработчиков
ex **exp** **export**	Извлечь модуль, но не создавать рабочую копию
help	Получить справку
hi **his** **history**	Отобразить информацию для файлов из журнала
im **imp** **import**	Импортировать новые модули в репозиторий
lgn **login** **logon**	Зарегистрироваться (закэшировать пароль) для работы на удаленном сервере CVS

Таблица 14.20 (продолжение)

Команда	Описание
lo **log** **rlog**	Отобразить журнал работы с файлом (файлами)
logout	Завершить работу (удалить пароль) с удаленным сервером CVS
pa **patch** **rdiff**	Выпустить **diff**. Результат может быть использован в качестве ввода программы **patch**, автором которой является Ларри Уолл (Larry Wall). Команду необязательно выполнять в рабочем каталоге
re **rel** **release**	Выполнить фиксируемое удаление рабочего каталога
remove **rm** **delete**	Удалить файл или каталог из репозитория
rt **rtag** **rfreeze**	Присвоить метку определенной версии
st **stat** **status**	Отобразить подробную информацию о состоянии файлов
ta **tag** **freeze**	Присвоить метку файлам в репозитории
unedit	Прекратить редактирование файла и снова сделать его доступным только для чтения
up **upd** **update**	Синхронизировать рабочий каталог с репозиторием
watch	Управление настройками наблюдения. Используется для улучшения координации усилий разработчиков
watchers	Вывести список пользователей, наблюдающих за изменением файлов. Используется для улучшения координации усилий разработчиков

add

```
add
  [ -k kflag ]
  [ -m message ]
  file ...
```

Указать CVS, что файлы или каталоги должны быть добавлены в репозиторий. Действительное добавление происходит в момент выполнения команды **cvs commit**. Команда также может использоваться для восстановления файлов, которые были удалены командой **cvs remove**.

Смысл параметра **–k** тот же, что и обычно. Только один дополнительный параметр может использоваться с командой **add: –m** *message*. Этот параметр используется для задания описания файла (которое выдается при выполнении команды **log**).

annotate

```
annotate
  [ [ -D date | -r rev ] -f ]
  [ -l | -R ]
  file ...
```

CVS выводит отчет для каждой из строк указанного файла. Каждой строке предшествует информация о последних изменениях, внесенных в эту строку, включая номер версии, пользователя и дату. Если версия не указана, используется начало ветви.

Параметры **–D**, **–f**, **–l**, **–r** и **–R** имеют тот же смысл, что и обычно.

checkout

```
checkout
  [ -A ]
  [ -c | -s ]
  [ -d dir [ -N ] ]
  [ [ -D date | -r rev ] -f ]
  [ -j rev1 [ -j rev2 ] ]
  [ -k kflag ]
  [ -l | -R ]
  [ -n ]
  [ -p ]
  [ -P ]
  module ...
```

Скопировать файлы из репозитория в рабочий каталог.

Параметры **–D**, **–f**, **–k**, **–l**, **–r** и **–R** имеют тот же смысл, что и обычно. Дополнительные параметры перечислены в табл. 14.21.

Таблица 14.21. Параметры checkout

Параметр	Описание
–A	Удалить все «липкие» метки или даты
–c	Скопировать файл *module* на стандартный вывод
–d *dir*	Изменить имя каталога по умолчанию
–j *rev*	Объединить ветви

Таблица 14.21 (продолжение)

Параметр	Описание
–N	Не сокращать пути модулей
–p	Передавать файлы на стандартный вывод, разделяя их строками заголовков, в которых отображаются имена файлов, имена RCS-файлов и версии
–P	Удалить пустые каталоги
–s	Отображать состояние каждого из модулей на основе информации из файла *modules*

commit

```
commit
  [ -f | [ -l | -R ] ]
  [ -F file | -m message ]
  [ -n ]
  [ -r revision ]
  [ files ...]
```

Поместить измененные файлы из рабочего каталога в репозиторий.

Доступна стандартная семантика общих параметров клиентов **–l**, **–n**, **–r** и **–R**. Дополнительные параметры перечислены в табл. 14.22.

Таблица 14.22. Параметры commit

Параметр	Описание
–f	Принудительно поместить в репозиторий, даже если файлы не изменялись
–F *file*	Использовать содержимое указанного файла в качестве записи журнала изменений
–m *message*	Использовать указанное сообщение в качестве записи журнала изменений

Использование параметра **–r** делает версию «липкой», и для продолжения использования рабочего каталога требуется выполнить команду **admin –A**.

diff

```
diff
  [ -k kflag ]
  [ -l | -R ]
  [ format ]
  [ [ -r rev1 | -D date1 ] [ -r rev2 | -D date2 ] ]
  [ file ...]
```

Команда **diff** сравнивает две версии файла и отображает различия в формате, который определяется параметрами команды. По умолчанию извлеченная в рабочий каталог версия сравнивается с версией в репозитории, с которой была сделана копия.

Параметры **–D**, **–k**, **–l**, **–r** и **–R** имеют тот же смысл, что и обычно. Все параметры команды **diff**, описанные в главе 3 «Команды Linux», также можно использовать.

edit

```
edit
  [ -a action ]
  [ -l | -R ]
  [ file ...]
```

Команда **edit** используется в сочетании с **watch** для обеспечения улучшенной координации (сериализации) процесса разработки. Файл становится доступным для записи, а всем пользователям, наблюдающим за файлом, посылается соответствующее уведомление. Устанавливается временное наблюдение **(watch)**, которое автоматически снимается при выполнении команды **unedit** или **commit**.

Параметры **–l** и **–R** имеют тот же смысл, что и обычно. Существует только один дополнительный параметр для команды **edit**: **–a** *actions*. Этот параметр используется для указания действий, за которыми необходимо вести наблюдение. Доступные значения аргумента *action* перечислены в описании команды **watch**.

editors

```
editors
  [ -l | -R ]
  [ file ...]
```

Вывести список пользователей, работающих с указанными файлами. Работа пользователя с файлом регистрируется по выполнению команды **edit** для этих файлов. Если команда **edit** не была выполнена, пользователь не попадет в отображаемый список.

Параметры **–l** и **–R** имеют тот же смысл, что и обычно. См. также описание команды **watch** далее.

export

```
export
  [ -d dir [ -N ] ]
  [ -D date | -r rev ]
  [ -f ]
  [ -k kflag ]
  [ -l | -R ]
  [ -n ]
  [ -P ]
  module ...
```

Экспорт файлов из репозитория. Действие схоже с действием команды **checkout** с той разницей, что результатом не является рабочий каталог, т. е. не создаются подкаталоги *CVS*. Это может использоваться при подготовке дистрибутивов. Например:

```
user@localhost$ cvs export -r foo-1_0 -d foo-1.0 foo
user@localhost$ tar czf foo-1.0.tar.gz foo-1.0
```

Параметры **−D**, **−f**, **−k**, **−l**, **−n**, **−r** и **−R** имеют тот же смысл, что и обычно. Дополнительные параметры перечислены в табл. 14.23.

Таблица 14.23. Параметры export

Параметр	Описание
−d *dir*	Использовать *dir* в качестве имени каталога, а не модуля
−n	Не выполнять никакие связанные с извлечением программы
−N	Не сокращать пути

При извлечении единственного файла, расположенного на втором или третьем уровне структуры подкаталогов модуля, параметр **−N** может использоваться совместно с **−d** для предотвращения создания промежуточных каталогов.

help

```
help
```

Вывод полезной информации по использованию программы **cvs**.

history

```
history
  [ -a | -u user ]
  [ -b string ]
  [ -c ]
  [ -D date ]
  [ -e | -x type ]
  [ -f file | -m module | -n module | -p repository ]...
  [ -l ]
  [ -o ]
  [ -r rev ]
  [ -t tag ]
  [ -T ]
  [ -w ]
  [ -z zone ]
  [ file ...]
```

Вывод информации из истории команд. Чтобы воспользоваться командой **history**, необходимо прежде подготовить файл *history* в репозитории. Более подробно этот файл описан в разделе «Структура репозитория».

Действие параметров **–f**, **–l**, **–n** и **–p** для команды **history** отличается от обычного в CVS.

Параметры **–D** и **–r** имеют тот же смысл, что и обычно. Отображается история активности, которая происходила после создания указанной версии или после указанной даты. Дополнительные параметры приведены в табл. 14.24.

Таблица 14.24. Параметры history

Параметр	Описание
–a	Показывать историю для всех пользователей (по умолчанию только для текущего)
–b *str*	Погружение в историю до первой записи, содержащей строку *str* в имени файла, модуля или в пути репозитория
–c	Сообщать о каждом выполнении **commit**
–e	Сообщать обо всех событиях
–f *file*	Показать информацию о последнем событии для файла *file*
–l	Показать только последнее событие
–m *module*	Вывести полный отчет по указанному модулю
–n *module*	Сообщать о последнем событии для указанного модуля
–o	Сообщать информацию по извлеченным модулям
–p *repository*	Показать историю для определенного каталога репозитория
–t *tag*	Показать историю с того момента, как метка *tag* была в последний раз добавлена к файлу истории
–T	Сообщать информацию обо всех метках
–u *name*	Показать историю для указанного пользователя
–w	Показать историю только для текущего рабочего каталога
–w *zone*	Отображать время для указанного часового пояса *zone*
–x *type*	Отчет по специфическим типам событий. См. табл. 14.25

Параметр **–p** должен ограничивать отчет **history** записями для указанного каталога или каталогов (если указано несколько параметров **–p**), но в версии 1.10.8 он никак не влияет на вывод. Например, для того чтобы отобразить историю для модулей *CVSROOT* и *hello*, следует выполнить команду:

```
cvs history -p CVSROOT -p hello
```

Использование **–t** приводит к более быстрому получению результатов, чем использование **–r**, поскольку в первом случае поиск выполняется только в файле истории, а не во всех файлах RCS.

Типы записей, перечисленные в табл. 14.25, генерируются командами **update**.

Таблица 14.25. Типы записей истории, связанные с работой update

Тип	Описание
C	Слияние было необходимо, но выявлены конфликты, которые требуют вмешательства пользователя
G	Успешное автоматическое слияние
U	Рабочий файл скопирован из репозитория
W	Рабочая копия удалена

Типы записей, перечисленные в табл. 14.26, генерируются командами **commit**.

Таблица 14.26. Типы записей истории, связанные с работой commit

Тип	Описание
A	Первоначальное добавление
M	Файл изменен
R	Файл удален

Типы записей, перечисленные в табл. 14.27, генерируются различными командами.

Таблица 14.27. Прочие типы записей в истории

Тип	Команда
E	**export**
F	**release**
O	**checkout**
T	**rtag**

import

```
import
  [ -b branch ]
  [ -d ]
  [ -I pattern ]
  [ -k kflag ]
  [ -m message ]
  [ -W spec ]
  module
  vendor_tag
  release_tag ...
```

Импортирование целого каталога в репозиторий в качестве нового модуля. Используется для добавления в проект кода из внешних источников либо произвольного кода, который изначально создавался вне системы контроля версий CVS. Может быть указано несколько идентификаторов изданий (версий) *release_tag*; в таком случае для начальной версии будет создано несколько меток.

Параметр **–k** имеет тот же смысл, что и обычно. Дополнительные параметры перечислены в табл. 14.28.

Таблица 14.28. Параметры import

Параметр	Описание
–b *branch*	Импортировать в указанную ветвь
–d	Использовать время и дату модификации файла вместо текущего времени и даты в качестве времени и даты импортирования. Может использоваться только при локальном доступе к репозиторию
–I *pattern*	Маски файловых имен, которые должны игнорироваться
–m *message*	Использовать *message* в качестве записи для журнала изменений вместо вызова редактора
–W *spec*	Спецификация фильтров

Параметр **–k** будет действовать только для файлов, импортируемых в текущем сеансе. Режимы подстановки ключевых слов для файлов, уже находящихся в репозитории, не меняются.

При указании параметра **–W** аргумент *spec* имеет такой же формат, как и записи в административном файле *cvswrappers* (см. предшествующий раздел «Файл cvswrappers»).

В табл. 14.29 перечислены коды состояния, отображаемые командой **import**.

Таблица 14.29. Коды состояний import

Состояние	Описание
C	Зарегистрировано изменение. Файл существует в репозитории, а версия в рабочем каталоге отличается; необходимо произвести слияние
I	Файл игнорируется в соответствии с определениями из файла *.cvsignore*
L	Ссылка. Символические ссылки игнорируются CVS
N	Новый. Файл является новым. Был добавлен в репозиторий
U	Обновление. Файл существует в репозитории, а версия в рабочем каталоге не отличается

log

```
log
  [ -b ]
  [ -d dates ]
  [ -h ]
  [ -N ]
  [ -rrevisions ]
  [ -R ]
  [ -s state ]
  [ -t ]
  [ -wlogins ]
  [ file ...]
```

Вывести журнал событий для всех файлов.

Параметр **–l** имеет тот же смысл, что и обычно. Дополнительные параметры перечислены в табл. 14.30.

Таблица 14.30. Параметры log

Параметр	Описание
–b	Перечислять версии ветви по умолчанию
–d *dates*	Сообщить информацию для указанных дат
–h	Вывести только заголовки
–N	Не отображать метки
–r[*revisions***]**	Отчет только по перечисленным версиям. Пробела между **–r** и аргументом нет. Параметр без аргумента является указанием использовать последнюю версию ветви по умолчанию
–R	Выводить только имя файла RCS. Использование **–R** в данном случае имеет иной смысл, чем во всех остальных командах CVS, где параметр **–R** является предписанием рекурсивного обхода
–s *state*	Выводить только версии, находящиеся в указанном состоянии
–t	Выводить только заголовки и текст описания
–w*logins*	Отчет по помещению в репозиторий файлов перечисленными пользователями. Пробела между **–w** и аргументом нет

С параметром **–d** следует использовать спецификации дат, приведенные в табл. 14.31. Допустимо указание нескольких спецификаций, разделенных точкой с запятой.

Таблица 14.31. Задание диапазона дат для log

Спецификация	Описание
d1<*d2* или *d2*>*d1*	Версии, датированные интервалом времени *d1*–*d2*, исключая значения *d1* и *d2*
d1<=*d2* или *d2*>=*d1*	Версии, датированные интервалом времени *d1*–*d2*, включая значения *d1* и *d2*
<*d* или *d*>	Версии, датированные временем до *d*
<=*d* или *d*>=	Версии, датированные временем до *d* или *d*
d< или >*d*	Версии, датированные временем после *d*
d<= или >=*d*	Версии, датированные временем после *d* или *d*
d	Самая последняя версия, датированная датой *d* или ранее

Спецификации версий для использования с **–r** приведены в табл. 14.32.

Таблица 14.32. Задание версий для log

Спецификация	Описание
rev1: rev2	Версии между *rev1* и *rev2* включительно
:rev	Версии от начала ветви до *rev* включительно
rev:	Версии от *rev* до конца ветви включительно
branch	Все версии ветви
branch1: branch2	Все версии всех ветвей между *branch1* и *branch2* включительно
branch.	Самая последняя версия ветви

Для диапазона *rev1:rev2* считается ошибкой, если версии не принадлежат к одной и той же ветви.

login

```
login
```

Команда используется для регистрации перед началом работы с удаленными репозиториями. Введенный пароль сохраняется в файле *~/.cvspass*, поскольку соединение с сервером не поддерживается между вызовами.

logout

```
logout
```

Команда используется для завершения работы с удаленным репозиторием. Пароль, сохраненный в файле *~/.cvspass*, удаляется.

rdiff

```
rdiff
  [ -c | -s | -u ]
  [ { { -D date1 | -r rev1 } [ -D date2 | -r rev2 ] } | -t ]
  [ -f ]
  [ -l | -R ]
  [-V vn]
  file ...
```

Команда **rdiff** создает **patch**-файл, который может использоваться для преобразования каталога одного издания в каталог другого.

Параметры **–D**, **–f**, **–l**, **–r** и **–R** имеют тот же смысл, что и обычно. Дополнительные параметры перечислены в табл. 14.33.

Таблица 14.33. Параметры rdiff

Параметр	Описание
–c	Использовать формат **context diff** (режим по умолчанию)
–s	Вывести сводку измененных файлов вместо создания файла **patch**
–t	Показать различия между двумя последними версиями
–u	Использовать формат **unidiff**
–V *rcsver*	Вышел из употребления. Использовался для указания версии RCS, для которой необходимо эмулировать подстановку ключевых слов. (Расширение слов соответствует RCS версии 5)

release

```
release
  [ -d ]
  directory ...
```

При желании рабочие каталоги могут быть удалены без применения **cvs release**; использование команды **release** приведет к занесению соответствующей записи в журнал событий (если этот механизм надлежащим образом настроен). Кроме того, будет проверено расположение (рекурсивно) каждого из файлов рабочего каталога перед удалением. Это поможет предотвратить случайное удаление работы, еще не внесенной в репозиторий.

С командой **release** можно использовать только один параметр: **–d**. Указание этого параметра приведет к удалению рабочей копии модуля в случае отсутствия не внесенных в репозиторий изменений.

Новые каталоги (включая и любые файлы в них) в рабочем каталоге будут удалены при использовании параметра **–d** с командой **release**.

Коды состояний, приведенные в табл. 14.34, используются для описания местоположения файлов в репозитории и рабочем каталоге.

Таблица 14.34. Коды состояний release

Состояние	Описание
A	Добавлен файл в рабочий каталог (файл был создан и выполнена команда **cvs add**), но он еще не внесен в репозиторий командой **commit**
M	Копия файла в рабочем каталоге изменилась
P U	Доступно обновление. В репозитории существует более новая версия файла, а копия в рабочем каталоге не изменялась
R	Копия из рабочего каталога была удалена (удален файл и выполнена команда **cvs remove**), но изменения не внесены в репозиторий
?	Файл существует в рабочем каталоге, но не в репозитории

remove

```
remove
  [ -f ]
  [ -l | -R ]
  [ file ...]
```

Указать CVS, что файлы должны быть удалены из репозитория. Пока изменения не внесены в репозиторий, файлы реально не удаляются. Если вы передумали, используйте **cvs add** для восстановления файлов, которые были удалены таким образом.

Параметры **–l** и **–R** имеют тот же смысл, что и обычно. С командой **remove** может использоваться только один дополнительный параметр: **–f**. При его использовании файл удаляется сначала из рабочего каталога.

rtag

```
rtag
  [ -a ]
  [ -b ]
  [ -d ]
  [ -D date | -r rev ]
  [ -f ]
  [ -F ]
  [ -l | -R ]
  [ -n ]
  tag
  file ...
```

Присвоить метку определенной версии или набору файлов. Если файл уже связан с меткой для другой версии, **cvs rtag** выдаст соответствующее сообщение, если только не использован параметр **–F**. Эта команда не работает с

версиями файлов в рабочем каталоге (для этой цели используйте **cvs tag**), поэтому ее можно при желании использовать и вне рабочего каталога.

Параметры **–D**, **–f**, **–l**, **–r** и **–R** имеют тот же смысл, что и обычно. Дополнительные параметры перечислены в табл. 14.35.

Таблица 14.35. Параметры rtag

Параметр	Описание
–a	Искать удаленные файлы с указанной меткой «на чердаке» (каталог *Attic*)
–b	Присвоить метку ветви
–d	Удалить метку
–F	Принудительно перенести метку с текущей версии на указанную
–n	Не выполнять никакие программы, связанные с меткой из файла *modules*

status

```
status
  [ -l | -R ]
  [ -v ]
  [ file ...]
```

Отобразить состояние файлов.

Параметры **–l** и **–R** имеют то же значение, что и обычно. С командой **status** может также использоваться параметр **–v**, для того чтобы включить информацию о метке.

tag

```
tag
  [ -b ]
  [ -c ]
  [ -d ]
  [ -D date | -r rev ]
  [ -f ]
  [ -F ]
  [ -l | R ]
  tag
  [ file ...]
```

Присвоить метку версиям или набору файлов из рабочего каталога. Для перечисления существующих для файла меток можно использовать команду **status –v**.

Метка *tag* должен начинаться с буквы и состоять только из букв, цифр, символов дефиса (-) и подчеркивания (_). Поэтому при желании присвоить проекту *hello* метку `1.0` при выпуске его первой версии вам придется использовать вместо этого какую-нибудь другую метку, например `hello-1_0`.

Параметры **–D**, **–f**, **–l**, **–r** и **–R** имеют то же значение, что и обычно. Дополнительные параметры перечислены в табл. 14.36.

Таблица 14.36. Параметры tag

Параметр	Описание
–b	Произвести ветвление
–c	Проверка изменений. Убедиться, что файлы не были локально изменены, прежде чем присваивать метку
–d	Удалить метку
–F	Принудительно перенести метку с текущей версии на указанную

Поскольку при использовании параметра **–d** удаляется информация, которая может оказаться важной, рекомендуется использовать этот параметр только в тех случаях, когда это абсолютно необходимо. Как правило, предпочтительнее создать другую метку с похожим именем.

unedit

```
unedit
  [ -l | -R ]
  [ file ...]
```

Прекратить редактирование файла и снова сделать его доступным только для чтения. Наблюдателям посылается уведомление. Параметры **–l** и **–R** имеют тот же смысл, что и обычно.

update

```
update
  [ -A ]
  [ -d ]
  [ -D date | -r rev ]
  [ -f ]
  [ -I pattern ]
  [ -j rev1 [ -j rev2 ] ]
  [ -k kflag ]
  [ -l | -R ]
  [ -p ]
  [ -P ]
  [ -W spec ]
  [ file ...]
```

Обновить рабочий каталог, произведя слияние файлов с версиями из репозитория. Например:

```
cvs -n -q update -AdP
```

может использоваться для быстрого сравнения состояния рабочего каталога с вершиной ветви.

Параметры **–D**, **–f**, **–k**, **–l**, **–r** и **–R** имеют тот же смысл, что и обычно. Дополнительные параметры перечислены в табл. 14.37.

Таблица 14.37. Параметры update

Параметр	Описание
–A	Удалить «липкие» метки
–d	Создать и обновить новые каталоги
–I *pattern*	Маски файловых имен, которые должны игнорироваться
–j *revision*	Слияние изменений двух версий. Мнемоника для *join*
–p	Извлечь файлы на стандартный вывод
–P	Удалить пустые каталоги
–W *spec*	Спецификации фильтров

Использование **–D** и **–r** приводит соответственно к созданию «липких» дат и меток для обрабатываемых файлов (а одновременное использование **–p** с этими параметрами предотвращает «липкость»). Используйте параметр **–A** для удаления «липких» дат или меток.

Если две версии задаются при помощи параметра **–j**, различия между этими версиями вычисляются и добавляются в текущий файл. Если задана только одна версия, тогда общий предок копии файла из рабочего каталога и указанной версии используется в качестве основы для поиска различий, которые должны быть внесены в файл.

Например, представим себе проект, в котором существует экспериментальная ветвь и важные изменения в файл *foo.c* были внесены в процессе перехода от версии 1.2.2.1 к версии 1.2.2.2. Как только изменения начинают считаться стабильными, может возникнуть необходимость отразить изменения в основной ветви проекта. Следует выполнить следующую команду в каталоге с извлеченной основной версией:

```
user@localhost$ cvs update -j 1.2.2.1 -j 1.2.2.2 foo.c
```

CVS найдет различия между двумя версиями и добавит разницу к файлу в рабочем каталоге.

Аргумент *spec* для параметра **–W** имеет такой же формат, как и для записей в административном файле *cvswrappers*. (см. предшествующий раздел «Файл cvswrappers»).

Коды состояний, приведенные в табл. 14.38, используются для описания действий, которые были выполнены с каждым файлов репозитория и рабочего каталога.

Таблица 14.38. Коды состояний update

Код состояния	Описание
A	Добавление. Сервер не выполнил никаких действий, поскольку файл не существует в репозитории. Была выполнена команда **cvs add**, но не **cvs commit**
C	Конфликт. Копия в рабочем каталоге изменилась (редактировалась после извлечения из репозитория или изменилась с момента выполнения последней команды **commit**). В репозитории новая версия файла. При слиянии версии репозитория с версией из рабочего каталога возникли конфликты
M	Изменение. Копия в рабочем каталоге изменилась (редактировалась после извлечения из репозитория или изменилась с момента выполнения последней команды **commit**). Если в репозитории существовала новая версия, изменения в ней были успешно внесены в файл из рабочего каталога (без конфликтов)
P	Наложение исправления. Идентично U, но с указанием на то, что сервер использовал **patch**
R	Удаление. Сервер не выполнил никаких действий. Была выполнена команда **cvs remove**, но не **cvs commit**
U	Обновление. Файл обновлен
?	Файл существует в рабочем каталоге, но не в репозитории

watch

```
watch
  { { on | off } | { add | remove } [ -a action ] }
  [ -l | -R ]
  file ...
```

Команда **watch** управляет механизмом слежения за работой с файлами. По умолчанию CVS действует в режиме параллельной разработки, позволяя любому пользователю редактировать любой файл в произвольный момент времени. В CVS включен механизм **watch** для поддержки разработчиков, которым удобнее немедленно узнавать об изменениях, внесенных другими разработчиками, а не в процессе выполнения команды **update**.

Файл *CVSROOT/notify* определяет, каким способом производится доставка уведомлений.

В табл. 14.39 отражены подкоманды **watch** и их использование.

Таблица 14.39. Подкоманды watch

Подкоманда	Описание
add	Начать наблюдение за файлами
off	Выключить наблюдение
on	Включить наблюдение
remove	Закончить наблюдение за файлами

Параметры **–l** и **–R** имеют тот же смысл, что и обычно. С командой **watch** может быть использован только один дополнительный параметр: **–a**. Параметр **–a** используется в сочетании с одним из действий *action*, перечисленных в табл. 14.40.

Таблица 14.40. Действия watch

Действие	Описание
all	Все действия
commit	Помещение изменений в репозиторий
edit	Выполнение пользователем **cvs edit**
none	Не производить наблюдения. Используется командой **edit**
unedit	Пользователь выполнил команду **cvs unedit**, **cvs release** или удалил файл и выполнил **cvs update**, создав его заново

См. также описания команд **edit**, **editors**, **unedit** и **watchers**.

watchers

```
watchers
  [ -l | -R ]
  [ file ...]
```

Вывести список пользователей, наблюдающих за указанными файлами. Наблюдение определяется на основе выполнения пользователем команды **watch** для определенного файла (или набора файлов). Если команда **watch** никем не используется, выводится пустой список.

Параметры **–l** и **–R** имеют тот же смысл, что и обычно.

См. также **watch**.

Инструмент RCS

Система управления версиями RCS (Revision Control System) разработана для отслеживания изменений многочисленных файлов и попутного сокра-

щения занимаемого проектом дискового пространства. При помощи RCS можно автоматически сохранять и восстанавливать версии проекта, совмещать или сравнивать версии, вести полный журнал изменений и определять версии по ключевым словам. В этой главе описана версия 5.7, а также отличия от более ранних версий.

Обзор команд RCS

Три наиболее важные команды RCS:

ci Добавить версию (передать файл под управление RCS).

co Получить версию.

rcs Установить или изменить атрибуты файлов RCS.

Следующие две команды отображают информацию о файлах RCS:

ident

Получить значения ключевых слов из файла RCS.

rlog

Отобразить журнал изменений файла RCS.

Сравнивать файлы RCS можно при помощи следующих команд:

rcsdiff

Отобразить различия между версиями.

rcsmerge

Совместить изменения двух файлов RCS в третьем файле.

А эта команда облегчает управление настройками:

rcsclean

Удалить рабочие файлы, которые не были изменены.

Основные операции RCS

Обычно файлы RCS хранятся в подкаталоге **RCS**, поэтому первый шаг при использовании RCS выглядит так:

```
mkdir RCS
```

Затем существующий файл (или файлы) передается под управление RCS выполнением команды добавления:

```
ci file
```

В результате создается файл с именем *file,v* в каталоге **RCS**. *file,v* называется *RCS-файлом*, и в нем будут храниться все будущие изменения указанного файла *file*. Если команда **ci** выполняется для файла впервые, вам будет пред-

ложено ввести описание содержания файла. Затем указанный файл помещается в RCS как версия 1.1.

Для редактирования новой версии скопируйте существующую версию:

```
co -l file
```

По этой команде RCS создает копию *file* из файла RCS. Вы должны заблокировать файл параметром **–l**, чтобы его можно было изменять. Эта копия называется рабочим файлом. Когда редактирование закончено, можно записать изменения повторным добавлением рабочего файла:

```
ci file
```

На этот раз вам будет предложено ввести описание внесенных изменений, а затем файл помещается на хранение как версия 1.2. Обратите внимание, что рабочий файл, как правило, удаляется при помещении версии в архив RCS. Чтобы получить копию только для чтения, скопируйте файл без блокирования:

```
co file
```

Это полезно, если необходимо иметь под рукой копию для поиска или компиляции. В качестве сокращения описанных ранее операций **ci**/**co** можно использовать команду:

```
ci -u file
```

Файл *file* будет добавлен и немедленно будет создана копия, предназначенная только для чтения. Для сравнения рабочего файла и его последней версии можно использовать команду:

```
rcsdiff file
```

Существует еще одна полезная команда **rlog**, которая отображает журнал записей изменений для файла.

Администратор системы может использовать команду **rcs** для настройки поведения по умолчанию для RCS.

Общие стандарты RCS

В этом разделе рассматриваются:

- Подстановка ключевых слов
- Нумерация версий
- Задание даты
- Задание состояний
- Стандартные параметры и переменные окружения

Подстановка ключевых слов

RCS позволяет помещать переменные (в виде ключевых слов) в рабочие файлы. Эти переменные позже могут расширяться в примечания к версиям. Примечания можно использовать как встроенные комментарии в исходном файле или как текстовые строки, отображающиеся при печати результата. Создание примечаний к версиям с помощью подстановок ключевых слов подразумевает выполнение следующей процедуры:

1. Добавьте в рабочий файл одно из перечисленных в следующем разделе ключевых слов.
2. Добавьте файл в проект RCS.
3. Извлеките добавленный файл. При этом команда **co** дополняет каждое из ключевых слов его значением. Таким образом, вхождения строки:

   $*keyword*$

   заменяются строками вида:

   $*keyword:value*$

4. Последующие добавления и извлечения файла обновляют все значения существующих ключевых слов. Существующие значения заменяются новыми, если нет указания на обратное.

У многих команд RCS есть параметр **–k**, обеспечивающий большую гибкость при подстановке ключевых слов.

Ключевые слова

$Author$

Имя пользователя, добавившего новую версию.

$Date$

Дата и время обновления.

$Header$

Название, включающее полный путь файла RCS, версию обновления, дату, имя автора, состояние и (при блокировке) имя пользователя, блокировавшего файл.

Id

Идентично **$Header$**, но исключая полный путь файла RCS.

$Locker$

Имя пользователя, который заблокировал версию. Если файл не блокирован, ключевое слово имеет пустое значение.

Log

Текст, который был создан при добавлении файла для описания его изменений. Текст предваряется именем файла RCS, номером версии, именем автора и датой. Описания изменений накапливаются, а не перезаписываются.

$RCSfile$

Имя файла RCS (без пути к нему).

$Revision$

Присвоенный номер версии.

$Source$

Имя файла RCS, включая путь.

$State$

Состояние, установленное параметром **–s** команды **ci** или **rcs**.

Примеры значений

Допустим, файл */projects/new/chapter3* был добавлен и извлечен пользователем с именем *daniel*. Вот каким будет результат подстановки ключевых слов во второй по счету версии файла:

```
$Author: daniel $

$Date: 2000/02/25 18:21:10 $

$Header: /projects/new/chapter3,v 1.2 2000/02/25 18:21:10 daniel \
    Exp Locker: daniel $

$Id: chapter3,v 1.2 2000/02/25 18:21:10 daniel Exp Locker: daniel $

$Locker: daniel $

$Log: chapter3,v $
#Revision 1.2  2000/02/25  18:21:10   daniel
# Added section on error-handling
#
#Revision 1.1  2000/02/25  16:49:59   daniel
# Initial revision
#

$RCSfile:  chapter3,v $

$Revision: 1.2 $

$Source: /project/new/chapter3,v $

$State: Exp $
```

Нумерация версий

Команды RCS работают с последней версией файла, если нет явных указаний на иное. У некоторых из команд существует параметр **–r**, который используется для задания номера версии. Кроме того, со многими параметрами номер версии может указываться в качестве дополнительного аргумента. (В описаниях команд этот аргумент отображен как [*R*]). Номера версий могут состоять из четырех полей: издание (release), уровень, ветвь и порядковый номер, но обычно состоят только из двух – издания и уровня.

Например, извлечь версию 1.4 можно следующим образом:

```
co -l -r1.4 ch01
```

Когда эта версия снова будет добавлена в проект, она получит номер версии 1.5. Но можно сделать измененный файл следующим изданием. Например:

```
ci -r2 ch01
```

Будет создана версия 2.1 файла. Вы также можете создать ответвление предыдущей версии. Следующая команда создает версию 1.4.1.1:

```
ci -r1.4.1 ch01
```

Нумерация, впрочем, не является единственным способом обозначения версий. Возможно задание имен версий в виде текстовой метки посредством использования параметра **–n** команд **ci** или **rcs**. Также возможно задание этого имени в качестве аргумента любого параметра, принимающего номер версии. Например, вы можете добавлять свои программы на языке C, используя одну и ту же метку независимо от номера текущей версии:

```
ci -u -nPrototype *.c
```

Кроме того, начиная с версии RCS 5.6, существует возможность задания аргумента $, который предписывает создавать номер версии на основе значений ключевых слов рабочего файла. Например:

```
rcsdiff -r$ ch01
```

сравнивает файл *ch01* с поступившим обновлением. Также можно использовать имена вместе со специальными символами. Команда

```
rcs -nDraft:$ ch*
```

присваивает имя номерам версий, связанным с файлами нескольких глав.

Задание даты

Отметка времени для добавляемых версий производится на основе текущей даты и времени. Строки значений некоторых ключевых слов содержат дату. Даты могут передаваться в параметрах командам **ci**, **co** и **rlog**. По умолчанию в RCS используется следующий формат задания даты:

1999/10/16 02:00:00 *(год/месяц/день время)*

Часовой пояс по умолчанию – GMT (Greenwich Mean Time), время которого известно также как UTC (Coordinated Universal Time). Даты можно задавать в произвольном формате. Это позволяет использовать различные стили. Вот некоторые из самых распространенных вариантов, в которых отображено то же время, что и в вышеприведенном примере:

```
6:00 pm lt              (предполагается, что текущая дата – октябрь, 16, 1999)
2:00 AM, Oct. 16, 1999
Sat Oct 16 18:00:00 1999 LT
Sat Oct 16 18:00:00 PST 1999
```

Сочетание букв *lt* (в любом регистре) является указанием на местное время (в данном случае Pacific Standard Time). В третьей строке представлен формат **ctime** (плюс *LT*); четвертая строка соответствует формату команды **date**.

Задание состояний

В некоторых ситуациях, в особенности, если речь идет о проекте разработки программного обеспечения, существует необходимость получить сведения о текущем состоянии набора версий. Файлы RCS связываются со строками текста, описывающими состояние (*state*) этих файлов. Состоянием по умолчанию является **Exp** (экспериментальное). Среди прочих распространенных вариантов – **Stab** (стабильное) и **Rel** (завершенное). Эти слова определяются пользователем и не имеют специального внутреннего значения в RCS. Некоторые значения ключевых слов включают строки состояния. Кроме того, существует возможность передачи состояний в качестве аргументов параметров команд **ci**, **co**, **rcs** и **rlog**.

Стандартные параметры и переменные окружения

В RCS определяется переменная окружения RCSINIT, используемая для настройки параметров по умолчанию для команд RCS. Список параметров (перечисляемых через пробел), содержащийся в этой переменной, будет предварять параметры любой выполняемой команды RCS. В RCSINIT полезно включать три параметра: **–q**, **–V** и **–x**. Эти параметры можно назвать стандартными, поскольку они принимаются большинством команд RCS. Обратите внимание, что параметр **–V** является новым для версии RCS 5, а параметр **–x** появился в версии 5.6.

–q[*R*]

Тихий режим, без диагностики действий. Аргумент *R* определяет версию файла.

–V[*n*]

Эмуляция указанной (*n*) версии RCS. Параметр полезен при обмене файлами с системами, использующими другие версии RCS. *n* может принимать значения 3, 4 или 5. Если номер версии не указан, отображается текущая версия RCS.

−x*suffixes*

Указать альтернативный список суффиксов *suffixes* файлов RCS. Суффиксы разделяются символом /. В системах Unix файлы RCS обычно заканчиваются символами *,v*. Параметр **−x** предоставляет возможность изменить суффикс для тех систем, в которых запятая (,) в имени файла не допустима.

−z[*zone*]

Указать формат даты для подстановки ключевых слов. Параметр без аргумента приводит к выводу времени в формате UTC без указания часового пояса. Если указан аргумент *LT*, используется местный часовой пояс для вывода времени в формате ISO 8601 с указанием разницы с UTC-временем. Можно также определить численную разницу с UTC. Например, результатом задания **−z+4:30** будет строка типа:
1998-11-24 02:30:00+4:30.

Например, при помещении рабочего файла в файл RCS команда:

```
ci -x,v/  ch01      (второй суффикс является пустым)
```

производит поиск файлов RCS в следующем порядке:

```
RCS/ch01,v
ch01,v
RCS/ch01
```

Алфавитный перечень команд RCS

Подробную информацию о синтаксисе ключевых слов, номерах версий, датах, состояниях и стандартных параметрах можно найти в предыдущих разделах.

ci [*options*] *files* — **ci**

Добавление версий. **ci** сохраняет содержимое указанных рабочих файлов *files* в соответствующих файлах RCS. Обычно **ci** удаляет рабочий файл после его добавления. Если RCS-файл не существует, тогда рабочий файл является начальной версией. В этом случае создается RCS-файл, а пользователю предлагается ввести описание файла. Если RCS-файл существует, **ci** увеличивает номер версии и предлагает ввести текст, описывающий последние сделанные изменения. Начиная с RCS версии 5.6, при добавлении неизмененного файла версия остается прежней. В RCS предшествующих версий существует возможность добавить новую версию, ничем не отличающуюся от предыдущей.

Чаще всего используются взаимоисключающие параметры **−u**, **−l** и **−r**. Параметр **−u** позволяет сохранить копию рабочего

ci

файла в режиме только для чтения (например, для выполнения компиляции или поиска по файлу). Параметр **–l** используется для обновления версии и последующего немедленного извлечения файла с блокировкой. Это позволяет сохранять промежуточные изменения и продолжать работу с файлом (например, в длительном сеансе редактирования). Используйте **–r** для добавления файла с другим номером издания. **ci** работает со стандартными параметрами **–q**, **–V**, **–x** и **–z**.

