Beginning
Linux® Programming
4th Edition

Beginning Linux® Programming

4th Edition

Neil Matthew
Richard Stones

Wiley Publishing, Inc.

Нейл Мэтью
Ричард Стоунс

Основы программирования в Linux

4-е издание

Санкт-Петербург
«БХВ-Петербург»
2009

УДК 681.3.06
ББК 32.973.26-018.1
М97

Мэтью, Н.

М97 Основы программирования в Linux: Пер. с англ. / Н. Мэтью, Р. Стоунс. — 4-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2009. — 896 с.: ил.

ISBN 978-5-9775-0289-4

В четвертом издании популярного руководства даны основы программирования в операционной системе Linux. Рассмотрены: использование библиотек C/C++ и стандартных средств разработки, организация системных вызовов, файловый ввод/вывод, взаимодействие процессов, программирование средствами командной оболочки, создание графических пользовательских интерфейсов с помощью инструментальных средств GTK+ или Qt, применение сокетов и др. Описана компиляция программ, их компоновка с библиотеками и работа с терминальным вводом/выводом. Даны приемы написания приложений в средах GNOME® и KDE®, хранения данных с использованием СУБД MySQL® и отладки программ. Книга хорошо структурирована, что делает обучение легким и быстрым.

Для начинающих Linux-программистов

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Перевод с английского	*Татьяны Коротяевой*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.01.09.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 72,24.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.002108.02.07 от 28.02.2007 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-0-470-14762-7 (англ.)
ISBN 978-5-9775-0289-4 (рус.)

Оглавление

Об авторах

Нейл Мэтью (Neil Matthew) интересуется компьютерами и пишет для них программы с 1974 г. Выпускник университета г. Ноттингема по специальности "Математика", Нейл по-настоящему увлекается языками программирования и любит искать новые пути решения компьютерных проблем. Им разработаны системы программирования на языках BCPL, FP (Functional programming), Lisp, Prolog и структурированном BASIC. Он даже написал эмулятор микропроцессора 6502 для выполнения в системах UNIX программ для микрокомпьютера BBC.

Что касается опыта работы в UNIX, начиная с конца 1970 гг., Нейл испробовал все варианты, включая BSD UNIX, AT&T System V, Sun Solaris, IBM AIX, многие другие и, конечно, Linux. Он может утверждать, что занимается ОС Linux с августа 1993 г., когда обзавелся дистрибутивом из Канады Software Landing (SLS) на дискетах с версией ядра 0.99.11. Он применял компьютеры на базе Linux дома и на работе для осваивания языков C, C++, Icon, Prolog, Tcl и Java.

Все "домашние" проекты Нейла разработаны на базе Linux. Он утверждает, что Linux гораздо проще, поскольку поддерживает множество функциональных возможностей других систем, поэтому программы, предназначенные для систем BSD и System V, как правило, компилируются с небольшими изменениями или вообще без каких-либо изменений.

Сейчас Нейл работает в компании Celesio AG как архитектор ПО компании (Enterprise Architect), специализирующийся на разработке стратегии информационных технологий. У него есть профессиональный опыт технического консультирования, разработки программного обеспечения и контроля качества. Нейл также писал программы на языках C и C++ для встроенных систем реального времени.

Нейл женат на Кристине, у них двое детей, Александра и Адриан. Он живет в перестроенном амбаре (barn) в графстве Нортгемптоншир в Англии. К его любимым занятиям относятся решение головоломок на компьютере, музыка, научная фантастика, сквош и горный велосипед, только не подражайте ему в этом последнем.

Ричард Стоунс (Richard Stones) начал программировать в школе (раньше, чем он может вспомнить) на микрокомпьютере BBC, оснащенном микропроцессором 6502, который с помощью нескольких запчастей продолжал функционировать следующие 15 лет. Он закончил университет г. Ноттингема по специальности "Электроника", но решил, что программное обеспечение увлекательнее.

Он работал в разных компаниях, начиная от очень маленьких с десятком сотрудников и заканчивая очень большой, включая гиганта IT-сервисов, компанию EDS.

В эти годы он занимался рядом проектов, будь то коммуникации в режиме реального времени для систем бухгалтерского учета или большие справочные настольные системы. В настоящее время Рик работает архитектором информационных технологий, действуя как технический руководитель разных крупных проектов в большой панъевропейской компании.

Будучи отчасти лингвистом в программировании он писал программы на разных ассемблерах, на чистом, патентованном языке телекоммуникаций, названном SL-1, нескольких диалектах языка FORTRAN, языках Pascal, Perl, SQL и чуть-чуть на Python, C++ и C. (Под давлением он даже признался, что одно время считался не без оснований специалистом в Visual Basic, но старается не афишировать это временное помрачение рассудка.)

Рик живет в деревне графства Лестершир в Англии с женой Анной, детьми Дженнифер и Эндрю и кошкой. В свободное от работы время он увлекается классической музыкой, особенно старинной духовной музыкой, и фотографией и прилагает максимум усилий, чтобы найти время для игры на пианино.

Благодарности

Авторы хотели бы поблагодарить многих людей, помогавших сделать возможным издание книги.

Нейл хотел бы поблагодарить свою жену Кристину за понимание и своих детей Александру и Адриана, не слишком жаловавшихся на то, что отец проводит так много времени в своем кабинете, занимаясь писательством.

Рик благодарен своей жене Анне и своим детям, Дженнифер и Эндрю за их долготерпение по вечерам и в выходные дни, когда отец опять и опять принимался за работу над книгой.

Мы хотели бы выразить признательность сотрудникам издательства Wiley, которые помогли подготовить это четвертое издание к печати. Спасибо Кэрол Лонг (Carol Long) за запуск этого процесса и улаживание проблем, связанных с контрактами, особая благодарность Саре Шлаер (Sara Shlaer) за исключительную редакторскую работу и Тимоти Борончику (Timothy Boronczyk) за отличные технические рецензии. Мы также хотим поблагодарить Дженни Ватсон (Jenny Watson) за поиск средств для оплаты неожиданно возникавших дополнительных расходов и сопровождение книги на всех административных уровнях, Биллу Бартону (Bill Barton) за обеспечение надлежащих организации и презентации и Киму Коферу (Kim Cofer) за тщательную корректуру. Мы также очень признательны Эрику Фостеру-Джонсону (Eric Foster-Johnson) за его фантастическую работу над *главами 16* и *17*. Мы можем сказать, что благодаря стараниям всех вас книга стала лучше.

Мы также хотели бы поблагодарить наших работодателей, компании Scientific Generics, Mobicom и Celesio за поддержку во время подготовки всех четырех изданий книги.

В заключение нам хотелось бы выразить глубокую признательность двум важнейшим инициаторам, сделавшим возможным появление книги. Во-первых, Ричарду Столлмену (Richard Stallman) за отличные средства проекта GNU и идею среды со свободно распространяемым программным обеспечением, ставшей в наши дни реальностью благодаря GNU/Linux, и во-вторых, Линусу Торвальдсу (Linus Torvalds) за начатую и продолженную им совместную разработку, которая дает нам все улучшающееся ядро системы Linux.

Предисловие

У всех программистов есть своя груда записей и черновиков. Они собирают собственные примеры текстов программ, накопившиеся за время героических погружений в многочисленные руководства или добытые из сети Usenet, в которой порой даже дураки боятся блуждать. (Другая точка зрения состоит в том, что у всех дураков свободный доступ к Usenet, и они используют ее безостановочно.) Поэтому довольно странно, что так мало книг выпущено в подобном стиле. В интерактивном мире существует множество коротких документов, касающихся конкретных проблем программирования и администрирования по существу. В рамках проекта по созданию документации Linux выпущено множество документов, посвященных самым разным темам, начиная с установки ОС Linux и Windows на одной машине и заканчивая написанием вашей виртуальной машины Java для Linux. На самом деле, загляните на Web-сайт Linux Documentation Project (проект документации Linux) по адресу **http://www.tldp.org**.

С другой стороны, книжный мир кажется состоящим в основном из подробных и самых полных научных томов, читать которые у вас нет времени, или книг для новичков, которые вы покупаете в шутку друзьям. Есть очень немного книг, которые пытаются рассмотреть основы целого ряда полезных проблем. Предлагаемая книга — одна из них, это набор расшифрованных (попробуйте прочесть что-нибудь, написанное программистом от руки), отредактированных и собранных в логически оправданную структуру заметок и черновиков программистов.

Данное издание книги было проверено и исправлено в соответствии с современным уровнем разработок в ОС Linux.

Алан Кокс (Alan Cox)

Введение

Рады предложить вам легкое в использовании руководство по разработке программ для Linux и других UNIX-подобных операционных систем.

В этой книге мы стремимся дать вам представление о разнообразных понятиях, важных для разработчика, применяющего систему Linux, так сказать, основы. Слово "основы" скорее относится к ее содержимому, чем к уровню подготовки читателя. Мы построили книгу так, чтобы, несмотря на уже приобретенный вами опыт, вы узнали больше о том, что может предложить Linux. Программирование в ОС Linux — обширная область, и мы стараемся представить достаточно сведений, касающихся различных тем, чтобы дать вам прочные "основы" для усвоения каждой из них.