Параметры

–d[*date*]

Добавить файл с отметкой времени *date* или, если дата не указана, с информацией о времени последнего изменения.

–f[*R*]

Принудительно добавить версию, даже если изменений нет.

–I[*R*]

Диалоговый режим. Запрашивать данные у пользователя, даже если стандартным вводом не является терминал (например, если **ci** является частью конвейера). Параметр **–I** появился в версии RCS 5.

–i[*R*]

Создать (инициализировать) файл RCS и добавить его. Выдается предупреждение, если файл RCS уже существует.

–j[*R*]

Добавить файл без инициализации. Выдается сообщение об ошибке, если файл RCS еще не существует.

–k[*R*]

Присвоить номер версии, дату создания, состояние и имя автора в соответствии со значениями ключевых слов в рабочем файле, а не использовать данные, полученные из локального окружения. Параметр **–k** полезен для распределения программного обеспечения: ключевые слова являются отметками времени изменения, разделяемого всеми сайтами.

–l[*R*]

Выполнить **co –l** после добавления файла. Таким образом сохраняется заблокированный файл следующей версии.

−m*msg*

Использовать строку *msg* в качестве журнальной записи для всех добавляемых файлов. При добавлении нескольких файлов **ci** обычно запрашивает подтверждение на использование журнальной записи предыдущего файла. Параметр **−m** отменяет такие запросы.

−M[*R*]

Установить время изменения рабочего файла в соответствующее значение времени для извлеченного файла. Параметр **−M** (появился в RCS версии 5.6) может затруднить работу **make** и должен использоваться с осторожностью.

−n*name*

Связать текст *name* с новым номером версии.

−N*name*

Аналогично **−n**, но отменяет предыдущее определение *name*.

−r[*R*]

Добавить файл с номером версии *R*.

−r Сам по себе параметр устанавливает поведение по умолчанию при снятии блокировки и удалении рабочего файла. Этот параметр имеет больший приоритет, чем любой из параметров **−l** или **−u**, инициализированных псевдонимами интерпретатора или сценариями. Такое поведение **−r** специфично для **ci**.

−s*state*

Указать состояние *state* добавляемой версии.

−T Привести время изменения файла RCS в соответствие со временем изменения последней версии, если файл RCS не обновлялся позже.

−t*file*

Заменить описание файла RCS содержимым указанного файла *file*. В версии 5 это работает только для первой версии файла.

−t−*string*

Заменить описание файла RCS строкой *string*. В версии 5 это работает только для первой версии файла.

ci

−u[*R*]

Выполнить **co −u** после добавления файла. Это позволяет сохранить копию файла только для чтения.

−w*user*

Установить значение поля автора в *user* для добавляемой версии.

Примеры

Добавить файлы глав, используя одинаковые записи в журнале:

```
ci -m'First round edits' chap*
```

Добавить измененный файл **prog.c**, сохранив копию только для чтения:

```
ci -u prog.c
```

Начать уровень издания 2; создать синоним «Prototype» для версии 2.1:

```
ci -r2 -nPrototype prog.c
```

co

co [*options*] *files*

Извлечь одну из предварительно добавленных версий файла в соответствующий рабочий файл (либо вывести на стандартный вывод, если указан параметр **−p**). Если вы намереваетесь отредактировать рабочий файл и вновь добавить его, необходимо указать параметр **−l** для блокировки файла. **co** работает со стандартными параметрами **−q**, **−V** и **−x**.

Параметры

−d*date*

Извлечь самую позднюю версию, время добавления которой соответствует указанной дате *date* или ранее.

−f[*R*]

Принудительная перезапись рабочего файла.

−I[*R*]

Диалоговый режим; запрашивать ввод пользователя, даже если стандартным вводом не является терминал. (Параметр появился в RCS версии 5.)

–j*R2:R3*

Действие аналогично работе **rcsmerge**. Аргументы *R2* и *R3* определяют две версии, чьи изменения объединяются в третий файл: либо соответствующий рабочий файл, либо третью версию (любой аргумент *R*, заданный другими параметрами команды **co**).

–k*c*

Расширение ключевых слов в соответствии с флагом *c*, который может принимать следующие значения:

b Аналогично **o**, но действует в двоичном режиме, создавая ключевые слова предыдущей версии и их значения в двоичной форме.

k Расширение ключевых слов в имена полей (без значений). Полезно для пропуска примитивных различий при сравнении файлов.

kv Расширение ключевых слов в имена полей и значения (режим по умолчанию). Вставка имени пользователя, заблокировавшего файл, происходит только при выполнении **ci –l** или **co –l**.

kvl Аналогично **kv**, но вставка имени пользователя, заблокировавшего файл, происходит всегда.

o Расширение ключевых слов в имена полей и значения, присутствующие в предыдущей версии. Полезно при работе с двоичными файлами, в которых недопустимы замены подстрок.

v Расширение ключевых слов в значения (без собственно ключевых слов в качестве имен полей). Это предотвращает последующую подстановку ключевых слов, и поэтому не рекомендуется к использованию.

–l[*R*]

Аналогично **–r**, но извлеченная версия блокируется.

–M[*R*]

Установить время изменения рабочего файла в значение времени извлеченного файла. Параметр **–M** (появился в RCS версии 5.6) может затруднить работу **make**, и его следует использовать осторожно.

–p[*R*]

Отправить извлеченную версию на стандартный вывод, а не записывать ее в рабочий файл. Параметр полезен для перенаправления вывода или фильтрации.

co

−r[*R*]

Извлечь самую последнюю версию или, если задано *R*, последнюю версию не старше *R*.

−s*state*

Извлечь последнюю версию, имеющую указанное состояние *state*.

−T Сохранить время изменения RCS-файла, даже если блокировка добавляется или снимается.

−u[*R*]

Аналогично **−r**, но извлеченная версия разблокируется, если была заблокирована ранее.

−w[*user*]

Извлечь последнюю версию, которая была добавлена либо текущим пользователем, либо указанным (*user*).

Примеры

Выполнить сортировку последней сохраненной версии файла *file*:

```
co -p file | sort
```

Извлечь (и заблокировать для редактирования) все файлы с именами в верхнем регистре:

```
co -l [A-Z]*
```

Обратите внимание, что расширение маски имен файлов не даст результатов, если рабочая копия не находится в текущем каталоге. Поэтому этот пример работает, только если файлы были предварительно добавлены командой **ci −u**. И, наконец, вот еще несколько различных способов извлечь сразу несколько рабочих файлов (в текущем каталоге):

```
co -r3 *,v                  последние версии 3-го издания
co -r3 -wjim *,v            то же, но сделанное пользователем jim
co -d'May 5, 2 pm LT' *,v   последние версии, которые были созданы в указанную дату или ранее
co -rPrototype *,v          последние версии с именем Prototype
```

ident

ident [*option*] [*files*]

Извлечение комбинаций ключевое слово/значение из указанных файлов (*files*). Файлы могут быть текстовыми, объектными или файлами дампов.

ident

Параметр

−q Не выводить предупреждение, если требуемая информация по ключевым словам не найдена.

Примеры

Если файл **prog.c** скомпилирован и содержит следующую строку кода:

```
char rcsID[] = "$Author: george$"
```

выводится следующий результат:

```
% ident prog.c prog.o
prog.c:
     $Author: george $
prog.o:
     $Author: george $
```

Отобразить ключевые слова для всех файлов RCS (предупреждения не выводить):

```
co -p RCS/*,v | ident -q
```

rcs

rcs [*options*] <*files*

Административная команда для установки или замены атрибутов по умолчанию файлов RCS. Среди прочего **rcs** позволяет устанавливать жесткую блокировку (**−L**), удалять версии (**−o**), а также обходить блокировку, установленную **co** (параметры **−l** и **−u**). Для файлов RCS существует список доступа (создаваемый указанием параметра **−a**), в котором перечислены пользователи, имеющие право запускать **rcs**. Зачастую список доступа пуст; это означает, что **rcs** могут использовать все. Кроме того, вы всегда можете использовать **rcs**, если являетесь владельцем файла, привилегированным пользователем или выполняете **rcs** с параметром **−i**. **rcs** работает со стандартными параметрами **−q**, **−V**, **−x** и **−z**.

Параметры

−a*users*

Добавить пользователей *users* (перечисляются через запятую) в список доступа.

−A*otherfile*

Добавить список доступа *otherfile* к списку доступа для файлов *files*.

rcs

–b[*R*]

Установить ветвь версий по умолчанию в *R* или, если аргумент опущен, в самую высокую ветвь издания.

–c'*s*'

Использовать строку *s* в качестве символа комментария для ключевого слова **$Log**. По умолчанию **co** расширяет найденные ключевые слова **$Log** в комментарии, предваряемые символом #. Вы можете, например, установить *s* в значение .\" для файлов **troff** или установить *s* в значение * для программ на C. (Охватывающие последовательности /* и */ придется вставлять вручную до и после **$Log**.)

–e[*users*]

Удалить всех пользователей (или только указанного *users*) из списка доступа.

–i Создать (инициализировать) файл RCS, но не помещать в него первую версию.

–I Диалоговый режим. Запрашивать ввод пользователя, даже если стандартным вводом не является терминал. (Параметр появился в версии RCS 5.)

–k*c*

Использовать стиль *c* в качестве стиля по умолчанию для подстановки ключевых слов. (См. возможные значения *c* в описании команды **co**.) Указание параметра **–kkv** восстанавливает исходный стиль подстановок; все прочие стили не являются совместимыми с версией RCS 4 или более ранними.

–l[*R*]

Заблокировать версию *R* или последнюю версию. **–l** создает «ретро-блокировку», что полезно, если пользователь по ошибке произвел неверное извлечение версии командой **co** вместо **co –l**.

–L Включить жесткую блокировку (режим по умолчанию). Это означает, что все пользователи, включая и владельца RCS-файла, должны использовать **co –l** для редактирования файла. Жесткая блокировка рекомендуется при совместном редактировании файлов (см. **–U**).

–m*R:msg*

Использовать строку *msg* для замены записи в журнале, относящейся к версии *R*. (Параметр появился в RCS версии 5.6.)

−M Отключить посылку почтовых уведомлений при нарушении блокировки командой **rcs −u**. Параметр следует использовать только тогда, когда существуют иные способы предупредить пользователей о разблокировке их файлов.

−n*flags*

Добавить или удалить связь между версией и именем. Аргумент *flags* может принимать значения:

name:R

Связать имя *name* с версией *R*.

name:

Связать имя *name* с последней версией.

name

Удалить связь с именем *name*.

−N*flags*

Аналогично **−n**, но существующие имена *names* перезаписываются.

−o*R_list*

Удалить (считать устаревшими) версии, перечисленные в списке *R_list*. *R_list* может иметь вид: *R1*, *R1−R2*, *R1−* или *−R2*. Если задана ветвь, **−o** удаляет только последнюю версию этой ветви. В RCS версии 5.6 разделитель диапазона изменился на символ «:», однако символ «−» по-прежнему допустим.

−s*state*[:*R*]

Установить состояние (*state*) версии *R* (или последней версии).

−t[*file*]

Заменить описание файла RCS содержимым файла *file* или, если имя файла не указано, стандартным выводом.

−t−*string*

Заменить описание файла RCS строкой *string*. Время изменения файла RCS остается прежним, если версия не удаляется.

−T Сохранять время изменения файла RCS.

−u[*R*]

Дополнение **−l**: разблокировать версию, которая предварительно была извлечена командой **co −l**. Если извлече-

rcs

ние было произведено кем-то другим, вам будет предложено ввести причину нарушения блокировки. Сообщение отправляется по электронной почте пользователю, заблокировавшему файл.

−U Включить мягкую блокировку. Все, кроме владельца файла, должны использовать **co −l** для его редактирования (см. **−L**).

−V Вывести номер версии RCS.

−z*zone*

Установить часовой пояс по умолчанию для временных отметок, создаваемых командами **ci** и **co**.

Примеры

Связать метку **To_customer** с последней версией всех файлов RCS:

```
rcs -nTo_customer: RCS/*
```

Добавить трех пользователей в список доступа для файла **beatle_deals**:

```
rcs -ageorge,paul,ringo beatle_deals
```

Удалить версии с 1.2 по 1.5:

```
rcs -o1.2-1.5 doc
```

Заменить описание файла RCS значением переменной:

```
echo "$description" | rcs -t file
```

rcsclean

rcsclean [*options*] [*files*]

Сравнить извлеченные файлы с соответствующей последней версией или версией *R* (в зависимости от параметров командной строки). Если различия отсутствуют, рабочий файл удаляется. (Для поиска различий используйте **rcsdiff**.) Команда **rcsclean** полезна для файлов сборки (makefiles). Например, можно создать цель «clean-up» для обновления ваших каталогов. **rcsclean** также полезно использовать перед выполнением **rcsfreeze**. **rscclean** работает со стандартными параметрами **−q**, **−V**, **−x** и **−z**.

rcsclean

Параметры

−k*c*

При сравнении версий производить подстановку ключевых слов с использованием стиля *c* (значения стилей *c* см. в описании команды **co**).

−n[*R*]

Произвести диагностику действий, но реально ничего не делать.

−r[*R*]

Сравнивать с версией *R*. Версия может быть указана в качестве аргумента других параметров, поэтому параметр **−r** является избыточным.

−T Сохранять время изменения RCS-файла даже при наложении или снятии блокировки.

−u[*R*]

Разблокировать версию, если она идентична рабочему файлу.

Пример

Удалить неизмененные копии программы и заголовочных файлов:

```
rcsclean *.c *.h
```

rcsdiff

rcsdiff [*options*] [*diff_options*] *files*

Сравнение версий посредством **diff**. Версии указываются с помощью параметра **−r** следующим образом:

Количество указанных версий:	Производимое сравнение:
Отсутствуют	Рабочий файл сравнивается с последней версией
Одна	Рабочий файл сравнивается с указанной версией
Две	Версии сравниваются между собой

rcsdiff работает со стандартными параметрами **−q**, **−T**, **−V**, **−x** и **−z**, а также с параметрами *diff_options*, которые являются допустимыми параметрами **diff**. Выполнение **rcsdiff** завершается с кодом 0 (различия не найдены), 1 (различия существуют) или 2 (сбой по неизвестной причине).

rcsdiff

Параметры

−k*c*

При сравнении версий использовать указанный стиль *c* подстановки ключевых слов. (Значения стилей *c* см. в описании команды **co**.)

−r*R1*

Использовать версию *R1* для сравнения.

−r*R2*

Использовать версию *R2* для сравнения (**−r*R1*** также должна быть указана).

rcsmerge

rcsmerge [*options*] *file*

Тройственное слияние версий файлов. Две различные версии сравниваются, и все различия заносятся в рабочий файл (*file*). Необходимо указать одну или две версии для слияния (как правило, с помощью параметра **−r**). Наложения изменений (противоречия) обрабатываются тем же способом, что и в **merge**, путем помещения предупреждений в тело получаемого файла. **rcsmerge** работает со стандартными параметрами **−q**, **−V**, **−x** и **−z**. Выполнение **rcsmerge** завершается с кодом 0 (противоречия отсутствуют), 1 (существуют противоречия) или 2 (сбой по неизвестной причине).

Параметры

−k*c*

При сравнении версий использовать указанный стиль *c* подстановки ключевых слов. (Значения стилей *c* см. в описании команды **co**.)

−p[*R*]

Послать версию, полученную путем слияния, на стандартный вывод вместо перезаписи файла *file*.

−r[*R*]

Слияние с версией *R* или, если *R* опущено, слияние с последней версией.

Примеры

Предположим, существует необходимость добавить обновления старой версии (1.3) файла *prog.c*, но текущая версия файла уже 1.6. Изменения добавляются следующим образом:

rcsmerge

```
co -l prog.c
(редактируем последнюю версию добавлением обновлений
к версии 1.3, затем:)
rcsmerge -p r1.3 -r1.6 prog.c > prog.updated.c
```

Отменить изменения между версиями 3.5 и 3.2 и перезаписать рабочий файл:

```
rcsmerge -r3.5 -r3.2 chap08
```

rlog

rlog [*options*] *files*

Отображение идентифицирующей информации для RCS-файлов (*files*), включая запись в журнале, связанную с каждой версией, количество добавленных или удаленных строк, дату последнего добавления и т. д. Команда **rlog** без параметров отображает всю информацию. Воспользуйтесь параметрами для получения определенных данных. **rlog** работает со стандартными параметрами **–T**, **–V**, **–x** и **–z**.

Параметры

–b Сократить вывод; выводить информацию только о ветви по умолчанию.

–d*dates*

Вывести информацию о версиях, чьи даты добавления попадают в диапазон *dates* (список элементов разделяется точкой с запятой). Не забывайте использовать кавычки. Каждая дата может быть задана в виде:

date1 < *date2*

Выбрать версии между *date1* и *date2*, включая и собственно указанные.

date1 <

Выбрать версии, добавленные в день *date1* или позже.

date1

Выбрать версии, добавленные в день *date1* или ранее.

–h Отобразить начало обычного вывода **rlog**.

–l[*users*]

Вывести информацию только о блокированных версиях или, если указаны блокирующие пользователи (*lockers*), только о версиях, заблокированных этими пользователями (*users*).

–L Пропускать файлы, которые не заблокированы.

rlog

−N Не отображать метки.

−r[*list*]

Отображать информацию о конкретных версиях из списка *list* (элементы разделяются запятыми). Если список опущен, используется последняя версия. Элементы списка могут задаваться следующим образом:

R1 Выбрать версию *R1*. Если *R1* является ветвью, выбрать все версии этой ветви.

R1 . Если *R1* является ветвью, выбрать ее последнюю версию.

R1−*R2*

Выбрать версии с *R1* по *R2*.

−*R1* Выбрать версии от начала ветви и до версии *R1*.

R1− Выбрать версии от *R1* и до конца ветви.

В RCS версии 5.6 в качестве разделителя диапазона используется двоеточие (:), хотя и символ «−» (дефис) по-прежнему допустим.

−R Отобразить только имя файла RCS.

−s*states*

Отобразить информацию о версиях, чье состояние соответствует одному из состояний, перечисленных через запятую в списке *states*.

−t Аналогично **−h**, но отображается также и описание файла.

−w[*users*]

Отобразить информацию о версиях, добавленных каким-либо пользователем из списка *users* (перечисление в списке через запятую). Если имена пользователей *users* не указаны, используется имя текущего пользователя.

Примеры

Отобразить историю версии файла:

```
rlog RCS/*,v | more
```

Отобразить имена файлов RCS, которые заблокированы пользователем *daniel*:

```
rlog -R -L -ldaniel RCS/*
```

Отобразить часть «title» (без истории версии) рабочего файла:

```
rlog -t calc.c
```

Глава 15

GNOME

GNOME (GNU Network Object Model Environment, среда сетевой объектной модели GNU) – это дружественная к пользователю графическая среда, отвечающая за внешний вид рабочего стола и обеспечивающая единый способ взаимодействия между приложениями. GNOME – одна из двух популярных графических сред, распространенных в Linux. Она является стандартной в Red Hat, Debian и других популярных дистрибутивах. Вторая популярная среда для Linux носит название KDE и рассматривается в главе 16 «KDE».

GNOME не является оконным менеджером. Будучи графической средой, GNOME позволяет до мельчайших деталей настраивать пользовательский интерфейс и обеспечивает согласованную функциональность стандартных элементов графического интерфейса, таких как меню, панели инструментов и кнопки. Являясь пользовательской средой, GNOME использует растущее число GNOME-приложений для создания эффективного рабочего процесса.

Одной из самых интересных особенностей среды GNOME является архитектура, основанная на технологии CORBA, которая организует взаимодействие приложений на уровне разделения и встраивания используемых объектов. CORBA (Common Object Request Broker Architecture) – это общая архитектура брокеров запросов к объектам. Она определяет методы, используемые для взаимодействия объектов посредством брокера запросов (ORB, object request broker). Пакет ORB в настоящее время входит в GNOME под именем ORBit (*http://www.labs.redhat.com/orbit*). ORBit реализует функциональность, аналогичную той, что предоставляется Windows-технологиями COM и OLE. Например, таблица, созданная приложением **gnumeric** (программа GNOME для создания электронных таблиц), может стать объектом, встроенным в документ редактора AbiWord.

GNOME предоставляет оконному менеджеру управление особенностями окружения системы X Window. Оконный менеджер управляет размещением,

передвижением и внешним видом окон на экране пользователя. GNOME можно использовать с любым оконным менеджером, но при этом он должен быть совместимым с этой системой, чтобы обеспечить использование таких возможностей, как, например drag-and-drop. Самыми распространенными оконными менеджерами являются Sawfish и Enlightenment, они полностью совместимы со средой GNOME. Помимо них можно использовать оконные менеджеры IceWM, WindowMaker и AfterStep.

Обзор рабочего стола

На рис. 15.1 представлен вид рабочего стола среды GNOME по умолчанию. В левой части экрана размещаются пиктограммы, являющиеся ссылками для запуска приложений. Верхняя пиктограмма представляет собой символическую ссылку на домашний каталог пользователя, и двойной щелчок по ней открывает файловый менеджер GNOME, отображающий содержимое этого каталога. Другие пиктограммы соответствуют дисководу и приводу CD-ROM, а остальные являются ссылками на веб-страницы. Пиктограммы на рабочем столе могут использоваться для запуска любой программы в системе, для запуска приложения, соответствующего файлу, либо для просмотра содержимого каталога или страницы по адресу URL.

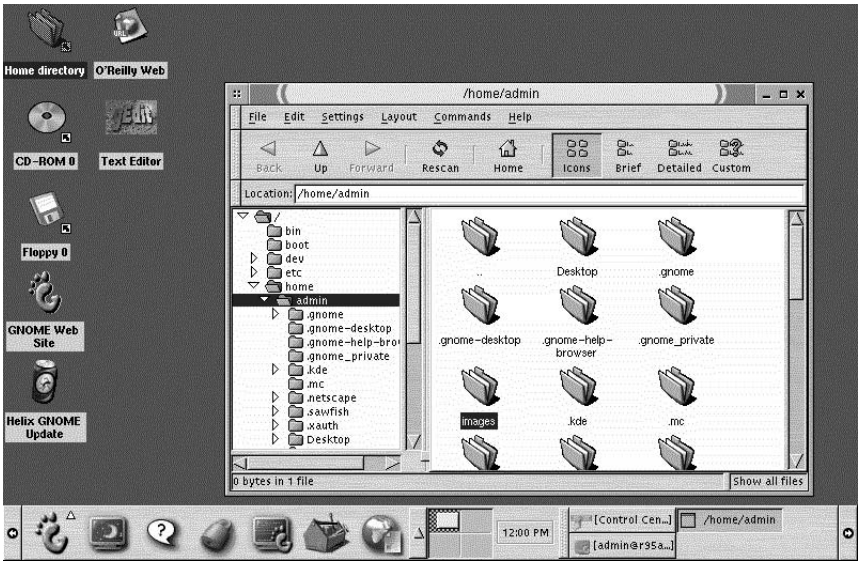


Рис. 15.1. Рабочий стол GNOME

В нижней части экрана находится панель GNOME, являющаяся основным средством навигации, запуска приложений и управления рабочим столом. Панель содержит кнопки вызова главного меню, справки, инструментов

настройки и броузера Netscape. На панель управления можно добавить кнопки, запускающие любые приложения в системе.

Кроме того, на панели запущены две специальные программы (называемые *апплетами*), которые помогают ориентироваться в пространстве рабочего стола. Апплет Desk Guide отображает пространство рабочего стола. Многие оконные менеджеры позволяют разделять рабочее пространство на ряд различных экранов (называемых *виртуальными рабочими столами* или *окнами просмотра*). Desk Guide обеспечивает микроотображение доступных рабочих столов и очертания окон, открытых на этих рабочих столах. Щелчок по уменьшенной версии рабочего стола приводит к переключению активной области просмотра.

Апплет Tasklist помогает отслеживать открытые окна. Для каждого открытого окна на панели отображается кнопка, щелчок по которой переключает фокус на выбранное окно или распахивает его, если окно находилось в минимизированном состоянии.

GNOME предоставляет невероятную гибкость в настройке среды рабочего стола. Щелчок правой кнопкой мыши практически на любом элементе вызывает всплывающее меню (называемое *контекстным меню*), которое содержит специальные действия, ассоциированные с данным элементом, и средство настройки его свойств. Общие установки настроек можно изменять из центра управления – GNOME Control Center. Получить доступ к этому инструменту можно следующим образом: щелкнуть по кнопке ящичка с инструментами на панели или выбрать в главном меню пункт Settings (Настройки), а затем GNOME Control Center (настройка отдельных приложений также доступна из этого меню).

Добавление пиктограмм на рабочий стол

Пиктограммы на рабочем столе предоставляют удобный (по двойному щелчку) доступ к наиболее важным файлам, приложениям и ссылкам. Элементы, отображаемые на рабочем столе, существуют в виде файлов в подкаталоге *.gnome-desktop* домашнего каталога пользователя. Все, что вы добавите в этот каталог, появится на рабочем столе.

Контекстное меню рабочего стола содержит пункт New, позволяющий добавлять различные типы элементов на рабочий стол.

Для того чтобы добавить пиктограмму, запускающую приложение, выберите New → Launcher (Кнопка запуска). В результате откроется диалоговое окно свойств элемента рабочего стола, представленное на рис. 15.2. Введите название пиктограммы (текст, отображаемый под пиктограммой), комментарий (текст всплывающей подсказки, появляющийся при наведении на пиктограмму указателя мыши) и собственно команду, используемую для запуска приложения. После нажатия кнопки OK новая пиктограмма запуска появляется на рабочем столе.

Для того чтобы добавить на рабочий стол пиктограмму, открывающую каталог, выберите New → Directory в контекстном меню рабочего стола. Введите

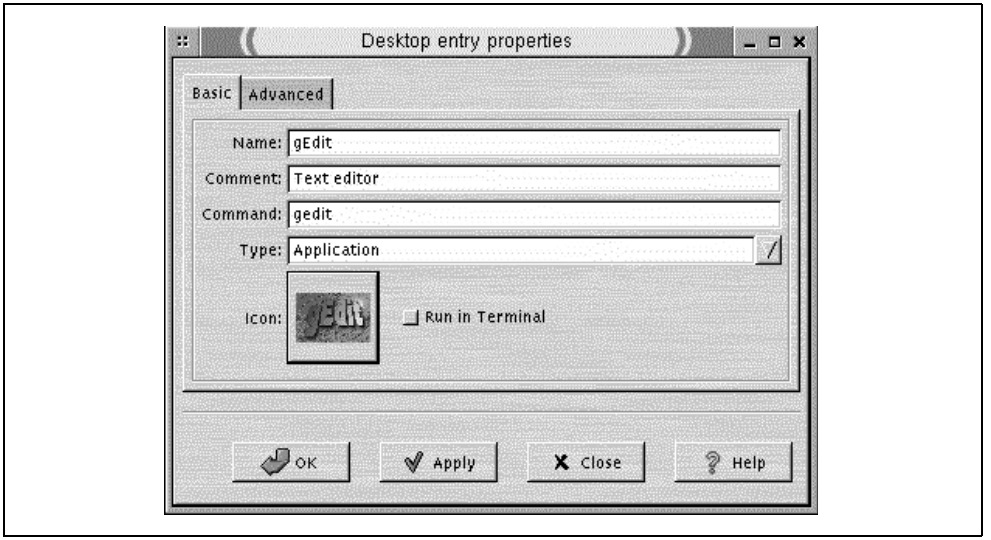


Рис. 15.2. Свойства элемента рабочего стола

имя каталога в диалоговом окне. Если указать абсолютное имя каталога, пиктограмма рабочего стола будет являться символической ссылкой. В случае относительного имени новый каталог будет создан в подкаталоге *~/.gnome-desktop*.

Для того чтобы добавить на рабочий стол ссылку на URL, выберите New → URL Link в контекстном меню рабочего стола. В открывшемся диалоговом окне необходимо ввести связываемый с пиктограммой URL и название пиктограммы. Нажмите OK, и пиктограмма появится на рабочем столе. Вы также можете щелкнуть и перетащить на рабочий стол любую ссылку, отображаемую в броузере Netscape.

Разумным применением пиктограмм на рабочем столе является создание ссылок на часто используемые файлы и каталоги. Ссылки проще всего создавать при помощи файлового менеджера (*gmc*). Если щелкнуть по элементу в файловом менеджере и перетащить его на рабочий стол, будет создана ссылка запуска для этого элемента. По существу это действие перемещает элемент в каталог *~/.gnome-desktop*. Если во время выделения и перемещения элемента удерживать клавишу <Ctrl>, элемент будет скопирован на рабочий стол. При использовании средней кнопки мыши (в качестве альтернативы можно удерживать клавишу <Alt>) во время выделения и перемещения элемента появится небольшое меню, которое позволит сделать выбор между перемещением, копированием или созданием символической ссылки для файла. Выбор пункта Link Here создаст на рабочем столе символическую ссылку, указывающую на исходное положение элемента. Для большинства файлов и каталогов такой способ создания ссылки является оптимальным.

Панель

Панель GNOME может содержать несколько различных типов объектов. Наиболее очевидными являются кнопки меню и запуска приложений. Вы также можете использовать кнопку открытия «выдвижного ящика» (drawer), который является как бы дополнительной панелью, содержащей расширенный набор кнопок запуска. Существует несколько специальных типов кнопок, используемых для завершения сеанса работы и блокирования экрана. И, наконец, на панели могут быть запущены небольшие программы, называемые апплетами, примерами которых являются Desk Guide и часы.

Настройки панели находятся в пункте Panel главного меню или контекстного меню, открываемого щелчком правой кнопки мыши по панели. Это меню позволяет добавить на панель новые кнопки запуска или апплеты, изменить стиль, размер и способ отображения панели или создать новые панели на рабочем столе.

Дополнительные панели

На рабочем столе можно создать более одной панели. Это имеет смысл при наличии различных наборов приложений, используемых для специфических, а не общих задач. Например, при большом объеме работы с графикой можно создать панель с кнопками запуска используемых графических приложений. Дополнительную панель можно создать, вызвав контекстное меню основной панели и выбрав пункт Add New Panel либо выбрав в основном меню Panel → Add New Panel. Существует пять различных типов панелей, доступных в подменю:

edge panel (панель по краю)

Стиль основной панели по умолчанию. Панель полностью занимает одну из сторон экрана. Кнопки со стрелками на концах панели позволяют сворачивать панель. Щелчок по кнопке со стрелкой, остающейся видимой, разворачивает панель вновь.

aligned panel (выровненная панель)

Панель аналогична предыдущей, но привязана к одному из углов экрана. Она принимает минимальный размер, достаточный для размещения кнопок и апплетов. Панель с привязкой сворачивается посредством щелчка по кнопке со стрелкой, расположенной на краю экрана. Дальняя от края экрана кнопка со стрелкой привязывает панель к противоположному краю.

sliding panel (скользящая панель)

Аналогична панели с привязкой, за тем исключением, что может быть передвинута вдоль края экрана. Скользящая панель не привязывается к углу.

floating panel (плавающая панель)

Как и следует ожидать, такая панель может размещаться в любом месте экрана.

menu panel (панель меню)

Особый тип панели, растянутой вдоль верхнего края экрана. Она представляет собой узкую полоску, которая содержит выпадающие меню, идентичные пунктам главного меню Programs (Приложения), Favorites (Избранное), Settings (Настройки) и Desktop (Рабочий стол).

Все панели, кроме панели меню, можно перемещать, нажав среднюю кнопку мыши (либо левую и правую кнопки одновременно) и перетащив ее в нужную часть экрана.

Любая панель может быть настроена индивидуально при помощи пункта Panel контекстного меню панели. Щелкните правой кнопкой мыши и выберите Panel → Properties. Здесь предоставляется возможность выбора из нескольких различных подпунктов меню. Подпункт Type (Тип) изменяет тип панели (хотя тип «панели меню» изменить нельзя). Подпункт Hiding Policy (Правила сворачивания) позволяет выбрать между установкой Explicit Hide (Явное указание), когда панель сворачивается при нажатии одной из кнопок со стрелками, и Auto Hide (Автоматически), когда панель сворачивается автоматически, если она не используется. Подпункт меню Hide Buttons (Отображение кнопок) позволяет запретить или разрешить отображение на панели кнопок сворачивания. Подпункт Size (Размер) устанавливает размер панели в пределах от маленькой до огромной. Подпункт Background Type (Тип фона) позволяет установить фон панели: либо какой-нибудь цвет, либо растровое изображение.

Для того чтобы получить доступ ко всем свойствам панели в одном диалоговом окне, выберите подпункт All Properties (Все свойства) этого меню. Глобальные установки свойств находятся в GNOME Control Center (Центр управления GNOME). Они описаны далее в этой главе.

Добавление на панель кнопки запуска приложения

Одним из удобных применений панелей является возможность создания пиктограмм запуска, которые дают возможность одним щелчком получить доступ к часто используемым приложениям. Для того чтобы добавить такую пиктограмму, щелкните по панели правой кнопкой мыши и выберите пункт меню Add New Launcher (Добавить новую кнопку запуска). Можно также щелкнуть правой кнопкой мыши по приложению в главном меню и выбрать пункт Add This Launcher to Panel (Добавить это приложение на панель).

В любом случае откроется диалоговое окно Create Launcher Applet. Необходимо ввести имя приложения, комментарий, используемый как всплывающая подсказка, и команду, которой будет запускаться приложение. Щелкните по кнопке пиктограммы, чтобы выбрать изображение для кнопки на пане-

ли. Если приложение будет работать в терминале, необходимо нажать кнопку Run in Terminal (Выполнять в терминале).

Благодаря функции drag-and-drop среды GNOME приложение можно поместить на панель многими способами. Например, можно щелкнуть по файлу приложения в окне файлового менеджера и перетащить его на панель. При этом будет открыто диалоговое окно Add New Launcher и создана новая кнопка запуска на панели. Кроме того, на панель можно перетащить пиктограмму с рабочего стола.

Свойства кнопки запуска можно изменить, выбрав пункт Properties в ее контекстном меню. Будет открыто окно свойств кнопки запуска, в котором можно изменить ее имя, комментарий, команду выполнения, тип приложения и связанное с кнопкой изображение.

Кнопки запуска могут размещаться на панели в любом месте и в любом порядке. Для того чтобы переместить кнопку, щелкните по ней правой кнопкой и выберите пункт Move (Переместить). Указатель мыши изменит вид, позволяя перетащить кнопку в другую точку панели. Щелкните мышью, чтобы закрепить новое положение кнопки.

Главное меню

По умолчанию на панели GNOME есть только одно меню – главное. Оно отображается при помощи первой кнопки слева с изображением гномьей лапы (иногда это меню называют «*меню с ногой*»). По умолчанию меню представлено набором пунктов и подменю, разбитых на разделы:

- Меню System (Системное) содержит элементы, связанные с установленными в системе GNOME-приложениями и утилитами. Они разделены на дополнительные подменю в соответствии с категориями, такими как Graphics (Графические), Utilities (Инструменты), Internet (Интернет) и т. д. Внизу системного меню находятся два дополнительных элемента: пункт Help вызывает средство просмотра файлов справки GNOME, а Run Command (Выполнить команду) открывает окно с единственной командной строкой для ввода команды. Меню System может изменяться только суперпользователем (пользователем с правами доступа **root**), и такое изменение затронет всех пользователей системы.
- Меню User (Пользовательское) по умолчанию пустое. Пользователи могут добавлять свои собственные элементы в этот раздел.
- Если в вашей системе установлены другие пакеты, такие как KDE или AnotherLevel, их меню по умолчанию также могут быть включены в главное меню.
- Элементы меню Panel соответствуют действиям и настройкам панели. Элементы данного меню содержатся также и в контекстном меню, вызываемым щелчком правой кнопки мыши по панели.

- Три элемента внизу меню позволяют заблокировать экран (для доступа к рабочему столу потребуется ввести пароль текущего пользователя), просмотреть информацию о версии панели и завершить работу со средой.

Обратите внимание на то, что самый верхний элемент любого меню или подменю не связан ни с каким действием. Это строка заголовка окна. Щелчок правой кнопкой мыши по заголовку открывает контекстное меню, которое позволяет:

Add this as drawer to panel (сделать меню «выдвижным ящиком» панели)

Текущее подменю преобразуется в «выдвижной ящик» на панели.

Add this as menu to panel (добавить подменю на панель)

Создает на панели кнопку запуска данного подменю. Следует помнить, что подменю главного меню и меню на панели – это одно и то же. Подменю можно редактировать специальным редактором меню, и изменения будут отображены в меню Panel.

Add this to personal menu (добавить к пользовательскому меню)

Копирует данное подменю в меню User (Пользовательское).

Изначальная (по умолчанию) настройка главного меню несколько затрудняет настройку под пользователя. Поскольку меню System задано и не может быть изменено (никем, кроме суперпользователя), вы можете изменять только пользовательское меню.

Наилучший способ получить полностью настраиваемое главное меню (помимо работы с привилегиями суперпользователя) – это скопировать нужные части меню System в меню User. Выполнив это, вы можете отредактировать меню User «под себя» в редакторе меню.

Свойства отображения меню

Изменяя свойства меню, можно задать отображение нужных меню. Щелкните правой кнопкой мыши по кнопке меню и выберите пункт Properties, чтобы открыть диалоговое окно Menu Properties, которое содержит настройки отображения меню System и User, а также всех прочих меню, существующих в вашей системе изначально. Для каждого меню можно выбрать режим полного отображения, отображения в виде подменю или вообще запретить отображение.

Редактирование меню

Редактор меню, изображенный на рис. 15.3, позволяет добавлять или удалять элементы меню и перемещать их. Для вызова редактора меню выберите в главном меню пункт Settings, а затем Menu Editor.

Редактор меню разделен на две части. В левом окне отображена иерархия пользовательского и системного меню, а также элементы, содержащиеся в этих меню. Вы можете изменить порядок следования элементов меню или перенести элементы в другие подменю, щелкая по ним мышью и перетаски-

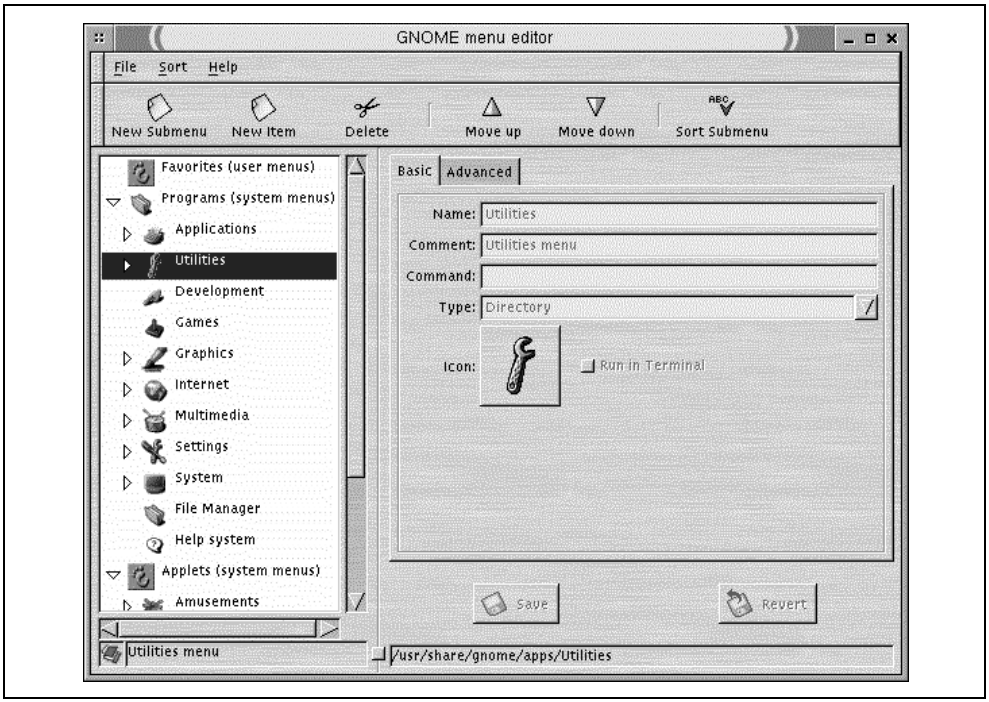


Рис. 15.3. Редактор меню

вая в нужное место в левом окне. В правом окне отображаются свойства элемента, выбранного в левом окне.

Кнопки на панели инструментов предоставляют пользователю возможность совершать над выделенным элементом меню следующие действия: добавлять новое подменю, добавлять новый элемент, удалять выделенные элементы, перемещать элементы вверх или вниз, а также упорядочивать подменю.

Для того чтобы добавить новое подменю, выберите меню, в которое вы хотите его поместить, и нажмите кнопку New Submenu (Новое подменю). Введите название подменю в правом окне и нажмите OK.

Для того чтобы добавить новый элемент в меню, выберите его местоположение в дереве меню в окне слева и нажмите кнопку New Item (Новый элемент). В правом окне отображаются настройки свойств элемента. Введите имя элемента, комментарий для всплывающей подсказки, команду, выполняемую при выборе элемента, и тип приложения. Нажмите OK, чтобы добавить элемент в меню.

Центр управления GNOME

Центр управления GNOME (GNOME Control Center) (рис. 15.4) – это то место, где выполняется большинство настроек среды вашего рабочего стола.

Центр управления открывается нажатием кнопки с изображением ящика с инструментами на панели или выбором пункта Utilities в главном меню. Центр управления содержит ряд приложений настройки, называемых *капплетами* (*capplets*), которые позволяют изменять различные настройки GNOME. Капплеты перечислены в левой части окна Control Center, и щелчок по названию приложения открывает капплет в правой части окна.

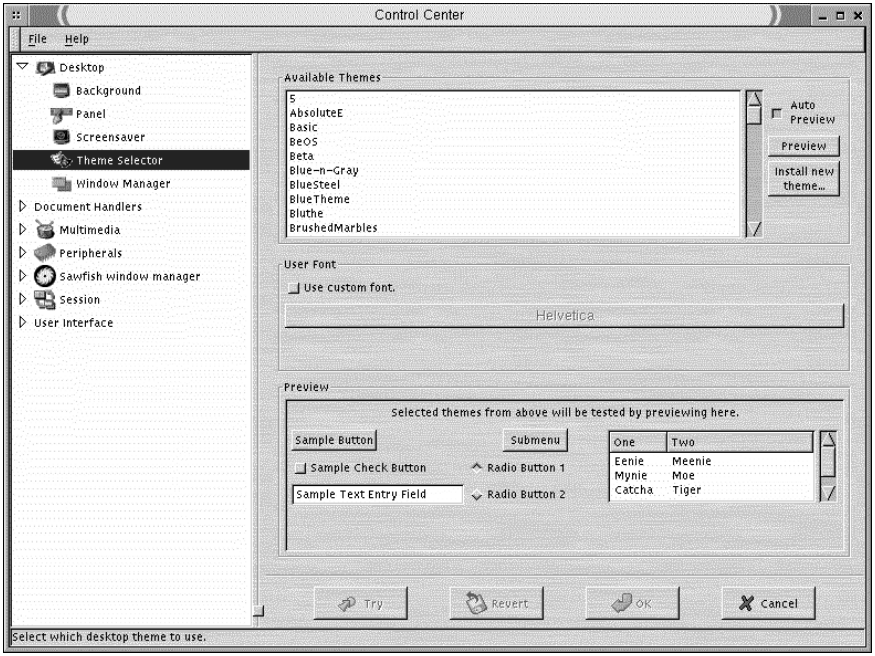


Рис. 15.4. Центр управления GNOME

Настройки рабочего стола

В этом разделе представлены общие настройки вида рабочего стола. В частности, можно выбрать используемый фон, экранную заставку, тему и оконный менеджер.

Фон

В разделе Background устанавливается фон рабочего стола. Можно выбрать изображение для обоев или какой-либо цвет. В разделе wallpaper (обои) нажмите кнопку Browse (Обзор), чтобы выбрать файл изображения на диске. Изображение может быть размножено на рабочем столе, отцентрировано или отмасштабировано. В разделе color можно выбрать сплошную заливку одним цветом либо горизонтальный или вертикальный двухцветный градиент. Основной и вспомогательный цвета выбираются щелчком по полям primary color и secondary color, соответственно.

Данный капплет устанавливает фон средствами GNOME. Фон можно установить и в оконном менеджере, и иногда эти настройки могут вступать в конфликт с настройками, заданными в GNOME. Для того чтобы убедиться в использовании фона GNOME, отметьте флажок Use GNOME to Set Background (Использовать GNOME для установки фона) в нижней части окна.

Экранная заставка

Раздел Screensaver содержит настройки экранной заставки. Вы можете выбрать экранную заставку из имеющихся (включая случайный выбор при включении), а также ввести количество минут бездействия, приводящее к вызову заставки, и уточнить необходимость ввода пароля при возвращении к рабочему столу. Здесь же доступны настройки управления питанием, если они поддерживаются системой.

Выбор темы

Темы (themes) обеспечивают единство стиля графических элементов и компонентов, используемых в GNOME. Несколько распространенных тем включены в пакет *gtk-engines*, а дополнительные темы можно загрузить с сайта *http://gtk.themes.org*.

Все темы, установленные в системе, перечислены в приложении настройки Theme Selector. Выбранную тему можно просмотреть в нижней части окна. При установленном флажке Auto Preview (Автопросмотр) предварительный просмотр выбранной темы будет включаться автоматически. В противном случае для просмотра темы необходимо нажать кнопку Preview.

Если вы загрузили тему и намереваетесь установить ее, нажмите кнопку Install New Theme (Установить новую тему). Введите имя, а также путь к файлу *.tar.gz* или *.tgz*, а затем нажмите OK. Новая тема будет установлена в каталог */usr/share/themes* и станет доступна для использования на рабочем столе.

Оконный менеджер

В этом разделе можно выбрать оконный менеджер для использования с GNOME и настроить его свойства, если у менеджера существует утилита настройки. По умолчанию GNOME работает с оконным менеджером Enlightenment. Можно использовать и другие оконные менеджеры, но необходимо убедиться, что они совместимы с архитектурой GNOME и не будут создавать проблем вашим приложениям.

В разделе Window Manager приводится список установленных оконных менеджеров, которые можно использовать. Можно добавить новый оконный менеджер к этому списку, нажав кнопку Add. В открывшемся диалоговом окне необходимо ввести имя нового оконного менеджера. При наличии у менеджера собственной утилиты настройки необходимо также ввести ее имя и месторасположение. Когда это будет сделано, внизу станет доступной кнопка Run the Configuration Tool for *wm-name* (Вызвать утилиту настройки *имя оконного менеджера*). Нажатие этой кнопки выполнит приложение настройки данного оконного менеджера.

Панель

Панели могут настраиваться индивидуально или все сразу. Для того чтобы изменить настройки отдельной панели, щелкните правой кнопкой по панели и выберите пункт This Panel's Properties (Свойства данной панели).

Большинство изменений в настройках панели производятся посредством инструмента глобальной настройки панели, который показан на рис. 15.5. Для вызова этого инструмента выберите пункт Global Preferences (Глобальные настройки) в меню Panel.

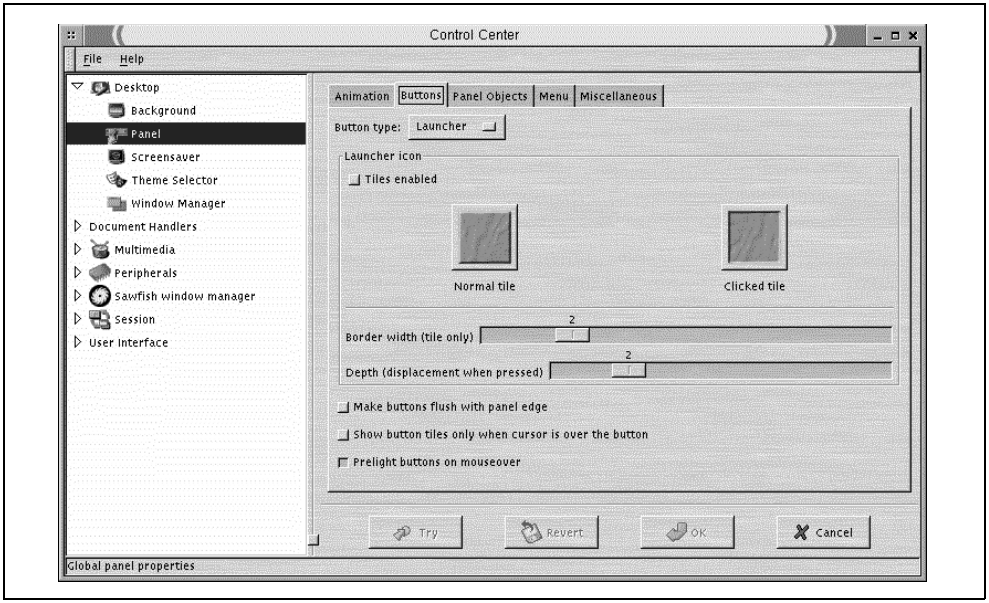


Рис. 15.5. Настройка свойств панели

Откроется раздел глобальной настройки панели Центра управления, который содержит следующие вкладки:

Анимация

Вкладка Animation содержит настройки анимированного движения панели, например, при ее сворачивании или открытии выдвижного ящика. В верхней части вкладки находится переключатель, позволяющий включать или отключать анимацию. Все остальные настройки производятся ползунками. Первые два ползунка определяют скорость сворачивания панели при автоматическом или принудительном (посредством кнопок со стрелками) сворачивании. Следующий ползунок регулирует скорость открытия и закрытия выдвижных ящиков. Для автоматически сворачивающихся панелей доступны дополнительные ползунки, определяющие период времени, по истечении которого панель сворачивается, и размер свернутой панели (в пикселах).

Кнопки

Вкладка Buttons позволяет настраивать внешний вид различных типов кнопок на панели (рис. 15.5). Тип кнопок для настройки можно выбрать в выпадающем меню.

Использование «плиток» позволяет придать пиктограммам панели квадратный фон вместо отображения пиктограмм прямо на фоне панели. Если разрешить использование «плитки» (при помощи кнопки вверху вкладки), можно выбрать изображение, которое будет использоваться в качестве фона для всех кнопок на панели. «Плитки» также используются для создания объемных кнопок на панели. Можно выбрать изображения для обычного и нажатого (как результат щелчка мышью) состояний кнопки. Для того чтобы выбрать файлы изображений, щелкните по изображениям «плитки» (Normal tile или Clicked tile), и откроется диалоговое окно выбора файла.

Два ползунка позволяют устанавливать ширину рамки (в пикселах) вокруг плитки и глубину вдавливания кнопки при нажатии.

Доступны следующие дополнительные настройки:

Make buttons flush with panel edge

Эта установка размещает кнопки по краям панели. Нет видимого пространства между кнопкой и фоном рабочего стола, хотя между кнопками панели интервалы есть.

Show button tiles only when cursor is over them

Данная установка делает «плитку» видимой только тогда, когда над кнопкой находится указатель мыши.

Prelight buttons on mouse-over

Эта установка обеспечивает «подсвечивание» кнопок, когда над ними находится указатель мыши.

Объекты панели

Вкладка Panel objects позволяет выбрать один из трех способов размещения пиктограмм. В режиме Switched Movement пиктограммы меняются местами с той, которая перемещается пользователем. Это режим по умолчанию. При выборе способа Free Movement пиктограммы не изменяют своего положения, если какая-либо из них перемещается пользователем. В этом случае пиктограммы могут перемещаться только на свободное место на панели. В режиме Push Movement пиктограммы при движении расталкивают другие пиктограммы, но не проходят над ними.

Ползунок Padding (Отбивка) устанавливает расстояние (в пикселах) между объектами на панели.

Меню

Вкладка Menu содержит глобальные установки меню.

Первый переключатель позволяет задать использование крупных пиктограмм для всех элементов меню, будь то папка, файл или приложение.

Второй переключатель Show ... Buttons включает/выключает каскадное отображение подменю. Если переключатель отмечен, кнопки с символами «...» отображаются для каждого элемента меню, имеющего подменю. Подменю отображается только при нажатии кнопки «...». Если переключатель *не* отмечен, используется автоматическое каскадное отображение меню.

Кнопка Show Pop-Up Menus Outside of Panels управляет размещением всплывающих меню относительно панели. Если кнопка нажата, всплывающие меню отображаются на рабочем столе таким образом, чтобы не перекрывать панель. Кнопка Keep Menus in Memory позволяет кэшировать недавно использовавшиеся меню в памяти, обеспечивая более быструю реакцию окружения рабочего стола.

В оставшейся части вкладки можно определить, какие разделы меню вы хотите отобразить. Существует возможность выбора: помещать ли разделы прямо в главное меню, в подменю или вообще запретить отображение разделов.

Разное

Вкладка Miscellaneous предоставляет доступ к целому ряду настроек панели:

Tooltips enabled

Разрешить отображение всплывающих подсказок для кнопок запуска.

Close drawer if launcher inside it is pressed

Закрыть «выдвижной ящик» после выбора одного из содержащихся в нем элементов. В противном случае ящик остается открытым до тех пор, пока вы не щелкнете по стрелке, закрывающей его.

Raise panels on mouse-over

Выдвинуть панель на передний план при нахождении указателя мыши над нею, если панель перекрывается другим окном.

Keep panel below windows

Позволить окнам GNOME-совместимых приложений отображаться поверх панели (в обычном режиме они не могут ее перекрывать).

Confirm the removal of panels with a dialog

Во всплывающем окне запрашивать подтверждение на удаление панели.

Обработчики документов

В разделе Document Handlers можно настроить по умолчанию программы, которые вызываются для определенных файлов, исходя из типа содержимого файлов или их URL.

Редактор по умолчанию

В разделе Default editor задается текстовый редактор, который вызывается по умолчанию при открытии файла, подлежащего редактированию, либо при выборе команды Edit из контекстного меню файла в файловом менеджере. Установленные в системе редакторы можно выбрать в выпадающем списке. Нажмите кнопку Run in Terminal Window, если выбранный редактор должен выполняться в окне терминала.

Типы Mime

Этот раздел позволяет устанавливать или редактировать типы mime. При помощи кнопки Add можно добавить новый тип. Откроется диалоговое окно, в котором необходимо уточнить категорию/вид mime-типа и расширение, связанное с ним. Дополнительно можно задать не более двух регулярных выражений, идентифицирующих mime-тип. Для редактирования mime-типа выделите его в списке и нажмите кнопку Edit. Откроется диалоговое окно, в котором можно определить пиктограмму, используемую для файлов этого типа, добавить или удалить расширения файлов и определить команды для открытия, просмотра и редактирования файлов этого типа. Для того чтобы удалить тип, выберите его из списка и нажмите кнопку Delete.

Обработчики URL

На странице URL handlers можно определить настройки для специальных пусковых URL-кнопок, используемых системой справки GNOME. Изначальные установки (по умолчанию) для протоколов HTTP, FTP и Mail уже сделаны, и, вероятнее всего, эти протоколы обрабатываются вашим веб-броузером (например Netscape). К специальным URL относятся *ghelp*, *info* и *man*, т. е. протоколы, соответствующие файлам справки GNOME, файлам *info* и файлам руководства *man*. По умолчанию эти протоколы обрабатываются средствами просмотра соответствующих типов файлов. Эти настройки лучше не изменять.

Мультимедиа

В разделе Multimedia расположены настройки, относящиеся к звуку в системе.

Звук

Страница Sound позволяет определить звуковую схему для различных событий. На вкладке General (Общие) можно включить или выключить все звуки, а на вкладке Sound Events (Звуковые события) представлен список доступных событий и звуковых файлов, связанных с ними. Выберите событие из списка и нажмите кнопку Play, чтобы прослушать связанный с ним звук. Введите в текстовое поле имя звукового файла, который необходимо связать с событием, либо используйте кнопку Browse (Обзор) для выбора звукового файла.

Устройства

В разделе Peripherals представлены настройки клавиатуры и мыши.

Клавиатура

В разделе Keyboard содержатся настройки автоповтора и звука. Можно установить скорость автоповтора при нажатии клавиши и задержку перед началом автоповтора. Можно также включить щелчки, издаваемые при нажатии клавиши, и регулировать их громкость. Настройки можно опробовать, напечатав что-либо в поле ввода Test Settings.

Три ползунка позволяют регулировать громкость, высоту тона и длительность клавиатурного сигнала. Нажмите кнопку Test, чтобы прослушать результат.

Мышь

Мышь может быть настроена для использования правшой или левшой. Можно также изменять скорость передвижения указателя и чувствительность мыши.

Сеанс

В разделе Session содержатся настройки программ, выполняемых в начале каждого сеанса работы.

Стартовый совет

Startup Hint – это программа, отображающая сообщение при каждом входе пользователя в систему. Советы могут содержать полезные, неизвестные пользователю мелочи, относящиеся к работе с GNOME. Эта программа также может использоваться для отображения забавных сообщений (fortunes) или «девизов дня».

Автозагрузка

Поскольку работа с GNOME происходит в сеансовом режиме, система запоминает, какие приложения были открыты, и автоматически вновь запускает их, когда пользователь входит в систему. Однако не все приложения поддерживают этот механизм GNOME. Если вы хотите, чтобы определенные приложения выполнялись автоматически при старте среды, можно перечислить их в разделе Startup Programs. Можно добавлять, редактировать и удалять автоматически загружаемые программы, а также получать сведения о запущенных в настоящий момент приложениях.

В верхней части окна находится кнопка, посредством которой контролируется отображение заставки при старте среды. Две дополнительные кнопки позволяют либо включить запросы подтверждения сохранения состояния сеанса при завершении работы, либо автоматически сохранять изменения.

Пользовательский интерфейс

В разделе User Interface находятся настройки различных компонентов окон приложений, таких как меню, панели инструментов, строки состояния и диалоговые окна.

Приложения

В разделе Applications доступны настройки, определяющие поведение меню, панелей инструментов и строк состояния приложений GNOME.

Панели меню в приложениях GNOME могут отделяться от окон и помещаться в любом месте рабочего стола. Этот режим установлен по умолчанию.

У отделяемого меню есть небольшая полоска на левом краю. Щелкнув по этой полоске, можно затем переместить меню в другое место рабочего стола. Для того чтобы объединить меню с окном, подтащите его обратно к заголовку окна приложения, и оно само встанет на место.

Для неотделяемых меню отключите переключатель Can Detach and Move Menus.

Точно такая же функциональность отделяемости есть по умолчанию и у подменю. Для того чтобы запретить эту функцию, отключите переключатель Submenus Can Be Torn Off.

Три настройки позволяют отображать меню, панели инструментов и кнопки панели инструментов с рельефными рамками. Рельефные рамки придают трехмерный вид компонентам; в противном случае они выглядят совершенно плоскими.

Помимо этого доступны два параметра для строк состояния, отображаемых внизу окна. Если в строке состояния используется индикатор хода работы, пользователь может настроить привязку индикатора к правому краю строки состояния. Если эта установка выключена, размещение индикатора определяется приложением.

Некоторые строки состояния могут иметь пользовательскую функциональность, определяемую приложением. Существующие настройки позволяют включить взаимодействие пользователя с такими строками состояния.

Диалоговые окна

Страница Dialogs поделена на разделы, посвященные виду диалоговых окон и их размещению.

Раздел Dialog Layout позволяет определить расположение кнопок диалогового окна при помощи следующих режимов: Default (По умолчанию), Spread Buttons Out (Расположить кнопки равномерно), Put Buttons on Edges (Расположить кнопки по краям), Left-justify Buttons (Кнопки выравниваются по левому краю) и Right-justify Buttons (Кнопки выравниваются по правому краю). Вы также можете включать или выключать пиктограммы для кнопок диалогового окна, а также предпочесть использовать строку состояния окна вместо отдельного диалогового окна, когда это возможно.

В разделе Dialog Behavior можно определить расположение диалоговых окон при помощи следующих параметров: Let Window Manager Decide (На усмотрение оконного менеджера), Center of the Screen (В центре экрана) и At the Mouse Pointer (Под указателем мыши). Можно также производить настройку всплывающих подсказок и включать автоматическое размещение диалогового окна над окном приложения, которому оно принадлежит.

MDI

В разделе многодокументного интерфейса MDI (Multiple Document Interface) определяется способ открытия приложением нескольких документов. Существует три стиля: Modal (Модальный), Toplevel (Верхнего уровня) и Notebook (Блокнот). По умолчанию установлен Modal. При выборе стиля Notebook можно указать расположение строки вкладок – по любому из краев окна.

Настройка оконного менеджера Sawfish

Оконный менеджер Sawfish является оконным менеджером по умолчанию, используемым с окружением Helix GNOME (которое вы видите на рисунках этой главы). Sawfish совместим со средой GNOME и не конфликтует с ее настройками. Helix-версия Центра управления GNOME включает раздел настроек для Sawfish вместо использования отдельного инструмента настройки. Раздел настроек Sawfish показан на рис. 15.6.

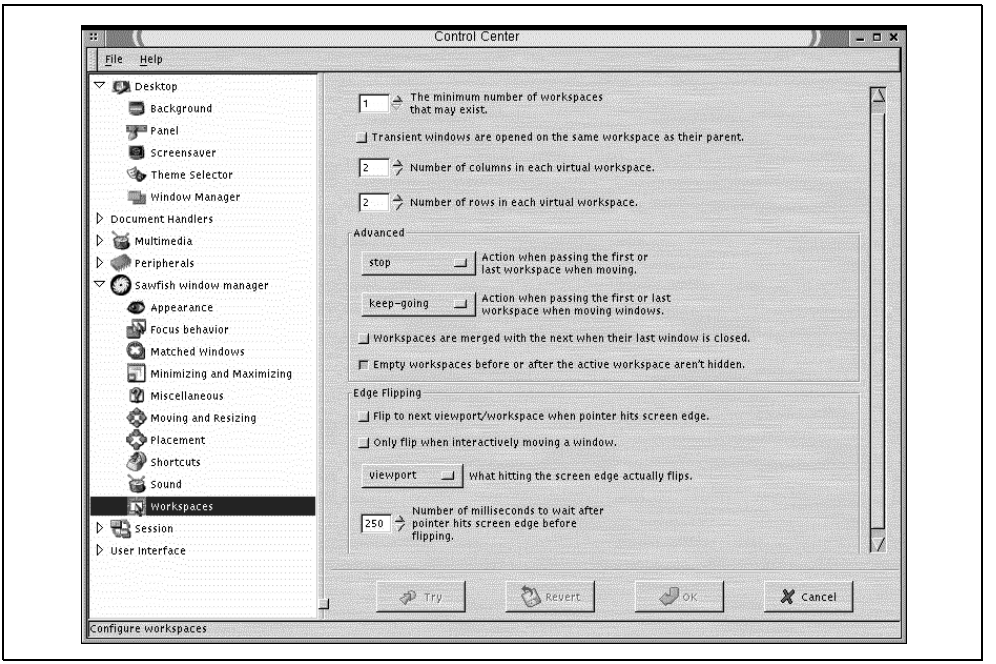


Рис. 15.6. Настройки Sawfish

В следующих разделах описаны настройки Sawfish, содержащиеся в Центре управления Helix GNOME.

Внешний вид

В разделе Appearance устанавливается вид окон по умолчанию, включая стиль рамки окна, шрифты, стиль перемещения.

Стиль рамки можно выбрать из выпадающего списка. Стиль рамки определяет внешний вид заголовков окон путем настройки вида кнопок и фона.

Некоторые всплывающие окна настроены так, что у них отсутствует заголовок окна. Принудительного отображения заголовка для всех окон можно добиться, нажав кнопку Decorate Transient Windows Similarly to Top-level Windows.

По умолчанию режим анимации для окон определяет отображение окон при минимизации. Вы можете выбрать следующие варианты: Solid (сплошной прямоугольник), Wireframe (только рамка) или None (анимация отсутствует).

Кнопка выбора шрифта позволяет определить шрифт по умолчанию для рабочего стола. Шрифт по умолчанию устанавливается темой рабочего стола GNOME, и Sawfish синхронизирует настройки, чтобы они совпадали.

Поведение фокуса

В разделе Focus behavior находятся настройки, определяющие, каким образом окно может получить фокус ввода.

В прокручиваемом списке в верхней части страницы определяется вариант перемещения фокуса указателем мыши. По умолчанию фокус передается окну по щелчку мышью в этом окне. Есть еще два варианта – передавать фокус только тогда, когда указатель мыши находится над окном («enter-exit»), или сохранять фокус в последнем окне, над которым был указатель мыши («enter-only»).

Три флажка используются для включения следующих правил для передачи фокуса: focus each window when first displayed (новые окна получают фокус ввода при первом отображении), transient windows inherit focus from their parent (дополнительные окна наследуют фокус от родителей) и raise windows when they are focused (помещать окно перед другими, когда оно получает фокус). Можно установить задержку (в миллисекундах) между получением фокуса и помещением окна на передний план.

Раздел более сложных настроек содержит следующее:

- Give focus to windows even when they haven't asked for it (Передавать фокус ввода, даже если окно этого не требует).
- Offset from left window edge when warping (Отступать от левого края окна при переходе).
- Offset from top window edge when warping (Отступать от верхнего края окна при переходе).
- Does click-to-focus mode pass the click through to the window? (Передается ли щелчок мыши, используемый для установки фокуса, в качестве обычного щелчка окну?)

Раздел Shade Hover содержит настройки, относящиеся к поведению скрытых окон. Скрытое окно по существу сворачивается, исчезает как тень в собственный заголовок. Обычно это достигается двойным щелчком по заголовку окна. Если включить режим Shade Hover, скрытое окно будет временно разворачиваться, если над его заголовком окажется указатель мыши. В следу-

ющем поле ввода можно задать задержку (в миллисекундах) до разворачивания окна. И последняя настройка в этом разделе разрешает автоматический переход окна на передний план, когда оно разворачивается из скрытого состояния.

Сворачивание и восстановление

В разделе Minimizing and Maximizing содержатся настройки, относящиеся к сворачиванию и восстановлению окон.

Windows are uniconified onto the current workspace

Свернутые окна восстанавливаются на текущем рабочем столе, если щелкнуть по кнопке в списке задач. Обычно в GNOME есть только один рабочий стол, разделенный на несколько экранов-ячеек, называемых окнами просмотра.

Windows are raised after being uniconified

Свернутые окна при восстановлении отображаются поверх всех остальных окон.

Windows are focused after being uniconified

Свернутые окна при восстановлении получают фокус.

Iconifying a window that's a member of a group removes the whole group

При сворачивании окна, входящего в определенную группу (например, приложение может открывать несколько порожденных окон, связанных с основным окном), сворачивается вся группа окон.

Uniconifying a window that's a member of a group restarts the whole group

При совместном использовании с предыдущим режимом окна свернутой группы будут открываться при восстановлении одного из окон группы.

Windows are uniconified to the current viewport

Свернутые окна будут открыты в текущем экране независимо от того, где они были прежде. В Sawfish каждый экран рабочего стола называется окном просмотра.

Доступны следующие настройки для максимизации размера окон:

Maximizing a window dimension always increases the size of that dimension

При максимизации окна по горизонтали или вертикали оно будет увеличено, даже если заполнит весь экран.

Raise windows when they're maximized

Максимизируемые окна автоматически попадают на передний план и получают фокус.

Let ignored windows be overlapped when filling windows

Если получено предписание игнорировать окно (посредством соответствующего пункта меню окна), оно будет перекрываться при заполнении.

Lock window geometry while the window is maximized

Запрещено изменять размер окна, если оно уже максимизировано.

Разное

В разделе Miscellaneous находятся различные настройки для окон, включая настройки отображения всплывающих подсказок.

Windows selected (normally by the Windows menu) are raised

Выбранное окно (посредством клавиатурного сокращения или меню рабочего стола) переводится на передний план и ему передается фокус.

Unshade selected windows

При выборе скрытого окна оно автоматически разворачивается.

Warp the mouse pointer to selected windows

Указатель мыши автоматически перемещается в выбранное окно.

Keep transient windows stacked above

Выпадающий список позволяет выбрать режим расположения порожденных окон. Возможные значения: None, Parents или All.

Update all windows when the default frame style is changed

Автоматически обновлять все окна при изменении стиля рамки окна.

Automatically reload themes when they are updated

При внесении изменений в тему вводить их в действие немедленно.

Group transient windows with their parents

Порожденные окна, например диалоговые, автоматически помещаются в группу родительского окна.

Raise windows when they are unshaded

При разворачивании окна получают фокус и отображаются поверх всех окон.

Для того чтобы включить всплывающие подсказки оконного менеджера, нажмите кнопку Display Tooltips for Window Frames.

Подсказка будет отображаться, пока указатель мыши находится над элементом. Можно задать автоматическое прекращение отображения подсказки, включив режим Remove Tooltips After a Period of Time. Интервалы времени до отображения и до удаления подсказки задаются в двух полях ввода.

Можно назначить шрифт, используемый в подсказках, а также цвета фона и текста.

Перемещение и изменение размеров

При перемещении окон доступны следующие методы их отображения: opaque (непрозрачное) и box (в виде рамки). Эти же методы применимы к процессу изменения размера окон.

Также доступны следующие переключатели:

- Raise windows being moved or resized interactively (Отображать перемещаемые окна и окна в процессе изменения их размера поверх всех остальных окон).
- Show the current position while moving windows interactively (Отображать текущее положение окна при перемещении).
- Show the current dimensions while resizing windows interactively (Отображать текущие размеры при их изменении).

Размещение

В разделе Placement содержатся настройки по умолчанию, относящиеся к размещению открываемых окон. Существуют правила размещения для обычных и порождаемых окон. Для каждого параметра доступен список возможных значений:

- Randomly (transient default) – случайным образом (значение по умолчанию для порождаемых окон)
- Interactively (Динамическое размещение)
- Centered (Центрировать при размещении)
- Centered-on-parent (Центрировать по родительскому окну)
- Under-pointer (Под указателем)
- None (Нет)
- First-fit (Произвольное размещение)
- Best-fit (window default) – оптимальное размещение (значение по умолчанию для окон)
- First-fit-or-interactive (Произвольное или динамическое размещение)

Для того чтобы отключить контроль приложений над размещением окон, необходимо отметить параметр Ignore Program-specific Window Positions. По умолчанию этот режим выключен.

Для произвольного и оптимального вариантов размещения можно указать минимальное расстояние (в пикселах) между краями окна.

Рабочее пространство

Sawfish позволяет иметь несколько рабочих пространств (рабочих столов), каждое из которых может быть разделено на ряд обзорных экранов, которые называются *окнами просмотра* в GNOME Desk Guide. Рабочие пространства (workspaces) можно рассматривать как рабочие столы, расположенные один над другим. Окна просмотра описываются сеткой строк и колонок.

Первый параметр на странице Workspaces устанавливает количество рабочих пространств. Для того чтобы определить сетку окон просмотра, необходимо задать количество колонок и строк для каждого рабочего пространства, придерживаясь установленных ограничений.

По умолчанию порождаемые (всплывающие) окна открываются в том же рабочем пространстве, в котором находится породившее их окно.

Раздел Advanced содержит настройки переключения между рабочими пространствами. Для переключения между пространствами или перемещения окон можно выбрать следующие правила, относящиеся к достижению границы рабочего пространства: stop (остановиться), wrap around (вернуться в начало, т. е. на первое рабочее пространство) или keep going (продолжать движение).

Самым простым способом переключения между окнами просмотра является использование апплета Desk Guide, расположенного на панели GNOME. Если вы хотите перемещаться между экранами с помощью указателя мыши, можно включить переход через край (раздел Edge Flipping). Переход через край настраивается двумя переключателями:

- Flip to next viewport/workspace when pointer hits screen edge (Переходить в следующее окно просмотра или в следующее рабочее пространство, когда указатель достигает края экрана).
- Flip only when interactively moving a window (Совершать переход только при перемещении окна).

Можно указать, что именно является целью перехода через край: следующее окно просмотра (по умолчанию) или следующее рабочее пространство. При желании установить задержку, после которой происходит переключение при достижении курсором края экрана, введите соответствующее число в миллисекундах.

Настройка оконного менеджера Enlightenment

Enlightenment – один из наиболее распространенных оконных менеджеров, используемых совместно с GNOME. Настройка оконного менеджера позволяет изменять поведение основных событий, происходящих на рабочем столе, таких как поведение мыши, размещение окон, стили рамок и количество виртуальных рабочих столов. Редактор настроек Enlightenment доступен из Центра управления GNOME. На странице оконных менеджеров выберите из списка Enlightenment, а затем щелкните по кнопке Run Configuration Tool for Enlightenment. Откроется окно настроек Enlightenment (рис. 15.7), в котором представлены следующие разделы:

Основные настройки

На странице Basic Options устанавливаются стили перемещения окон и фокуса ввода.

Move methods (стили перемещения)

Эти кнопки позволяют выбрать стиль прорисовывания окна при перемещении. Доступны следующие стили: Opaque (Непрозрачный), Lined (Линии), Box (Рамка), Shaded (Затененный), Semi-Solid (Полусплошной) и Translucent (Полупрозрачный).

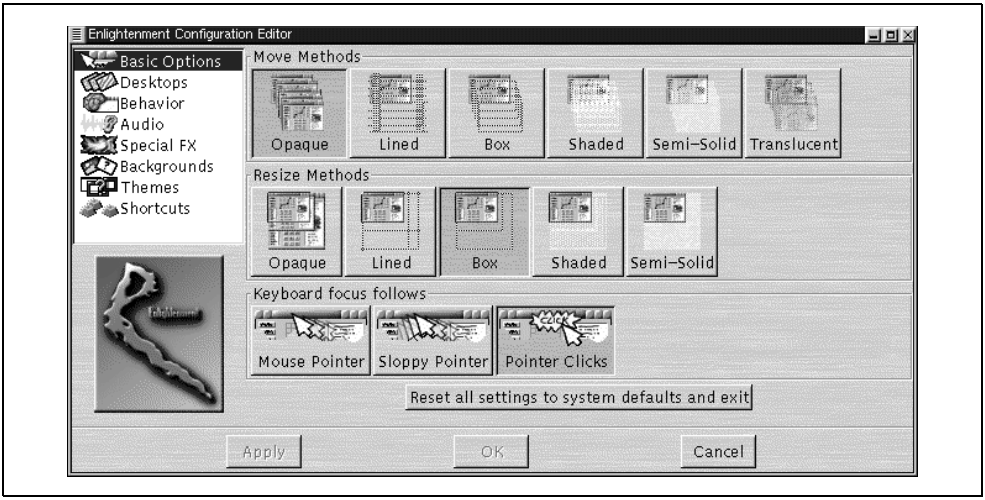


Рис. 15.7. Средство настройки Enlightenment

Resize methods (стили изменения размера)

Эти кнопки позволяют определить стиль прорисовывания окна при изменении размера. Доступны следующие стили: Opaque, Lined, Box, Shaded и Semi-Solid.

Keyboard focus follows (фокус ввода)

Эти кнопки позволяют определить, как фокус ввода связан с действиями мыши. Доступны варианты: Mouse Pointer (фокус следует за указателем мыши), Sloppy Pointer (фокус следует за указателем мыши с небольшой задержкой) и Pointer Clicks (фокус переходит к окну, по которому выполнен щелчок мышью).

Рабочие столы

В разделе Desktops доступны настройки пространства рабочего стола. Можно создать несколько виртуальных рабочих столов с их собственными экранами и переключаться между ними по необходимости. Два ползунка позволяют задать размеры виртуального рабочего стола (количество строк и столбцов экрана). Включение Edge Flip позволит переключаться на соседний рабочий стол, когда указатель мыши пересечет границу текущего экрана. Имеется возможность установить задержку перехода (в миллисекундах). Настройка Separate Desktops позволяет создать несколько «слоеных» рабочих столов (взамен многоэкранных окон). Обратите внимание, что апплет GNOME Desk Guide не поддерживает «слоеные» рабочие столы.

Поведение

Раздел Behavior содержит параметры для настройки дополнительных свойств фокуса ввода в новых окнах, а также различные настройки, относящиеся к

всплывающим подсказкам и размещению порождаемых окон. Вкладка Advanced Focus содержит следующие пункты:

All new windows that appear get the keyboard focus

Новые окна автоматически получают фокус ввода.

All new pop-up windows get the keyboard focus

Новые порождаемые окна автоматически получают фокус ввода.

Only new pop-up windows whose owner is focused get the keyboard focus

Порождаемое окно получает фокус только в том случае, если оно порождено окном с активным фокусом.

Raise windows when switching focus with keyboard

Свернутые окна восстанавливаются при получении фокуса.

Send the pointer to windows when switching focus with the keyboard

Указатель мыши автоматически перемещается в окно, получающее фокус ввода.

Вкладка Miscellaneous содержит следующие настройки:

Tooltips

Нажмите кнопку Enable, чтобы разрешить отображение всплывающих подсказок, когда указатель находится над пиктограммой. Длительность отображения подсказки в секундах может быть установлена с помощью ползунка.

Transient pop-up windows appear together with leader

Всплывающие окна привязываются при отображении к родительскому окну.

Switch to where pop-up window appears

Автоматически переключаться в экран рабочего стола, в котором открылось новое порожденное окно.