Для кого эта книга?

Если вы программист, стремящийся повысить квалификацию за счет функциональных возможностей, предлагаемых разработчикам Linux (или UNIX), уделить больше внимания программированию и расширить применение ваших приложений в системе Linux, вы выбрали нужную книгу. Понятные объяснения и практический на основе пошаговых проверок подход поможет вам быстро повысить профессиональный уровень и освоить все ключевые методы.

Мы полагаем, что у вас есть некоторый опыт программирования на языках C и/или C++ в ОС Windows или какой-нибудь другой операционной системе, но мы старались сохранить простоту приведенных в книге примеров, чтобы для их понимания не требовалось слишком высокой квалификации в программировании на C. Все явные сопоставления методов программирования в Linux с приемами программирования на языках C/C++ отмечены в тексте книги.

> ***Примечание***
>
> Особое примечание для тех, кто впервые знакомится с ОС Linux. Эта книга не об установке и настройке Linux. Если вы хотите узнать больше об администрировании системы Linux, возможно вы захотите познакомиться с дополнительными книгами.

Поскольку книга задумана как учебник, знакомящий с различными средствами и наборами функций/библиотек, доступных вам в большинстве систем Linux, а также удобный справочник, к которому вы можете обращаться время от времени, она отличается четкой и понятной формой изложения, исчерпывающими пояснениями и подробными примерами.

Чему посвящена книга?

У книги есть ряд задач, перечисленных далее.

- Научить применению стандартных библиотек языка C в ОС Linux и других средств, описанных в разных стандартах Linux и UNIX.
- Показать, как использовать большинство стандартных средств разработки Linux.
- Дать краткий обзор способов хранения данных под управлением Linux с помощью СУБД DBM и MySQL.
- Показать, как создавать графические интерфейсы пользователя на базе графической системы X Window System. Мы воспользуемся библиотеками GTK (основы графической среды GNOME) и Qt (основы графической среды KDE).
- Поддержать вас и дать вам практические навыки разработки собственных реальных приложений.

В ходе обсуждения этих тем мы знакомим вас с теорией программирования и затем иллюстрируем ее соответствующими примерами и понятными пояснениями. Таким образом, вы сможете быстро освоить тему и потом при необходимости вернуться к ней, чтобы освежить все существенные ее составляющие.

Помимо маленьких примеров, разработанных в основном для иллюстрации работы ряда функций или некоторых элементов теории в действии, в книге рассматривается большой пример: разработка простого приложения для работы с базой данных, хранящей информацию о коллекции аудиокомпакт-дисков. По мере накопления ваших знаний вы сможете разрабатывать, переделывать и расширять проект в соответствии со своими желаниями. Несмотря на сказанное, приложению для работы с компакт-дисками не отводится главная роль в главах книги, и вы можете пропустить посвященные ему страницы, но нам кажется, что оно предлагает полезные, более сложные примеры применения обсуждаемых нами методов. Оно также позволяет наилучшим образом проиллюстрировать все более сложные темы по мере их рассмотрения. Наше первое обсуждение данного приложения появится в конце *главы 2* и покажет, как организован сценарий командной оболочки очень большого объема, как командная оболочка обрабатывает ввод пользователя и как она может формировать меню, хранить и искать данные.

После повторения основных правил компиляции программ, компоновки их с библиотеками и доступа к интерактивным руководствам вы проникните на время в командные оболочки. Затем вы перейдете к программированию на языке C, здесь мы обсудим работу с файлами, получение данных из окружения ОС Linux, обработку терминального ввода и вывода и библиотеку curses (позволяющую легче обрабатывать интерактивный ввод и вывод). После этого вы будете готовы взяться за новую реализацию приложения для работы с компакт-дисками на языке C. Проект приложения останется прежним, но в программном коде будет использована библиотека curses для создания экранного пользовательского интерфейса.

После этого мы обсудим управление данными. Знакомство с библиотекой базы данных dbm, к которой мы обратимся в нескольких последующих главах, — доста-

точное основание для переделки приложения, но на этот раз вместе с проектом. В следующей главе рассматривается хранение данных в реляционной базе данных средствами СУРБД MySQL и позже мы также повторно применим эти методы хранения данных, поэтому вы сможете сравнить разные способы управления данными. Размер новых версий приложения таков, что нам далее придется иметь дело с такими практическими задачами, как отладка, контроль исходного текста программы, распространение программного обеспечения и make-файлы.

Вы также узнаете, как могут взаимодействовать разные процессы в ОС Linux и как программы в Linux могут применять сокеты для поддержки сетевых соединений разных машин на базе протоколов TCP/IP, включая проблемы, касающиеся связи машин с процессорами разной архитектуры.

После изложения основ программирования в Linux мы обсуждаем создание программ в графическом режиме. Этому посвящены две главы, в которых сначала рассматривается комплект инструментальных средств GTK+, лежащий в основе графической среды GNOME, а затем комплект Qt, лежащий в основе графической среды KDE.

Заканчиваем мы кратким рассмотрением стандартов, делающих системы Linux разных поставщиков настолько подобными, что мы можем легко переходить из одной системы в другую и писать программы, которые будут работать в разных дистрибутивах Linux.

Как вы догадываетесь, в книге есть и многое другое, но мы надеемся, что этот краткий обзор даст вам достаточное представление о материале, который мы обсуждаем.

Что вам потребуется для использования книги?

В этой книге мы привьем вам вкус к программированию в Linux. Для того чтобы извлечь максимум пользы, нужно в процессе чтения выполнять примеры. Они служат хорошей практической базой и наверняка вдохновят вас на создание собственных программ. Мы надеемся, что вы будете читать эту книгу и одновременно экспериментировать в установленной у вас системе Linux.

Существуют варианты Linux для самых разных систем. Адаптируемость Linux такова, что предприимчивые люди заставляют ее работать в том или ином виде на любом оборудовании, имеющем процессор! Примеры включают системы на базе процессоров Alpha, ARM, IBM Cell, Itanium, PA-RISC, PowerPC, SPARC, SuperH и ЦП 68k, а также на базе различных процессоров класса *x*86 с 32- и 64-разрядными версиями.

Мы писали эту книгу и разрабатывали примеры в двух системах Linux с разными спецификациями, поэтому мы уверены, что если у вас есть работающая система Linux, вы сможете найти хорошее применение этой книге. Более того, пока проходило техническое рецензирование книги, мы проверили программный код в других версиях Linux.

Работая над книгой, мы сначала главным образом использовали системы на базе процессоров *x*86, хотя мало что из описанного в книге характерно только для *x*86. Несмотря на то, что можно успешно запускать Linux на PC 486 с 8 Мбайт RAM, для успешной работы современного дистрибутива Linux и выполнения примеров из этой книги мы советуем выбрать современную версию одного из наиболее популярных дистрибутивов Linux, например Fedora, openSUSE или Ubuntu, и проверить их аппаратные рекомендации.

Что касается требований к программному обеспечению, мы полагаем, что вы используете современную версию предпочитаемого вами дистрибутива Linux и, чтобы поддерживать систему на современном уровне и иметь самые свежие исправления найденных ошибок, применяете текущий набор обновлений, которые большинство поставщиков делают доступными интерактивно в виде автоматических обновлений. Linux и комплект инструментальных средств проекта GNU выпускаются на условиях GNU General Public License (GPL) (Общедоступной лицензии проекта GNU). Большинство других компонентов типичного дистрибутива Linux ссылаются либо на GPL, либо на одну из множества других лицензий Open Source (открытый или свободно распространяемый программный код), и это означает, что у них есть определенные характеристики, одна из которых — свобода. У них всегда есть исходный программный код, и никто не может отнять эту свободу. Дополнительную информацию о GPL см. на Web-сайте **http://www.gnu.org/licenses/**, а определение Open Source и разные применяемые лицензии — на Web-сайте **http://www.opensource.org**. В случае GNU/Linux у вас всегда будет возможность технической поддержки либо благодаря самостоятельной работе с исходным программным кодом, либо за счет найма стороннего специалиста или обращения к одному из поставщиков, предлагающих платную техническую поддержку.

Исходный программный код

Для работы с примерами книги можно ввести программный код вручную или воспользоваться сопроводительными файлами с исходным текстом примеров. Весь программный код, применяемый в книге, можно найти на Web-сайте **http://www.wrox.com**. Открыв главную страницу сайта, просто найдите заголовок книги (либо с помощью поля **Search** (Поиск), либо используя один из списков заголовков) и на странице с описанием книги щелкните кнопкой мыши ссылку **Download Code** для того, чтобы получить весь программный код примеров.

> ***Примечание***
>
> Поскольку у многих книг похожие заголовки, легче всего найти нужную книгу по номеру ISBN (International Standard Book Number); ISBN этой книги (оригинальной) — 978-0-470-14762-7.

После загрузки программного кода из Интернета просто распакуйте его своей любимой программой сжатия. Вы также можете перейти на главную страницу загрузки программного кода издательства Wrox **http://www.wrox.com/dynamic/books/download.aspx**, для того чтобы просмотреть код к данной книге и ко всем остальным книгам издательства.

Замечание, касающееся программного кода примеров

Мы старались представить примеры программ и фрагменты программного кода, наилучшим образом иллюстрирующие темы и понятия, обсуждаемые в тексте. Для того чтобы сделать новую функциональную возможность, которая рассматривается, максимально понятной, мы допускали одну-две вольности в стиле программирования.

В основном мы не всегда проверяли результаты, возвращаемые каждой вызываемой нами функцией, на соответствие ожидаемым. В профессиональном программном коде для реальных приложений мы всегда выполняли бы такую проверку, и вы должны внедрять строгие методы, касающиеся обработки ошибок. (В *главе 3* мы обсуждаем некоторые способы обнаружения и обработки ошибок.)

Общедоступная лицензия проекта GNU

Исходный программный код книги сделан доступным на условиях Общедоступной лицензии проекта GNU версии 2 (GNU General Public License, version 2), опубликованной на Web-странице **http://www.gnu.org/licenses/old-licenses/gpl-2.0.html**. Приведенное далее положение о разрешении и правах применяется ко всему программному коду данной книги.

```
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
(Это программа — свободно распространяемое программное обеспечение; вы можете
распространять ее и/или изменять на условиях Общедоступной лицензии GNU,
опубликованной Фондом свободного программного обеспечения; либо версии 2 этой
лицензии, либо (по вашему усмотрению) любой более свежей версии.)

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
(Эта программа распространяется в расчете на ее полезность, но без каких-либо
гарантий, даже без подразумеваемой гарантии ТОВАРНОГО СОСТОЯНИЯ ПРИ ПРОДАЖЕ И
ПРИГОДНОСТИ ДЛЯ ИСПОЛЬЗОВАНИЯ В КОНКРЕТНЫХ ЦЕЛЯХ. Более подробную информацию
см. в Общедоступной лицензии проекта GNU.)

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
(Вы должны были получить копию Общедоступной лицензии GNU вместе с этой
программой; если этого не произошло, напишите в Фонд свободного программного
обеспечения по адресу Free Software Foundation, Inc., 59 Temple Place, Suite
330, Boston, MA 02111-1307 USA)
```

Стилевое оформление, принятое в книге

Для того чтобы помочь вам максимально усвоить текст и следить за тем, что происходит, мы применили во всей книге ряд стилевых оформлений и выделений текста.

> ***ПРИМЕЧАНИЕ***
>
> Фрагменты, подобные данному, содержат требующую запоминания критически важную информацию, которая непосредственно относится к окружающему тексту, а также советы, подсказки, особенности и замечания, касающиеся текущего обсуждения.

Когда вводятся *важные понятия*, мы выделяем их курсивом. Символы, которые вы **должны ввести**, выделяются жирным моноширинным шрифтом. Элементы интерфейса выделены полужирным шрифтом. Комбинации клавиш обозначаются следующим образом: <Ctrl>+<A>.

Программный код и терминальные сеансы мы приводим тремя разными способами:

```
$ who
root tty1 Sep 10 16:12
rick tty2 Sep 10 16:10
```

Верхняя строка приведенного кода — это командная строка, а остальные строки отображаются в обычном стиле. Знак `$` — приглашение (если для ввода команды требуется суперпользователь, приглашение обозначается знаком `#`); жирным шрифтом помечается текст, который вы должны ввести, и для выполнения команды следует нажать клавишу <Enter> (или <Return>). Любой последующий текст, набранный тем же шрифтом, но без выделения жирным, — это вывод обозначенной жирным шрифтом команды. В приведенном примере вы вводите команду `who` и видите ее вывод в двух строках, расположенных под ней.

Прототипы функций и структуры, определенные в системе Linux, приводятся жирным шрифтом, как показано далее:

```
#include <stdio.h>
int printf (const char *format, ...);
```

В программном коде наших примеров строки с выделенным фоном указывают на новый важный материал, например, так:

```
/* Это новый и важный материал, и соответствующий код выглядит так. */
```

если код выглядит так, как показано далее (без выделения фоном), он менее важен:

```
/* Этот код уже встречался, и он выглядит так. */
```

Часто, когда программа создается на протяжении главы, только что добавленный код первый раз приводится без фона. Например, новая программа будет выглядеть следующим образом.

```
/* Программный код примера */
/* Это строка завершения. */
```

Если позже в этой главе мы добавим в нее новые строки, она будет выглядеть так:

```
/* Программный код примера. */
/* В эти строки */
/* добавляется новый код */
/* Это строка завершения. */
```

И последнее, принятое в книге стилевое оформление, о котором следует упомянуть, — все примеры программного кода начинаются с заголовка "Упражнение", который помогает разделить код там, где это полезно, выделить его составные части и показать, как развивается приложение. Когда это важно, мы после программного кода включаем раздел "Как это работает" для пояснения основных мест в тексте программы, касающихся изложенной перед ним теории. Мы считаем, что эти два приема помогают разбить наиболее громоздкие листинги на легко перевариваемые кусочки.

Ошибки

Мы приложили максимум усилий, чтобы избежать ошибок в тексте и программном коде. Но никто не идеален, и ошибки есть. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, получив ваше сообщение о ней. Сообщив об ошибке, вы, быть может, убережете другого читателя от многочасового разочарования и в то же время поможете нам, предоставив информацию еще более высокого качества.

Для поиска страницы с ошибками, найденными в этой книге, перейдите на Web-сайт **http://www.wrox.com** и найдите заголовок с помощью поля **Search** (Поиск) или одного из списков заголовков. Далее на странице с выходными данными книги щелкните кнопкой мыши ссылку **Errata** (Ошибки). Вы попадете на страницу, отображающую все ошибки, представленные на рассмотрение и опубликованные редакторами издательства Wrox. На Web-странице **www.wrox.com/misc-pages /booklist.shtml** можно найти полный список книг, включающий ссылки на ошибки, найденные в каждой книге.

Если вы не обнаружили "свою" ошибку на странице **Errata** (Ошибки), перейдите на страницу **www.wrox.com/contact/techsupport.shtml** и заполните форму для отправки нам найденной вами ошибки. Мы проверим присланную информацию и, если согласимся с ней, опубликуем сообщение на странице с ошибками, найденными в книге, и исправим ее в последующих изданиях книги.

Сайт *p2p.wrox.com*

Для обмена мнениями с авторами и такими же, как вы, читателями присоединяйтесь к форумам P2P (Programmer to Programmer) на Web-сайте **p2p.wrox.com**. Форумы — это система на основе Web-технологии, предназначенная для отправки вашего сообщения, относящегося к книгам издательства Wrox и родственным технологиям, и обмена мнениями с другими читателями и пользователями этих техно-

логий. Форумы предлагают функцию подписки для отправки вам по электронной почте по мере поступления новых сообщений, относящихся к выбранным вами и интересующих вас темам. На этих форумах представлены авторы и редакторы Wrox и другие специалисты, работающие в области информационных технологий.

На Web-сайте **http://p2p.wrox.com** вы найдете ряд разных форумов, которые помогут вам не только во время чтения книги, но и в процессе разработки ваших собственных приложений. Для присоединения к форумам выполните следующие действия:

1. Перейдите на Web-сайт **p2p.wrox.com** и щелкните кнопкой мыши ссылку **Register** (Зарегистрироваться).
2. Прочтите условия пользования и щелкните мышью кнопку **Agree** (Принять).
3. Введите необходимую для присоединения к форуму информацию и любую необязательную информацию, которую хотите предоставить, и щелкните мышью кнопку **Submit** (Отправить).
4. Вы получите по электронной почте сообщение со сведениями, описывающими, как проверить вашу учетную запись и завершить процесс присоединения к форуму.

> ***Примечание***
>
> Читать сообщения на форумах вы сможете и без регистрации в P2P, но для того чтобы отправлять собственные сообщения, придется зарегистрироваться.

После присоединения к форуму вы можете посылать новые сообщения и отвечать на сообщения, посланные другими пользователями. Читать сообщения можно будет в любое время, находясь в Web-пространстве. Если вы хотите получать по электронной почте новые сообщения, появляющиеся на конкретном форуме, щелкните мышью пиктограмму **Subscribe to this Forum** (Подписаться на этот форум), расположенную рядом с именем форума в списке форумов.

Для получения дополнительной информации о правилах использования системы Wrox P2P непременно прочтите P2P FAQ (часто задаваемые вопросы) и получите ответы о работе программного обеспечения форумов и ответы на общие вопросы, касающиеся P2P и книг издательства Wrox. Для чтения этих вопросов и ответов щелкните мышью ссылку **FAQ** на любой странице P2P.

1

Приступая к работе

В этой главе вы узнаете, что такое ОС Linux и как она связана со своим прообразом — ОС UNIX, познакомитесь с функциями и средствами, предоставляемыми средой разработки программ в ОС Linux, и напишите и выполните свою первую программу. Попутно вы получите представление о:

- UNIX, Linux и проекте GNU;
- программах и языках программирования в ОС Linux;
- способах поиска ресурсов разработки;
- статических и совместно используемых библиотеках;
- теоретических основах ОС UNIX.