Display icons when windows are iconified

Отображать маленькую пиктограмму на экране, когда окно свернуто. Отключите этот режим, поскольку он дублирует функциональность панели GNOME.

Place windows manually

При открытии нового окна указатель мыши автоматически устанавливается на его заголовок, предоставляя пользователю возможность переместить и зафиксировать окно в нужном месте.

Audio

Эта страница позволяет включить звуковые эффекты Enlightenment. Звуки для отдельных событий могут выбираться пользователем или устанавливаться в качестве части темы рабочего стола.

Спецэффекты

Раздел Special FX содержит настройки движения компонентов рабочего стола и включает следующие параметры:

Window Sliding Methods

Помимо ниже перечисленных настроек, здесь присутствуют такие же параметры, как на странице Basic Options:

- Windows slide in when they appear (Окна при появлении плавно скользят по экрану).
- Windows slide about during window cleanup (Окна скользят при обновлении изображения).
- Desktops slide in when changing desktops (Плавное переключение между рабочими столами).
- Window shading speed (pixels/sec) (Скорость сокрытия окна в пикселах/сек.).

Drag bar

Нажмите эту кнопку, чтобы разрешить перетаскивание заголовка, и выберите его местоположение: left (слева), right (справа), top (сверху) и bottom (по нижнему краю).

Animate menus

Нажмите эту кнопку, чтобы включить плавную анимацию в меню.

Reduce Refresh

Нажмите эту кнопку, чтобы уменьшить частоту обновления рабочего стола.

Фон

В разделе Backgrounds можно выбрать фоновое изображение для каждого из имеющихся виртуальных рабочих столов. Выберите конкретный рабочий стол, а затем файл изображения из списка доступных. Существует также кнопка No Background (Без фонового изображения). Для того чтобы добавить изображение в список, нажмите кнопку Add New... и выберите файл.

Кнопка High Quality Rendering for Background позволяет отображать фоновые изображения с максимальным разрешением и с максимальной глубиной цвета, установленными на вашем мониторе.

Фоновые изображения рабочих столов, которые не использовались какое-то время, могут удаляться из памяти. Для этого необходимо нажать кнопку Minutes After Which to Expunge Unviewed Backgrounds from Memory и установить количество минут хранения в памяти при помощи ползунка.

Помните, что установка фонового изображения в Enlightenment имеет больший приоритет, чем аналогичные настройки GNOME.

Темы

В разделе Themes можно выбрать тему для оконного менеджера. Темы обеспечивают неповторимый и согласованный вид компонентов окон и фона. Темы Enlightenment устанавливаются в каталог */usr/share/enlightenment/themes*. Любое количество пользовательских тем можно загрузить с *http://e.themes.org*. Если вы используете тему рабочего стола в среде GNOME, выберите основную тему оконного менеджера, чтобы не создавать нежелательных противоречий в стиле рабочего стола.

Горячие клавиши

Раздел Shortcuts позволяет связать клавиатурные комбинации с определенными действиями Enlightenment. В разделе представлен список уже определенных клавиатурных сокращений. Можно изменить сокращение, выбрав его из списка, присвоив нужный модификатор и нажав кнопку Change. В появившемся окне необходимо ввести клавишу для использования в данном сокращении.

Глава 16

KDE

KDE (K Desktop Environment) – это свободно распространяемая среда, разработчики которой стремятся к созданию дружественного к пользователю, современного окружения рабочего стола для Unix- и Linux-систем, обладающего единством стиля. KDE – это не просто оконный менеджер вроде **fvwm**, это целая система, в которой функции оконного менеджера интегрированы в графическую конфигурацию. В интерфейсе KDE широко используется механизм drag-and-drop, так что вы можете перетащить пиктограмму текстового файла из файлового менеджера в окно текстового редактора, чтобы открыть этот файл. Полная интеграция приложений KDE с сетевыми сервисами обеспечивает прозрачный доступ к файлам на других компьютерах или FTP-узлах и работу с ними как с локальными файлами.

В KDE также реализована стандартная система справки на основе HTML. Приложение, имеющее кнопку Help, позволяет открывать связанный с ними файл справки в средстве просмотра таких файлов.

Одной из целей KDE является предоставление пользователю удобных графических интерфейсов рабочего стола для получения информации и настройки системы. Центр управления KDE (KDE Control Center) является основным инструментом для настройки рабочего стола и приложений, а также источником информации по важным компонентам системы. Информационный модуль Центра управления позволяет получать данные о состоянии процессора, памяти, шины PCI и сетевых устройств.

Существует большое количество эффективных в работе приложений, разработанных на основе возможностей, предоставляемых KDE. Основной пакет KDE содержит такие программы, как почтовый клиент, календарь и записная книжка, проигрыватель компакт-дисков, средства просмотра изображений, чат-клиенты и многое другое.

Большинство дистрибутивов Linux включают KDE и позволяют сделать эту среду основным рабочим окружением уже при установке системы. Для самостоятельной установки KDE необходимо загрузить и установить соответствующие пакеты, которые можно найти по адресу *ftp://ftp.kde.org* и в других местах.

Для того чтобы сделать KDE основной рабочей средой, найдите файлы инициализации X Window в своем домашнем каталоге. В зависимости от дистрибутива эти файлы могут называться *.xinitrc*, *.xsession*, *.Xclients*. Если ни один из этих файлов не существует, создайте файл *.xinitrc*. Удалите из файла все упоминания о других оконных менеджерах и добавьте команду **startkde** в последнюю строку файла. Убедитесь, что каталоги KDE упомянуты в переменной PATH. По умолчанию пакеты устанавливаются в каталог */opt/kde*. Следует помнить, что в некоторых дистрибутивах используется другой путь к KDE, но он обычно настраивается стандартной установкой.

Обзор рабочего стола

На рис. 16.1 приведен типичный рабочий стол KDE. Полоска в верхней части экрана – это панель задач. В ней перечисляются приложения, работающие в среде. Основная панель расположена внизу экрана. Она содержит кнопку вызова главного меню, список окон, переключатель рабочих столов, а также кнопки для запуска приложений.

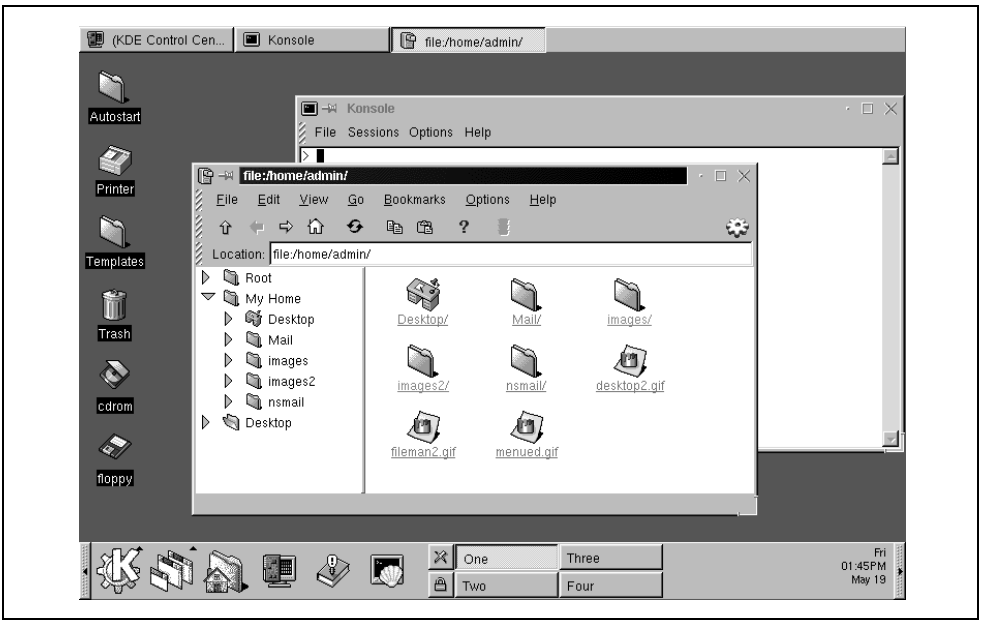


Рис. 16.1. Рабочий стол KDE

На рабочем столе отображаются окна открытых приложений и пиктограммы, которые можно использовать для запуска приложений «одним щелчком». По умолчанию на рабочем столе расположен определенный набор значков, два из которых связаны с папками. Первая папка – Templates (Шаблоны) – содержит набор файлов, служащих для создания ссылок на рабочем столе. Папка Autostart (Автозапуск) содержит ссылки на приложения, которые запускаются автоматически в начале сеанса работы. Пиктограмма Trash связана с особой папкой рабочего стола, в которую можно перетаскивать файлы, подлежащие удалению. Также существуют пиктограммы, связанные с приводом CD-ROM и дисководами.

Окна приложений

Каждое окно в KDE имеет заголовок с одинаковыми кнопками в правой части, которые предназначены для сворачивания, распахивания и закрытия окон. В левой части заголовка расположена маленькая пиктограмма (или дефис, если пиктограмма не определена приложением) и кнопка, которая похожа на канцелярскую. Щелчок по пиктограмме вызывает меню окна, которое содержит варианты различных действий, доступных для данного окна, в том числе и перемещение окна на другой рабочий стол. Пиктограмма в виде канцелярской кнопки служит для открепления или прикрепления окна к экрану. Если щелкнуть по ней, окно станет «липким» и будет присутствовать на всех виртуальных рабочих столах. Если окно является прикрепленным, отображается нажатая кнопка. Повторное нажатие открепляет окно.

Меню окна содержит стандартные команды: Maximize, Iconify, Move, Resize, Sticky и Close, а также команду, которая позволяет переместить окно на другой виртуальный рабочий стол.

kfm – файловый менеджер KDE

Одним из наиболее важных приложений среды KDE является **kfm** – графический файловый менеджер и броузер Интернета. Помимо этого **kfm** управляет рабочим столом KDE и размещенными на нем пиктограммами. Если нажать кнопку Home на панели или щелкнуть по пиктограмме Autostart на рабочем столе, откроется окно **kfm**, в котором отображается содержимое каталога. На рис. 16.2 показано окно **kfm** для домашнего каталога. Файлы и каталоги по умолчанию отображаются в виде пиктограмм, но с помощью меню View можно настроить параметры отображения информации.

Интерфейс **kfm** происходит от веб-броузеров. Панель инструментов содержит кнопки Back (Возврат) и Forward (Вперед) для навигации по выбранному каталогу, кнопки home, reload (обновить) и stop. В поле Location отображаются URL как для сетевых адресов, так и для имен локальных файлов. В **kfm** встроена качественная поддержка отображения HTML-документов, хотя и не такая надежная, как в коммерческих веб-броузерах. Для тех, кому не нужны примочки и продвинутые возможности, такие как апплеты Java или выполнение сценариев, **kfm** будет отличным простым веб-броузером. Он да-

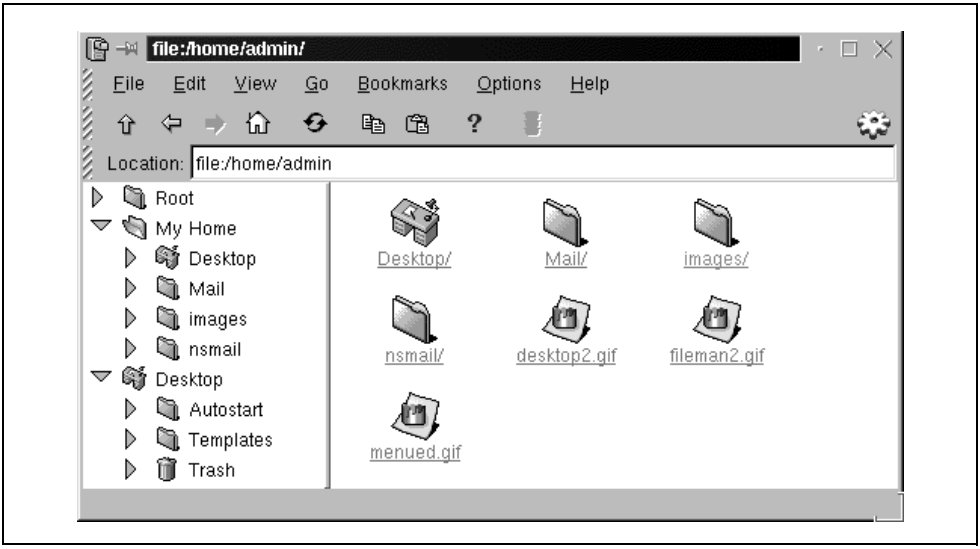


Рис. 16.2. kfm – файловый менеджер KDE

же позволяет сохранять адреса в виде закладок и использовать файлы HTTP cookies (файлы предпочтений).

В дополнение к управлению файлами **kfm** отвечает также за функциональность рабочего стола. В начале рабочего сеанса KDE **kfm** просматривает содержимое каталога *Desktop* (Рабочий стол). Все файлы или папки будут представлены на рабочем столе в виде пиктограмм, являющихся ссылками. Элементы, содержащиеся в папке *Autostart*, будут запущены. Для настройки ссылок на рабочем столе в **kfm** используются текстовые файлы с суффиксом *.kdelnk*, описываемые ниже в данной главе.

Создание ссылки на рабочем столе

Существует несколько способов создания ссылки на рабочем столе. Самый простой – щелкнуть правой кнопкой мыши на рабочем столе и выбрать пункт New контекстного меню, в котором содержится ряд элементов, соответствующих типам ссылок: Folder (Папка), File System Device (Файловая система), FTP URL, Mime Type (Mime-тип), Application (Приложение), Internet Address (Адрес Интернета) и WWW URL. Когда пользователь сделает выбор, на рабочем столе будет создана стандартная пиктограмма соответствующего типа и откроется окно свойств для созданной ссылки.

Окно свойств может быть разным для различных типов ссылок, но в любом случае необходимо указать имя файла ссылки, имя пиктограммы, а также выполняемую команду или расположение файла. Кроме того, можно установить права доступа для файла ссылки и выбрать другую пиктограмму.

Следующий пример иллюстрирует создание ссылки рабочего стола для приложения Kedit (текстовый редактор). Сначала щелкните правой кнопкой мыши на рабочем столе и выберите пункт New → Application. На рабочем столе появится стандартная пиктограмма KDE, изображающая шестеренку и обозначенная как «Application» (Приложение), после чего будет открыто окно свойств.

Вкладка General (Общие настройки), показанная на рис. 16.3, содержит имя по умолчанию для *kdelnk*-файла и другую информацию о файле. Измените имя файла, чтобы оно соответствовало назначению ссылки; в данном примере это будет *Kedit.kdelnk*.

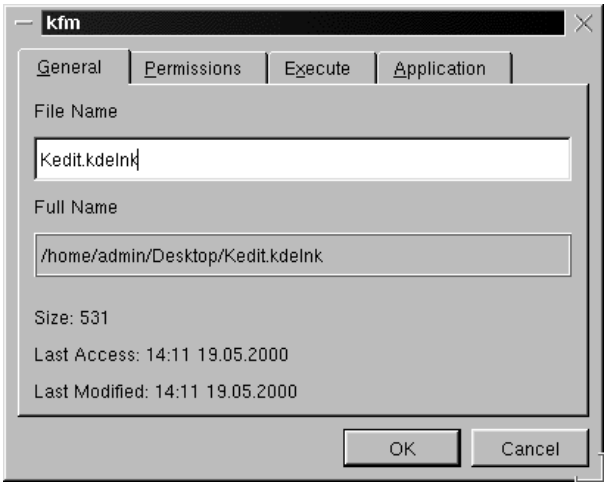


Рис. 16.3. Свойства ссылки рабочего стола, вкладка General

Правами доступа, позволяющими работать с файлом ссылки, обладает пользователь, который этот файл создал. Права доступа можно изменить с помощью вкладки Permissions. Теперь необходимо определить команду, предназначенную для выполнения приложения. На вкладке Execute (Выполнить), показанной на рис. 16.4, введите команду либо нажмите кнопку Browse (Обзор), чтобы найти файл. Здесь же может быть изменена пиктограмма для ссылки путем нажатия кнопки с изображением текущей пиктограммы. В результате открывается окно, содержащее набор стандартных пиктограмм KDE, доступных в данной системе. Выберите одну из них и нажмите OK.

И, наконец, необходимо ввести имя ссылки и текст всплывающей подсказки. Заполните поле Comment на вкладке Application (Приложение), показанной на рис. 16.5, текстом, описывающим приложение. Введите имя пиктограммы (отображаемое непосредственно под ней на рабочем столе) в поле Name. Чтобы закончить настройку, нажмите кнопку OK.

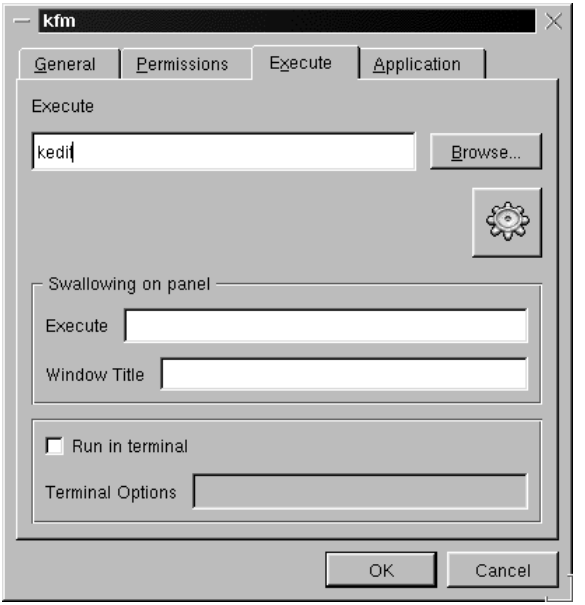


Рис. 16.4. Свойства ссылки рабочего стола, вкладка Execute

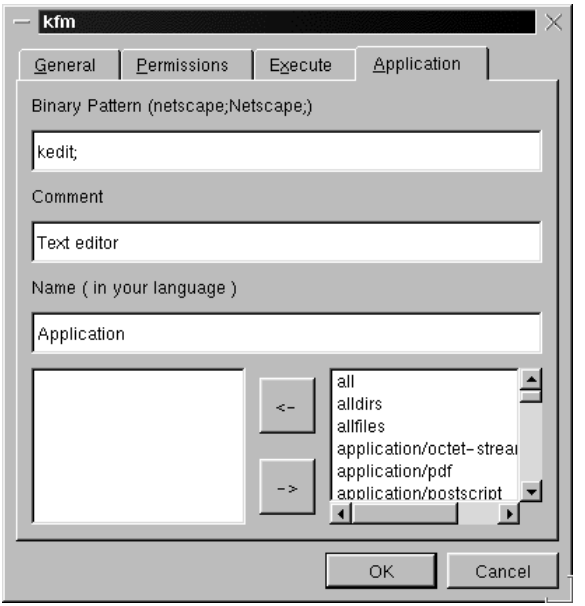


Рис. 16.5. Свойства ссылки рабочего стола, вкладка Application

Для настройки адресов URL или Интернета доступны все те же свойства, кроме стандартных имен и пиктограмм. Для этого типа ссылок необходимо указывать URL на вкладке URL (рис. 16.6).

Рис. 16.6. Свойства ссылки рабочего стола, вкладка URL

Для новой ссылки на устройство определите имя ссылки и, при необходимости, права доступа. На вкладке Device определите местонахождение устройства, например */dev/fd3*, и точку монтирования. Укажите тип файловой системы в поле Filesystems (например, default, msdos и т. д.).

Также можно создать ссылку на рабочем столе, перетащив на него элемент (любой файл или каталог) из окна файлового менеджера. После перетаскивания элемента на рабочий стол появляется диалоговое окно, предлагающее скопировать (copy), переместить (move) или создать ссылку (link) на элемент. Копирование создает простую копию элемента в каталоге *Desktop* (Рабочий стол). Перемещение удаляет исходный элемент и помещает его в каталог *Desktop*. Если выбрать опцию link, то пиктограмма будет содержать символическую ссылку, указывающую на выбранный элемент.

Папка Desktop и kdelnk-файлы

Все отображаемые элементы рабочего стола находятся в папке *~/Desktop*. Если открыть эту папку в файловом менеджере, можно обнаружить каталоги *Templates*, *Autostart*, а также файлы *.kdelnk* для привода CD-ROM и других созданных ссылок. При запуске KDE сканирует содержимое каталога *Desktop* и создает пиктограммы для всех элементов.

Ссылки рабочего стола, связанные с приложениями, URL-адресами или файлами, настраиваются без участия пользователя посредством файлов *.kdelnk*. Это обычные текстовые файлы, которые содержат установленную для ссылок (посредством диалоговых окон настройки) информацию. И хотя вся настройка ссылок рабочего стола замечательно производится с помощью всплывающих окон настройки, содержимое *.kdelnk*-файлов может заинтере-

совать более опытных пользователей. Ниже приводится пример такого файла для ссылки на текстовый редактор Kedit:

```
# KDE Config File
[KDE Desktop Entry]
Name[ ]=Kedit
Exec=kedit
Type=Application
Comment[ ]=Text editor
BinaryPattern=kedit;Kedit
Icon=exec.xpm
TerminalOptions=
Path=
Terminal=0
MimeType=
SwallowExec=
SwallowTitle=
Name[fi]=Sovellus
Name[hr]=Program
Name[sl]=Uporabniki program
Name[pl]=Aplikacja
...
```

Как можно заметить, формат файла прост и очевиден. Каждый из элементов, определенных в свойствах, располагается на отдельной строке. Параметр `Type` определяет тип файла ссылки. В данном примере Kedit является приложением. `Type=URL` означает, что речь идет о ссылке, связанной с адресом Интернета. Параметр `Name` определяет имя приложения, `Exec` — выполняемую команду, `Icon` — имя файла отображаемой пиктограммы и т. д. Неопределенные параметры имеют пустые значения. Дополнительные строки параметров имени приложения устанавливаются в файлах шаблонов и определяют альтернативные имена для других языков на тот случай, если произойдет переключение используемого в среде языка.

Основная панель и панель задач

Программа **kpanel** отвечает за работу основной панели и панели задач. Основная панель представляет собой полоску в нижней части экрана, содержащую управляющие элементы. Панель предназначена для поиска и запуска приложений, а также навигации по окнам и рабочим столам. Она содержит меню, в котором установленные приложения KDE организуются в подменю, средство навигации в файловой системе, а также «пейджер» рабочего стола (desktop pager). На панель могут добавляться новые кнопки для запуска приложений, открытия каталогов и URL-адресов.

Пейджер и список окон

Как и большинство оконных менеджеров, **kwm** (оконный менеджер KDE) позволяет разделять рабочее пространство на несколько рабочих столов.

Окна приложений могут открываться на различных рабочих столах, обеспечивая, таким образом, больший порядок в рабочем пространстве. Переключение между рабочими столами можно осуществить с помощью кнопок на панели или посредством списка окон. На панели отображается сетка кнопок – по одной на каждый виртуальный рабочий стол. Каждой кнопке присвоено имя (по умолчанию – One, Two, Three и т. д.). Нажатие кнопки переключает область просмотра на указанный рабочий стол.

Если нажать кнопку списка окон, будет отображено меню, разбитое на секции (по одной на каждый рабочий стол) и элементы (каждый элемент соответствует определенному окну, а список окон также доступен по щелчку средней кнопкой мыши по рабочему столу). К примеру, если второй рабочий стол содержит открытое окно файлового менеджера, можно выбрать это окно в списке окон, после чего произойдет переключение на второй рабочий стол, а окно файлового менеджера станет активным.

Количество виртуальных рабочих столов и их имена можно определить в Центре управления в разделе Desktops модуля настройки панели.

Панель задач

Панель задач расположена в верхней части экрана. Она облегчает наблюдение за запущенными приложениями. Панель задач содержит кнопки, каждая из которых связана с окном открытого приложения. Если щелкнуть по кнопке, окно приложения станет активным. Активному окну соответствует нажатая кнопка на панели задач. Свернутое окно можно восстановить нажатием кнопки на панели инструментов. Если окно свернуто, соответствующий текст на вкладке панели задач берется в скобки.

Если для сворачивания основной панели используются кнопки со стрелками, на панели инструментов отображаются кнопки доступа к главному меню, дисковому навигатору и списку окон. Эти кнопки исчезают при восстановлении панели.

В модуле настройки панели в Центре управления можно изменять размещение панели задач на экране или вовсе отключить ее отображение. Можно настроить автоматическое сворачивание неиспользуемой панели.

Создание на панели ссылки на приложение

Самый простой способ добавить кнопку приложения на панель – перетащить туда пиктограмму с рабочего стола. Произойдет копирование соответствующей ссылки. Любое приложение из главного меню можно легко добавить на панель. Сначала в главном меню выберите пункт Panel → Add Application, а затем – нужный элемент из предложенного списка. Перечисляются те приложения, которые доступны в главном меню.

Для того чтобы запретить (или, наоборот, разрешить) отображение некоторых стандартных кнопок панели, можно воспользоваться меню панели. Пункты disk navigator (дисковый навигатор) и window list (список окон) меню

панели позволяют переключать отображение кнопок. Если пиктограмма элемента меню «вдавлена», кнопки отображаются. Чтобы запретить отображение, необходимо просто выбрать соответствующий пункт меню.

Выполнение приложений на панели

Подчиненное приложение – это программа, которая выполняется на панели, а не в отдельном окне. Подчиненное приложение может быть небольшой утилитой, которая, к примеру, отслеживает активность сетевых соединений или наличие новых почтовых сообщений. На вкладке Execute в окне свойств в разделе Swallowing on Panel введите команду, которая должна выполняться, и текст, который должен фигурировать в заголовке окна приложения. Панель определяет приложения, которые необходимо подчинять, исходя из текста в заголовке окна.

Центр управления KDE

Центр управления KDE содержит ряд инструментов настройки, называемых *модулями*. Они позволяют изменять и просматривать информацию о системе. Пользователь может настроить рабочий стол, оконный менеджер, устройства ввода и другие важные компоненты системы. Центр управления разделен на два окна: в левом отображается иерархия установленных модулей, а в правом – выбранные модули. Также существует возможность отображения каждого модуля в отдельном окне. Для этого в меню Options (Параметры) выключите выбранный по умолчанию пункт Swallow Modules.

Приложения

Модули Applications позволяют настраивать работу наиболее важных составляющих KDE, таких как менеджер сеансов, **kfm** и панель.

Менеджер сеансов (только для суперпользователя)

Менеджер сеансов KDE (login manager) – это программа, которая заведует графическим окном для входа в систему. Этот модуль позволяет изменить графический стиль экрана, а также установить некоторые стандартные параметры, такие как предварительный список пользователей и доступные окружения для сеансов работы.

На вкладке Appearance (Вид) можно изменить строку приветствия, которая отображается на экране, и выбрать логотип. Выпадающее меню позволяет выбрать стиль окна для входа в систему: стиль Windows либо стиль Motif. Параметр Language (Язык) на этой же вкладке позволяет выбрать кодировку по умолчанию для менеджера сеансов.

Вкладка Fonts (Шрифты) позволяет определить стили и размеры шрифтов для экранных сообщений: Greeting (Приветствие), Fail (Ошибка) и Standard (Стандартное). Выберите тип сообщения, для которого необходимо выполнить настройку, затем нажмите кнопку Change Font (Изменить шрифт). В

открывшемся окне можно выбрать шрифт из перечня доступных, а также определить его размер. Нажатие кнопки OK закрывает окно. Выбранный шрифт отображается на этой же вкладке в окне Example (Пример).

Вкладка Background (Фон) позволяет выбрать фоновое изображение для экрана входа в систему. Как и в случае с фоном для рабочего стола, можно выбрать как заливку одним цветом, так и изображение.

Вкладка Users позволяет вывести список пользователей на экране входа в систему. Для каждого пользователя отображается регистрационное имя и логотип. Когда пользователь щелкает по своему логотипу, имя пользователя автоматически вписывается в поле ввода, но пароль все равно должен вводиться вручную. Вкладка содержит списки всех пользователей, выбранных пользователей и неотображаемых пользователей. Можно выбирать имена из списков и использовать кнопки со стрелками для перемещения имен из одного списка в другой. Существует два режима задания отображения. Можно поместить имена в список выбранных пользователей и нажать кнопку Show Only Selected Users (Отображать только выбранных пользователей) либо отредактировать список всех пользователей и нажать кнопку Show All Users But No-Show Users (Все пользователи кроме неотображаемых). Неотображаемым пользователям соответствуют идентификаторы, предназначенные для работы с системными ресурсами, доступ к которым должен быть ограничен (например, **root**, **news** и **nobody**).

Вкладка Sessions (Сеансы) содержит настройки сеансов работы. Раскрывающийся список в верхней части вкладки позволяет определить, кому разрешено останавливать работу системы. Доступные варианты: None (Никому), All (Всем пользователям), Root Only (Только суперпользователю) и Console Only (Только пользователю с консоли). В разделе Commands можно определить команды, используемые для останова и перезапуска системы. Раздел Session Types (Типы сеансов) позволяет настроить список типов сеансов, доступных пользователю, входящему в систему. Список Default (Стандартный) содержит различные среды, уже доступные в системе, такие как KDE и GNOME. Типы можно как добавлять в список, так и удалять из него.

Менеджер файлов

В модуле File manager содержатся настройки **kfm** – файлового менеджера и веб-броузера KDE. Первая вкладка содержит параметры шрифтов. Размер шрифта может быть установлен кнопками small (мелкий шрифт), medium (средний) или large (крупный). Стили шрифтов, используемые при отображении файлов или веб-страниц, устанавливаются настройками Standard Font (Стандартный шрифт) и Fixed Font (Моноширинный шрифт). Возможно указание кодировки.

На вкладке Color (Цвет) доступны настройки цветов фона, текста и ссылок. Для ссылок можно определить два различных цвета: для просмотренных ссылок (запись о которых имеется в журнале URL-адресов) и новых ссылок. Здесь же доступны настройки курсора (над ссылкой) и подчеркивания ссылок. Пользователь может также задать стандартные цвета для HTML-доку-

ментов, которые будут иметь больший приоритет, чем цвета, определенные непосредственно в документах.

Вкладка Other (Прочее) позволяет определить настройки, специфичные для URL-адресов. Если при просмотре документа по определенному адресу или содержимого локального каталога определить некоторые свойства просмотра, такие как отображение меню или свойства обзора, то эти настройки можно сохранить до следующего визита на этот URL. Кнопка Tree View Follows Navigation, будучи нажатой, приводит к автоматическому отображению на левой панели файлового менеджера содержимого папок, открытых в данный момент на правой панели. И еще одна настройка этой вкладки позволяет выбрать стандартные программы, используемые файловым менеджером в качестве терминала, если пользователь выбирает пункт Open Terminal из меню File, а также редактор, предназначенный для просмотра исходных текстов документов.

Веб-броузер

Модуль Web-browser содержит настройки, используемые **kfm** для навигации по веб-пространству.

Если система находится за брандмауэром, вам может понадобиться использовать прокси-сервер для служб HTTP и FTP. Вкладка Proxy позволяет включить доступ через прокси-сервер и указать необходимые для этого параметры. Отметьте пункт Use Proxy (Использовать proxy), а затем введите имя узла или IP-адрес прокси-сервера и номер используемого порта. Работа с некоторыми сайтами может быть некорректной при доступе через прокси-сервер. В поле No Proxy For можно перечислить узлы и домены, к которым необходимо обращаться напрямую, а не через прокси-сервер.

Вкладка HTTP позволяет определить значения некоторых параметров, посылаемых броузером в теле HTTP-запросов. Поле Accept Languages (Допустимые языки) содержит перечень двухбуквенных сокращений, которые определяют языки документов, принимаемые веб-броузерами. Сокращения (например, «en» для английского языка или «fr» – для французского) могут разделяться запятыми или пробелами. Порядок элементов списка определяет предпочтительность документов на различных языках. Это значение определяет HTTP-заголовки Accept-Language.

Поле Accept Character Sets содержит перечень кодовых страниц, которые броузер может принимать (значение определяет содержимое заголовка HTTP-запроса Accept-Charset). Список должен содержать стандартизованные указатели на кодировки, разделяемые пробелами или запятыми. При помощи кнопки Assume HTML можно указать броузеру, что документы, тип которых не до конца определен в заголовке ответа сервера, следует обрабатывать как HTML-содержащие документы.

Вкладка User Agent (Клиент пользователя) позволяет определить соответствующую строку, которую броузер посылает сайтам, не способным опознать **kfm** или другой веб-броузер, не являющийся распространенным коммерческим продуктом. Эта строка содержит имя и номер версии программы-клиен-

та, которая посылает запрос. На момент написания этой книги строкой по умолчанию для **kfm** является «Konqueror/1.1.2». Чтобы определить другое значение для конкретного сервера, введите имя сервера в поле On Server, а затем строку, определяющую значение user-agent, в поле Login As. Все настройки перечислены в поле Known Bindings. Для того чтобы удалить настройку, выберите ее в списке и нажмите кнопку Delete.

Вкладка Cookies определяет, как **kfm** будет работать с файлами cookies. Эти файлы игнорируются, если не выбран переключатель Enable Cookies в верхней части вкладки. Необходимо определить стандартное правило работы с файлами cookies. Доступные варианты: принимать все cookies (accept), отвергать все (reject) или же каждый раз запрашивать необходимые действия у пользователя (ask). Можно также определить правила для отдельных доменов. Введите имя домена в поле Change Domain Accept Policy, а затем выберите правило – Accept, Ask или Reject. Нажмите кнопку Change, чтобы сохранить изменения. Новая запись появится в списке Domain Specific Settings. Чтобы удалить запись, выберите ее из списка и нажмите кнопку Delete.

Панель

Модуль Panel отвечает за работу панели, панели инструментов, виртуальных рабочих столов и дискового навигатора.

Вкладка Panel содержит настройки, относящиеся к размещению панели. Панель можно расположить вдоль нижнего, верхнего, правого или левого края экрана. Размер панели может принимать значения Tiny (Крохотный), Normal (Обычный) или Large (Большой). Здесь же доступны настройки отображения панели задач. Если требуется, чтобы панель задач не отображалась, выберите в списке пункт Hidden (Скрытая). Кнопки Top и Bottom позволяют задать отображение панели по верхнему или по нижнему краю экрана. При отображении по нижнему краю экрана панель задач будет располагаться прямо над основной панелью. И, наконец, вариант Top/Left позволяет отображать панель задач в верхнем левом углу экрана, при этом кнопки задач располагаются одна над другой.

Вкладка Options (Настройки) разделена на три части (рис. 16.7). Раздел Menu Tooltips (Всплывающие подсказки меню) содержит переключатель отображения всплывающих подсказок для элементов меню. Количество времени, которое должно пройти, прежде чем появится подсказка, можно установить при помощи ползунка. Раздел Visuals (Эффекты) позволяет задать автоматическое сворачивание неиспользуемых панелей – главной и панели задач. Для этого необходимо установить флажки Auto Hide Panel (Автоматически сворачивать панель) и Auto Hide Taskbar (Автоматически сворачивать панель задач). Для каждой панели при включенном режиме доступны два ползунка: Delay, определяющий временной интервал, в течение которого панель не использовалась (т. е. в ее границах отсутствовал указатель мыши), и Speed, определяющий скорость сворачивания и восстановления панели. Флажок Animate Show/Hide отвечает за наличие спецэффекта «скольжения» при сворачивании или восстановлении панели с помощью кнопок со стрелками. Скорость анимации определяется соседним ползунком.

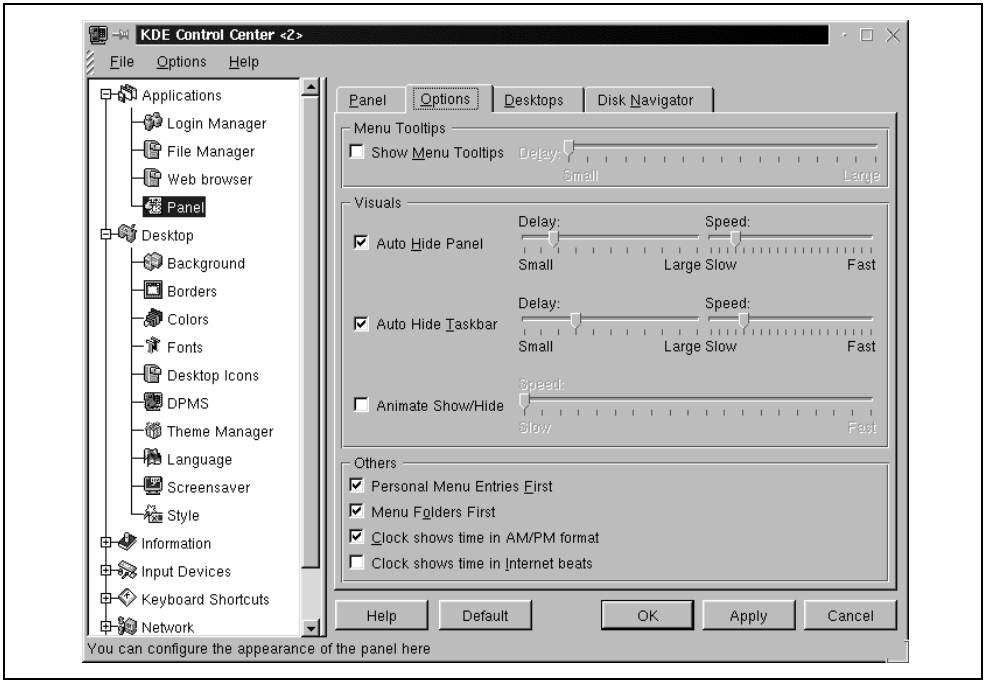


Рис. 16.7. Настройка параметров панели

На вкладке Options (Настройки) содержатся следующие дополнительные параметры (рис. 16.7):

Personal Menu Entries First

Установите данный флажок, чтобы предписать отображение пользовательского раздела меню вместо стандартного системного в самом начале главного меню. Системные меню в этом случае будут содержаться в стандартном подменю.

Menu Folders First

Данный флажок следует установить, чтобы задать отображение папок над остальными элементами в пользовательском меню. Эта настройка не принимается во внимание при изменении порядка расположения элементов меню с помощью соответствующего редактора.

Clock shows time in AM/PM format

Установка этого флажка позволяет выводить время на часах в панели в формате AM/PM. Обычно время отображается в 24-часовом формате.

Clock shows time in internet beats

Этот параметр позволяет выводить время на часах в панели в Swatch-формате, в котором день разделен на 1000 отметок. Часовой пояс принимает значение GMT+1.