Введение в UNIX, Linux и проект GNU

В последние годы ОС Linux стала заметным явлением. И дня не проходит без того или иного упоминания Linux в электронных средствах массовой информации. Мы потеряли счет приложениям, которые стали доступны в ОС Linux и внедрившим ее организациям, включая некоторые министерства и городские администрации. Основные поставщики компьютерного оборудования, такие как корпорации IBM и Dell, поддерживают ОС Linux, а крупнейшие разработчики программного обеспечения, например компания Oracle, обеспечивают выполнение своих программ в ОС Linux. Linux стала по-настоящему конкурентоспособной операционной системой, особенно на серверном рынке.

Своим успехом она обязана системам и приложениям — предшественникам: ОС UNIX и программному обеспечению GNU. В этом разделе рассматривается, как появилась ОС Linux и каковы ее корни.

Что такое ОС UNIX?

Операционная система UNIX первоначально была разработана в компании Bell Laboratories, бывшей в то время частью телекоммуникационного гиганта, компании AT&T. Разработанная в 1970-х гг. для мини-компьютеров PDP корпорации Digital Equipment ОС UNIX стала очень популярной многопользовательской, многозадачной операционной системой для самых разных аппаратных платформ, начиная с рабочих станций PC и заканчивая многопроцессорными серверами и суперкомпьютерами.

Краткая история ОС UNIX

Строго говоря, UNIX — это торговое название, контролируемое организацией Open Group и относящееся к компьютерной операционной системе, соответствующей определенной спецификации. В этой спецификации, именуемой "The Single UNIX Specification" ("Единая спецификация UNIX"), определены имена, интерфейсы и поведение всех обязательных функций операционной системы UNIX. Данная спецификация в значительной степени представляет собой расширенный набор более ранних спецификаций, стандартов P1003 или POSIX (Portable Operating System Interface, интерфейс переносимой операционной системы), разработанных IEEE (Institute of Electrical and Electronic Engineers, Институт инженеров по электротехнике и радиоэлектронике).

Существует много коммерческих UNIX-подобных систем, таких как AIX корпорации IBM, UX компании HP и Solaris компании Sun Microsystems. Некоторые системы, например FreeBSD и Linux, свободно распространяются. В настоящее время спецификации Open Group удовлетворяют лишь несколько операционных систем, что позволяет предлагать их на рынке с названием UNIX.

В прошлом совместимость разных систем UNIX была реальной проблемой, хотя стандарт POSIX и оказывал неоценимую помощь в ее решении. В наши дни следование нескольким простым правилам сделало возможным создание приложений, работающих под управлением всех UNIX и UNIX-подобных систем. Более подробную информацию о стандартах ОС Linux и UNIX вы сможете найти в *главе 18*.

Идеология UNIX

В последующих главах мы надеемся представить особенности программирования в ОС Linux (а следовательно, и в UNIX). Несмотря на то, что в большинстве своем программирование на языке C одинаково на разных платформах, у разработчиков Linux и UNIX есть свой взгляд на разработку программ и операционных систем.

В операционной системе UNIX, а значит и в Linux, поощряется определенный стиль программирования. Далее перечислены некоторые характеристики, общие для типовых программ и систем UNIX.

- **Простота.** Многие из наиболее полезных утилит UNIX очень просты и как результат малы и понятны. KISS (Keep It Small and Simple, сохраняйте программу маленькой и простой) — отличный подход, которому следует научиться. Чем больше и сложнее система, тем наверняка в ней больше сложных ошибок, и отладка превращается в тяжелую работу, которой хотелось бы избежать.

- **Узкая направленность.** Зачастую лучше сделать программу, хорошо выполняющую одну задачу, чем включать в каждую функцию полный набор нужного и ненужного. "Раздутую" программу трудно использовать и поддерживать ее работоспособность. Одноцелевые программы легче усовершенствовать при появлении улучшенных алгоритмов или интерфейсов. В ОС UNIX при необходимости выполнения трудных задач чаще комбинируются маленькие утилиты, чем делается попытка в одной большой программе предусмотреть все потребности пользователя.
- **Многократно используемые компоненты.** Превращайте ядро вашего приложения в доступную библиотеку. Хорошо документированные библиотеки с простыми, но гибкими программными интерфейсами могут помочь другим пользователям разрабатывать различные варианты или применять методы в новых сферах приложения. К примерам можно отнести библиотеку базы данных dbm, представляющую собой пакет функций многократного использования, а не единую программу управления базой данных.
- **Фильтры.** Многие приложения UNIX могут применяться как фильтры. Они преобразуют свой ввод и формируют вывод. Как вы увидите в дальнейшем, ОС UNIX обладает функциональными возможностями, позволяющими разрабатывать очень сложные приложения из других UNIX-программ путем комбинирования их оригинальными способами. Конечно, подобное многократное использование возможно благодаря методам разработки, упоминавшимся ранее.
- **Открытые файловые форматы.** Наиболее удачные и популярные UNIX-программы применяют файлы конфигурации и файлы данных в виде обычного текста ASCII или файла на языке XML. Если в разрабатываемой вами программе можно использовать любой из этих форматов — это хороший выбор. Он позволит другим пользователям применить стандартные средства при изменении или поиске элементов конфигурации и разрабатывать новые средства для выполнения новых функций обработки файлов данных. Хорошим примером такого подхода может служить система перекрестных ссылок исходного кода ctags, записывающая сведения о местоположении символа в виде регулярного выражения, подходящего для использования программами поиска.
- **Гибкость.** Вы не можете точно предусмотреть заранее, как изобретательные пользователи будут применять вашу программу. Программируя, попытайтесь быть настолько гибким, насколько это возможно. Старайтесь избегать любых ограничений размеров полей или числа записей. Если можно, пишите программу в расчете на применение в сети, способную выполняться одинаково хорошо при сетевом вызове и на локальной машине. Никогда не думайте, что вы знаете все о потребностях будущего пользователя.

Что такое Linux?

Как вы уже, возможно, знаете, Linux — это свободно распространяемая реализация UNIX-подобного ядра, низкоуровневой сердцевины операционной системы. Поскольку прообразом ОС Linux стала система UNIX, Linux- и UNIX-программы

очень похожи. В действительности почти все программы, написанные для ОС UNIX, могут быть скомпилированы и выполнены в ОС Linux. Кроме того, некоторые коммерческие приложения, продаваемые для коммерческих версий UNIX, могут выполняться без изменения их двоичного кода в системах под управлением Linux.

ОС Linux была разработана Линусом Торвальдсом (Linus Torvalds) из Университета г. Хельсинки совместно с программистами UNIX, оказывавшими ему помощь по Интернету. Работа начиналась как хобби, а вдохновителем стала ОС Minix Энди Таненбаума (Andy Tanenbaum), маленькая UNIX-подобная система. Со временем Linux выросла, превратившись в сложную самостоятельную систему. Ее цель — отказ от патентованного кода и применение только свободно распространяемого программного кода.

В настоящее время ОС Linux существует для широкого набора компьютерных систем с разными типами процессоров, включая PC на 16- и 32-битных процессорах Intel *x*86 и совместимых с ними процессорах; рабочие станции и серверы на процессорах Sun SPARC, IBM PowerPC, AMD Opteron и Intel Itanium и даже некоторые карманные компьютеры PDA и игровые приставки Playstation 2 и 3 фирмы Sony. Если у устройства есть процессор, кто-то где-нибудь пытается добыть ОС Linux, выполняющуюся на этом процессоре!

Проект GNU и Фонд свободного ПО

ОС Linux обязана своим существованием совместным усилиям множества людей. Ядро операционной системы само по себе образует лишь малую часть пригодной к использованию системы разработки. Коммерческие системы UNIX традиционно снабжаются приложениями, обеспечивающими системные сервисы и средства. Для систем Linux подобные дополнительные программы написаны множеством разных программистов и распространяются они свободно.

Linux-сообщество (совместно с другими людьми) поддерживает идею свободного программного обеспечения (ПО), т. е. свободного от ограничений и подчиняющегося Общедоступной лицензии проекта GNU (GNU General Public License, GPL). (GNU означает GNU's Not UNIX (GNU не UNIX).) Несмотря на то, что получение программного обеспечения может быть небесплатным, это ПО может использоваться как угодно и обычно распространяется в виде исходного программного кода.

Фонд свободного программного обеспечения (Free Software Foundation) был организован Ричардом Столлменом (Richard Stallman) — автором GNU Emacs, одного из самых известных текстовых редакторов для ОС UNIX и других систем. Столлмен — автор концепции свободного программного обеспечения и организатор проекта GNU, попытки создания операционной системы и среды разработки, совместимой с ОС UNIX, но не подверженной ограничениям, связанным с торговой маркой UNIX и предоставлением исходного программного кода. В любой момент может оказаться, что проект GNU сильно отличается от UNIX способами поддержки аппаратных средств и управления исполняемыми программами, но он будет продолжать поддерживать приложения в стиле UNIX.

Проект GNU уже снабдил программистское сообщество множеством приложений, сильно напоминающих компоненты, входящие в системы UNIX. Все эти программы, называемые программным обеспечением GNU, распространяются в соответствии с Общедоступной лицензией GNU (GPL), копию которой можно найти на сайте **http://www.gnu.org**. В этой лицензии вводится понятие "авторского “лева” (copyleft)" (в противоположность авторскому праву ("copyright")). Авторское "лево" задумано как препятствие установлению каких-либо ограничений на использование свободного программного обеспечения.

Далее приведены основные примеры ПО проекта GNU, распространяемого в соответствии с лицензией GPL:

- пакет компиляторов GCC (GNU Compiler Collection), включающий компилятор GNU C;
- G++ — компилятор C++, включающий как часть GCC;
- GDB — отладчик на уровне исходного кода;
- GNU make — версия UNIX-автосборщика make;
- Bison — генератор синтаксических анализаторов, совместимый с генератором компиляторов UNIX yacc;
- bash — командная оболочка;
- GNU Emacs — текстовый редактор и среда разработки.

Кроме того, было разработано и распространено на принципах свободного ПО и под контролем лицензии GPL множество других пакетов, включая электронные таблицы, средства управления программным кодом, компиляторы, интерпретаторы, интернет-средства, программы обработки графических объектов, например, графический редактор Gimp и две законченные объектно-ориентированные среды разработки: GNOME и KDE. Мы обсудим GNOME и KDE в *главах 16* и *17*.

Сейчас существует такое множество доступного свободного программного обеспечения, что если добавить к нему ядро Linux, можно сказать, что благодаря Linux достигнута основная цель проекта GNU — свободная UNIX-подобная система. Признавая вклад программного обеспечения проекта GNU, многие люди теперь, как правило, называют Linux-системы GNU/Linux.

Более подробную информацию о концепции свободного программного обеспечения можно получить на Web-сайте **http://www.gnu.org**.

Дистрибутивы Linux

Как мы уже упоминали, Linux — это только ядро. Вы можете получить исходный программный код ядра, откомпилировать его и установить на машину, а затем получить и установить много другого свободного программного обеспечения для завершения установки ОС Linux. Такие установки часто называют *системами Linux*, т. к. они содержат много программ помимо ядра. Большинство утилит приходит от проекта GNU Фонда свободного ПО.

Понятно, что создание системы Linux только из исходного программного кода — трудное дело. К счастью, многие люди подготовили готовые к установке дистрибутивы (часто называемые разновидностями (flavor)), обычно загружаемые из Интернета или с CD/DVD-накопителей и содержащие не только ядро, но и множество других программных средств и утилит. Часто в их состав входит реализация X Window System — графической оболочки, общей для множества систем UNIX. Дистрибутивы обычно снабжаются программой установки и дополнительной документацией (как правило, все на компакт-дисках), чтобы помочь вам установить собственную систему Linux. К некоторым хорошо известным дистрибутивам, в особенности для семейства процессоров Intel *x*86, относятся дистрибутивы Red Hat Enterprise Linux и его усовершенствованный сообществом родственник Fedora, Novell SUSE Linux и свободно распространяемый вариант openSUSE, Ubuntu Linux, Slackware, Gentoo и Debian GNU/Linux. Подробную информацию о множестве других дистрибутивов можно найти на Web-сайте DistroWatch по адресу **http://distrowatch.com**.

Программирование в ОС Linux

Многие думают, что программирование в Linux означает применение языка программирования C. Известно, что ОС UNIX первоначально была написана на C и что большинство UNIX-приложений были написаны на языке C. Но для программистов ОС Linux, или UNIX, C — не единственно возможный вариант. Далее в книге мы назовем пару альтернатив.

> ***Примечание***
>
> На самом деле первая версия UNIX была написана в 1969 г. на ассемблере PDP 7. Язык C был задуман Деннисом Ритчи (Dennis Ritchie) примерно в это время, и в 1973 г. он вместе с Кеном Томпсоном (Ken Tompson) по существу переписал на C все ядро UNIX, совершив настоящий подвиг в эпоху разработки системного программного обеспечения на языке ассемблера.

В системах Linux доступен широкий диапазон языков программирования, многие из них свободно распространяются и есть на компакт-дисках или в архивах на FTP-сайтах в Интернете. Далее перечислена часть языков программирования, доступных программистам Linux:

- ❒ Ada;
- ❒ C;
- ❒ C++;
- ❒ Eiffel;
- ❒ Forth;
- ❒ Fortran;
- ❒ Icon;
- ❒ Java;
- ❒ JavaScript;
- ❒ Lisp;
- ❒ Modula 2;
- ❒ Modula 3;
- ❒ Oberon;
- ❒ Objective C;
- ❒ Pascal;
- ❒ Perl;
- ❒ PostScript;
- ❒ Prolog;
- ❒ Python;
- ❒ Ruby;
- ❒ Smalltalk;
- ❒ PHP;
- ❒ Tcl/Tk;
- ❒ Bourne Shell.

В *главе 2* мы покажем, как применять оболочку Linux для разработки приложений малого и среднего размера. В оставшейся части книги мы сконцентрируемся главным образом на языке C и уделим основное внимание изучению программных интерфейсов ОС Linux с точки зрения программиста, поэтому мы рассчитываем на знание читателей языка программирования C.

Linux-программы

Linux-приложения представлены файлами двух типов: исполняемыми (executable) и сценариями или пакетными файлами (script). Исполняемые файлы — это программы, которые могут непосредственно выполняться на компьютере; они соответствуют файлам ОС Windows с расширением exe. Сценарии или пакетные файлы — это наборы команд для выполнения другой программой, интерпретатором. Они соответствуют в ОС Windows файлам с расширением bat или cmd или интерпретируемым программам на языке Basic.

ОС Linux не требует, чтобы исполняемые или пакетные файлы имели определенные имена или какие-либо расширения. Для обозначения файла как способной выполняться программы применяются атрибуты файловой системы, которые будут обсуждаться в *главе 2*. В ОС Linux вы можете заменять пакетные файлы откомпилированными программами (и наоборот), не оказывая влияния на другие программы или пользователей, которые обращаются к ним. На уровне пользователя, по сути, между ними нет разницы.

В процессе регистрации в системе Linux вы взаимодействуете с программой командной оболочки (часто bash), которая запускает программы так же, как это делает оболочка командной строки в ОС Windows. Она находит запрашиваемые вами программы по имени, выполняя поиск файла с тем же именем в заданном наборе каталогов. Каталоги, предназначенные для поиска, хранятся в переменной оболочки `PATH`, так же как в ОС Windows. Путь поиска (который вы можете пополнять) настраивается вашим системным администратором и обычно содержит стандартные каталоги, в которых сохраняются системные программы. К ним относятся:

- /bin — бинарные файлы (binaries), программы, применяемые для загрузки системы;
- /usr/bin — пользовательские библиотеки, стандартные программы, доступные пользователям;
- /usr/local/bin — локальные библиотеки, программы, относящиеся к этапу инициализации.

Если войти в систему как администратор, например с именем `root`, можно использовать переменную `PATH`, которая включает каталоги с хранящимися системными программами, такие как /sbin и /usr/sbin.

Необязательные компоненты операционной системы и приложения сторонних производителей могут устанавливаться в подкаталоги /opt, а добавить инсталляционные программы в вашу переменную `PATH` можно через пользовательские инсталляционные сценарии.

Примечание

Не стоит удалять каталоги из переменной PATH, пока нет полной уверенности в результате, который будет получен.

Обратите внимание на то, что в ОС Linux, как и UNIX, для разделения отдельных элементов в переменной PATH применяется символ двоеточия (:) в отличие от символа точки с запятой, используемого в ОС MS-DOS и Windows. (ОС UNIX сделала выбор первой, поэтому спрашивайте, почему отличается Windows, а не почему в UNIX все не так!) Далее приведен пример переменной PATH:

```
/usr/local/bin:/bin:/usr/bin:.:/home/neil/bin:/usr/X11R6/bin
```

В этой переменной PATH содержатся каталоги для хранения стандартных программ, текущий каталог (.), исходный каталог пользователя и каталог графической оболочки X Window System.

Запомните, в ОС Linux используется прямой слэш (/) для отделения имен каталогов в полном имени файла в отличие от обратного слэша (\), применяемого в ОС Windows. И снова ОС UNIX выбирала первой.

Текстовые редакторы

Для ввода и набора примеров программного кода, приведенных в книге, вам понадобится текстовый редактор. В типовых системах Linux есть большой выбор таких программ. У многих пользователей популярен редактор vi.

Оба автора предпочитают Emacs, поэтому мы предлагаем потратить немного времени на знакомство с основными функциями этого мощного редактора. Почти во все дистрибутивы ОС Linux Emacs включен как необязательный пакет, который можно установить. Кроме того, вы можете получить его на Web-сайте GNU по адресу **http://www.gnu.org** или же взять версию для графических сред разработки на Web-сайте XEmacs по адресу **http://www.xemacs.org**.

Для того чтобы узнать больше о редакторе Emacs, можно воспользоваться его интерактивным средством обучения. Начните с выполнения команды emacs, затем нажмите комбинацию клавиш <Ctrl>+<H> с последующим вводом символа t для доступа к этому средству. У редактора Emacs есть также полное руководство. Для получения дополнительной информации о нем в редакторе Emacs нажмите комбинацию клавиш <Ctrl>+<H> с последующим вводом символа i. В некоторых версиях Emacs может быть меню, предоставляющее доступ к средству обучения и полному руководству.