Вкладка Desktops (Рабочие столы) позволяет определить количество виртуальных рабочих столов, которые составляют рабочее пространство. По умолчанию определены 4 стола, максимум – 8. Ползунок Visible (Видимый) определяет количество отображаемых рабочих столов (доступны только четные числа). Рабочие столы перечисляются и нумеруются в интервале от 1 до 8. При включенном списке рабочих столов рядом с ним располагается поле ввода, которое позволяет определить метку для рабочего стола, используемую в проводнике панели. Ползунок Width определяет ширину кнопок проводника, отображаемых на панели.

Вкладка Disk Navigator (Дисковый навигатор) содержит настройки дискового навигатора меню и панели задач. В разделе History (Журнал) этой вкладки расположены две кнопки: Edit Personal и Edit Shared, которые позволяют редактировать содержимое пользовательского и общего разделов дискового навигатора с помощью файлового менеджера.

Кнопка Edit Personal открывает **kfm** для каталога *~/.kde/share/apps/kdisknav*. Эта папка в домашнем каталоге пользователя содержит *kdelnk*-файлы для элементов, отображаемых в пользовательском разделе Personal дискового навигатора. Кнопка Edit Shared открывает **kfm** для каталога */usr/share/apps/kdisknav*, содержащего *kdelnk*-файлы для общего раздела Shared. Данные в этом каталоге может редактировать только привилегированный пользователь, всем остальным кнопка недоступна.

Дополнительные настройки определяют количество записей для папок и файлов, которое отображается в разделе дискового навигатора Recent, а также максимальное количество файлов, которое может отображаться для папки.

Раздел вкладки Disk Navigator под названием Options (Настройки) содержит следующие настройки: Show Dot Files (отображать скрытые файлы, имена которых начинаются с точки), Show Shared Section (Отображать общий раздел), Show Recent Section (Отображать раздел «недавнее»), Show Personal Section (Отображать пользовательский раздел), Show Option Entry (Отображать параметры) и Ignore Case When Sorting (Игнорировать регистр символов при сортировке). Здесь также можно определить используемый в качестве стандартного эмулятор терминала.

Вид рабочего стола

Модули Desktop определяют внешний вид рабочей среды (фон, шрифты или цвета для окон). Можно также с помощью менеджера тем (Theme Manager) определить сразу все параметры в соответствии с одним из установленных пакетов тем.

Выбор фона

Модуль Background позволяет определить фон рабочего стола. Можно задать фон для каждого виртуального рабочего стола или один общий фон для всех столов. Выберите из списка рабочий стол, для которого необходимо изменить фон, либо отметьте поле Common Background (Общий фон). Можно вы-

брать цвет фона или изображение. Вариант One Color (Один цвет) позволяет залить фон сплошным цветом, а вариант Two Color (Два цвета) – смешать два цвета. Выберите нужные цвета в палитрах цветов. Кнопка Setup открывает диалоговое окно для двухцветной настройки, предоставляя выбор между вертикальным, горизонтальным или узорным смешением цветов.

Для использования в качестве фона графического изображения выберите его из раскрывающегося списка или нажмите кнопку Browse (Обзор), чтобы выбрать изображение из файлов системы. Расположение изображения контролируется дополнительной настройкой Arrangement, предоставляющей эффекты от простого размножения до симметричного зеркального отображения. Можное выбрать и случайный фон. В таком случае для каждого сеанса работы будет использоваться новое изображение.

Границы

В модуле Borders определяются свойства границ окон и экрана. Активные границы рабочего стола позволяют переключаться между рабочими столами перемещением указателя мыши за соответствующую границу. Отметьте поле Active desktop borders, чтобы включить этот режим. Ползунок Desktop switch-delay позволяет задать интервал времени до непосредственного переключения на соседний рабочий стол. Значение должно быть удобным для работы и достаточно большим, чтобы не происходило переключение на другие рабочие столы по ошибке. Для того чтобы указатель мыши после переключения оказывался в центре экрана, отметьте поле Move Pointer Towards Center After Switch.

Параметр Magic Borders определяет зоны «привязки» вокруг окон и по краям экрана. Ширина зоны задается количеством точек, отделяющих перемещаемое в зону окно от края экрана или границы ближайшего окна.

Цвета

Модуль Colors позволяет определять цвета различных компонентов, из которых состоят окна приложений. Цвета могут выбираться на основе схемы, уже определенной в системе. Доступные схемы перечислены здесь же. Необходимо выбрать схему из списка и нажать кнопку Add (Добавить) для замены текущей схемы. Цвет каждого компонента может определяться отдельно. В верхней части вкладки отображены примеры компонентов окон. Чтобы выбрать компонент для настройки, например, активный заголовок окна или фон окна, необходимо щелкнуть по нему мышью. Компоненты также можно выбирать из отдельного списка. Щелкните по палитре цветов, расположенной под списком, и выберите в открывшемся диалоговом окне нужный цвет. Контрастность компонента можно изменить с помощью ползунка Contrast.

Шрифты

Модуль Fonts позволяет определить стандартные шрифты. Доступно изменение шрифта для следующих параметров: основной шрифт (general font), шрифт фиксированной ширины (fixed-width font), шрифт для заголовков окон (window titles), шрифт кнопок панели (panel buttons) и шрифт для ча-

сов панели. Шрифт можно выбрать из выпадающего списка и определить его начертание – полужирное или курсивное. Можно изменить размер шрифта и, при необходимости, кодировку. Внизу окна отображается образец выбранного шрифта.

Пиктограммы рабочего стола

Модуль Desktop icons содержит настройки, относящиеся к отображению пиктограмм на рабочем столе. Пиктограммы могут помещаться в клетках невидимой сетки, которой расчерчен рабочий стол. Параметр grid spacing определяет обязательный отступ для каждой пиктограммы. Два регулятора позволяют изменить значение этого отступа по горизонтали и вертикали.

По умолчанию фон подписей к пиктограммам прозрачный, т. е. поверхность рабочего стола видна под ними. Для того чтобы сделать подписи непрозрачными, снимите отметку с поля Transparent Text for Desktop Icons. Цвет текста в метках можно определить, нажав кнопку Icon Foreground Color. Если фон подписей непрозрачен, то их цвет также может быть определен нажатием кнопки Icon Background Color.

И последняя настройка этой вкладки позволяет разрешить отображение скрытых файлов (имена которых начинаются с точки) на рабочем столе.

Система управления питанием монитора

DPMS – это аббревиатура Display Power Management System (Система управления питанием монитора). Если аппаратная часть компьютера поддерживает управление питанием, то можно включить программное управление, нажав кнопку DPMS enabled. После этого могут быть заданы интервалы времени бездействия пользователя, по истечении которых система переходит в режим пониженного энергопотребления, затем в режим приостановки и, наконец, просто выключается.

Менеджер тем

Темы (модуль Theme Manager) позволяют придать единство стиля элементам рабочего пространства, избавляя пользователя от необходимости настраивать каждый элемент отдельно. Тема может определять цветовую схему для окон, стили шрифтов, пиктограмм, фона и даже звуковое сопровождение работы. В стандартную поставку KDE входит несколько тем, а все остальные доступны для загрузки по адресу *http://kde.themes.org*.

На вкладке Installer (Средство установки) (рис. 16.8) перечислены установленные в системе темы. Здесь отображаются общесистемные темы, которые хранятся в подкаталоге */share/apps/kthememgr/Themes* стандартного каталога KDE, и локальные темы, хранящиеся в каталоге *~/.kde/share/apps/kthememgr/Themes* в домашнем каталоге пользователя. Локальными темами являются те, которые устанавливаются и изменяются пользователем и хранятся в его домашнем каталоге. Пользователь может редактировать и со-

хранять локальные темы. Общесистемные темы не могут быть изменены обычными пользователями.

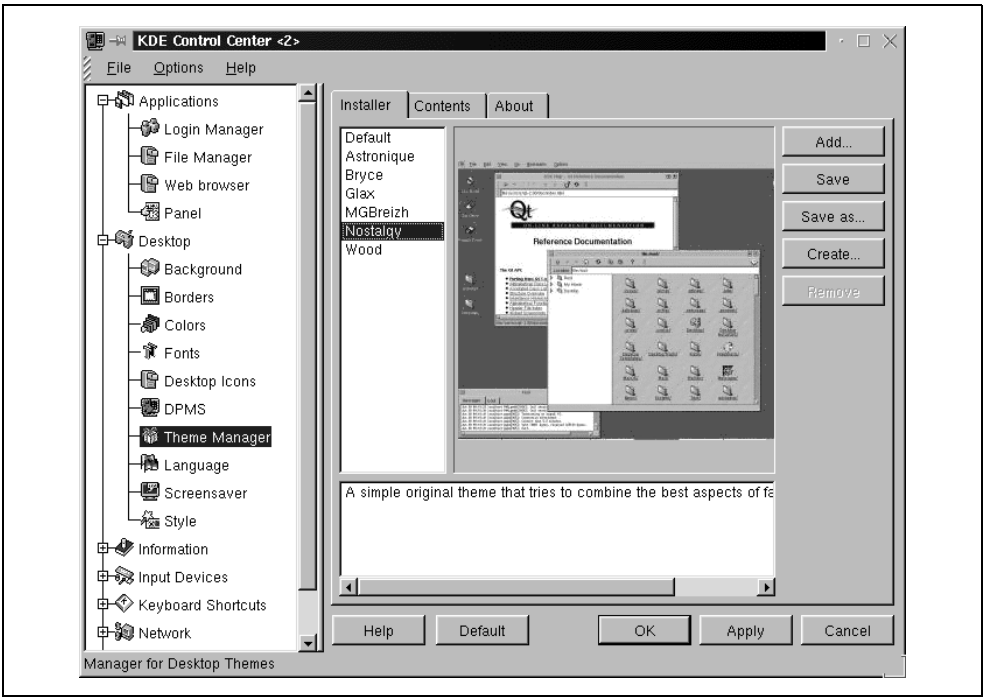


Рис. 16.8. Установка тем

При выборе темы из списка на вкладке появится изображение, показывающее рабочий стол с данной темой в действии, а также краткий сопроводительный текст. Выберите тему и нажмите кнопку Apply (Применить). Тема будет скопирована в рабочий каталог пользователя для тем (*~/.kde/share/apps/kthememgr/Work*).

Вкладка Installer также предоставляет средство управления установленными темами. Для установки новой темы нажмите кнопку Add. Укажите имя *.tgz*-файла темы и его расположение в диалоговом окне и нажмите кнопку OK. Кнопка Save позволяет сохранить только что настроенную тему или общесистемную тему в качестве локальной. Нажав кнопку Save As, можно сохранить используемую тему в качестве отдельного локального пакета без изменения исходной темы. Кнопка Create работает аналогично и позволяет скопировать текущую рабочую тему в новый локальный пакет с темой. Кнопка Remove удаляет локальную тему либо деактивизирует общесистемную.

На вкладке Contents (Содержимое) перечислены компоненты, настройки которых могут изменяться темой. Тема может содержать настройки не для всех перечисленных компонентов. Если тема содержит настройку определенного компонента, он входит в список доступных. В ином случае компо-

нент входит в список неопределенных. Переключатели позволяют определить, какие из определений компонентов темы необходимо использовать. Если оставить какую-либо настройку для определенного компонента выключенной, будет действовать соответствующая настройка из предыдущей темы, при условии, что ее информация все еще доступна в рабочем каталоге менеджера тем. Чтобы этого не произошло, активизируйте компонент, даже если он находится в списке неопределенных; в этом случае будут использованы стандартные настройки.

Вкладка About содержит информацию о версии выбранной темы и ее авторе.

Язык

Модуль Language позволяет определить выбор языка (locale) для программ пользователя. Выпадающие списки позволяют определить до трех языков, если они доступны.

Экранная заставка

В модуле Screensaver можно выбрать экранную заставку. Список доступных заставок приводится рядом с окном предварительного просмотра. Выбрав пункт No Screensaver в начале списка, вы откажетесь от заставки. Кнопка Setup открывает диалоговое окно, которое содержит специфические для данной заставки настройки. Чтобы проверить работу заставки в полноэкранном режиме, нажмите кнопку Test.

Раздел Settings (Настройки) позволяет установить интервал времени бездействия пользователя, по истечении которого происходит включение заставки. Введите значение этого параметра в минутах в поле Wait For. В поле Require Password можно ввести свой пароль, необходимый для отключения заставки и возврата к работе. Поле Show Password as Stars управляет маскированием текста вводимого пароля символами * (обычно вводимый текст не отображается). Настройка Priority (Приоритет) позволяет изменять приоритет работающего процесса заставки. Если машина активно работает в качестве сервера, установите небольшое значение приоритета, чтобы работа заставки не понижала производительность других программ.

Стиль

Модуль Style содержит несколько настроек, управляющих отображением окон и пиктограмм. Кнопка Draw Widgets in the Style of Windows 95 включает стиль компонентов окон, схожий со стилем, используемым в операционной системе Windows. Кнопка Menu bar on Top of the Screen in the Style of MacOS помещает меню окна в верхней части экрана. Настройка Apply Fonts and Colors to Non-KDE Apps обеспечивает использование выбранных стилей в программах, которые не разрабатывались специально под среду KDE.

В разделе Icon Style (Стиль пиктограмм) определяется размер пиктограмм, который может принимать значение Normal (Обычный) или Large (Крупный). Имеется возможность задать размер пиктограмм панели, файлового менеджера и рабочего стола, а также размер всех остальных пиктограмм.

Информационные модули

Модули Information позволяют просматривать сведения о состоянии различных частей системы. Настройки в этих модулях отсутствуют, но если необходимо просмотреть информацию о процессоре машины или об установленных PCI-устройствах, воспользуйтесь этими модулями. Доступны сведения по следующим системным компонентам:

- Devices (Устройства)
- DMA-Channels (DMA-каналы)
- Interrupts (Прерывания)
- IO-Ports (Порты ввода-вывода)
- Memory (Память)
- Partitions (Разделы)
- PCI (PCI-устройства)
- Processor (Процессор)
- SCSI (SCSI-устройства)
- Samba Status (Состояние Samba)
- Sound (Звук)
- X-Server (Сервер X Window System)

Устройства ввода

Модули раздела Input Devices позволяют выполнять настройку клавиатуры и мыши.

Клавиатура

В модуле Keyboard можно настроить автоповтор клавиатуры. Чтобы включить автоповтор, выберите поле On, а затем используйте ползунок для установки громкости щелчков, издаваемых при нажатии клавиш. Чтобы выключить щелчки, установите значение ползунка в 0.

Мышь

Данный модуль контролирует движение и раскладку кнопок мыши. Ползунок Acceleration (Ускорение) определяет скорость движения указателя мыши по экрану. Ползунок Threshold (Порог) устанавливает расстояние (в точках), на которое должна переместиться мышь, прежде чем начнется движение указателя. Параметры Button Mapping позволяют настроить раскладку кнопок для пользователя-правши или пользователя-левши.

Сочетания клавиш

Модули Keyboard Shortcuts позволяют выбрать схему для сочетаний клавиш – стандартную схему KDE или текущую, если вы изменяли настройки. В ниж-

нем разделе вкладки доступны для редактирования существующие сочетания клавиш. Действие можно не привязывать к какой-либо клавише либо привязать к стандартной или произвольной клавише. Для изменения сочетания отметьте соответствующий модификатор (Shift, Ctrl или Alt) либо снимите отметку со всех модификаторов, если модификация клавиши не нужна. Щелкните по кнопке, которая соответствует клавише, а затем нажмите эту клавишу на клавиатуре. Кнопка Save Changes позволяет сохранить изменения в текущей схеме.

Общие сочетания

В следующей таблице приведены клавиатурные комбинации, связанные с глобальными событиями для окон:

Сочетание	Действие
Alt-Esc, Ctrl-Esc	Отобразить список запущенных приложений
Alt-Tab, Alt-Shift-Tab	Переключение между окнами рабочего стола
Ctrl-Tab, Ctrl-Shift-Tab	Переключение между виртуальными рабочими столами
Alt-F2	Запустить командную строку
Alt-F3	Открыть управляющее меню текущего окна
Alt-F4	Закрыть текущее окно
Ctrl-F[1..8]	Переключиться на виртуальный рабочий стол с соответствующим номером
Ctrl-Alt-Esc	Принудительно закрыть текущее окно

Стандартные сочетания

Стандартные сочетания клавиш связаны со стандартными действиями, совершаемыми в большинстве приложений KDE, таких как файловый менеджер или графический текстовый редактор.

Сочетание	Действие	Сочетание	Действие
Ctrl-W	Закрыть	PageDown	Следующий
Ctrl-C	Скопировать	PageUp	Предыдущий
Ctrl-X	Вырезать	Ctrl-End	Конец документа
Ctrl-V	Поместить	Ctrl-Home	Начало документа
Ctrl-P	Печатать	Ctrl-O	Открыть
Ctrl-R	Замена	Ctrl-F	Поиск
Ctrl-Insert	Вставка	Ctrl-S	Сохранение
Ctrl-N	Новый	Ctrl-Z	Отмена
F1	Справка	Ctrl-Q	Выход

Звук

Модули этого раздела позволяют выполнить настройку клавиатурного сигнала и других звуков системы.

Сигнал

Этот модуль содержит настройки системного сигнала.

Ползунок Volume позволяет определить громкость звука, ползунок Pitch – высоту звука в герцах. Длительность звучания определяется ползунком Duration. Результат можно прослушать, нажав кнопку Test.

Системные звуки

Темы рабочего стола могут определять проигрывание звуковых *.wav*-файлов при наступлении определенных событий, таких как вывод окна на передний план или нажатие кнопки в диалоговом окне. Модуль System Sounds позволяет включить и настроить звуковые события. Включение производится отметкой поля Enable System Sounds.

Вкладка содержит две панели. В левой отображается список событий, к которым могут привязываться звуки, а в правой – список *.wav*-файлов, которые могут быть привязаны к событиям. Выбор события в левой панели приведет к автоматической подсветке имени файла, который с ним связан. Чтобы привязать звуковой файл к событию, выберите действие из списка событий, а затем звуковой файл на правой панели. Кнопка Test позволяет прослушать выбранный звуковой файл. Для сохранения изменений нажмите кнопку Apply.

Поведение окон

Модули раздела Window Behavior позволяют определить вид и функции заголовков окон, действия, связанные с нажатиями кнопок мыши, а также правила для передачи фокуса ввода и размещения окон.

Дополнительные настройки

В разделе Keyboard and Mouse (Клавиатура и мышь) вкладки Advanced (Дополнительно) содержатся следующие настройки:

Ctrl-Tab walks through desktops

Это поле связывает комбинацию клавиш <Ctrl>+<Tab> с последовательным переключением рабочих столов. По умолчанию настройка включена.

Alt-Tab is limited to current desktop

Данное поле ограничивает действие сочетания <Alt>+<Tab> переключением только между окнами текущего рабочего стола. По умолчанию настройка включена.

Alt-Tab mode

Эта настройка определяет стиль использования сочетания клавиш <Alt>+<Tab> для переключения между окнами. Вариант KDE setting определяет визуальный стиль. При нажатии комбинации <Alt>+<Tab> появляется список, в котором перечисляются существующие окна. В варианте CDE setting происходит немедленное переключение.

Grab the Right Mouse Button

Данная настройка по умолчанию включена. Правая кнопка мыши по умолчанию контролируется KDE. Для работы некоторых не-KDE приложений необходимо, чтобы они контролировали правую кнопку мыши самостоятельно. Если вы работаете с такими приложениями, отключите данную настройку.

Раздел вкладки Advanced под названием Filters содержит настройки, которые позволяют придавать окнам стиль или функциональность, исходя из их заголовков или классов. Раскрывающийся список содержит параметры, которые могут быть определены для «фильтруемого» окна. Рядом с меткой Windows Will (Окна...) можно выбрать следующие параметры:

- Have tiny decorations (имеют минимум декораций)
- Have no decorations (не имеют декораций)
- Never gain focus (никогда не получают фокус ввода)
- Start as sticky (изначально «липкие»)
- Be excluded from session management (исключаются из работы в сеансе)

Выберите параметр, затем определите окно в следующем разделе, который начинается словами «If They Match the Following» (Если они соответствуют следующему). Здесь доступны поля Titles (Заголовки) и Classes (Классы). Можно задать одно свойство или оба. Введите текст заголовка окна приложения в поле Title, а имя класса приложения – в поле Classes. (Обычно имя начинается с заглавной буквы и предваряется буквой X, например XEmacs для окна Emacs.) Может быть определено несколько заголовков или классов окна.

Кнопки

Этот модуль позволяет определить раскладку кнопок, которые фигурируют на заголовках окон. Существует пять кнопок: свернуть (точка), максимизировать (квадрат), приклеить (канцелярская кнопка), закрыть (X) и меню (дефис или пиктограмма, определяемая приложением). Каждая кнопка имеет три варианта размещения: left (слева), right (справа) или off (выключена). Можно поместить не более трех кнопок на одной стороне заголовка окна.

Мышь

Модуль Mouse позволяет определить действия мыши, связанные с различными компонентами окон. Раскрывающиеся меню содержат несколько параметров, таких как raise (вывести на передний план) или lower (вывести на зад-

ний план), для левой, средней и правой кнопок мыши (рис. 16.9). В каждой категории доступны следующие варианты:

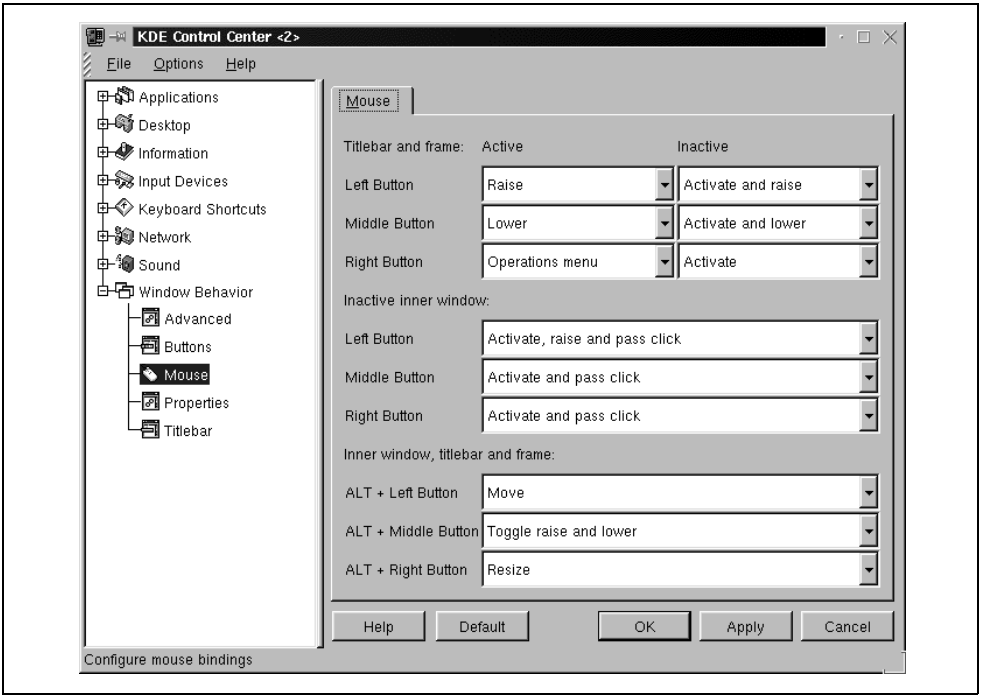


Рис. 16.9. Настройка действий для кнопок мыши

For active titlebars and frames (*Для активных заголовков и фреймов*)

Raise (На передний план)

Lower (На задний план)

Operations menu (Меню операций)

Toggle raise and lower (Переключение состояний)

For inactive titlebars and frames (*Для неактивных заголовков и фреймов*)

Activate (Активизировать)

Activate and raise (Активизировать и вывести на передний план)

Activate and lower (Активизировать и вывести на задний план)

For an inactive inner window (*Для неактивных дополнительных окон*)

Activate, raise and pass click (Активизировать, вывести на передний план и передать нажатие)

Activate and pass click (Активизировать и передать нажатие)

Activate (Активизировать)

Activate and raise (Активизировать и вывести на передний план)

For titlebars and frames of inner windows (*Для заголовков и фреймов дополнительных окон*)

Move (Переместить)

Toggle raise and lower (Переключение состояний)

Resize (Изменить размер)

Raise (На передний план)

Lower (На задний план)

Nothing (Нет действия)

Свойства

На вкладке Properties определяются параметры для окон, а также правила размещения и передачи фокуса. В разделе Windows (Окна) содержатся следующие настройки:

Vertical maximization only by default

Отметьте этот параметр, чтобы максимизировать окно только по высоте, оставляя ширину неизменной при выполнении команды «максимизировать».

Display content in moving windows

Отображать содержимое перемещаемых окон.

Display content in resizing window

Отображать содержимое окон во время изменения размера.

Resize animation

Этот ползунок контролирует анимацию сворачиваемых или максимизируемых окон. Предельное левое положение ползунка (None) отключает анимацию – окна будут сворачиваться и распахиваться мгновенно. Сдвигая ползунок вправо, можно определить различную скорость анимации.

Раздел Placement Policy (Правила размещения) вкладки Properties содержит выпадающий список с пятью параметрами, которые определяют правила размещения новых окон на рабочем столе:

Smart (*По умолчанию*)

При размещении новых окон система стремится избегать наложения одного окна на другое.

Cascade (*Каскад*)

Новые окна размещаются каскадом – каждое следующее окно помещается с определенным сдвигом относительно предыдущего.

Interactive (*Диалог*)

Эта настройка позволяет осуществлять «допустимое наложение». Если окно может быть размещено на рабочем столе без перекрытия определенного процента площади другого окна, оно будет размещено автоматичес-

ки. В противном случае вам будет предоставлена возможность самостоятельно разместить окно.

Random (*Случайным образом*)

Новое окно размещается на рабочем столе случайным образом.

Manual (*Вручную*)

Окно открывается, автоматически активизируется режим перемещения окна, а указатель мыши помещается над заголовком. Пользователю остается переместить окно в нужное положение и закрепить его нажатием кнопки мыши.

Раздел Focus Policy (Правила передачи фокуса) позволяет определить обстоятельства, при которых окно получает фокус ввода и его заголовок становится активным. Раскрывающийся список предоставляет выбор из четырех стилей. По умолчанию установлено правило Click to Focus, которое передает окну фокус ввода и выводит его на передний план, только когда пользователь щелкает по окну мышью. Все остальные правила связаны с перемещением указателя мыши и активизируют режимы Auto Raise (Автоматический перевод на передний план) или Click Raise (Перевод на передний план по щелчку).

Click to focus

Правило по умолчанию. Для передачи фокуса и перевода окна на передний план необходимо щелкнуть мышью в зоне окна. Это правило использует графический стиль комбинации клавиш <Alt>+<Tab> для переключения между окнами рабочего стола.

Focus follows mouse

Окно получает фокус, если указатель мыши оказывается в его пределах. Окно не переходит на передний план, если не отмечено поле Auto Raise. Окно сохранит фокус ввода, пока указатель мыши не окажется в пределах другого окна.

Classic focus follows mouse

Аналогично варианту Focus Follows Mouse, за исключением того, что окно теряет фокус, когда указатель мыши покидает его пределы. Если указатель не находится в пределах какого-либо окна, никакое из окон не владеет фокусом ввода. Нажатие клавиш <Alt>+<Tab> производит прямое переключение между окнами без отображения их списка.

Classic sloppy focus

Аналогично варианту Focus Follows Mouse – окно не теряет фокус, если указатель мыши покидает его пределы. Разница лишь в том, что используется прямое переключение между окнами по <Alt>+<Tab>.

Для всех нестандартных вариантов необходимо выбрать правило Auto Raise или Click Raise, чтобы иметь возможность помещать окна на передний план. Click Raise выводит окно на передний план по нажатию кнопки мыши на окне.

Auto Raise выводит окно на передний план после небольшой паузы, длительность которой можно определить с помощью ползунка Delay.

Заголовки окон

Вкладка Titlebar содержит многочисленные настройки представления и функциональности заголовков окон. Параметр Title Alignment (Выравнивание заголовка) определяет положение текста в заголовке окна: Left (По левому краю), Middle (По центру), Right (По правому краю).

Настройки раздела Appearance (Представление) отвечают за визуальные декорации заголовков:

Shaded Vertically

Двухцветный фон с вертикальным градиентом.

Shaded Horizontally

Двухцветный фон с горизонтальным градиентом.

Plain

Сплошной цвет фона.

Pixmap

Фоновое изображение для заголовка окна. В этом варианте становятся доступными настройки Pixmap, которые позволяют выбрать изображение для активного и неактивного состояния заголовка. Если для простоты восприятия вы желаете запретить отображение фона под текстом в заголовке, отметьте поле, обозначенное No Pixmap Under Text.

Отмеченное поле Active Title has Shaded Frame придает затененный трехмерный вид активным заголовкам.

Параметры Mouse Action определяют действие, связанное с двойным щелчком по заголовку окна. Доступны следующие варианты: (Un)Shade – Восстановить/Скрыть (по умолчанию), (Un)Maximize – Восстановить/Максимизировать, Iconify – Свернуть, (Un)Sticky – Обратить «липкость» и Close – Закрыть. Варианты с префиксом (Un) являются обратимыми действиями.

Для слишком длинных текстов в заголовках может быть включена анимация прокрутки, скорость которой устанавливается соответствующим ползунком.

Глава 17

Альтернативный оконный менеджер fvwm2

Одним из наиболее притягательных свойств Linux-систем является их гибкость, независимость от доминирующих в отрасли стандартов и степень контроля пользователя над рабочей средой. В большинство дистрибутивов Linux, как правило, входит графическая рабочая среда, укомплектованная удобными инструментами, меню и обладающая единством стиля. Наиболее широко распространены среды GNOME и KDE, настройка которых подробно описана в главе 15 «GNOME» и главе 16 «KDE», соответственно.

Обе графические среды предоставляют пользователю, работающему с Unix-системой на персональном компьютере, особенный PC-интерфейс. Для некоторых людей это является положительным моментом. Но тем, кому не нравится, что экран захламлен диалоговыми окнами, и кого не смущает необходимость редактирования файлов настройки, оконный менеджер **fvwm2** может сослужить неплохую службу.

fvwm2 – это второе поколение оконного менеджера под названием **fvwm**. При этом не очень ясно, что означают буквы *fv* в названии. Разумный вариант для *v* – *virtual*, виртуальный. Менеджер **fvwm** предшествовал средам GNOME и KDE в качестве программы, предлагающей виртуальные рабочие столы как средство расширения рабочего пространства. Однако значение буквы *f* стало предметом догадок. При этом среди текущего состава команды разработчиков много любителей кошек, которые утверждают, что *f* – значит «кошачий» (*feline*).

Первое, что необходимо понять при освоении работы с **fvwm**, – это то, что рабочий стол может быть гораздо больше площади экрана пользователя. По сути дела, благодаря **fvwm2** пользователь может располагать целыми акрами пространства рабочего стола в форме виртуальных экранов или страниц.

В типичном окружении, конфигурируемом по умолчанию, имеется единственный рабочий стол, состоящий из четырех виртуальных экранов/страниц, упорядоченных в таблице размером два на два экрана.

Вы можете работать с приложениями на любой из экранных страниц и путешествовать по рабочему столу различными способами. Те, кого стандартная среда не устраивает, всегда могут произвольно изменить размер сетки виртуальных экранов. Может быть, три на два? Никаких проблем.

Если и этого мало, пользователю доступны многочисленные рабочие столы, каждый из которых содержит несколько страниц! Можно использовать разные рабочие столы для работы с различными приложениями или различными проектами, кому как нравится. **fvwm2** предоставляет средства перемещения в том пространстве, которое вы организуете для себя сами.

Менеджер **fvwm2** позволяет настроить и многие другие детали, наиболее важные из которых мы рассмотрим в этой главе. По существу, можно сказать одно: речь идет о максимуме рабочего пространства и предельной гибкости.

Запуск fvwm2

В большинстве дистрибутивов Linux поставляется достаточно свежая версия **fvwm2**. При работе с GNOME или KDE наиболее простой способ переключиться в **fvwm2** заключается в следующем:

1. Вызовите оконный менеджер в стартовом файле X-клиента. (В зависимости от системы файл может называться *.xinitrc*, *.startx*, *.xsession* или *.Xclients*.)
2. После модификации файла необходимо перезапустить систему X Window.

Вот фрагмент простого стартового файла, модифицированного для работы с **fvwm2**:

```
xterm -geometry +50+0 &
xterm -geometry -0+0&
fvwm2 &
xterm -title login -iconic
```

Хотя гипотетически возможно одновременно использовать GNOME или KDE и **fvwm2**, но при этом происходит широкое перекрытие предоставляемой функциональности. Получившийся уродливый гибрид не имеет к **fvwm** никакого отношения.

Файлы настройки

Ключ к гибкости **fvwm2** – в файлах настройки, читаемых при запуске или перезапуске менеджера. Общесистемному файлу настройки присвоено имя *system.fvwm2rc*, и он обычно находится в каталоге */etc/X11/fvwm2*.

Типичный файл *system.fvwm2rc*, который входит в дистрибутив, должен создавать простую, но готовую к работе среду. Этот вариант рассмотрен в следующем разделе. Нет никаких гарантий, что файл в вашей системе создает такое же окружение, но этого достаточно, чтобы уловить идею.

Для того чтобы подстроить систему под свои нужды, необходимо сделать копию файла *system.fvwm2rc*, дать ей имя *.fvwm2rc* и поместить в свой домашний каталог. Файл настройки пользователя имеет больший приоритет, чем общесистемный. Изменяя файл *.fvwm2rc*, можно настраивать **fvwm2** по необходимости.

Существует несколько простых правил редактирования пользовательского файла *.fvwm2rc*. Во-первых, каждая строка, начинающаяся с символа «#», является комментарием, т. е. игнорируется при чтении. Во-вторых, символ «+» в начале строки является указанием повторить первые операторы предшествующей строки. Эта возможность проиллюстрирована в разделе «Как сделать FvwmWinList частью стандартного окружения». И последнее, о чем стоить помнить: жизнь станет проще, если вы будете вносить собственные определения в файл настройки, руководствуясь его текущей структурой. Например, для того чтобы определить сочетания клавиш, добавьте строки в раздел файла, который посвящен именно этому.

Гибкость настройки **fvwm2** является одновременно и его достоинством, и недостатком. Достоинство заключается в том, что пользователь может изменять вид и поведение **fvwm2** в исключительно большом количестве аспектов. Это же качество плавно переходит в недостаток. У оконного менеджера десятки настроек, многие из которых просты и удобны, другие же, наоборот, сложны и даже загадочны. В результате работа с файлом настройки может смутить человека, непривычного к подобным вещам. Чрезмерное обилие возможностей иногда просто мешает работать.

На страницах руководства по **fvwm2** перечислены все параметры настройки и приведены примеры их использования. Кроме того, вам доступны страницы руководства по так называемым модулям **fvwm2**, речь о которых пойдет в следующем разделе. Веб-сайт *http://www.fvwm.org* является превосходным источником документации по оконному менеджеру **fvwm**, новостей, исходных текстов и обновлений, связанных с ним.

Эта глава поможет быстро освоить работу с **fvwm2**, выполнить основную настройку полезных особенностей, а также сделать сложные, но удобные добавления к **fvwm2**.

Модульный подход

Менеджер **fvwm2** разрабатывался таким образом, чтобы заинтересованный программист (или будущий программист) имел возможность создавать новые компоненты, так называемые *модули*. Типичным модулем является пейджер (FvwmPager), который отображает карту пространства рабочего стола и предоставляет способ перемещения по ней (речь об этом пойдет чуть

ниже). Пейджер является стандартным модулем практически во всех вариантах настройки.

Другим полезным модулем является FvwmWinList. Не будучи таким «вездесущим», как пейджер, он является не менее необходимым. FvwmWinList представляет собой небольшое окно, в котором отображается список всех окон на всех страницах всех рабочих столов. WinList – это другое средство для навигации, которое позволяет передавать фокус любому запущенному приложению и переключать страницу, чтобы пользователь мог воспользоваться приложением. Подробности – чуть позже.

Модуль, по существу, является независимой от **fvwm2** программой, но она взаимодействует с менеджером, передавая ему команды, которые должны быть выполнены. Во многих настройках **fvwm2** существует корневое меню (Root menu), в котором есть подменю FvwmModules, позволяющее запускать нужные программы. (Разумеется, список модулей в этом меню можно изменить.) Кроме того, путем редактирования файла *.fvwm2rc* можно определить иные способы запуска модулей (по нажатию определенных клавиш, по определенным событиям и т. д.).

Поскольку модуль является отдельной программой, пользователи могут создавать свои собственные модули, не изменяя при этом **fvwm2** необратимым образом. Следует, однако, иметь в виду, что **fvwm2** необходимо настроить, чтобы получить возможность вызова модуля; модуль не может быть запущен из командной строки. Заметьте, что в то время как некоторые модули, такие как пейджер, используются на протяжении всего сеанса работы, другие модули могут просто выполнять необходимые действия и завершаться (например, RefreshWindow). Поскольку модули являются самостоятельными программами, у многих из них есть собственные страницы руководства (man).

Как вводить в действие новые настройки оконного менеджера

Если файл *.fvwm2rc* был изменен, просто перезапустите **fvwm2**, чтобы изменения вступили в силу. В большинстве настроенных окружений есть специальный пункт меню, который позволяет перезапустить оконный менеджер. Готовая настройка, с которой мы начали, содержит пункт Exit fvwm2 в корневом меню. Если выбрать этот пункт, откроется подменю Really quit fvwm? (Действительно завершить работу?), содержащее несколько пунктов, включая Restart fvwm2 (Перезапустить fvwm2). Если выбрать этот пункт, изменения в настройках вступят в силу. Более медленный, но столь же эффективный способ ввода настроек в действие состоит в том, чтобы завершить сеанс работы в системе X Window и запустить ее снова (подразумевается, что стартовый файл сеанса включает запуск **fvwm2**).

Краткий обзор среды fvwm

В любом окружении рабочего стола, предоставляющем возможность иметь несколько виртуальных экранов/страниц, работать можно лишь с одной страницей единовременно. Но **fvwm2** облегчает переключение между страницами, позволяет запускать приложения на различных страницах и перемещать окна приложений с одной страницы на другую. Если требуется часто обращаться к одному и тому же окну, можно сделать его доступным одновременно на всех рабочих столах. (Позже мы еще вернемся к понятию «липких» окон.) При этом пользователь не ограничен одной страницей, да и окно даже может выходить за пределы экрана.

Обратите внимание на длинную горизонтальную панель в нижнем правом углу нашего тестового окружения (рис. 17.1). Эта панель представляет собой модуль **FvwmButtons** (также называемый *панелью кнопок*). Модуль FvwmButtons обычно применяется для группировки инструментов и кнопок вызова приложений, которые часто используются в работе. Зачастую это другие модули **fvwm**.

В нашем примере настройки модуль FvwmButtons содержит два других модуля: менеджер пиктограмм (FvwmIconMan) в левой части панели и пейджер

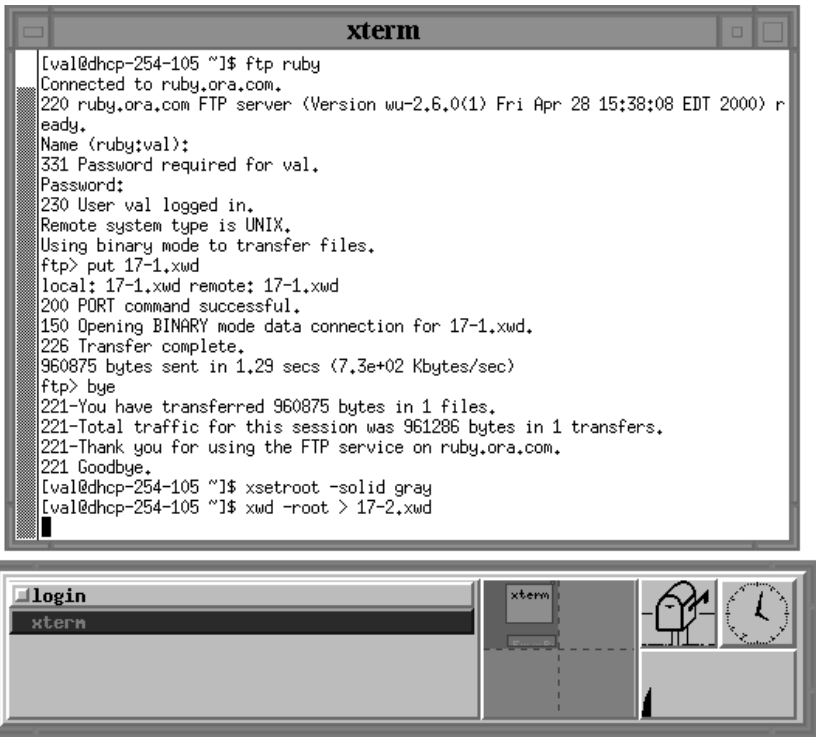


Рис. 17.1. Типичное окружение fvwm2

(FvwmPager) – в правой. С правого края FvwmButtons расположены три маленьких окна приложений: **xbiff** (почтовый ящик, по которому можно определить, что пришли новые почтовые сообщения), **xclock** и **xload** (график загрузки системы).

Менеджер пиктограмм и пейджер – это инструменты, позволяющие наблюдать за происходящими в среде событиями, а также управлять окнами выполняемых приложений. Менеджер пиктограмм отображает строку для каждого из обычных окон, доступных на рабочем столе. Если окно свернуто, запись для него предваряется квадратиком, имеющим трехмерный вид. Сворачивать и восстанавливать окна в текущем окне можно щелчком по кнопке соответствующих записей в менеджере пиктограмм. (Менеджер пиктограмм отображает только окна на текущей странице; инструментом, позволяющим просматривать список приложений для всех рабочих столов, является модуль FvwmWinList, описанный далее в этой главе.)

Модуль «пейджер» можно считать крохотным зеркалом, в котором отражается весь рабочий стол (или все столы). В типичном окружении, состоящем из одного рабочего стола (2×2 экранных страницы), пейджер отображает сетку из четырех разделов, расчерченную пунктирными линиями. Разделы соответствуют четырем виртуальным страницам экрана. (Если рабочих столов несколько, для их разделения используется сплошная линия. О том, как создать несколько рабочих столов, рассказано в разделе «Работа с несколькими рабочими столами» далее в этой главе.)

Каждое выполняемое приложение появляется в виде миниатюрной копии в окне пейджера. Маленькие окна приложений достаточно сложно опознать, когда они настолько малы, но, приглядевшись, их все-таки можно заметить в окне пейджера. Уменьшенная версия большого окна (как у *xterm*) должна иметь читаемый заголовок.

Все операции, выполняемые над окнами рабочего стола, – перемещение, сворачивание, изменение размера и т. д. – отражаются в окне пейджера. Но это больше, чем монитор активности, это еще и инструмент. Уменьшенные копии окон можно перемещать, в результате чего перемещаются соответствующие им окна. Пейджер позволяет перемещать окна с одной страницы на другую и выбирать область, отображаемую на мониторе (которая может не совпадать точно со страницей).

Помимо возможности управления рабочим столом в **fvwm2** часто настраивается система меню, которая начинается с корневого меню. Нажатие левой кнопки мыши в основном окне менеджера должно приводить к появлению этого меню. Корневое меню является удобным способом вызова нового окна с эмулятором терминала. Если вы начали работу с окружением, по умолчанию существующим в системе, то, вероятно, в корневом меню содержатся такие подменю, как Fvwm Modules, Fvwm Window Ops (в котором доступны пункты, позволяющие перемещать окна, изменять их размер и закрывать их), Fvwm Simple Config Ops (для изменения настроек передачи фокуса, работы механизма страниц и т. д.) и Exit Fvwm (для перезапуска менеджера или завершения работы с ним, запуска другого менеджера и т. д.).

Эта глава построена с расчетом на то, что читатель владеет основными навыками работы с оконным менеджером. Мы не будем учить вас пользоваться пейджером или пунктами меню, но покажем, как изменять количество отображаемых рабочих столов, как добавлять элементы в меню, настраивать сочетания клавиш и выполнять некоторые полезные настройки.

Настройка передачи фокуса по щелчку

Большинство версий **fvwm2** настроены на использование фокусной модели указателя мыши (FocusFollowsMouse или MouseFocus в файле настройки). Это означает, что необходимо переместить указатель в пределы окна, чтобы сделать возможным ввод данных в этом окне, использование меню и т. д. Однако в этом отношении **fvwm2** более гибок, чем прочие оконные менеджеры.

Существует два других правила передачи фокуса: `ClickToFocus`, требующее щелкнуть указателем мыши внутри окна, и очень удобное правило `SloppyFocus`, которое работает следующим образом: последнее окно теряет фокус не при выводе указателя за его пределы, а лишь когда указатель окажется в области другого окна, которое в этот момент и получает фокус. В частности, это может быть удобно при работе с окнами эмулятора терминала (**xterm** и **rxvt**). Можно переместить указатель за пределы окна (намеренно или случайно) и при этом осуществлять нормальный ввод.

Что больше всего радует в способах, предоставляемых **fvwm2** для работы с фокусом ввода, так это возможность смешивать их и определять правила передачи фокуса для различных типов окон. Все настройки, относящиеся к фокусу, являются аргументами переменной `Style`. (Эта переменная принимает несколько аргументов, определяющих вид и поведение определенного окна или компонента оконного менеджера.)

В следующем фрагменте файла настройки первая строка определяет фокусировку по умолчанию (по указателю мыши) для всех приложений (символ «*» является маской). Последующие строки определяют исключения из этого правила. Панель кнопок лучше работает с правилом `ClickToFocus`, как и **xman** (средство просмотра страниц руководства), и **xmag** (инструмент увеличения). Два эмулятора терминала выигрывают при использовании «запаздывающего» фокуса.

```
Style "*"           FocusFollowsMouse
Style "FvwmButtons" Icon toolbox.xpm, ClickToFocus
Style "xman"        Icon xman.xpm, RandomPlacement, ClickToFocus
Style "xmag"        Icon mag_glass.xpm, RandomPlacement, ClickToFocus
Style "XTerm"       Icon xterm.xpm, SloppyFocus, IconBox -70 1 -1 -140
Style "rxvt"        Icon term.xpm, SloppyFocus, IconBox -70 1 -1 -140
```

(Более подробно параметры этой мощной настройки описаны на страницах руководства по **fvwm2**. См. Style.)

В нашей тестовой настройке подменю Simple Config Ops корневого меню содержит три элемента, которые позволяют изменять правила фокусировки «на ходу» только для текущего сеанса работы:

- Sloppy Focus
- Click to Focus
- Focus Follows Mouse

Обратите внимание, что эти пункты имеют более высокий приоритет, чем созданные в файле настройки *.fvwm2rc*. Для того чтобы восстановить правила из файла настройки, необходимо перезапустить **fvwm2**.

Автоматический перевод окна с фокусом на передний план

Если используется фокусировка по указателю мыши (FocusFollowsMouse), можно подумать о применении модуля FvwmAuto для автоматического перевода на передний план окна, получившего фокус. Если добавить следующую строчку в файл *.fvwm2rc*, окно будет автоматически попадать на передний план, после того как указатель мыши пробудет в нем 200 миллисекунд:

```
Module FvwmAuto 200
```

Задержка на 1/5 секунды важна и делает работу FvwmAuto более эффективной. Если активно переключение фокуса по указателю, автоматический перевод окон на передний план может создать на экране хаос. Указатель, перемещаемый по экрану, проходит через несколько окон, которые поочередно переводятся на передний план, словно тасуемая колода карт. Задержка позволяет предотвратить проявление этого эффекта при достаточно быстром перемещении указателя в нужное окно.

Если по умолчанию действует правило `ClickToFocus`, то автоматический перевод окон на передний план включен, и вам не нужно вносить какие-либо изменения в файл настройки.

Разумеется, пользователи, работающие с модулем FvwmWinList, избавлены от необходимости редактировать файл *.fvwm2rc* и даже убирать указатель с модуля WinList, поскольку при передаче фокуса через FvwmWinList окно автоматически оказывается на переднем плане.

Одно из ваших меню можно так настроить, чтобы оно содержало элементы, позволяющие «на ходу» включать и выключать автоматический перевод окон на передний план. В некоторых стандартных настройках для этой цели предусмотрены пункты AutoRaise и Stop AutoRaise подменю Fvwm Modules.

Изменение размера рабочего стола

Во многих стандартных настройках рабочий стол имеет две страницы в ширину и две в высоту, что в файле настройки определяется следующим образом:

```
DeskTopSize 2x2
```

Размер рабочего стола легко изменить, отредактировав значения размерности для сетки страниц. Так, следующая строка определяет рабочий стол, который имеет три страницы в ширину и две в высоту:

```
DeskTopSize 3x2
```

Совершенно необязательно, чтобы и по горизонтали и по вертикали было несколько страниц. Можно определить рабочий стол, в котором страницы будут помещены «в столбик»:

```
DeskTopSize 1x3
```

Да и вообще, иметь несколько страниц совершенно необязательно:

```
DeskTopSize 1x1
```

Впрочем, зачем тогда использовать **fvwm2**?

Конечно же, количество определяемых страниц зависит от потребности в пространстве рабочего стола, от стиля работы и от того, сколько будет рабочих столов (см. следующий раздел). Если рабочих столов несколько, каждый из них определяется переменной `DeskTopSize`. Поэтому если вам нужен один рабочий стол для работы и один – для игр, то при определении количества страниц лучше проявить умеренность. Два рабочих стола, размером три на три каждый, обеспечивают достаточное пространство (18 страниц), в котором вполне можно заблудиться. Однако людям, работающим с графикой, может понадобиться и больше.

Работа с несколькими рабочими столами

Для работы с несколькими рабочими столами необходимо просто настроить пейджер на отображение необходимого их количества. Все столы будут иметь одинаковое количество страниц, определяемое переменной `DeskTopSize` (см. предыдущий раздел).

Чтобы использовать несколько рабочих столов, следует отредактировать строку, которая выглядит примерно так:

```
*FvwmButtons(2x2 Frame 2 Swallow(UseOld) "FvwmPager" "Module FvwmPager 0 0")
```

Эта строка включает модуль Pager (пейджер) в состав модуля FvwmButtons (панель кнопок). Два числа в конце строки определяют диапазон видимых рабочих столов. Первый рабочий стол имеет порядковый номер 0; последний рабочий стол в данном случае тоже имеет номер 0, т. к. стол всего один.

Чтобы получить два рабочих стола, измените последнее число на 1:

```
*FvwmButtons(1x2 Frame 2 Swallow(UseOld) "FvwmPager" "Module FvwmPager 0 1")
```

Следующая строка создаст модуль Pager с четырьмя рабочими столами, последний из которых имеет номер 3:

```
*FvwmButtons(1x2 Frame 2 Swallow(UseOld) "FvwmPager" "Module FvwmPager 0 3")
```

Немногим пользователям требуется такой объем рабочего пространства. Но даже если вы добавите всего один рабочий стол, имейте в виду, что, возможно, придется изменить размеры пейджера и, соответственно, размеры панели кнопок, на которой он располагается, чтобы все ваши рабочие столы не занимали слишком много места. Также может потребоваться освободить место для размещения пейджера.

Существует несколько параметров, с которыми можно экспериментировать в целях освобождения места для пейджера, показывающего несколько рабочих столов:

- Размеры панели кнопок (модуль FvwmButtons)
- Количество колонок, на которое разделена панель кнопок
- Сколько из этих колонок занимает пейджер

Типичный модуль FvwmButtons занимает 520 точек в ширину и 100 точек в высоту:

```
*FvwmButtonsGeometry 520x100-1-1
```

Он также может быть настроен для отображения в две строки и пять колонок (размеры которых полностью зависят от геометрии FvwmButtons):

```
*FvwmButtons(Frame 2 Padding 2 2 Container(Rows 2 Columns 5 Frame 1 Padding 10 0))
```

В данной конкретной настройке пейджер занимает одну колонку в двух строках модуля FvwmButtons :

```
*FvwmButtons(1x2 Frame 2 Swallow(UseOld) "FvwmPager" "Module FvwmPager 0 1")
```

Менеджер пиктограмм занимает три колонки:

```
*FvwmButtons(3x2 Frame 2 Swallow "FvwmIconMan" "Module FvwmIconMan")
```

И оставшаяся колонка отводится под приложения рабочего стола (например, **xbiff**, **xclock**, **xload**), которые выполняются в контейнере (Container) в модуле FvwmButtons:

```
*FvwmButtons(1x2 Frame 0 Container(Rows 2 Columns 2 Frame 0))
*FvwmButtons(Frame 2 Swallow(UseOld,NoHints,Respawn) "xbiff" 'Exec exec xbiff -bg
bisque3')
*FvwmButtons(Frame 3 Swallow(UseOld,NoHints,Respawn) "xclock" 'Exec exec xclock -bg
bisque3 -fg black -hd black -hl black -padding 0 update 1')
*FvwmButtons(2x1 Frame 2 Swallow(UseOld,NoHints,Respawn) "xload" 'Exec exec xload -
bg
bisque3 -fg black -update 5 -nolabel')
```

Обратите внимание, что контейнер имеет дальнейшее разделение на две строки и две колонки, в которых могут располагаться приложения. Надеемся, это не запутает читателей (разбор файла настройки требует внимания).

Вернемся к нескольким рабочим столам. Если вам нужны два рабочих стола, прежде всего измените второе число в диапазоне на единицу:

```
*FvwmButtons(1x2 Frame 2 Swallow(UseOld) "FvwmPager" "Module FvwmPager 0 1")
```

Затем установите размер окна пейджера таким, чтобы он был достаточен для нормального отображения этих столов, увеличьте ширину панели кнопок. Вот дополнительная сотня к числу 520, с которого мы начали:

```
*FvwmButtonsGeometry 620x100-1-1
```

Теперь перераспределим пять доступных колонок так, чтобы менеджер пиктограмм занимал только две колонки (а не три, как было вначале), и отдадим освободившуюся колонку пейджеру. Раздел для приложений по-прежнем будет занимать одну колонку.

```
*FvwmButtons(2x2 Frame 2 Swallow "FvwmIconMan" "Module FvwmIconMan")
*FvwmButtons(2x2 Frame 2 Swallow(UseOld) "FvwmPager" "Module FvwmPager 0 0")
*FvwmButtons(1x2 Frame 0 Container(Rows 2 Columns 2 Frame 0))
```

На рис. 17.2 приведен новый вид панели кнопок с двумя рабочими столами. Это только пример отдельной настройки. Представьте себе, учитывая свои нужды и размеры дисплея, насколько сложным может стать процесс настройки. Но все изменения могут быть протестированы простым перезапуском оконного менеджера.

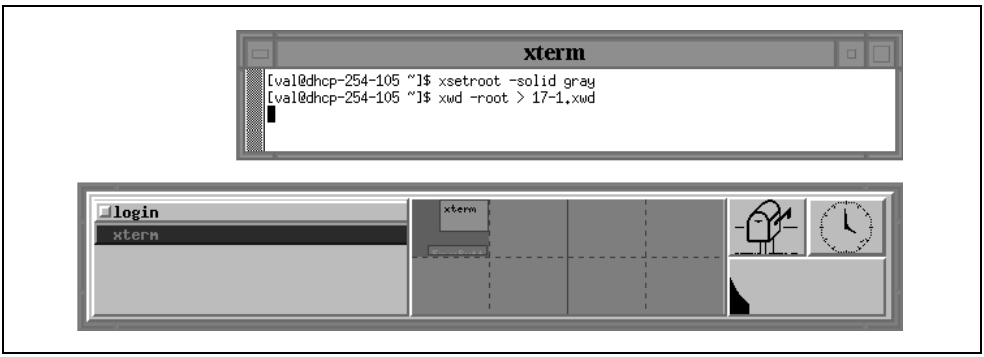


Рис. 17.2. Измененный модуль FvwmButtons содержит два рабочих стола в окне пейджера

Отображение окна на всех виртуальных страницах

Окно, отображаемое на всех виртуальных страницах, называется липким, потому что оно кажется прилипшим к стеклу. Некоторые окна обозначены как липкие в файле *system.fvwm2rc*, например: **xbiff**, программы, имена которых оканчиваются на *lock* (**xclock** и **oclock**), а также все модули **fvwm2** (поскольку пользователю необходимы панель кнопок, пейджер и т. д. на каждой странице).

Для того чтобы видеть окно на экране, независимо от того, какая это виртуальная страница, необходимо указать это в файле *~/.fvwm2rc*. Спецификация требует использования переменной Style, за которой следует имя клиента и параметр с легко запоминающимся именем Sticky (липкий). Переменная Style также служит для определения самых различных характеристик. Вот некоторые из строк, которые можно встретить в файле настройки. Строки, среди прочего, определяют, что окна являются липкими.

```
Style "xbiff"       NoTitle, Sticky, WindowListSkip, ClickToFocus
Style "*lock"       NoTitle, NoHandles, Sticky, WindowListSkip, ClickToFocus
Style "Fvwm*"       NoTitle,  Sticky, WindowListSkip
```

Обратите внимание, что переменная Style распознает маску (*), что позволяет расширить влияние определения. Fvwm* означает, что действие команды распространяется на все модули **fvwm2**.

Попробуйте добавить следующую строку, которая предписывает липкость приложению по имени **xpostit**:

```
Style "xpostit"      Sticky
```

Вы, вероятно, захотите определить для **xpostit** правило фокусировки ClickToFocus, а также скрыть его в списке окон FvwmWinList. Следующая строка определяет именно такие эффекты:

```
Style "xpostit"     Sticky, WindowListSkip, ClickToFocus
```

В большинстве случаев липкими делают только маленькие окна, запускаемые в единственном экземпляре (и часто используемые). Не очень практично иметь на каждой странице эмулятор терминала, подобный *xterm*; он занимает слишком много места. Однако если вам необходимо всегда иметь под рукой такую программу, как *xterm*, не забудьте дать этой копии программы характерное имя, используя параметр **–name**.

Например, в стартовом файле сеанса X Window можно запускать *xterm* с именем **mailwindow**:

```
xterm -name mailwindow &
```

Теперь сделайте так, чтобы это окно присутствовало на всех страницах, добавив следующую строку в файл *~/.fvwm2rc*:

```
Style "mailwindow"  Sticky
```

Для того чтобы временно сделать определенное окно липким, найдите меню Fvwm Window Ops в корневом меню. Обычно оно содержит переключатель (Un)Stick a Window. Впрочем, можно добавить эти пункты и самостоятельно. Как это сделать, рассказано в разделе «Настройка меню» далее в этой главе.

Открытие окон на разных рабочих столах и виртуальных страницах

Существует очевидный, примитивный способ запуска приложения на различных рабочих столах и различных страницах рабочего стола: нужно переключиться в нужную область (используя пейджер, сочетания клавиш или любой другой метод), открыть окно эмулятора терминала (используя корневое меню), а затем выполнить любую программу. В результате приложение запускается на нужной странице нужного рабочего стола.

Но есть и два автоматизированных способа добиться того же самого. Первый из них: нужно определить в файле *~/.fvwm2rc*, что указанные программы при запуске должны автоматически открываться на определенных рабочих столах/страницах. Делается это посредством переменной **fvwm2** `Style`, которая имеет два относящихся к теме параметра: `StartsOnPage` и `StartsOnDesk`.

Здесь начинается сложное. Параметр `StartsOnPage` может принимать до трех чисел в качестве аргументов. Если аргумент только один, он определяет номер рабочего стола, на котором должно быть открыто окно. Если аргументов три, второй и третий дополнительно определяют страницу с помощью координат X и Y. Чуть позже мы к этому вернемся (просто пара аргументов интерпретируется как координаты страницы X и Y).

А что же переменная `StartsOnDesk`, тесно связанная с предыдущей? Еще больше сложностей. `StartsOnDesk` принимает только один аргумент – номер рабочего стола. Но поскольку этот номер может быть установлен переменной `StartsOnPage` вместе с адресом конкретной страницы, на практике нет необходимости использовать `StartsOnDesk`. Единственное, для чего это может понадобиться, так это для того, чтобы определения в файле *.fvwm2rc* были как можно более понятными.

Вернемся к адресации рабочих столов и страниц. Рассмотрим схему задания адресов на рабочем столе, размер которого – две на две страницы. Так же, как первый рабочий стол имеет номер 0, первая страница рабочего стола определяется координатами (0,0). Следующая страница справа – кординатами (1,0); третья страница по часовой стрелке (нижний правый квадрант) – (1,1); четвертая страница по часовой стрелке (нижний левый квадрант) – (0,1).

Если предположить, что существует по меньшей мере два рабочих стола по четыре страницы каждый, следующее определение является указанием при запуске **xterm** под именем «bigxterm» открывать окно в нижнем левом квадранте (0,1) второго рабочего стола (с номером 1):

```
Style "bigxterm"     StartsOnPage 1 0 1
```

Изменив таким образом свой файл *.fvwm2rc* и перезапустив **fvwm2**, вы можете выполнить команду:

```
xterm -name bigxterm &
```

и убедиться, что окно открылось именно там, где ему было предписано.

Можно добиться такого же эффекта, используя синтаксис определения ресурсов X Window в командной строке. Эта стратегия, в принципе, может оказаться немного более практичной, чем размещение определений в файле *.fvwm2rc*, поскольку не придется беспокоиться об обработке нескольких копий различных программ (например, bigxterm, littlexterm, mailwindow или еще каких-то). Параметр **–xrm** (опознаваемый многими X-клиентами) позволяет задавать строку X-ресурсов следующим образом:

```
xterm -xrm '*Page: 1 0 1' &
```

Можно поместить набор таких строк в стартовый файл сеанса X Window, чтобы открывать приложения там, где они требуются, уже при входе в систему.

С виду оба этих метода открытия окон на различных рабочих столах или страницах (переменная `Style` и `StartsOnPage/StartsOnDesk` либо параметр командной строки **–xrm**) приводят к одинаковым результатам, но в действительности существует тонкое различие. При использовании переменной `Style` и указании только номера рабочего стола окно открывается на первой странице (0,0) этого стола. Если указать ту же информацию в командной строке, используя **–xrm**, конечная страница нового окна будет зависеть от того, на какой странице вы находитесь, выполняя эту команду. Новое окно появляется на той же странице окна, в которой выполнена команда.

Есть еще альтернатива, если существует необходимость открыть окно на другой странице текущего рабочего стола. Выполните приложение с параметром **–geometry**, задав достаточно большие значения координат, чтобы окно поместилось на нужной странице. Если уже создан рабочий стол размером три на три, следующая команда откроет окно на средней странице (девятистраничной сетки):

```
xterm -geometry +1200+1200 &
```

При этом следует иметь в виду, что специфичные характеристики дисплея играют большую роль в вычислении расстояний, и угадать их непросто.

Настройка скорости перемещения указателя между страницами

Если при перемещении по рабочему столу с помощью указателя мыши, вам кажется, что переход с одной страницы на другую происходит слишком быстро (или слишком медленно), обратитесь к удачно названной переменной `EdgeResistance` (Сопротивление края) в файле настройки, которая позволяет определять, насколько легко будет переместить указатель за границы текущей страницы.

Переменная принимает два аргумента. Первый относится непосредственно к указанной проблеме и определяет количество миллисекунд, по истечении которого происходит переход на соседнюю страницу. Второй параметр относится к способу перемещения окна между страницами: он определяет коли-

чество точек за краем экрана, после прохождения которого окно частично перейдет на соседнюю страницу. Типичные установки по умолчанию:

```
EdgeResistance 250 10
```

Некоторые жалуются, что значение EdgeResistance слишком мало и может вызывать случайный переход на соседнюю страницу, что, согласитесь, неудобно. В этом случае попробуйте увеличить значение первого аргумента:

```
EdgeResistance 500 10
```

Значение аргумента из диапазона 500–1000 определяет повышенное сопротивление. Предельным значением является число 10000, которое делает переход на соседнюю страницу в принципе невозможным.

Если, наоборот, указатель «отскакивает» от края, как от стенки, попробуйте уменьшить значение первого аргумента:

```
EdgeResistance 100 10
```

В типичной конфигурации **fvwm2** по умолчанию имеется ряд пунктов, позволяющих «на лету» изменять параметры переключения между страницами. Ряд таких пунктов находится в подменю Fvwm Simple Config Ops корневого меню. Таким образом, есть возможность включать и выключать переход между страницами при помощи пунктов Full Paging On и All Paging Off.

Пункт All Paging Off (Выключить переход между страницами) действует согласно названию, а именно: ограничивает указатель мыши текущей страницей. Это может быть удобно, если вы намереваетесь поработать какое-то время именно с текущей страницей и не хотите беспокоиться из-за возможного перехода на соседнюю. Чтобы снова включить переходы, воспользуйтесь пунктом Full Paging On.

Существуют другие пункты, различным образом ограничивающие функциональность перехода между страницами (например, Horizontal Paging Only (Только горизонтальные переходы), Vertical Paging Only (Только вертикальные переходы)).

Пункт Partial Paging позволяет перемещать указатель таким образом, чтобы в поле обзора попадали части двух соседних страниц; видимая зона будет подсвечена в окне пейджера.

Пункт Full Paging & Edge Wrap расширяет диапазон возможностей перехода между страницами. В обычных условиях указатель мыши не может переходить за край рабочего стола. При выборе этого пункта меню указатель может быть перенесен за край стола, в результате чего произойдет циклический переход и указатель появится на странице с другой стороны рабочего стола (по горизонтали или по вертикали). Таким образом, если указатель находится на верхней правой странице четырехстраничного стола и вы переносите указатель за правый край, произойдет циклический переход на верхнюю левую страницу этого стола.

В основе работы всех этих пунктов меню лежит переменная `EdgeScroll`. Вот параметры `EdgeScroll`, соответствующие различным пунктам меню:

```
Full Paging ON              EdgeScroll 100 100
All Paging OFF              EdgeScroll 0 0
Horizontal Paging Only      EdgeScroll 100 0
Vertical Paging Only        EdgeScroll 0 100
Partial Paging              EdgeScroll 50 50
Full Paging & Edge Wrap     EdgeScroll 100000 100000
```

Два аргумента `EdgeScroll` определяют в процентном отношении объем прокрутки страницы при достижении края экрана. Первый аргумент – для горизонтального перемещения, второй – для вертикального. Если горизонтальное или вертикальное значение умножается на число 1000, прокрутка вызывает циклический переход на противоположный край рабочего стола. Параметр `EdgeScroll 100000 100000` является указанием выполнять циклические переходы при прокрутке в любом направлении.

Если вы не полагаетесь на меню, добавьте соответствующую строку с определением `EdgeScroll` в свой файл *.fvwm2rc*. Подробнее читайте на страницах руководства по **fvwm2** и в файле *system.fvwm2rc*.

Обратите внимание, что ни один из вышеприведенных вариантов не позволяет переключаться между рабочими столами. В следующем разделе мы расскажем, как настроить сочетания клавиш для этой цели.

Создание сочетаний клавиш

Примитивный файл *system.fvwm2rc*, с которым мы начали работу, содержит незначительное количество определений сочетаний клавиш. Но если вы предпочитаете пользоваться клавиатурой, то можете в своем файле *.fvwm2rc* с легкостью определить сочетания клавиш, которые позволят выполнять самые разнообразные действия.

Сочетания клавиш для навигации по рабочему столу

Пейджер – это отличный инструмент для навигации по одному или нескольким рабочим столам. Однако многие люди испытывают ненависть к использованию мыши. Настройка сочетаний клавиш дает им удобный альтернативный способ навигации.

Добавьте следующие строки в свой файл *.fvwm2rc*, чтобы создать комбинации клавиш для перехода на следующую страницу в любом направлении в пределах одного рабочего стола, используя <Ctrl> и клавиши управления курсором. Окно обзора прокручивается в направлении, определяемом клавишей управления курсором.

В каждом определении используется переменная `Key` и ее аргументы:

1. Название клавиши.

2. Контекст (место), в котором она должна использоваться.
3. Все модификаторы (которые также должны находиться в нажатом состоянии).
4. Действие, выполняемое по клавише или сочетанию клавиш.

Так, в следующем примере первая строка с определением связывает нажатие клавиши <стрелка влево> в любом контексте «A» при нажатой клавише <Ctrl> («C») с прокруткой экрана на одну страницу влево на текущем рабочем столе:

```
# Press arrow + Control in any context
# to scroll by one page in the direction of the arrow
# (В любом контексте нажмите клавишу управления курсором и <Ctrl>,
# чтобы переместиться на одну страницу в направлении стрелки)
Key Left            A     C       Scroll -100 0
Key Right           A     C       Scroll +100 +0
Key Up              A     C       Scroll +0   -100
Key Down            A     C       Scroll +0   +100
```

В табл. 17.1 приведены выполняемые при этом действия.

Таблица 17.1. Сочетания клавиш для навигации по страницам

Сочетание	Переход
<Ctrl>+<стрелка вправо>	На страницу вправо
<Ctrl>+<стрелка влево>	На страницу влево
<Ctrl>+<стрелка вверх>	На страницу вверх
<Ctrl>+<стрелка вниз>	На страницу вниз

Переменная Scroll принимает те же аргументы, что и EdgeScroll, использование которых описано ранее в этой главе в разделе «Настройка скорости перемещения указателя между страницами». За более подробной информацией обращайтесь к страницам руководства **fvwm2**. Примите во внимание, что определенные нами сочетания клавиш позволяют перемещаться в пределах одного рабочего стола, но не позволяют перейти на другой рабочий стол. С этим вопросом мы разберемся чуть позже.

Вот еще одна допустимая комбинация. Она определяет последовательную смену страниц рабочего стола и возврат (в итоге) к первой странице. Здесь так же в произвольном контексте следует нажимать клавишу <Tab> совместно с <Ctrl>. Строка с определением выглядит следующим образом:

```
# Press Tab + Control in any context to scroll
# by one page with wrap scrolling
Key Tab         A       C       Scroll 100000 0
```

В табл. 17.2 приведены выполняемые при этом действия.

Таблица 17.2. Еще одно сочетание клавиш для перехода между страницами

Комбинация	Переход
<Ctrl>+<Tab>	На следующую страницу рабочего стола

Поскольку окна приложений могут частично находиться на соседних страницах, иногда необходимо, чтобы видимая на экране страница не точно совпадала со страницей рабочего стола (вам также может понадобиться одновременно видеть происходящее на двух страницах). Следующие сочетания клавиш позволяют прокручивать страницу на десятую часть ее размера. Вместо клавиши <Ctrl> в этих сочетаниях используется клавиша <Meta>. Это символическое имя реальной клавиши, выполняющей функцию Meta, варьируется от клавиатуры к клавиатуре. Во многих случаях эту функцию выполняет клавиша <Alt>. Вот строки определений для файла настройки:

```
## Press arrow + meta key in any context
## to scroll by 1/10 of a page in the direction of arrow
Key Left            A    M    Scroll -10 +0
Key Right           A    M    Scroll +10 +0
Key Up              A    M    Scroll +0   -10
Key Down            A    M    Scroll +0   +10
```

В табл. 17.3 приведены выполняемые при этом действия.

Таблица 17.3. Комбинации клавиш для прокрутки 1/10 части страницы

Комбинация	Переход
<Meta>+<стрелка влево>	На десятую часть страницы влево
<Meta>+<стрелка вправо>	На десятую часть страницы вправо
<Meta>+<стрелка вверх>	На десятую часть страницы вверх
<Meta>+<стрелка вниз>	На десятую часть страницы вниз

Если рабочих столов больше одного, можно определить сочетания клавиш для переключения между столами. Следующие два сочетания позволяют переключаться между столами в окружении с двумя столами:

```
## Press Control + Return in any context
## to scroll forward by 1 desktop
Key Return         A       C       Desk 1 1 1
## Press Shift + Control + Return in any context
## to scroll back by 1 desktop
Key Return         A       SC      Desk -1 0 0
```

В табл. 17.4 приведены выполняемые при этом действия.

Таблица 17.4. Сочетания клавиш для перехода между рабочими столами

Комбинация	Переход
<Ctrl>+<Enter>	На следующий рабочий стол
<Shift>+<Ctrl>+<Enter>	На предыдущий рабочий стол

Второй и третий аргументы переменной Desk ограничивают переходы таким образом, чтобы запретить переход за пределы рабочих столов. (Гипотетически можно перейти за пределы видимости пейджера!) Если рабочих столов не два, а больше, эти определения придется изменить. Подробную информацию о переменной Desk можно найти на страницах руководства по **fvwm2**.

Перемещение указателя с помощью клавиш

В предыдущем разделе мы рассмотрели сочетания клавиш, которые могут быть определены для прокрутки рабочего стола. Но также можно определить комбинации для перемещения указателя по рабочему столу. Конечно, такой способ годится только для людей, ненавидящих пользоваться мышью. Но даже если вы просто предпочитаете пользоваться клавиатурой, а не мышью, эти сочетания могут быть полезными. Они определяются при помощи переменной CursorMove, описанной также на страницах руководства по **fvwm2**.

Горячие клавиши первой группы перемещают указатель на одну десятую часть экрана за один раз. Первая строка связывает нажатие левой курсорной клавиши в произвольном контексте «A» при нажатых клавишах <Shift> («S») и <Meta> («M») с перемещением курсора на одну десятую часть размера экрана в направлении стрелки на клавише:

```
## Press Shift + Meta + arrow in any context
## to move the pointer by 1/10 of a page in direction of arrow
Key Left            A    SM    CursorMove -10  +0
Key Right           A    SM    CursorMove +10  +0
Key Up              A    SM    CursorMove +0   -10
Key Down            A    SM    CursorMove +0   +10
```

В табл. 17.5 приведены выполняемые при этом действия.

Таблица 17.5. Сочетания клавиш для перемещения указателя на 1/10 часть страницы

Комбинация	Перемещение указателя
<Meta>+< Shift>+<стрелка влево>	На одну десятую часть страницы влево
<Meta>+< Shift>+<стрелка вправо>	На одну десятую часть страницы вправо
<Meta>+< Shift>+<стрелка вверх>	На одну десятую часть страницы вверх
<Meta>+< Shift>+<стрелка вниз>	На одну десятую часть страницы вниз

Для того чтобы получить столь же тонкий контроль над указателем, как при работе с мышью, нужно определить сочетания, перемещающие указатель на один процент размера страницы за одно нажатие:

```
## Press Shift + Control + arrow in any context
## to move the pointer by 1% of a page in direction of arrow
Key Left            A    SC    CursorMove -1 0
Key Right           A    SC    CursorMove +1 +0
Key Up              A    SC    CursorMove +0   -1
Key Down            A    SC    CursorMove +0   +1
```

В табл. 17.6 приведены выполняемые при этом действия.

Таблица 17.6. Сочетания клавиш для перемещения указателя на один процент размера страницы

Комбинация	Перемещение указателя
< Shift>+<Ctrl>+<стрелка влево>	На одну сотую часть страницы влево
< Shift>+<Ctrl>+<стрелка вправо>	На одну сотую часть страницы вправо
< Shift>+<Ctrl>+<стрелка вверх>	На одну сотую часть страницы вверх
< Shift>+<Ctrl>+<стрелка вниз>	На одну сотую часть страницы вниз

Сочетания клавиш для меню и управления окнами

До сих пор мы ограничивали назначение сочетаний клавиш навигацией по рабочему пространству и перемещением указателя. Но можно определять сочетания клавиш, связанные с любой функцией оконного менеджера.

Вот некоторые примеры сочетаний, предназначенных для выполнения простых операций над окнами и отображения меню:

```
# Keyboard accelerators
Key F1          A       M       Iconify
Key F2          A       M       Move
Key F3          A       M       Resize
Key F4          A       M       Popup "RootMenu"
Key F5          A       M       Popup "Misc-Ops"
Key F6          A       M       Popup "Utilities"
Key F7          A       M       Popup "Module-Popup"
Key F10         A       M       Restart fvwm2
Key F12         A       SM      Close
```

Это просто примеры; пользователь может создать собственные комбинации и заставить их делать, что угодно. Но эти сочетания позволят нам рассмотреть некоторые из доступных возможностей, а также потенциальные проблемы.

Первая из строк этих определений связывает нажатие функциональной клавиши <F1> при нажатой клавише <Meta> («M») и указателе в произвольном

контексте «A» со сворачиванием или восстановлением окна, владеющего фокусом ввода. <Meta>+<F2> позволяет начать перемещение окна, владеющего фокусом, а <Meta>+<F3> – изменение размера окна (клавиша <Meta> описана в разделе «Сочетания клавиш для навигации по рабочему столу» ранее в этой главе).

Обратите внимание, что если вы приняли сочетания для перемещения указателя (описанные в предыдущем разделе), то можете выполнять операции перемещения окна и изменения его размеров, пользуясь только клавиатурой. Инициируйте операцию перемещения, нажав клавиши <Meta>+<F2>, а затем перетащите контур окна, перемещая изображение указателя с помощью соответствующих сочетаний клавиш. Нажмите клавишу <Enter>, чтобы завершить операцию.

И, наконец, мы настроили функциональные клавиши, связав их с открытием четырех различных меню, содержимое которых также определено в файле *.fvwm2rc*. Когда меню появляется, можно использовать клавиши <стрелка вверх> и <стрелка вниз> для выбора элемента из меню, клавиши <стрелка вправо> и <стрелка влево> – для входа в подменю и выхода из них (каскадные меню), а клавишу <Enter> – для выбора пункта. Клавиша <Esc> закрывает меню без выполнения какого-либо пункта.