Компилятор языка C

В системах, соответствующих стандарту POSIX, компилятор языка C называется c89. Раньше компилятор языка C назывался просто cc. Шли годы, разные поставщики продавали UNIX-подобные системы с компиляторами C, обладающими разными функциями и параметрами, но очень часто все также названными cc.

Когда создавался стандарт POSIX, выяснилось, что невозможно определить стандартную команду cc, которая была бы совместима со всеми этими разработками.

Вместо этого комитет решил создать новую стандартную команду для компилятора языка C — `c89`. Если эта команда представлена, она всегда использует одни и те же опции независимо от машины.

В системах Linux, которые на деле пытаются следовать стандартам, можно обнаружить, что все или некоторые из команд `c89`, `cc` и `gcc` ссылаются на системный компилятор языка C, обычно компилятор GNU C или gcc. В системах UNIX компилятор языка C почти всегда называется cc.

В этой книге мы используем gcc, поскольку он поставляется в дистрибутивах Linux и потому что он поддерживает для языка C синтаксис стандарта ANSI. Если когда-нибудь вы обнаружите, что в вашей системе нет gcc, мы советуем получить его и установить. Найти его вы можете по адресу **http://www.gnu.org**. Всюду, где мы используем в книге команду `gcc`, просто заменяйте ее подходящей командой вашей системы.

Упражнение 1.1. Ваша первая Linux-программа на языке C

В этом примере вы начнете разработку в ОС Linux с помощью языка C, написав, откомпилировав и выполнив свою первую Linux-программу. Ею, кстати, может стать самая известная из всех программ для начинающих — программа, выводящая сообщение "Hello World" ("Привет, мир").

1. Далее приводится текст файла hello.c:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  printf("Hello World\n");
  exit(0);
}
```

2. Теперь откомпилируйте, скомпонуйте и выполните вашу программу.

```
$ gcc –o hello.c
$ ./hello
Hello World
$
```

Как это работает

Вы запустили компилятор GNU C (в Linux, вероятнее всего, он будет доступен и как `cc`), который оттранслировал исходный код на языке C в исполняемый файл, названный hello. Вы выполнили программу, и она вывела на экран приветствие. Это наипростейший из существующих примеров, но если вы смогли с помощью вашей системы добраться до этого места, то сможете откомпилировать и выполнить и остальные примеры из книги. Если же программа не сработала, убедитесь в том, что в вашей системе установлен компилятор языка C. Например, во многих

дистрибутивах Linux есть установочная опция, названная **Software Development** (Разработка ПО) (или что-то похожее), которую следует выбрать для установки необходимых пакетов.

Поскольку это первая выполненная вами программа, самое время обратить внимание на некоторые основные положения. Программа hello, вероятно, должна быть в вашем исходном каталоге. Если в переменную `PATH` не включена ссылка на ваш исходный каталог, оболочка не сможет найти программу hello. Более того, если один из каталогов в переменной `PATH` содержит другую программу, названную hello, вместо вашей будет выполнена эта программа. То же самое произойдет, если такой каталог упомянут в переменной `PATH` раньше вашего исходного каталога. Для решения этой потенциальной проблемы можно снабдить имена программ префиксом `./` (например, `./hello`). Данный префикс сообщает оболочке о необходимости выполнить программу с заданным именем, находящуюся в текущем каталоге. (Точка — это условное название текущего каталога.)

Если вы забыли опцию `-o name`, которая указывает компилятору, куда поместить исполняемый файл, компилятор поместит его в файл с именем a.out (что означает ассемблерный вывод). Не забудьте поискать файл с именем a.out, если вы уверены, что скомпилировали программу, а найти ее не можете! Когда ОС UNIX только появилась, пользователи, хотевшие играть в ней в игры, часто запускали их как файл с именем a.out, чтобы не быть пойманными системным администратором, и некоторые установки ОС UNIX традиционно удаляют каждый вечер все файлы с именем a.out.

Маршрутная карта системы разработки

Разработчику ОС Linux важно знать кое-что о том, где размещаются средства и ресурсы разработки. В следующих разделах дан краткий обзор некоторых важных каталогов и файлов.

Приложения

Приложения обычно хранятся в отведенных для них каталогах. Приложения, предоставляемые системой для общего использования, включая средства разработки программ, находятся в каталоге /usr/bin. Приложения, добавленные системными администраторами для конкретного хост-компьютера или локальной сети, часто хранятся в каталоге /usr/local/bin или /opt.

Администраторы предпочитают /opt и /usr/local, потому что они хранят предоставляемые конкретными поставщиками файлы и свежие дополнения отдельно от приложений, предоставляемых системой. Подобная организация хранения файлов может помочь, когда придет время обновлять операционную систему, т. к. в этом случае потребуется сберечь только каталоги /opt и /usr/local. Мы рекомендуем компилировать в ветви иерархии /usr/local только системные приложения общего назначения, предназначенные для запуска и доступа к требуемым файлам. Для разрабатываемых программ и личных приложений лучше всего применять папку в вашем исходном каталоге.

Дополнительные средства и системы программирования могут иметь собственные структуры каталогов и каталоги программ. Важнейшая среди них — графическая оболочка X Window System, которая обычно устанавливается в каталог /usr/X11 или каталог /usr/bin/X11. В дистрибутивах Linux, как правило, применяется версия X.Org Foundation графической оболочки X Window System, базирующаяся на модификации Revision 7 (X11R7). В других UNIX-подобных системах могут быть выбраны иные версии X Window System, устанавливаемые в другие каталоги, например, каталог /usr/openwin для оболочки Open Windows компании Sun в системе Solaris.

Программа системного драйвера компилятора GNU, gcc (которую вы использовали в предыдущем упражнении) обычно помещается в каталог usr/bin или usr/local/bin, но она будет запускать различные поддерживающие компиляцию приложения из других каталогов. Эти каталоги задаются во время компиляции самого компилятора и зависят от типа хост-компьютера. В системах Linux это может быть зависящий от конкретной версии подкаталог /usr/lib/gcc/. На одной из машин одного из авторов во время написания книги это был подкаталог /usr/lib/gcc/i586-suse-linux/4.1.3. В нем хранятся отдельные проходы компилятора GNU C/C++ и специфические заголовочные файлы GNU.

Заголовочные файлы

В процессе программирования на языке C и других языках вам потребуются заголовочные файлы или файлы заголовков для включения определений констант и объявлений вызовов системных и библиотечных функций. В случае языка C эти файлы почти всегда находятся в каталоге /usr/include и его подкаталогах. Заголовочные файлы, зависящие от конкретного воплощения запущенной вами ОС Linux, вы, как правило, найдете в каталогах /usr/include/sys и /usr/include/linux.

У других систем программирования тоже есть заголовочные файлы, хранящиеся в каталогах, которые автоматически находятся соответствующим компилятором. Примерами могут служить каталоги /usr/include/X11 для графической оболочки X Window System и /usr/include/c++ для языка GNU C++.

Вы можете использовать заголовочные файлы из подкаталогов или нестандартных мест хранения, указав флаг `-I` (для include) в строке вызова компилятора языка C. Например, команда

```
$ gcc -I/usr/openwin/include fred.c
```

заставит искать заголовочные файлы, использованные в программе fred.c, в стандартных каталогах и в каталоге /usr/openwin/include. Для получения дополнительных сведений обратитесь к руководству компилятора C (`man gcc`).

Искать заголовочные файлы с конкретными определениями и прототипами конкретных функций часто удобно с помощью команды `grep`. Предположим, вам нужно знать имя из директив `#define`, используемое для возврата из программы статуса завершения. Просто замените каталог на /usr/include и примените `grep` для поиска предполагаемой части имени следующим образом:

```
$ grep EXIT_ *.h
...
stdlib.h#define    EXIT_FAILURE    1    /*Failing exit status. */
stdlib.h#define    EXIT_SUCCESS    0    /*Successful exit status. */
...
$
```

В этом случае команда `grep` ищет в каталоге все файлы с именами, заканчивающимися на .h, со строкой `EXIT_`. В данном примере она нашла (среди прочих) нужное вам определение в файле stdlib.h.

Библиотечные файлы

Библиотеки — это наборы заранее откомпилированных функций, написанных в расчете на многократное использование. Обычно они состоят из наборов связанных функций, предназначенных для решения общей задачи. Примерами могут служить библиотеки функций работы с экраном (библиотеки curses и ncurses) и процедуры доступа к базе данных (библиотека dbm). В последующих главах мы познакомим вас с некоторыми библиотеками.

Стандартные системные библиотеки обычно хранятся в каталогах /lib и /usr/lib. Компилятору языка C (или, точнее, компоновщику) необходимо сообщить, в каких библиотеках искать, поскольку по умолчанию он ищет только в стандартной библиотеке C. Это пережиток, пришедший к нам из того времени, когда компьютеры были медленными и циклы ЦПУ были дороги. Недостаточно поместить библиотеку в стандартный каталог и ждать, что компилятор найдет ее; библиотеки должны следовать очень специфическим правилам именования и быть упомянуты в командной строке.

Имя файла библиотеки всегда начинается с символов *lib*. Далее следует часть, указывающая на назначение библиотеки (например, *c* для библиотеки C или *m* для математической библиотеки). Последняя часть имени начинается с точки (.) и задает тип библиотеки:

- .a — для традиционных статических библиотек;
- .so — для совместно используемых библиотек *(см. далее)*.

Обычно библиотеки существуют в статическом и совместно используемом форматах, как покажет быстрый просмотр каталога командой `ls /usr/lib`. Вы можете заставить компилятор искать библиотеку, задав полное имя ее файла или применив флаг `-l`. Например, команда