Поскольку мы постоянно занимаемся настройкой **fvwm2**, то связали сочетание <Meta>+<F10> с перезапуском оконного менеджера. Это гораздо быстрее, чем работать с меню.

Также мы создали сочетание, закрывающее окно, которое владеет фокусом ввода: <Shift>+<Meta>+<F12>. Разумеется, очень удобно иметь возможность избавиться от окна с помощью клавиатуры, но нельзя допустить, чтобы это было слишком просто выполнить, т. к. это можно сделать и по ошибке. Необходимость нажатия дополнительной клавиши <Shift> и назначение самой последней функциональной клавиши <F12> требует осознанных действий для закрытия окна таким способом.

Настройка меню

Среди возможностей и функций, определяемых в файле настроек, есть и меню с их содержимым. Файл *system.fvwm2rc* обычно определяет набор меню, который может быть полезен большому количеству людей. Но по существу, набор пунктов меню зависит целиком от пользователя, который может остановить свой выбор и на стандартной конфигурации.

Типичное корневое меню определяется следующим образом:

```
AddToMenu RootMenu    "Root Menu"            Title
+                     "XTerm"                Exec exec xterm
+                     "Rxvt"                 Exec exec rxvt
+                      ""                     Nop
+                      "Remote Logins"        Popup Remote-Logins
+                      ""                     Nop
```

```
+                    "Utilities"              Popup Utilities
+                    ""                       Nop
+                    "Fvwm Modules"           Popup Module-Popup
+                    "Fvwm Window Ops"        Popup Window-Ops
+                    "Fvwm Simple Config Ops" Popup Misc-Ops
+                    ""                       Nop
+                    "Refresh Screen"         Refresh
+                    "Recapture Screen"       Recapture
+                    ""                       Nop
+                    "Exit Fvwm"              Popup Quit-Verify
```

Переменная AddToMenu предназначена для создания меню. Первый аргумент определяет имя меню, в данном случае – RootMenu. Имена меню используются для ссылки на меню в других точках файла настройки (например, при указании сочетания клавиш, открывающего данное меню). Обратите внимание, что переменная AddToMenu и имя меню повторяются в каждой строке, определяющей меню, на что и указывает символ «+».

Каждая строка определения создает строку меню; это может быть заголовок меню, элемент меню, пустая строка либо разделитель. Третья составляющая каждой строки определяет текст, который содержится в элементе меню. Четвертая составляющая определяет функцию оконного менеджера, связанную с элементом.

В первой строке нашего примера определяется заголовок меню. Строки с пустыми текстовыми полями (""") и привязанными к ним функциями Nop («No operation», операция отсутствует) используются для создания строк-разделителей в меню.

Имеет смысл чуть более пристально взглянуть на функцию Pop-up, которая определяет отображение меню, а в качестве аргумента получает его имя. Если Pop-up используется из другого меню, создаются подменю (или каскадные меню). В этом примере у корневого меню есть шесть подменю, которые называются Remote-Logins, Utilities, Module-Popup, Window-Ops, Misc-Ops и Quit-Verify. Эти меню также будут определены посредством указания команды AddToMenu где-либо в файле настройки.

При создании собственных меню или изменении существующих в качестве примера можно использовать меню из файла *system.fvwm2rc*, а в качестве справки – страницы руководства по **fvwm2**. Довольно просто заменить строки определений в шаблоне меню, и ненамного сложнее написать свое меню с нуля.

Также можно изменять способ отображения меню. Не всем, например, нравится дерево каскадных меню, растущее из корневого. В предыдущем разделе мы настроили некоторые функциональные клавиши, связав их с отображением определенных меню. Это одна из возможностей. Вместо этого можно использовать кнопки мыши для отображения различных меню. В типичной стандартной настройке первая кнопка мыши связана с корневым меню, а вторая – с меню Window Ops. Но поскольку бо́льшая часть функций меню Window Ops (например, Move, Resize, Iconify) доступна непосредственно в свойст-

вах окон, вместо этого можно связать вторую кнопку с отображением другого меню (например, Utilities).

FvwmWinList: переключение фокуса

FvwmWinList – это модуль **fvwm2**, который позволяет следить за всеми окнами приложений на многостраничных рабочих столах. Обычно для переключения фокуса настраивается WinList, но гипотетически он может использоваться для выполнения других операций.

Во многих типичных окружениях FvwmWinList можно запустить из меню Fvwm Modules (которое зачастую является подменю корневого). Если вместо этого требуется настроить **fvwm** на автоматический запуск WinList, обратитесь к разделу «Как сделать FvwmWinList частью стандартного окружения» данной главы. (Вы также можете определить сочетание клавиш для запуска модуля WinList; за подробностями обратитесь к разделу «Создание сочетаний клавиш».)

Внешний вид и поведение модуля FvwmWinList можно настроить. В следующем разделе мы рассмотрим типичные строки определения для этого модуля. Если выбрать эту настройку, FvwmWinList выполняет следующие операции:

Щелчок первой кнопкой мыши:

Переключить фокус на указанное окно. Если окно свернуто, восстановить его. Произвести переключение страниц таким образом, чтобы была отображена страница с окном.

Щелчок второй кнопкой мыши:

Свернуть/восстановить окно; отображаемая страница не изменяется.

Щелчок третьей кнопки мыши:

Отобразить всплывающую панель, содержащую информацию об окне (например, размеры в точках, является ли окно липким, основным или порожденным и т. д.) Панель закрывается по щелчку любой кнопкой мыши на ней.

Интересная особенность модуля WinList заключается в том, что ни одна из этих команд не перемещает указатель в окно, получающее фокус. Вместо этого указатель остается на строке WinList, соответствующей этому окну.

Гипотетически можно просто оставить указатель на модуле FvwmWinList и осуществлять всю навигацию с его помощью, кроме тех случаев, когда необходимо работать с модулем FvwmButtons или другим окном, которое отсутствует в списке WinList.

Использование FvwmWinList с несколькими экземплярами одного окна

Основное ограничение модуля FvwmWinList заключается в том, что достаточно трудно определить, где какое окно в списке. Каждая запись FvwmWinList состоит из текста, который появляется в заголовке соответствующего окна (если заголовок подавляется, то используется текст, который был бы отображен в заголовке). Если запущено несколько копий одной программы одновременно, например, несколько копий **xterm**, в списке FvwmWinList все они будут выглядеть одинаково. Отличаться будут только свернутые окна, текст для которых заключается в скобки.

Тем, кто будет много работать с FvwmWinList, вероятно, имеет смысл задавать разные заголовки для различных экземпляров одного приложения. Проделать это можно с помощью стандартных X-параметров **–title** и **–name**. Следует при этом иметь в виду, что **–title** изменяет текст только в заголовке, а **–name** изменяет непосредственно имя приложения, таким образом влияя на ресурсы и присвоение значений в файле настройки.

Как сделать FvwmWinList частью стандартного окружения

Если вы намереваетесь сделать модуль FvwmWinList частью своего стандартного окружения, отредактируйте файл настройки, чтобы выполнять его при инициализации и перезапуске оконного менеджера. В следующем примере для этих целей мы добавили третью и шестую строки:

```
AddToFunc InitFunction    "I" Module FvwmButtons
+           "I" exec xsetroot -mod 2 2 -fg \#554055 -bg \#705070
+           "I" Module FvwmWinList

AddToFunc RestartFunction "I" Module FvwmButtons
+           "I" exec xsetroot -mod 2 2 -fg \#554055 -bg \#705070
+           "I" Module FvwmWinList
```

Они предписывают загружать модуль FvwmWinList при запуске и перезапуске оконного менеджера. Окно появляется в нижнем левом углу экрана.

В качестве альтернативы можно сделать FvwmWinList всплывающим меню. Следующее определение связывает модуль с третьей кнопкой мыши, нажатой в зоне главного окна (это может оказаться не так удобно, как наличие постоянно запущенного модуля).

```
Mouse 3         R       A       Module FvwmWinList Transient
```

Отдельным вопросом является определение поведения и внешнего вида модуля. Как и для ряда других модулей (FvwmButtonBox, FvwmPager и т. д.), различные характеристики FvwmWinList определяются в других местах файла настройки. Ниже приводятся типичные строки:

```
#########################FvwmWinList#########################
*FvwmWinListBack #908090
*FvwmWinListFore Black
```

```
*FvwmWinListFont -adobe-helvetica-bold-r-*-*-10-*-*-*-*-*-*-*
*FvwmWinListAction Click1 Iconify -1,Focus
*FvwmWinListAction Click2 Iconify
*FvwmWinListAction Click3 Module "FvwmIdent" FvwmIdent
*FvwmWinListUseSkipList
*FvwmWinListGeometry +0-1
```

Первые три строки определяют цвет фона, цвет текста и используемый в приложении шрифт. Следующие три привязывают нажатия на различные кнопки мыши к определенным действиям внутри WinList. UseSkipList является предписанием для WinList не отображать строки для окон, которым присвоена стилевая классификация WindowListSkip. Как правило, таким образом скрываются все модули, в результате чего они не отображаются в списке FvwmWinList. Последняя строка определяет точку экрана, в которой должно появляться окно (левый нижний угол).

WinList является липким окном, т. е. оно доступно на каждом рабочем столе. Файл настройки может вводить в заблуждение. Данное свойство определяется в другом месте файла посредством переменной `Style`:

```
Style "Fvwm*"       NoTitle, Sticky, WindowListSkip
```

Эта строка определяет свойства липкости, отсутствия заголовка и отсутствия всех модулей (включая WinList) в списке окон FvwmWinList. В случае же с модулем FvwmWinList, появись он в собственном списке, это было бы более чем странно.

Алфавитный указатель

Специальные символы

- (дефис), метасимвол шаблона поиска, 636
! (восклицательный знак)
    команда (ex), 693
    метасимвол интерпретатора bash, 524
#
    команда (csh/tsch), 609
    команда интерпретатора bash, 545
#!
    команда (csh/tcsh), 609
    команда интерпретатора bash, 545
#, команда (sed)
    комментарии, 704
$ (знак доллара)
    метасимвол шаблона поиска, 636
$Author$, ключевое слово (RCS), 785
$COMSPEC, переменная окружения, 755
$CVS_CLIENT_LOG, переменная окружения, 755
$CVS_IGNORE_REMOTE_ROOT, переменная окружения, 755
$CVS_PASSFILE, переменная окружения, 755
$CVS_RCMD_PORT, переменная окружения, 755
$CVS_RSH, переменная окружения, 755
$CVS_SERVER, переменная окружения, 756
$CVS_SERVER_SLEEP, переменная окружения, 756
$CVSEDITOR, переменная окружения, 756
$CVSIGNORE, переменная окружения, 756
$CVSREAD, переменная окружения, 756
$CVSROOT, переменная окружения, 756
$CVSUMASK, переменная окружения, 756
$CVSWRAPPERS, переменная окружения, 756
$Date$, ключевое слово (RCS), 785
$EDITOR, переменная окружения, 756
$Header$, ключевое слово (RCS), 785
$HOME, переменная окружения, 756
$HOMEDRIVE, переменная окружения, 756
$HOMEPATH, переменная окружения, 756
Id, ключевое слово (RCS), 785
$Locker$, ключевое слово (RCS), 785
Log, ключевое слово (RCS), 786
$PATH, переменная окружения, 756
$RCSBIN, переменная окружения, 756
$RCSfile$, ключевое слово (RCS), 786
$Revision$, ключевое слово (RCS), 786
$Source$, ключевое слово (RCS), 786
$State$, ключевое слово (RCS), 786
$TEMP, переменная окружения, 756
&, команда (ex), 694
(квадратные скобки)
    метасимвол шаблона поиска, 636
    метасимволы интерпретатора bash, 524
* (звездочка)
    метасимвол интерпретатора, 523
+, метасимвол шаблона поиска, 637
. (точка)
    команда интерпретатора bash, 545
    метасимвол шаблона поиска, 636
.cvsignore, файл настройки, 757
.cvspass, файл, 757
.cvsrc, файл, 757
.cvswrappers, файл настройки, 757

.fvwm2rc, файл настройки, 859
автоматическое поднятие окна с фокусом, 864
клавиатурные сокращения, 872
липкие окна, 868
открытие приложений на разных рабочих столах, 869
переменная EdgeScroll, 872
.gnome-desktop, директория, 805
.kdelnk, файлы, 836
дисковый навигатор, 844
.mailrc, файл, 285
/dev/initrd, файл, 479
/etc/group, файл, 193
/etc/gshadow, файл, 194
/etc/lilo.conf, файл, 462, 477
/etc/passwd
изменение информации учетных записей, 433
создание новых записей, 431
/etc/passwd, файл, 515
изменение информации, 447
удаление записей, 432
удаление ошибочных записей, 339
/etc/rpmrc, файл, 483
/etc/shadow, файл, 339
:
команда (csh/tcsh), 609
команда (sed), 704
команда интерпретатора bash, 545
:set, команда (vi), 695
<>, команда (ex), 694
= (знак равенства)
команда (ex), 693
команда (sed), 704
присваивание в gawk, 720
? (вопросительный знак)
метасимвол интерпретатора bash, 523
метасимвол шаблона поиска, 637
\ (обратный слэш)
маскировка символов, 637
^ (диакритический символ)
метасимвол шаблона поиска, 636
|, метасимвол шаблона поиска, 637
~, команда (ex), 694
~/.exrc, файл, 664

A

a, команда (sed), 704
abbrev, команда (ex), 682
ad, add, команды (CVS), 766
addsuffix, переменная (csh/tcsh), 585
AddToMenu, переменная (fvwm2), 878
adm, admin, команды (CVS), 751
параметр -o, предупреждение об использовании, 752
admin, команды (CVS)
задание диапазонов, 752
agetty, команда, 49
alias, команда (bash), 545
alias, команда (csh/tcsh), 609
alloc, команда (csh/tcsh), 610
ann, annotate, команды (CVS), 767
apmd, команда, 50
apmd_proxy, команда, 50
append, команда (ex), 683
Applications, модули, 839
apt-cdrom, команда (Debian), 497
apt-get, команда (Debian), 494, 498
ar, команда, 52
arch, команда, 53
args, команда (ex), 683
argv, переменная (csh/tcsh), 586
arp, команда, 53
as, команда, 54
at, команда Linux, 55
atan2, команда (gawk), 721
atq, команда, 57
atrm, команда, 58
autocorrect, переменная (csh/tcsh), 586
autoexpand, переменная (csh/tcsh), 586
autolist, переменная (csh/tcsh), 586
autologout, переменная (csh/tcsh), 586
awk, программ, 638
awk, программа, 634, 635, 713

B

b, команда (sed), 705
backslash_quote, переменная (csh/tcsh), 586
backslash_quote, переменная (csh/tcsh), 586
Backspace, клавиша
в Emacs, 641
интерпретатор bash, 524
badblocks, команда, 58
banner, команда, 58
Base, директория (CVS), 759
basename, команда, 59
Baserev, файл (CVS), 759
bash, интерпретатор, 518, 519

встроенные команды, 544
bash, команда, 59
batch, команда, 59
bc, язык, 60
bdflush, команда, 429
bg, команда (bash), 546
bg, команда (csh/tcsh), 610
biff, команда, 66
BIND (Berkeley Internet Name Domain), 39
bind, команда (bash), 546
bindkey, команда (tcsh), 611
bison, команда, 66
bootpd, команда (TCP/IP), 67
bootpgw, команда Linux, 69
bootptest, команда (TCP/IP), 70
break, команда (bash), 547
break, команда (csh/tcsh), 612
break, команда (gawk), 721
breaksw, команда (csh/tcsh), 612
BSD (Berkeley Software Distribution), 18, 21
built-in, команда (bash), 547
built-ins, команда (tcsh), 612
bye, команда (tcsh), 612
bzip2, команда, 70
bzip2recover, команда, 70

C

c++, команда (см. g++, команда), 71
C++, язык программирования, 169
отладка, 179
c, команда (sed), 705
C, язык программирования
компилятор gcc, 170
отладка, 179
препроцессор, 94
cal, команда, 71
cardctl, команда, 72
cardmgr, команда, 73
case, команда (csh/tcsh), 612
cat, команда, 74
cd, команда (bash), 548
cd, команда (csh/tcsh), 612
cd, команда (ex), 683
cfdisk, команда, 75
change, команда (ex), 683
chattr, команда, 76
chdir, команда (csh/tcsh), 613
Checkin.prog, файл (CVS), 759
checkout, команда (CVS), 767
checkoutlist, файл (CVS), 737
chfn, команда (NFS/NIS), 77
chgrp, команда, 78
chmod, команда, 79
chown, команда, 81
chpasswd, команда, 82
chroot, команда, 83
chsh, команда, 83
ci, команда (RCS), 789
cksum, команда, 84
clear, команда, 84
ClickToFocus, правило (fvwm2), 863
close, команда (gawk), 721
cmp, команда, 84
co, команда (RCS), 792
col, команда, 85
colcrt, команда, 85
colrm, команда, 86
column, команда, 86
com, commit, команды (CVS), 768
comm, команда, 86
command, команда (bash), 548
commitinfo, файл (CVS), 738
comp.os.linux.admin, 11
comp.os.linux.announce, 11
comp.os.linux.development, 11
comp.os.linux.help, 11
comp.os.linux.misc, 11
comp.os.linux.networking, 11
complete, команда (tcsh), 613
complete, переменная (csh/tcsh), 586
compress, команда, 87, 198
config, файл (CVS), 738
continue, команда (bash), 548
continue, команда (csh/tcsh), 614
continue, команда (gawk), 721
Cookies, вкладка (Центр управления KDE), 842
copy, команда (ex), 683
CORBA и GNOME, 803
correct, переменная (csh/tcsh), 586
cos, команда (gawk), 722
cp, команда, 88
cpio, команда, 89
cpp, препроцессор, 94
интерфейс imake, 210
CRC (Cyclic Redundancy Check), 84
cron, команда, 99
crontab, команда, 100
csh, интерпретатор, 101, 518

csh, интерпретатор команд, 575
csplit, команда, 101
ctags, команда, 103
CursorMove, переменная (fvwm2), 875
cut, команда, 105
CVS, инструмент, 20, 729
  безопасность, 734
  ветвление, 731
  возможные проблемы, 733
  задание диапазонов, 752
  изменение директорий/файлов, 746
  импортирование из PVCS, 749
  импортирование из RCS, 748
  импортирование из SCCS, 749
  импортирование моментальных копий кода, 747
  импортирование проектов, 746
  инициализация, 753
  информация о настройках, 738
  команда release -d, предупреждение, 776
  команды администратора, 750
  логические модули, 743
  механизм блокировки, 730
  механизм слияния, 730
  общие параметры, 732
  основные понятия, 729
  параметры клиентов, 759
  переменные окружения, 755
  подстановка ключевых слов, 763
  пользовательские команды, 765
  посылка уведомлений об изменениях, 741
  путь к репозиторию, 744, 750, 754
  рабочая директория, 729
  разделяемая рабочая директория, 749
  репозиторий, 729
  сервер, 750
  создание репозитория, 734
  справочник администратора, 733
  справочник пользователя, 754
  структура рабочей директории, 758
  структура репозитория, 735
  файлы настройки клиента, 757
  формат команд, 732
  форматы дат, 762
cvsignore, административный файл, 739
CVSROOT, директория, 735
  административные файлы, 736
cvswrappers, административный файл, 740
cwd, переменная (csh/tcsh), 586
C-команды Emacs, 652
C-команды редактора Emacs
  команда C-s, 642
специальные символы
  расширения print в bc, 62

D

D, команда (sed), 705
d, команда (sed), 705
date, команда, 106
dbm-файлы, 294
dd, команда, 109
debugfs, команда, 111
declare, команда (bash), 548
default, команда (csh/tcsh), 614
Del, клавиша
  в Emacs, 641
  интерпретатор bash, 524
delete, команда (ex), 684
delete, команда (gawk), 722
depmod, команда, 114
Desk Guide, апплет, 805
Desk, переменная (fvwm2), 875
Desktop, модули (KDE), 844
Desktop, папка KDE, 836
df, команда, 115
diff, команда, 116
diff, команда (CVS), 768
diff3, команда, 120
dip, команда, 122
dirname, команда, 126
dirs, команда (bash), 549
dirs, команда (csh/tcsh), 614
dirstack, переменная (csh/tcsh), 586
disown, команда (bash), 549
dmesg, команда, 126
DNS (Domain Name System), 39
dnsdomainname, команда (TCP/IP), 126
do, команда (gawk), 722
domainname, команда (NFS/NIS), 126
dosfsck, команда, 126
dpkg, команда (Debian), 494, 502
dpkg-deb, команда (Debian), 494, 508
dpkg-split, команда, 510
DPMS (Display Power Management System), 846
dselect, команда (Debian), 494, 512
du, команда, 127

dumpe2fs, команда, 129
dumpkeys, команда, 129
dunique, переменная (csh/tcsh), 587

E

e2fsck, команда, 131
echo, команда, 131
echo, команда (bash), 549
echo, команда (csh/tcsh), 615
echo, переменная (csh/tcsh), 587
echo_style, переменная (csh/tcsh), 587
echotc, команда (csh/tcsh), 615
ed, редактор, 634, 635, 638
EdgeResistance, переменная (fvwm2), 870
EdgeScroll, переменная (fvwm2), 872
edit, команда (CVS), 769
edit, команда (ex), 684
editors, команда (CVS), 769
egrep, команда, 132, 639
    метасимволы, 634, 635
ellipsis, переменная (csh/tcsh), 587
Emacs, редактор, 20, 133
Emacs, режим (csh/tcsh), 604
Enlightenment, оконный менеджер
    виртуальные рабочие столы, настройка, 828
    выбор темы, 829
    клавиатурные сокращения, 829
    компоненты рабочего стола, перемещение, 828
    отображение всплывающих подсказок, 827
    поведение фокуса ввода, 826
    создание виртуальных рабочих столов, 826
    управление звуковыми событиями, 827
    фокус ввода и перемещение окон, 825
Entries, файл (CVS), 759
env, команда, 133
escape–последовательности, отображение, 85
etags, команда, 133
eval, команда (csh/tcsh), 616
ex, редактор, 135, 638, 681
exec, команда (bash), 551
exec, команда (csh/tcsh), 617
exit, команда (bash), 551
exit, команда (csh/tcsh), 617
exit, команда (gawk), 722
exp, команда (gawk), 722
expand, команда, 135
export, команда (bash), 552
export, команда (CVS), 769
expr, команда, 136
ext2, файловая система
    команда e2fsck, 131
    настройка параметров работы, 422
    создание на устройстве, 301
ext2,файловая система
    отладка, 111
exusage, команда (ex), 684

F

false, команда, 139
fc, команда (bash), 542, 552
fdformat, команда, 139
fdisk, команда, 139
fetchmail, команда, 140
fflush, команда (gawk), 722
fg, команда (bash), 553
fg, команда (csh/tcsh), 617
fgrep, команда, 144
fignore, переменная (csh/tcsh), 587
File Transfer Protocol (FTP), 160
file, команда, 144
file, команда (ex), 684
filec, переменная (csh), 587
filetest, команда (tcsh), 618
find, команда, 145
finger, команда, 77, 151, 447
fingerd, команда, 152
flex, команда, 152
fmt, команда, 154
FocusFollowsMouse, правило (fvwm2), 863
fold, команда, 155
for, команда (gawk), 722
for, циклы, 553
foreach, команда (csh/tcsh), 618
formail, команда, 155
free, команда, 158
fsck, команда, 158
fsck.ext2, команда, 126
fsck.minix, команда, 159
FSF (Free Software Foundation), 21
ftp, команда, 160
ftpd, команда (TCP/IP), 167

function, команда (gawk), 723
fuser, команда, 168
fvwm2, 857
ClickToFocus, 863
FocusFollowsMouse, 863
FvwmAuto, 864
FvwmButtons, 861, 865
FvwmIconMan, 861
FvwmWinList, 860, 879
Pager, 860, 865
SloppyFocus, 863
WinList, 879
автоматическое поднятие окна с фокусом, 864
ввод в действие новых настроек, 860
виртуальные рабочие столы, 858
виртуальные экраны, 861
изменение меню, 878
изменение размеров рабочего стола, 864
инструменты рабочего стола, 861
клавиатурные сокращения, 872
корневое меню, 862
липкие окна, 867
менеджер иконок, 861
модуль FvwmPager, 859
настройка, 877
открытие приложений на разных рабочих столах, 869
перезапуск, 860
переключение из GNOME или KDE, 858
переменная AddToMenu, 878
переменная CursorMove, 875
переменная Desk, 875
переменная EdgeResistance, 870
переменная EdgeScroll, 872
переменная Scroll, 873
перемещение указателя, 870
перемещение указателя с помощью клавиш, 875
правила передачи фокуса, 863
работа с набором рабочих столов, 865
сетка, 858, 862, 864
система меню, 862
файл .fvwm2rc, 859
файлы настройки, 858
fvwm2, оконный менеджер, 857
FvwmAuto, модуль, 864
FvwmButtons, модуль, 861
работа с комплектом рабочих столов, 865
FvwmIconMan, модуль, 861
FvwmPager, модуль, 859
FvwmWinList, модуль, 860, 879
как часть стандартного окружения, 880

G

g++, команда, 169
G, команда (sed), 706
g, команда (sed), 706
gated, демон, 38, 169
gawk, язык сценариев, 170, 713
gcc, компилятор, 170
gdb, отладчик, 179
gdc, команда, 182
General Public License (GPL), 21
gensub, команда (gawk), 723
getkeycodes, команда, 185
getline, команда (gawk), 723
getopts, команда (bash), 554
getty, команда, 185
ghostscript, команда, 195
gid, переменная (csh/tcsh), 587
glob, команда (csh/tcsh), 618
global, команда (ex), 684
GNOME, 803
Control Center, 811
автозагрузка, 818
апплеты, 805, 807
выбор темы, 813
главное меню, 809
закладка Animation, 814
закладка Buttons, 815
закладка Menu, 815
закладка Miscellaneous, 816
закладка Panel Objects, 815
и CORBA, 803
используемые оконные менеджеры, 804
настройка, 814
настройка клавиатуры, 818
настройка мыши, 818
настройка панелей, 808, 814
настройка пользовательского интерфейса, 818
настройки Multimedia, 817
настройки рабочего стола, 811, 812
настройки, параметры, 805
обработчики документов, 816

приложение настройки, 812
редактор по умолчанию, 817
создание дополнительных панелей, 807
создание иконок рабочего стола, 805
создание кнопок запуска приложений, 808
создание ссылок, 806
файлы справки, 817
Центр Управления, 811
экранные заставки, 813
GNOME-RPM, 492
gnorpm, команда, 493
GNU C, препроцессор, 94
GNU, проект, 18
goto, команда (csh/tcsh), 619
GPL (General Public License), 21
gprof, команда, 186
grep, команда, 187, 639
метасимволы, 634, 635
groff, команда, 190
group, переменная (csh/tcsh), 587
groupadd, команда, 193
groupdel, команда, 193
groupmod, команда, 193
groups, команда, 194
grpck, команда, 194
grpconv, команда, 195
grpunconv, команда, 195
gs, команда, 195
GSS-API, путь к репозиторию, 754
gsub, команда (gawk), 724
gunzip, команда, 197
gzexe, команда, 197
gzip, команда, 197

H

H, команда (sed), 707
h, команда (sed), 707
halt, команда, 199
hangup, 316
hash, команда (bash), 554
hashstat, команда (csh/tcsh), 619
head, команда, 200
Helix GNOME, среда, 820
help, команда (CVS), 770
help, команда (ex), 685
histdup, переменная (csh/tcsh), 587
histfile, переменная (csh/tcsh), 587
histlit, переменная (csh/tcsh), 587
history, команда (csh/tcsh), 619
history, команда (CVS), 770
типы записей, 772
history, переменная (csh/tcsh), 587
history, файл (CVS), 740
host, команда, 201
hostid, команда, 204
hostname, команда, 204
hup, команда (tcsh), 620
hwclock, команда, 205

I

i, команда (sed), 707
ibase, ключевое слово, 61
icmpinfo, команда (TCP/IP), 206
id, команда, 207
ident, команда (RCS), 794
identd, команда (TCP/IP), 207
if, команда (bash), 555
if, команда (csh/tcsh), 620
if, команда (gawk), 724
ifconfig, команда, 208
ifconfig, команда (Linux), 40
ignoreeof, переменная (csh/tcsh), 587
imake, интерфейс, 210
Imakefile, файл, 214
imapd, команда (TCP/IP), 214
implicitcd, переменная (csh/tcsh), 588
import, команда (CVS), 772
коды состояний, 773
index, команда (gawk), 724
inetd, демон, 215
inetd.conf, файл, 754
info, команда, 216
Information, модули (KDE), 849
init, команда, 217
init, команда (CVS), 753
initrd и RAM-диски, 479
Input Devices, модули (KDE), 849
insert, команда (ex), 685
insmod, команда, 218
install, команда, 218
int, команда (gawk), 724
ipchains, инструмент, 219
ipchains, инструмент, ветвь 2.2 ядра Linux, 225
ipchains–restore, команда, 224
ipchains–save, команда, 225
ipfwadm, команда, ветвь 2.0 ядра

Linux, 225
iptables, команда (ядра ветви 2.4), 41
iptables, команда, ветвь 2.4 ядра Linux, 225, 230
iptables-restore, команда, 241
iptables-save, команда, 43, 241
IP-адреса, 37
IP–адреса
ipfwadm, 225
IRC-сеть OpenProjects, 11
ispell, команда, 242, 243

J

jobs, команда (bash), 556
jobs, команда (csh/tcsh), 621
join, команда, 243
join, команда (ex), 685

K

k, команда (ex), 685
kbd_mode, команда, 244
kbdrate, команда, 245
KDE
kfm (файловый менеджер KDE), 832
kwm (оконный менеджер KDE), 837
виртуальные рабочие столы, 837, 844
всплывающие подсказки, отображение, 842
выбор цветовой схемы, 845
дисковый навигатор, 844
заголовки окон, 851
менеджер сеансов, 839
менеджер тем, 846
модули Desktop, 844
модули Information, 849
модули Input Devices, 849
модули Keyboard Shortcuts, 849
настройка клавиатуры, 849, 851
настройка мыши, 849
настройка свойств границ, 845
настройки, 840
настройки шрифтов, 845
окна приложений, 832
отображение, 846
панель, 837
панель задач, 838
папка Desktop, 836
правила передачи фокуса, 855
проводник рабочего стола, 837
работа с cookie-записями, 842
создание кнопок приложений на панели, 838
список окон, 838
ссылки, 832
установка, 831
функциональность web-браузера, 841
Центр управления, 839
экранные заставки, 848
KDE (K Desktop Environment), 283, 480, 830
Kerberos 4, путь к репозиторию, 754
kerneld, команда, 245
Keyboard Shortcuts, модули (KDE), 849
kill, команда, 246
kill, команда (bash), 556
kill, команда (csh/tcsh), 621
killall, команда, 246
killall5, команда, 247
klogd, команда, 247
kpanel, программа, 837
ksyms, команда, 249
kwm (оконный менеджер KDE), 837

L

l, команда (sed), 708
last, ключевое слово, 61
lastlog, команда, 249
ld, редактор связей, 249
ldconfig, команда, 252
ldd, команда, 253
length, команда (gawk), 724
less, команда, 253
let, команда (bash), 557
LILO (Linux Loader), 460
общие параметры, 463
параметры загрузочного образа, 468
параметры, передаваемые ядру, 469
файл настройки, 462
LILO, загрузчик, 19
lilo, команда, 462
параметры, 471
limit, команда (csh/tcsh), 622
Linux
двойная загрузка систем Windows, 475
двойная загрузка, системы Windows, 461
исходные тексты, 21
команды, 18

Linux Documentation Project, 10
Linux Magazine, 10
Linux User Groups (LUG), 12
list, команда (ex), 685
listflags, переменная (csh/tcsh), 588
listjobs, переменная (csh/tcsh), 588
listlinks, переменная (csh/tcsh), 588
listmax, переменная (csh/tcsh), 588
ln, команда, 261
Loadlin, 19
    загрузка из MS-DOS, 473
    параметры, 474
local, команда (bash), 557
locate, команда, 262
lockfile, команда, 263
log, команда (csh/tcsh), 623
log, команда (CVS), 774
    задание диапазонов дат, 774
log, команда (gawk), 724
logger, команда (TCP/IP), 264
login, команда, 264
login, команда (csh/tcsh), 623
login, команда (CVS), 775
loginfo, файл (CVS), 740
logname, команда, 265
logout, команда (bash), 557
logout, команда (csh/tcsh), 623
logout, команда (CVS), 775
logrotate, команда, 265
look, команда, 267
lpc, команда, 268
lpd, демон, 269
lpq, команда, 271
lpr, команда, 271, 272
lprm, команда, 273
lpstat, команда, 273
lptest, команда, 274
ls, команда, 274
lsattr, команда, 278
ls-F, команда (tcsh), 623
lsmod, команда, 279
LUG (Linux User Groups), 12

M

m4, процессор, 279
mail, 280
    почтовые псевдонимы, 334
    преобразование ввода в формат почтового ящика, 155
mail, переменная (csh/tcsh), 588
mailq, команда, 286
mailstats, команда, 287
make, инструмент, 287
    интерфейс imake, 210
makedbm, команда, 294
makedepend, инструмент (imake), 212
makedirhier, инструмент (imake), 213
makemap, команда, 295
man, команда, 296
manpath, команда, 298
map, команда (ex), 685
mark, команда (ex), 686
match, команда (gawk), 724
matchbeep, переменная (csh/tcsh), 588
MDI и GNOME, 820
merge, команда, 298
mesg, команда, 299
mimencode, команда, 300
mime-типы, настройка (GNOME), 817
MINIX
    файловая система, 159
MINIX, файловая система, 303
mkdir, команда, 300
mke2fs, команда, 301
mkexrc, команда (ex), 686
mkfs, команда, 302
mkfs.minix, команда, 303
mklost+found, команда, 303
mkraid, команда, 303
mkswap, команда, 303
modprobe, команда, 304
modules, файл (CVS), 741
more, команда, 304
mount, команда, 307
move, команда (ex), 686
Multimedia, настройки GNOME, 817
mv, команда, 310
M-команды Emacs, 655

N

N, команда (sed), 708
n, команда (sed), 708
named, демон, 311
namei, команда, 312
netdate, команда (TCP/IP), 313
netfilter (правила отбора), 230
netstat, команда (TCP/IP), 313
newgrp, команда, 314
newgrp, команда (csh/tcsh), 623
newusers, команда, 314

next, команда (ex), 686
next, команда (gawk), 725
nextfile, команда (gawk), 725
NFS (Network File System), 43
nice, команда, 315
nice, команда (csh/tcsh), 624
NIS
    команды администрирования, 31
NIS (Network Information Service)
    команды, 446
NIS (Network Information System), 45
nm, команда, 315
noclobber, переменная (csh/tcsh), 588
noglob, переменная (csh/tcsh), 588
nohup, команда, 316
nohup, команда (csh/tcsh), 624
nonomatch, переменная (csh/tcsh), 588
notify, команда (csh/tcsh), 624
notify, файл (CVS), 743
Notify, файлы (CVS), 759
nslookup, инструмент (TCP/IP), 316
number, команда (ex), 687

O

obase, ключевое слово, 61
onintr, команда (csh/tcsh), 624
Online-поддержка, 11
open, команда (ex), 687
OpenProjects, IRC-сеть, 11
ORBit (пакет) и GNOME, 803
owd, переменная (csh/tcsh), 589

P

P, команда (sed), 709
p, команда (sed), 709
Pager, модуль (fvwm2), 860
    работа с комплектом рабочих столов, 865
passwd, файл (CVS), 744
paste, команда, 320
patch, команда, 320
pathchk, команда, 323
PCMCIA, гнезда, 72
Perl
    сценарии, 496
    язык, 325
PID (идентификаторы процессов), 246, 325
pidof, команда, 325
ping, команда, 325
popd, команда (bash), 558
popd, команда (csh/tcsh), 625
POSIX, списки символов, 637
powerd, демон, 329
PPP (протокол узел-узел), 329
PPP, протокол точка-точка, 40
pppd, демон, 329
pr, команда, 331
praliases, команда, 334
preserve, команда (ex), 687
previous, команда (ex), 687
print, команда (ex), 687
print, команда (gawk), 725
printenv, команда, 334
printenv, команда (csh/tcsh), 625
printexitvalue, переменная (csh/tcsh), 589
printf, команда, 334
printf, команда (bash), 558
printf, команда (gawk), 725
PRIO_MAX и PRIO_MIN, переменные, 347
prompt2, переменная (csh/tcsh), 589
promptchars, переменная (csh/tcsh), 589
ps, команда, 334
pserver (CVS), 754
    безопасность, 734
    доступ к репозиторию, 744, 750
    настройка, 734
psupdate, команда, 339
pushd, команда (bash), 558
pushd, команда (csh/tcsh), 625
pushdsilent, переменная (csh/tcsh), 589
pushdtohome, переменная (csh/tcsh), 589
put, команда (ex), 687
PVCS, импортирование файлов, 749
pwck, команда, 339
pwconv, команда, 339
pwd, команда, 340
pwd, команда (bash), 559

Q

q, команда (sed), 709
quit, команда (ex), 688

R

r, команда (sed), 709
raidstart, команда, 340

raidstop, команда, 340
RAID-устройства
запуск и останов, 340
настройка, 303
ramsize, команда, 341
RAM-диски и initrd, 479
rand, команда (gawk), 726
ranlib, команда, 341
RARP (протокол обратного разрешения адресов), 341
rarp, команда, 341
rcp, команда, 342
RCS, инструмент, 20, 782
импортирование файлов в CVS, 748
механизм блокировки, 730
нумерация версий, 787
основные понятия, 729
подстановка ключевых слов, 785
создание поддиректории RCS, 783
rcs, команда (RCS), 795
rcsclean, команда (RCS), 798
rcsdiff, команда (RCS), 799
rcsinfo, файл (CVS), 745
RCSINIT, переменная окружения (RCS), 788
rcsmerge, команда (RCS), 800
rdate, команда (TCP/IP), 342
rdev, команда, 343
rdiff, команда (CVS), 776
rdist, команда, 343
rdistd, демон, 346
read, команда (bash), 559
read, команда (ex), 688
readers, файл (CVS), 745
readonly, команда (bash), 560
reboot, команда, 347
recexact, переменная (csh/tcsh), 589
recognize_only_executables, переменная (csh/tcsh), 589
recover, команда (ex), 688
Red Hat Package Manager (RPM), 482
GNOME-RPM, 492
Red Hat, менеджер пакетов (RMP), 357
rehash, команда (csh/tcsh), 626
release, команда (CVS), 776
remove, команда (CVS), 777
renice, команда, 347
repeat, команда (csh/tcsh), 626
Repository, файл (CVS), 759
reset, команда, 348
return, команда (bash), 561
return, команда (gawk), 726
rewind, команда (ex), 688
rexecd, демон, 348
rlog, команда (RCS), 801
rlogin, команда, 349
rlogind, демон, 350
rm, команда, 351
rmail, команда (TCP/IP), 351
rmdir, команда, 352
rmmod, команда, 352
Root, файл (CVS), 759
rootflags, команда, 352
route, команда (TCP/IP), 353
routed, демон, 38, 354
RPC (Remote Procedure Call), 47, 355
rpcgen, команда, 354
rpcinfo, команда (NFS/NIS), 355
rpm, команда, 357, 482
параметры FTP/HTTP, 490
параметры выбора информации, 487
параметры выбора пакетов, 486
параметры проверки подписи, 489
параметры создания пакетов, 491
параметры удаления пакетов, 488
параметры установки/обновления, 484
rprompt, переменная (csh/tcsh), 589
rsh, команда, 357
rshd, демон, 357
rstat, команда, 358
rt, rtag, команды (CVS), 777
runlevel, система, 358
run-parts, команда, 358
ruptime, команда, 358
rusers, команда, 359
rwall, команда, 359
rwho, команда, 359

S

s, команда (sed), 710
savehist, переменная (csh/tcsh), 590
Sawfish, оконный менеджер
всплывающие подсказки, отображение, 823
обновление окон, 823
перемещение и изменение размеров, 823
поведение фокуса ввода, 821
рабочее пространство, 824

размещение новых окон, 824
сворачивание и восстановление окон, 822
скрытые окна, 821
scale, ключевое слово, 61
SCCS, импортирование файлов, 749
sched, команда (csh/tcsh), 627
sched, переменная (csh/tcsh), 590
script, команда, 360
script, команда (ex), 688
Scroll, переменная (fvwm2), 873
sed, инструмент, 20
sed, редактор, 360, 638, 699
команды (конспект), 704
метасимволы, 634, 635
select, команда (bash), 561
sendmail
команда mailq, 286
команда mailstats, 287
команда makemap, 295
команда praliases, 334
sendmail, команда, 360
set, команда (bash), 561
set, команда (csh/tcsh), 627
set, команда (ex), 689
setenv, команда (csh/tcsh), 627
setfdprm, команда, 367
setsid, команда, 367
settc, команда (csh/tcsh), 628
setty, команда (tcsh), 628
sh, командный интерпретатор, 368
shar, команда, 368
shell, команда (ex), 689
shell, переменная (csh/tcsh), 590
shift, команда (bash), 565
shift, команда (csh/tcsh), 628
showmount, команда (NFS/NIS), 371
shutdown, команда, 372
sin, команда (gawk), 726
size, команда, 373
slattach, команда (TCP/IP), 373
sleep, команда, 374
SLIP (межсетевой протокол для последовательного канала), 40
SloppyFocus, правило (fvwm2), 863
sort, команда, 374
source, команда (bash), 565
source, команда (csh/tcsh), 629
source, команда (ex), 689
split, команда, 376
split, команда (gawk), 726
sprintf, команда (gawk), 727
sqrt, команда (gawk), 727
srand, команда (gawk), 727
StartsOnDesk, параметр (fvwm2), 869
StartsOnPage, параметр (fvwm2), 869
stat, команда, 377
status, команда (CVS), 778
status, переменная (csh/tcsh), 590
stop, команда (csh/tcsh), 629
stop, команда (ex), 689
strace, команда, 377
strfile, команда, 380
strftime, команда (gawk), 727
strings, команда, 381
strip, команда, 382
stty, команда, 382
Style, переменная (fvwm2)
липкие окна, 868
открытие приложений на разных страницах рабочего стола, 869
правила передачи фокуса, 863
su, команда, 389
sub, команда (gawk), 727
substitute, команда (ex), 689
substr, команда (gawk), 727
sum, команда, 390
suspend, команда (bash), 565
suspend, команда (csh/tcsh), 629
suspend, команда (ex), 690
swapdev, команда, 390
swapon, swapoff, команды, 391
switch, команда (csh/tcsh), 629
symlinks, переменная (csh/tcsh), 590
sync, команда, 391
sysklogd, демон, 391
syslogd, демон, 392
systat, команда, 393
system, команда (gawk), 728
system.fvwm2rc, файл настройки, 858
изменение меню, 877
липкие окна, 867
systime, команда (gawk), 728

T

t, команда (ex), 690
t, команда (sed), 710
ta, tag, команды (CVS), 778
tac, команда, 393
tag, команда (ex), 690
Tag, файл (CVS), 759
taginfo, файл (CVS), 745

tagnext, команда (ex), 691
tagpop, команда (ex), 691
tagprev, команда (ex), 691
tagtop, команда (ex), 691
tail, команда, 394
talk, команда, 395
talkd, демон, 395
tar, команда, 395
Tasklist, апплет, 805
TCP/IP, 35, 400
tcpd, демон, 400
tcpdchk, команда (TCP/IP), 400
tcpdmatch, команда (TCP/IP), 401
tcsh, интерпретатор, 518
tcsh, интерпретатор команд, 575
tcsh, командный интерпретатор, 401
tee, команда, 401
telinit, команда, 402
telltc, команда (csh/tcsh), 630
telnet, 403
telnetd, демон, 411
Template, файл (CVS), 759
test, команда
  Linux, 412
test, команда (bash), 565
TFTP (Trivial File Transfer Protocol), 414
tftpd, демон, 415
time, команда (csh/tcsh), 630
time, переменная (csh/tcsh), 590
times, команда (bash), 569
tload, команда, 415
tolower, команда (gawk), 728
top, команда, 416
touch, команда, 418
toupper, команда (gawk), 728
tperiod, переменная (csh/tcsh), 590
tr, команда, 419
traceroute, команда (TCP/IP), 421
trap, команда (bash), 569
Trivial File Transfer Protocol, 415
Trivial File Transfer Protocol (TFTP), 414
troff, команда, 190, 422
true, команда, 422
tune2fs, команда, 422
tunelp, команда, 424
type, команда (bash), 570
typeset, команда (bash), 571

U

ul, команда, 425
ulimit, команда (bash), 571
umask, команда (bash), 572
umask, команда (csh/tcsh), 630
umount, команда, 425
unabbreviate, команда (ex), 691
unalias, команда (bash), 572
unalias, команда (csh/tcsh), 630
uncomplete, команда (tcsh), 631
uncompress, команда, 426
undo, команда (ex), 691
unedit, команда (CVS), 779
unexpand, команда, 426
unhash, команда (csh/tcsh), 631
uniq, команда, 427
Unix, 16
unlimit, команда (csh/tcsh), 631
unmap, команда (ex), 691
unset, команда (bash), 572
unset, команда (csh/tcsh), 631
unsetenv, команда (csh/tcsh), 631
unshar, команда, 428
unstr, команда, 380
until, команда (bash), 573
update, команда, 429
update, команда (CVS), 779
  коды состояний, 781
Update.prog, файл (CVS), 759
uptime, команда, 430
URL-обработчики, настройка (GNOME), 817
URL-ссылки
  настройка цветов, 840
  создание на рабочем столе KDE, 833
  создание новых на рабочем столе GNOME, 806
User Agent, вкладка (Центр управления KDE), 841
useradd, команда, 430
userdel, команда, 432
usermod, команда, 433
users, команда, 434
users, файл (CVS), 745
usleep, команда, 434
uudecode, команда, 434
uuencode, команда, 434

V

v, команда (ex), 691
vacation, команда, 435
verbose, переменная (csh/tcsh), 590
verifymsg, файл (CVS), 745
version, команда (ex), 691
version, переменная (csh/tcsh), 590
vi, команда (ex), 692
vi, командный режим, 665
vi, команды строки состояния, 667
vi, редактор, 20, 436, 664
    метасимволы, 634, 635
vi, режим (csh/tcsh), 605
vi, режим вставки, 665
vidmode, команда, 436
visiblebell, переменная (csh/tcsh), 590
visual, команда (ex), 692
viusage, команда (ex), 692

W

w, команда, 437
w, команда (sed), 711
wait, команда (bash), 573
wait, команда (csh/tcsh), 631
wall, команда, 438
watch, команда (CVS)
    файл notify, 743
watch, команды (CVS), 781
watch, переменная (csh/tcsh), 591
watchers, команда (CVS), 782
watchlog, команда (csh/tcsh), 631
wc, команда, 438
web-браузера, функциональность (KDE), 841
whatis, команда, 439
where, команда (tcsh), 632
whereis, команда, 439
which, команда, 440
which, команда (csh/tcsh), 632
while, команда (bash), 573
while, команда (csh/tcsh), 632
while, команда (gawk), 728
who, команда, 441
whoami, команда, 442
Windows 95/98, двойная загрузка с Linux, 461
Windows NT/2000
    двойная загрузка с Linux, 475
WinList, модуль (fvwm2), 879
    как часть стандартного окружения, 881
wordchars, переменная (csh/tcsh), 591
wq, команда (ex), 692
write, команда, 442
write, команда (ex), 692
writers, файл (CVS), 746

X

x, команда (sed), 712
xargs, команда, 443
xbiff, программа, 862, 867
xclock, программа, 862, 867
xit, команда (ex), 693
xload, программа, 862
xmkmf, инструмент (imake), 213
xterm, 19

Y

y, команда (sed), 712
yacc, команда, 445
yank, команда (ex), 693
yes, команда, 445
ypbind, команда (NFS/NIS), 446
ypcat, команда (NFS/NIS), 446
ypchfn, команда (NFS/NIS), 447
ypchsh, команда (NFS/NIS), 447
ypinit, команда (NFS/NIS), 447
ypmatch, команда (NFS/NIS), 448
yppasswd, команда (NFS/NIS), 448
yppoll, команда (NFS/NIS), 449
yppush, команда (NFS/NIS), 449
ypserv, команда (NFS/NIS), 450
ypset, команда (NFS/NIS), 450
ypwhich, команда (NFS/NIS), 451
ypxfr, команда (NFS/NIS), 451

Z

z, команда (ex), 693
zcat, команда, 197, 452
zcmp, команда, 452
zdiff, команда, 453
zdump, команда, 453
zforce, команда, 453
zgrep, команда, 453
zic, команда, 453
zmore, команда, 457
znew, команда, 458

А

абзацы в Emacs, 649
Автозагрузка программ
  GNOME, 818
автоматическое поднятие окна с фокусом, fvwm2, 864
административные файлы CVS, 735
  переменные, 736
Адреса
  IP, 37
адреса
  копий строк в sed, 701
  строк, редактор ex, 681
адресация копий (sed), 701
Анимация, настройка
  для панели GNOME, 814
анимация, настройка
  оконный менеджер Enlightenment, 828
  оконный менеджер Sawfish, 821
Аппаратная часть
  команды, 28
апплеты
  Desk Guide, 805, 824
  Tasklist, 805
  панель GNOME, 807
арифметика
  язык bc, 60
Арифметические выражения, 537
архивные файлы, 52, 89, 395
архитектуры, тип, 53
Атрибуты файла, 76
атрибуты файлов, 278

Б

Безопасность
  ipchains, 219
  ipfwadm, 225
  iptables, 230
  брандмауэры и сокрытие адресов, 41
  команды администрирования, 32
библиотеки
  просмотр содержимого, 315
  связывание, 249
  создание файлов, 52
блокировки, механизм, в инструментах CVS/RCS, 730
Брандмауэры, 41
  ipchains, 219
  ipfwadm, 225
  iptables, 230
буферы
  запись на диск, 391
  команды Emacs, 648

В

ввод
  аргументов команд, 443
  команды со стандартного ввода, 59
  преобразование в формат почтового ящика, 155
  преобразование и вывод, 419
  режимы ввода-вывода терминала, 382
ввода-вывода, обработка в sed, 703
версии, нумерация (RCS), 787
Версии, управление
  CVS, инструмент, 729
ветви, управление версиями, 732
  команда admin, 752
  команда log, 775
Виртуальные рабочие столы
  оконный менеджер Enlightenment, 826
виртуальные рабочие столы, 844
  GNOME, 805
  KDE, 837
  оконный менеджер Enlightenment, 828
виртуальные рабочие столы, fvwm2, 858
виртуальные экраны, fvwm2, 861
Владение файлами, 81
внешний способ доступа к репозиторию, 754
волшебный файл, 144
восьмеричные числа прав доступа, 80
всплывающие подсказки
  отображение в оконном менеджере Sawfish, 823
всплывающие подсказки, отображение
  KDE, 842
  оконный менеджер Enlightenment, 827
Вставка
  команды vi, 672
Вторая расширенная файловая система
  и команда e2fsck, 131

вторая расширенная файловая система
настройка параметров работы, 422
отладка, 111
создание на устройстве, 301
вход в систему, 350
Вывод
постраничный просмотр, 304
файлов, 393
вывод
команда echo (bash), 549
команда tee, 401
отображение содержимого файлов, 394
печать файлов на стандартный вывод, 74
постраничный просмотр, 253
преобразованного ввода, 419
режимы ввода-вывода терминала, 382
выгрузка модулей, 352
выравнивание, команды Emacs, 649
Вычисление выражений, 537
вычисление выражений, 136, 412

Г

Главная загрузочная запись (MBR), 460
главное меню, GNOME, 809
Гнезда PCMCIA, управление, 72
границы, свойства (KDE), 845
график загрузки системы, построение, 415
группа, 193
Группы, 78, 314
группы, 78, 314
отображение пользовательской, 194
Группы пользователей, 12

Д

Дата и время
часовые пояса, 453
дата и время
задание в RCS, 787
календарь, 71
команда uptime, 430
ожидание, 374
планирование выполнения команд, 55
текущие, 106
установка, 313
файлы преобразования, 453
форматы дат CVS, 762
часовые пояса, 453
Двойная загрузка
Linux и Windows NT/2000, 475
двойная загрузка
Linux и Windows 95/98, 461
Демоны, 33
команды вызова, 27
маршрутизации, 38
дефис (-), метасимвол шаблона поиска, 636
Демоны
NFS, 44
Диагностирование
TCP/IP, 41
Диакритический символ ^
метасимвол шаблона поиска, 636
Диалоги
пользовательский интерфейс в GNOME, 819
диалоговый режим интерпретатора, 515
Директивы cpp, 97
директории
lost+found, 303
в стеке, 549
выполнение всех сценариев, 358
копирование в, 218
отображение имен, 126
отображение содержимого, 274
переименование, 310
смена, 548
создание, 300
текущая рабочая, 340
удаление, 352
Дискеты
загрузка системы, 459
диски
загрузочная информация, 460
информация об использовании, 341
копирование файловых архивов, 89
параметры, 367, 464
разбивка на разделы, 139
разбиение на разделы, 75
сброс буферов файловой системы, 391
форматирование, 139
дисковый навигатор, KDE, 844
Дистрибутив Berkeley Software (BSD), 21

дистрибутив Berkeley Software (BSD), 18
дистрибутивы ПО, распространение, 357
дистрибутивы ПО, установка, 357
Документация
TCP/IP, 36
страницы руководства, 52
документация
Emacs, 651
информационные файлы, 216
страницы руководства, 296
доллар, знак ($)
метасимвол шаблона поиска, 636
Домены, 45
дополнение
команды csh и tcsh, 603
дополнения
команды в bash, 526
Доступ к репозиторию
способы, 754

Ж

журнал команд
csh и tcsh, 599
интерпретатор bash, 538
команда fc, 552
работа в интерпретаторе bash, 525, 528

З

завершение работы
команды редактора vi, 675
с редактором vi, 675
завершение работы с ex, 669
заголовки окон
KDE, 851
оконного менеджера Sawfish, 821
загрузка модулей, 218, 304
выгрузка, 352
Загрузка, методы, 459
загрузочные параметры, 473, 477
загрузочный сектор, 460
загрузчик LILO, 19
загрузчики, 460
задания, 57
фоновые/приоритетные, 546, 553, 608
задания, выполнение по указанной дате/времени, 100
запись
на диск, 391
на терминал, 299
зарегистрированные в системе пользователи
перечисление, 359
звездочка (*)
метасимвол интерпретатора, 523
звуковые события
оконный менеджер Enlightenment, 827
звуковые события, настройка
GNOME, 817

И

ИБП, источник бесперебойного питания, 329
Идентификаторы
строковые, 731
идентификаторы, строковые
команда rtag, 777
Иерархические модули, 742
иконки рабочего стола
отображение в KDE, 846
создание в GNOME, 805
Имена
доменные, 39
команда etags, 134
имена
команда ctags, 103
пользователей, 265
создание псевдонимов файлов, 261
узлов, 204
файлов и директорий, 310
имена файлов, поиск по маске, 578
имена файлов, поиск по шаблону, 634
Инструмент RCS, 782
Интерпретаторы
переменные окружения bash, 534
смена командного интерпретатора, 83
сценарии, 496
интерпретаторы, 514
демон rshd, 357
изменение регистрационного, 447
команды специального режима Emacs, 649
работа в редакторе vi, 676
создание для действительного

идентификатора пользователя, 389
сценарии, 515
Интерпретаторы команд, 19
интерпретаторы, удаленные, программы доступа, 757

К

календарь, 71
Карты NIS, 45
карты, NIS, 448, 449
каталог
корневой, 83
клавиатура
автоповтор нажатий клавиш, 245
команда dumpkeys, 129
настройка в GNOME, 818
настройка в KDE, 849, 851
режим, 244
клавиатурные сокращения, 849
fvwm2, 872
оконный менеджер Enlightenment, 829
клавиатурные схемы, KDE, 850
классификация файлов по типам, 144
клиент, файлы настройки, 757
ключевые слова RCS, 785
ключевые слова, подстановка в CVS, 763
кнопки запуска
создание на панели, 808
кнопки запуска приложений
добавление приложений на панель GNOME, 808
Кнопки, настройка
для панели GNOME, 815
кнопок, панель (fvwm2), 861
Кода, моментальные копии - импортирование, 747
кодированные файлы, воссоздание исходных, 434
колонки
слияние строк текста в, 320
удаление, 86, 105
форматирование ввода, 86
команда egrep, 638
команда grep, 638
Командная строка
синтаксис для редактора ex, 668
командная строка
команды редактирования csh/tcsh, 604
обработка аргументов, 554
редактирование в bash, 524
синтаксис для редактора sed, 700
синтаксис для редактора vi, 667
синтаксис использования gawk, 714
Командный режим vi
конспект команд, 678
Команды
FTP, 160
Linux, 18
администратора CVS, 750
администрирование TCP/IP, 34
диагностирования TCP/IP, 41
менеджера пакетов Debian, 497, 789
редактора ex, 681
команды
csh и tcsh, 580
gdb (GNU debugger), 179
mail, 280
NIS, 446
telnet, 404
TFTP, 414
выполнение на удаленном узле, 357
выполнение после hangup, 316
инструмент nslookup, 316
интерпретатор, 516
интерпретатора bash, 530, 544
поиск, 554
пользователя CVS, 765
псевдонимы, 545
редактор sed, 701
редактора vi, 666
со стандартного ввода, 59
языка gawk, 721
Команды в стиле Emacs, 524
команды ветвления, sed, 703
команды навигации
редактор Emacs, 644
команды отступа в Emacs, 649
компилятор gcc, 170
компиляция исходных файлов C, 170
контрольная сумма (по CRC), 84
контрольная сумма, вычисление, 390
Конференции, Usenet, 11
Конфликты и слияние
управление версиями, 730
Копирование
команды vi, 673
копирование
файлов, 88, 342

копирование и вставка, 645
sed, 703
в редакторе vi, 674
копирование файлов, 109, 218
корневое меню fvwm2
изменение, 877
Корневое меню, fvwm2, 862

Л

липкие окна, fvwm2, 861, 867
лицензирование, 21
логические модули (CVS), 743
логические операторы, 137
локальный способ доступа к репозиторию, 754

М

макрокоманды
Emacs, 650
Макросы
команды etags, 133
команды vi, 676
макросы
инструмент make, 290
команда ctags, 103
процессор m4, 279
Маршрутизация, 38, 353
маски имен файлов
cvsignore, 739
маскировка, 529, 578
массивы
в gawk, 720
машина
тип архитектуры, 53
Менеджер иконок, модуль (fvwm2), 861
Менеджер пакетов Debian, 494
инструменты для работы с пакетами, 494
команды, 497, 789
состояние пакета/выбора, 495
сценарии Perl и интерпретатора, 496
файлы, 495
флаги пакетов, 496
менеджер сеансов KDE, 839
менеджеры пакетов, 481
меню, 809
меню панели, GNOME, 808
Метасимволы
поиск имен файлов, 634
метасимволы, 634
поиск имен файлов, 578
метасимволы замены, 640
модули
выгрузка, 352
загрузка, 218, 304
оконный менеджер fvwm2, 859
перечисление, 279
Центр управления KDE, 839
модульные ядра, 479
Монтирование файловых систем, 44
монтирование файловых систем, 307
размонтирование, 425
Мышь
настройка в GNOME, 818
настройка в KDE, 849

Н

Наборы правил (брандмауэра)
ipchains, 219
iptables, 230
Навигация, команды
редактор vi, 670
Настройка TCP/IP, 40
Настройка рабочего стола, 18
настройка сеанса работы с Linux, 515
настройки меню
панель GNOME, 815
нумерация версий (RCS), 787

О

области текста в Emacs, 646
обработка файлов журналов, 265
обратные переносы строк, отображение, 85
Обратный слэш
маскировка символов, 637
общение
с пользователями, 442
Общие клавиатурные комбинации, KDE, 850
объектные файлы
профилировочная информация, 186
размеры, 373
создание, 54
таблицы имен, 315
объявления переменных и массивов в

gawk, 720
Обычные модули, 742
Окна
всплывающие подсказки, отображение, 823, 827
обновление, 823
перемещение и изменение размеров, 823
поведение фокуса ввода, 821, 826
размещение новых, 824
скрытые, 821
окна
всплывающие подсказки, отображение, 842
изменение размера и перемещение, 854
команды Emacs, 648
настройка стандартного поведения, 820
настройка фокуса ввода, 855
настройка фокуса и перемещения, 826
правила размещения новых, KDE, 854
сворачивание и восстановление, 822
сворачивание/восстановление, 854
окнами просмотра GNOME
рабочее пространство Sawfish, 824
окнами просмотра, GNOME, 805
окон, список в KDE, 838
Оконные менеджеры
fvwm2, 857
Sawfish, 820
оконные менеджеры
Enlightenment, 825
kwm (KDE), 837
выбор для GNOME, 813
используемые с GNOME, 804
оконный менеджер Sawfish
настройка внешнего вида окон, 820
Операторы
интерпретатора bash, 540
языка bc, 64
операторы, 136
csh и tcsh, 595
интерпретатора bash, 535
языка сценариев gawk, 719
операторы подстановки
интерпретатора bash, 535
интерпретаторы csh и tcsh, 582
орфография, 242
откат, команды Emacs, 646
Отладка
файловой системы ext2, 111
отладка
gdb (GNU debugger), 179
Отображение
файлов на стандартный вывод, 393
отображение
обратных переносов строк, 85
отображение данных в обратном порядке, 393
отслеживание версий файлов, 731

П

пакетные флаги, 496
пакетов, менеджеры, 481
память, 158
использование дискового пространства, 127
свободное дисковое пространство, 115
Панель задач, KDE, 838
панель кнопок fvwm2
работа с комплектом рабочих столов, 865
панель меню, рабочий стол GNOME, 808
панель по краю, рабочий стол GNOME, 807
панель с привязкой, рабочий стол GNOME, 807
Параграфы в Emacs, 646
параметров, файл - Loadlin, 473
Параметры
диска, 464
параметры
загрузки, 473, 477
пароли, 339, 448
передача данных, проверка, 390
передача файлов, 160, 414
переименование файлов и директорий, 310
Переменные
интерпретатора bash, 534
переменные
csh и tcsh, 582
в языке сценариев gawk, 720
языка сценариев gawk, 718
переменные интерпретатора
экспортирование значений, 552
переменные окружения, 133, 593

инструмент CVS, 755
получение значений, 334
Перемещение курсора
команды vi, 670
перемещение курсора
команды Emacs, 644
перенаправление
в csh и tcsh, 581
интерпретатор bash, 531
Перестановки текста в Emacs, 646
Печать, 32, 269
демон lpd, 269
проверочная, 274
состояние очереди заданий, 273
строк, 380
печать
вывод символов плакатного размера, 58
значений переменных окружения, 334
настройка параметров, 424
строк, 334
файлов на стандартный вывод, 74
Печать, команды, 23
пиктограммы рабочего стола
управление перемещением/размещением на панели GNOME, 815
плюс, знак (+), метасимвол шаблона поиска, 637
поглощение модулей окном, 839
Поддержка, online, 11
подстановка команд в csh/tcsh, 599
подстановка слов в csh/tcsh, 600
подстановки, операторы
в интерпретаторе bash, 542
подсчет слов, 438
подсчет элементов текстового файла, 438
подчеркивание, 425
поиск
и замена, 640
команд, 554
команды Emacs, 647
команды редактора vi, 671
по содержимому файла, 187, 267, 453
по содержимому файлов, 132, 144
сбойных секторов, 58
специальные символы, 636
страницы руководства, 52, 439
файлов, 146, 262
поиск по шаблону, 633
в языке gawk, 716
операторы bash, 536
поиск различий, команды, 22
Поиск, команды, 24
Пользователи
вход в систему, 264
добавление, 314
команды администрирования, 32
учетные записи NIS, 46
пользователи
вход в систему, 349
идентификатор пользователя, 442
команда finger, 151
команда su, 389
общение, 395
общение с, 442
отчет по, 359
перечисление, 194, 434
перечисление для определенного узла, 359
получение информации о, 207, 441
посылка сообщений, 359, 438
смена групповой принадлежности, 314
создание учетных записей, 430
уведомление о пришедшей почте, 66
удаление учетных записей, 432
пользовательский интерфейс, настройка в GNOME, 818
Пользовательское меню, GNOME, 809
помещение директорий в стек, 549
последовательные соединения, 373
посылка сообщения
пользователям, 438
Почта
команды администрирования, 29
уведомление о новых сообщениях, 66
Почтовые сообщения
автоматический ответ, 435
переменные интерпретатора bash, 538
почтовые сообщения
кодирование файлов данных, 434
получение с почтовых серверов, 140
удаленная обработка, 351
пошагового поиска, команды, 647
права владения, файлы, 81

Права доступа к файлам
задание в восьмеричном виде, 80
права доступа к файлам
группы, 78, 314
изменение, 78
Правил, наборы (брандмауэры), 42
Правила отбора (netfilter), 230
преобразование файлов .Z в .gz, 458
прерывание
команды Emacs, 646
Приглашение командной строки
настройка, 543
приложения
открытие на разных рабочих столах (fvwm2), 869
пользовательский интерфейс в GNOME, 818
приложения настройки, GNOME, 812
принудительное завершение процессов, 246
приоритет сообщения, 248
приоритетные задания, 553, 608
приоритеты процессов, 315, 347
приостановка, редактор ex, 694
Пробелы
отступы в Emacs, 649
пробелы
команда colcrt, 85
команда fmt, 154
команды отступа в Emacs, 649
преобразование в табуляторы и обратно, 135, 426
проверка орфографии, 242
Проверочный текст
печать, 274
Программирование в интерпретаторе, команды, 24
Программирование, команды, 23
программы
сценарии интерпретатора, 515
проект GNU, 18
Проект Документирования Linux, 10
прозвонка (ping) узлов, 325
производительность
график средней загрузки системы, 415
загрузка процессора, 416
установка скорости клавиатурного автоповтора, 245
Протоколы, TCP/IP, 35, 38
профилировочная информация для объектного файла, 186
процедуры языка gawk, 717
Процессы
идентификаторы (PID), 246
команды управления, 32
приоритеты, 315
процессы, 334
идентификаторы (PID), 325
идентификация по файлу или файловой системе, 168
интенсивность использования процессорных ресурсов, 416
команда exec, 551
останов, 199
принудительное завершение, 246
приоритеты, 347
сопроцессы bash, 533
Псевдонимов, модули, 742
Псевдонимы
специальные, в tcsh, 603
пусковые иконки
создание на рабочем столе в GNOME, 805
пути
к страницам руководства, 298
текущая рабочая директория, 340
путь к репозиторию
pserver, 744, 750

Р

Работа с заданиями, 608
Работа с сетями, 33
ipchains, 219
ipfwadm, 225
iptables, 230
определение состояния сети, 313
получение информации о состоянии узла, 393
Работа с текстом, команды, 25
Работа с файлами, команды, 22
рабочего стола, проводник в KDE, 837
Рабочее пространство оконного менеджера Sawfish, 824
Рабочие среды
KDE, 283, 480, 830
рабочие столы (fvwm2), 865
рабочий стол, инструменты (fvwm2), 861
разбивка дисков на разделы, 139
разбивка файлов, 101
разбивка файлов на секции, 376

разбиение жестких дисков, 75
развилки, управление версиями, 732
раздел подкачки (swap), 303, 390
разделы
    страницы руководства, 298
размер блока
    изменение, 109
Размонтирование файловых систем, 425
размонтирование файловых систем, 425
разрыв строк текста, 155
регистр (заглавные)
    преобразование, 110
регистра (преобразование)
    команды Emacs, 647
регистрационные имена пользователей, 265
Регулярные выражения, 536
регулярные выражения
    поиск в файлах, 132
    поиск по содержимому файла, 187
Редактирование
    команды vi, 672
редактирование
    в командной строке, 702
    командной строки
        csh и tcsh, 604
редактирование в стиле vi, 527
редактор Emacs, 20, 641
редактор vi, 20
    настройка, 695
Редактор меню, GNOME, 810
редактор, по умолчанию в GNOME, 817
режим построчного редактирования (bash), 541
режим работы системы, 358, 402
Режимы, Emacs, 643

С

свободная панель, рабочий стол GNOME, 808
связывание клавиш, 546
связь по последовательным линиям, 40
Сектора
    поиск сбойных, 58
сектора
    загрузочный, 460
секции
    разбивка файлов, 376
    файлов, 101
семафоров, файлы, 263
Серверы
    NFS, 43
    NIS, 45
серверы
    демон rshd, 357
    получение информации, 371
сетевые команды, 22
Сети
    команды администрирования, 31
сетка, fvwm2, 858, 862, 864
сжатие
    преобразование .Z в .gz, 458
Сжатие файлов
    команда bzip2, 70
    команда compress, 87
    команда gzexe, 197
сжатие файлов
    команда gzip, 452
символические ссылки, 261
Символы
    специального режима интерпретатора в Emacs, 649
символы
    подсчет количества в файле, 438
    подчеркивание, 425
    преобразование в строке, 419
    скорость автоповтора, 245
Система
    команды администрирования, 32
система
    контрольные сообщения, 126
    режим работы, 358, 402
    состояние, 441, 539
    статистика использования, 437
    установка времени, 313
система меню, fvwm2, 862
система, информация о состоянии, KDE, 849
Системное меню, GNOME, 809
скользящая панель, рабочий стол GNOME, 807
Слияния, механизм, в CVS, 730
слова, сокращения в Emacs, 648
словарь, орфография, 242
Служба имен, 39
сокращений, команды в Emacs, 648
Сокрытие адресов, 41
    ipchains, 219
    ipfwadm, 225

iptables, 230
сообщения
посылка всем пользователям, 438
посылка всем пользователям узла, 359
посылка на терминал, 299
системные, 126, 247
сопроцессы, 533
сортировка содержимого файлов, 374
Состояние системы, команды, 25
состояния, версий (RCS), 788
сохранение файла, команды vi, 675
Специальные символы
команда tr, 419
специальные символы
интерпретатора bash, 529
команда colcrt, 85
справка по командам Emacs, 651
Сравнение файлов, 116
сравнение файлов, 84, 86, 453
cc, команда (см. gcc, компилятор), 75
ссылки
создание новых на рабочем столе GNOME, 806
Ссылки рабочего стола
настройка цветов, 840
ссылки рабочего стола
добавление приложений на панель KDE, 838
настройка свойств в KDE, 836
создание в KDE, 833
Стандартные клавиатурные комбинации, KDE, 850
Столлман, Ричард, 21
страницы руководства
поиск, 52, 439
страницы руководства (man), 296
Строки
печать, 380
строки
печать, 334
преобразование символов, 419
Строки, текст
разрыв, 155
сравнение, 116
строки, текст
печать, 200, 394
подсчет, 438
поиск, 132, 187, 267
разрыв, 155
слияние в колонки, 320
сравнение, 86
удаление идентичных, 427
Строковые идентификаторы, 731
строчная адресация (ex), 681
Сценарии
интерпретатора/Perl, 496
сценарии
выполнение всех в директории, 358
завершение, 551

Т

текст
проверка орфографии, 242
разрыв строк, 155
текущая
рабочая директория, 340
текущее
время и дата, 106
темы, выбор
GNOME, 813
KDE, 846
оконный менеджер Enlightenment, 829
терминалы
getty и agetty, 49, 185
очистка экрана, 84, 348
посылка сообщений, 299
режимы ввода-вывода, 382
тэги
команда rtag, 777
перемещение, 752

У

Удаление
команды vi, 673
удаление
директорий, 352
заданий из очереди, 58
заданий из очереди печати, 273
идентичных строк текста, 427
колонок, 105
команды Emacs, 645
обработка файлов журналов, 265
файлов, 351
Удаление колонок, 86
удаленная
передача файлов между узлами, 414
удаленное
выполнение команд, 357
копирование файлов, 342

удаленные
интерпретаторы (CVS), 757
удаленный
доступ к системе, 349
узел, машина, 204
Узлы
команды администрирования, 29
получение информации, 393
узлы
время непрерывной работы, 358
отслеживание маршрутов пакетов, 421
перечисление работающих пользователей, 359
получение информации, 201, 358
посылка сообщения всем пользователям, 359
прозвонка (ping), 325
синхронизация версий файлов, 343
удаленное выполнение команд, 357
удаленный вход в систему, 350
удаленный доступ, 349
указатель мыши, фокусная модель (fvwm2), 863
указатель, перемещение с помошью клавиш (fvwm2), 875
Управление версиями
CVS, инструмент, 729
управление версиями
инструмент RCS, 782
управление питанием, KDE, 846
условные выражения, 537
Установка
команды администрирования, 29
устранение проблем
редактор Emacs, 642

Ф

файл .plan, 151
файл .project, 151
файл /etc/groups, 78
файл /etc/passwd, 82
изменение информации, 77
Файловые системы
NFS, 43
идентификация использующих процессов, 168
команды администрирования, 29
проверка, 158
создание, 301
файловые системы
MINIX, 159, 303
MS–DOS, проверка, 126
вторая расширенная, 111, 131, 301, 422
монтирование, 307
перезагрузка, 347
размонтирование, 425
сброс буферов на диск, 391
создание, 302
файловых имен, маски
файл rcsinfo, 745
файл taginfo, 745
Файлы
атрибуты, 76
идентификация использующих процессов, 168
объединение изменений (CVS), 730
печать, 74, 269, 393
постраничный просмотр, 304
преобразование, 109
преобразование .Z в .gz, 458
работа в редакторе vi, 675
сжатие, 87, 426
слияние строк, 243
сравнение, 116
управление пакетами в Debian, 495
файлы
атрибуты, 77, 278
выполняемые, 440
вычисление контрольной суммы, 390
зависимости, 114
инструмент CVS, 729
Инструмент RCS, 782
интерпретатора bash, 523
классификация по типам, 144
кодирование для пересылки почтой, 434
кодированные, воссоздание исходных, 434
команды редактора Emacs, 644
копирование, 88, 109, 342
настройки imake, 213
обновление времени доступа и изменения, 418
передача, 160, 414
переименование, 310
перечисление, 274
печать, 200
подсчет элементов, 438

поиск, 145, 262
поиск по содержимому, 132, 144, 187, 267, 453
постраничное форматирование, 331
постраничный просмотр, 253
почтовые, 286
преобразования времени, 453
проверка, 84, 323
проверки, 412
разбивка на секции, 101, 376
сжатие, 70, 197, 452
синхронизация версий между узлами, 343
слияние строк текста, 320
сортировка содержимого, 374
сравнение, 84, 86, 453
удаление, 351
удаление идентичных строк, 427
файлы журналов
обработка, 265
флаги пакетов, 496
фокус ввода
оконный менеджер Sawfish, 821
фокус ввода в окнах
оконный менеджер Enlightenment, 826
фокус ввода, поведение в окнах
fvwm2, 863
фокус, поведение
KDE, 855
Фон рабочего стола
настройка в менеджере окон Enlightenment, 828
фон рабочего стола
настройка в KDE, 844
настройки GNOME, 812
Фонд Свободного ПО (FSF), 21
фоновые задания, 546, 608
формат
даты и времени, 106
форматирование
диска, 139
почтовый ящик, преобразование ввода, 155
приглашения tcsh, 591
файлов, постраничное, 331
форматирование приглашения tcsh, 591
Функции
команда etags, 133
функции
в языке bc, 63
инструмент make, 291
команда ctags, 103
связывание функций, 546

Х

Хранение, команды, 25

Ц

цветовые схемы, KDE, 845
Целостность системы, 32
Центр Управления, 811
центрирование, команды Emacs, 650
Цепи (наборы правил), 42
ipchains, 219
iptables, 230
Циклы, 547
циклы, 548, 553

Ч

Часы и время, 27
числа
арифметические операции bash, 539
вычисление выражений, 136
номера/адреса строк, 681
нумерация строк и адресация, 672
язык bc, 60

Ш

Шлюзы, 38
шрифты, настройка (KDE), 845

Э

экранные заставки, настройка GNOME, 813
экранные заставки, настройка в KDE, 848
Экспортирование файловых систем, 44

Я

Ядра, 16
брандмауэры и сокрытие адресов, 42
команды администрирования, 30
ядра
ipfwadm, 225

iptables, 230
модульные, 479
отображение экспортируемых имен, 249
параметры, передаваемые при загрузке, 477
установка видеорежима, 436

По договору между издательством «Символ-Плюс» и Интернет-магазином «Books.Ru – Книги России» единственный легальный способ получения данного файла с книгой ISBN 5-93286-012-X, название «Linux. Справочник, 3-е издание» – покупка в Интернет-магазине «Books.Ru – Книги России». Если Вы получили данный файл каким-либо другим образом, Вы нарушили международное законодательство и законодательство Российской Федерации об охране авторского права. Вам необходимо удалить данный файл, а также сообщить издательству «Символ-Плюс» (piracy@symbol.ru), где именно Вы получили данный файл.