```
$ gcc -o fred fred.c /usr/lib/libm.a
```

сообщает компилятору о необходимости компилировать файл fred.c и искать разрешения ссылок на функции в библиотеке математических функций в дополнение к стандартной библиотеке C. Аналогичного результата можно добиться с помощью следующей команды:

```
$ gcc -o fred fred.c -lm
```

`-lm` (без пробела между символами `l` и `m`) — это сокращенное обозначение (сокращенные формы очень ценятся в UNIX-кругах) библиотеки с именем libm.a, хранящейся в одном из стандартных библиотечных каталогов (в данном случае /usr/lib). Дополнительное преимущество обозначения `-lm` в том, что компилятор автоматически выберет совместно используемую библиотеку, если она существует.

Несмотря на то что библиотеки, как и заголовочные файлы, обычно хранятся в стандартных каталогах, вы можете добавить каталоги для поиска, указав флаг `-L` (заглавная буква) в команде вызова компилятора. Например, команда

```
$ gcc -o x11fred -L/usr/openwin/lib x11fred.c -lX11
```

будет компилировать и компоновать программу x11fred, используя версию библиотеки libX11, найденную в каталоге /usr/openwin/lib.

Статические библиотеки

Простейшая форма библиотеки — это коллекция объектных файлов, хранящихся вместе в виде, готовом к использованию. Когда программе нужна функция, содержащаяся в библиотеке, в нее включают заголовочный файл, объявляющий эту функцию. За соединение программного кода и библиотеки в единый исполняемый файл отвечают компилятор и компоновщик. Вы только должны применить опцию `-l` для указания нужных библиотек, отличных от стандартной библиотеки С исполняющей системы.

Статические библиотеки, также называемые архивами, в соответствии с принятыми соглашениями имеют окончание .a. Например, lib/libc.a и /usr/lib/libX11 для библиотек С и X11 соответственно.

Вы можете очень легко создавать и поддерживать собственные статические библиотеки с помощью программы ar (для создания архивов) и отдельно компилировать функции с помощью команды `gcc -c`. Старайтесь, насколько это возможно, хранить функции в отдельных исходных файлах. Если функциям нужен доступ к общим данным, вы можете поместить данные в один исходный файл и использовать статические переменные, объявленные в этом файле.

Упражнение 1.2. Статические библиотеки

В этом упражнении вы создадите свою маленькую библиотеку, содержащую две функции, и затем используете одну из функций в примере программы. Функции называются `fred` и `bill` и просто выводят приветствия.

1. Сначала создайте отдельные исходные файлы (как не удивительно, названные fred.c и bill.c) для каждой функции.

   Далее приведен первый из них:

   ```
   #include <stdio.h>

   void fred(int arg)
   {
     printf("fred: you passed %d\n", arg);
   }
   ```

А это второй:

```
#include <stdio.h>

void bill(char *arg)
{
  printf("bill: you passed %s\n", arg);
}
```

2. Вы можете отдельно откомпилировать эти функции и создать объектные файлы, готовые к включению в библиотеку. Для этого запустите компилятор C с опцией `-c`, которая помешает компилятору создать законченную программу. Попытка создать законченную программу окажется безуспешной, т. к. вы не определили функцию с именем `main`.

```
$ gcc -c bill.c fred.c
$ ls *.o
bill.o fred.o
```

3. Теперь напишите программу, вызывающую функцию `bill`. Прежде всего, хорошо бы создать заголовочный файл для вашей библиотеки. В нем будут объявлены функции из вашей библиотеки, и он будет включаться во все приложения, которые захотят применить вашу библиотеку. В файлы fred.c и bill.c тоже хорошо бы включить заголовочный файл, чтобы помочь компилятору обнаружить любые ошибки.

```
/*
Это файл lib.h. В нем объявлены пользовательские функции fred and bill
*/[1]

void bill(char *);
void fred(int);
```

4. Вызывающая программа (program.c) может быть очень простой. Она включает заголовочный файл и вызов из библиотеки одной из функций.

```
#include <stdlib.h>
#include "lib.h"

int main()
{
  bill("Hello World");
  exit(0);
}
```

5. Теперь можно откомпилировать и протестировать программу. Для этого задайте компилятору явно объектные файлы и попросите его откомпилировать ваш файл и связать его с ранее откомпилированным объектным модулем bill.o.

---

[1] Здесь и далее для удобства читателей комментарии переведены на русский язык. Возможность ввода знаков кириллицы зависит от выбранного дистрибутива Linux и текстового редактора. — *Пер.*

```
$ gcc -c program.c
$ gcc -o program program.o bill.o
$ ./program
bill: we passed Hello World
$
```

6. Для создания архива и включения в него ваших объектных файлов используйте программу ar. Программа называется ar, поскольку она создает архивы или коллекции отдельных файлов, помещая их все в один большой файл. Имейте в виду, что программу ar можно применять для создания архивов из файлов любого типа. (Как многие утилиты UNIX, ar — универсальное средство.)

```
$ ar crv libfoo.a bill.o fred.o
a - bill.o
a - fred.o
```

7. Библиотека создана, и в нее добавлены два объектных файла. Для того чтобы успешно применять библиотеку в некоторых системах, в особенности в производных от Berkeley UNIX, требуется создать для библиотеки индекс содержимого архива или список вложенных в библиотеку функций и переменных (table of contents). Сделайте это с помощью команды `ranlib`. В ОС Linux при использовании программных средств разработки GNU этот шаг не является необходимым (но и не приносит вреда).

```
$ ranlib libfoo.a
```

Теперь ваша библиотека готова к использованию. Вы можете добавить следующий список файлов, которые должен обработать компилятор для создания вашей программы:

```
$ gcc -o program program.o libfoo.a
$ ./program
bill: we passed Hello world
$
```

Можно было бы применить для доступа к библиотеке флаг `-l`, но т. к. она хранится не в одном из стандартных каталогов, вы должны сообщить компилятору место поиска с помощью флага `-L` следующим образом:

```
$ gcc -o program program.o -L. -lfoo
```

Опция `-L` заставляет компилятор искать библиотеки в текущем каталоге (`.`). Опция `–lfoo` сообщает компилятору, что нужно использовать библиотеку с именем libfoo.a (или совместно используемую библиотеку libfoo.so, если она есть). Для того чтобы посмотреть, какие функции включены в объектный файл, библиотеку или исполняемую программу, можно применить команду `nm`. Если вы взглянете на файлы program и libfoo.a, то увидите, что библиотека содержит обе функции: `fred` и `bill`, а файл program — только функцию `bill`. Когда создается программа, в нее включаются из библиотеки только те функции, которые ей действительно нужны. Вставка

заголовочного файла, содержащего объявления всех функций библиотеки, не вызывает включения в конечную программу целиком всей библиотеки.

Если вы знакомы с разработкой программ в ОС Windows, то поймете, что в ОС UNIX существует ряд прямых аналогий, перечисленных в табл. 1.1.

Таблица 1.1

Элемент	UNIX	Windows
Объектный модуль	func.o	FUNC.OBJ
Статическая библиотека	lib.a	LIB.LIB
Программа	program	PROGRAM.EXE

Совместно используемые библиотеки

У статических библиотек один недостаток — когда вы запускаете много приложений одновременно и все они используют функции из одной библиотеки, в результате образуется множество копий одних и тех же функций в памяти и множество реальных копий функций в самих файлах программ. Это может привести к потреблению большого объема полезной памяти и дискового пространства.

Устранить этот недостаток могут многие системы UNIX и Linux с поддержкой совместно используемых или разделяемых библиотек. Подробное обсуждение совместно используемых библиотек и их реализации в разных ОС не входило в нашу задачу, поэтому ограничимся только реализацией их в ОС Linux.

Совместно используемые библиотеки хранятся в тех же каталогах, что и статические, но у имен файлов совместно используемых библиотек другой суффикс. В типовой системе Linux имя совместно используемой версии стандартной библиотеки математических функций — /lib/libm.so.

Когда программа применяет совместно используемую библиотеку, она компонуется таким образом, что содержит не код функции как таковой, а ссылки на совместно используемый код, который станет доступен на этапе выполнения. Когда окончательная программа загружается в память для выполнения, ссылки на функции разрешаются, и выполняются вызовы совместно используемой библиотеки, которая будет загружаться в память по мере необходимости.

В этом случае система предоставляет возможность многим приложениям одновременно использовать единственную копию совместно используемой библиотеки и хранить ее на диске в единственном экземпляре. Дополнительным преимуществом служит возможность обновления совместно используемой библиотеки независимо от базирующихся на ней приложений. Применяются символические ссылки из файла /lib/libm.so на текущую версию библиотеки (/lib/libm.so.*N*, где *N* — основной номер версии — 6 во время написания книги). Когда ОС Linux запускает приложение, она учитывает номер версии библиотеки, требующийся приложению, чтобы не дать ведущим новым версиям библиотеки испортить более старые приложения.

ПРИМЕЧАНИЕ

Вывод в следующем примере получен из дистрибутива SUSE 10.3. Если вы применяете другой дистрибутив, ваш вывод может слегка отличаться.

В системах Linux программа (динамический загрузчик), отвечающая за загрузку совместно используемых библиотек и разрешение ссылок на функции в клиентских программах, называется ld.so и может присутствовать в системе как ld-linux.so.2, или li-lsb.so.2, или li-lsb.so.3. Дополнительные каталоги поиска совместно используемых библиотек настраиваются в файле /etc/ld.so.conf, который после внесения изменений (например, если добавляются совместно используемые библиотеки X11 при установке графической оболочки X Window System) следует обработать командой `ldconfig`.

Запустив утилиту `ldd`, вы можете увидеть, какие совместно используемые библиотеки требуются программе. Например, если вы попытаетесь выполнить утилиту для приложения из нашего примера, то получите следующие строки:

```
$ ldd program
        linux-gate.so.1 =>   (0xffffe000)
        libc.so.6  =>  /lib/libc.so.6 (oxb7db4000)
        /lib/ld-linux.so.2  (0xb7efc000)
```

В этом случае вы видите стандартную библиотеку C (libc) как совместно используемую (.so). Программе требуется основная версия 6. В других системах UNIX принимаются аналогичные меры для получения доступа к совместно используемым библиотекам. Подробную информацию можно найти в системной документации.

Во многом совместно используемые библиотеки аналогичны динамически подключаемым библиотекам в ОС Windows. Библиотеки с расширением .so соответствуют файлам с расширением dll и требуются во время выполнения, а библиотеки с расширением .a аналогичны файлам с расширением lib, которые включаются в исполняемые программы.

Получение справки

Подавляющее большинство систем Linux хорошо документировано в отношении системных программных интерфейсов и стандартных утилит. Это на самом деле так, потому что со времени первых систем UNIX программистов призывали снабжать свои приложения справочными материалами или описаниями. Эти справочные страницы (man pages), которые иногда предоставлялись в печатной форме, всегда доступны и в электронном виде.

Доступ к интерактивным справочным руководствам обеспечивает команда `man`. Эти справочные руководства отличаются качеством и деталями. Одни могут просто переадресовать читателя к другой, более полной документации, в то время как другие дают полный перечень всех опций и команд, поддерживаемых утилитой. В любом случае справочные руководства — подходящий способ знакомства с приложением.

Комплект программного обеспечения проекта GNU и другое свободное ПО используют интерактивную систему документации, названную info. Вы можете просмотреть всю документацию в интерактивном режиме с помощью специальной программы info или команды `info` редактора emacs. Достоинство системы info в том, что вы можете перемещаться по разделам документации с помощью связей и перекрестных ссылок, переходя непосредственно к нужному вам разделу. Для разработчика документации преимущество в том, что справочные файлы могут автоматически генерироваться из того же источника, что и печатные или набранные на клавиатуре страницы документации.

Упражнение 1.3. Справочные руководства и система info

Давайте познакомимся с документацией для компилятора GNU C (gcc).

1. Сначала посмотрим на справочное руководство.

```
$ man gcc
GCC(1)                GNU                GCC(1)

NAME
       gcc — GNU project C and C++ compiler
SYNOPSIS
       gcc [-c|-S|-E] [-std=standard]
           [-g] [-pg] [-Olevel]
           [-Wwarn...] [-pedantic]
           [-Idir...] [-Ldir...]
           [-Dmacro[=defn]...] [-Umacro]
           [-foption...] [-mmachine-option...]
           [-o outfile] infile...

       Only the most useful options are listed here; see below
       for the remainder. g++ accepts mostly the same options as
       gcc.

DESCRIPTION
       When you invoke GCC, it normally does preprocessing, com
       pilation, assembly and linking. The ``overall options''
       allow you to stop this process at an intermediate stage.
       For example, the -c option says not to run the linker.
       Then the output consists of object files output by the
       assembler.

       Other options are passed on to one stage of processing.
       Some options control the preprocessor and others the com
       piler itself. Yet other options control the assembler and
       linker; most of these are not documented here, since we
       rarely need to use any of them.
...
```

Если хотите, можно прочесть об опциях, поддерживаемых транслятором. В этом случае справочное руководство очень длинное, хотя содержит лишь малую часть полной документации по компилятору GNU C (и C++).

При чтении справочных страниц можно использовать клавишу <Пробел> для перехода к следующей странице, клавишу <Enter> (или клавишу <Return>, если на вашей клавиатуре применяется эта клавиша вместо <Enter>) для перехода к следующей строке и клавишу <q> для полного выхода из программы.

2. Для получения более подробной информации о компиляторе GNU C можно попробовать применить команду `info`.

```
$ info gcc
File: gcc.info, Node: Top, Next: G++ and GCC, Up: (DIR)
Introduction
************

   This manual documents how to use the GNU compilers, as well as their
features and incompatibilities, and how to report bugs. It corresponds to
GCC version 4.1.3. The internals of the GNU compilers, including how to port
them to new targets and some information about how to write front ends for
new languages, are documented in a separate manual.
*Note Introduction: (gccint)Top.

* Menu:

* G++ and GCC:: You can compile C or C++ Applications.
* Standards:: Language standards supported by GCC.
* Invoking GCC:: Command options supported by `gcc'.
* C Implementation:: How GCC implements the ISO C specification.
* C Extensions:: GNU extensions to the C language family.
* C++ Extensions:: GNU extensions to the C++ language.
* Objective-C:: GNU Objective-C runtime features.
* Compatibility:: Binary Compatibility
--zz-Info: (gcc.info.gz)Top, 39 lines --Top-------------------------
Welcome to Info version 4.8. Type ? for help, m for menu item.
```

Вам предоставляется длинное меню опций, которые можно выбирать для перемещения в полной текстовой версии документации. Элементы меню и иерархия страниц позволяют находить нужные разделы в очень большом документе. На бумаге документация к компилятору GNU C занимает многие сотни страниц.

У системы info есть собственная справка, конечно, в формате страниц info. Если нажать комбинацию клавиш <Ctrl>+<H>, можно познакомиться со справочным руководством, включающим средства обучения пользованию системой info. Программа info входит в состав многих дистрибутивов Linux и может устанавливаться в других ОС UNIX.

Резюме

В этой вводной главе мы познакомились с программированием в ОС Linux и другими компонентами ОС Linux, общими с патентованными системами UNIX. Мы отметили огромное разнообразие систем программирования, доступных UNIX-разработчикам. Мы также представили простые программу и библиотеку, чтобы показать базовые средства языка C и сравнить их с эквивалентными средствами в ОС Windows.

2

Программирование средствами командной оболочки

Начав книгу с программирования в ОС Linux на языке C, теперь мы сделаем отступление и остановимся на написании программ в командной оболочке. Почему? ОС Linux не относится к системам, у которых интерфейс командной строки — запоздалое детище графического интерфейса. У систем UNIX, прообраза Linux, первоначально вообще не было графического интерфейса; все выполнялось из командной строки. Поэтому оболочка командной строки UNIX все время развивалась и превратилась в очень мощный инструмент. Эти свойства перекочевали и в Linux, и некоторые самые серьезные задачи вы можете выполнить наиболее легким способом именно из командной оболочки. Поскольку она так важна для ОС Linux и столь полезна для автоматизации простых задач, программирование средствами командной оболочки рассматривается прежде всего.

В этой главе мы познакомим вас с синтаксисом, структурами и командами, доступными при программировании в командной оболочке, как правило, используя интерактивные (основанные на экранах) примеры. Они помогут продемонстрировать функциональные возможности командной оболочки и собственные действия. Мы также бросим беглый взгляд на пару особенно полезных утилит режима командной строки, часто вызываемых из командной оболочки: grep и find. Рассматривая утилиту grep, мы познакомимся с основными положениями, касающимися регулярных выражений, которые появляются в утилитах ОС Linux и языках программирования, таких как Perl, Ruby и PHP. В конце главы вы узнаете, как писать настоящие сценарии, которые будут перепрограммироваться и расширяться на языке C на протяжении всей книги. В этой главе рассматриваются следующие темы:

- что такое командная оболочка;
- теоретические основы;
- тонкости синтаксиса: переменные, условия и управление программой;

- списки;
- функции;
- команды и их выполнение;
- встроенные (here) документы;
- отладка;
- утилита grep и регулярные выражения;
- утилита find.

Если вы столкнулись со сложным сценарием командной оболочки при администрировании системы, хотите смоделировать вашу последнюю грандиозную (но удивительно простую) идею или просто повысить производительность какой-либо итеративной задачи, эта глава вам будет полезна.

Почему программа в командной оболочке?

Одна из причин применения командной оболочки — возможность быстрого и простого программирования. Более того, командная оболочка всегда есть даже в самых упрощенных установках ОС Linux, поэтому благодаря простому моделированию вы сможете понять, работает ли ваша идея. Командная оболочка очень удобна и для небольших утилит, выполняющих относительно простую задачу, для которой производительность менее важна, чем простые настройка, сопровождение и переносимость. Вы можете применять оболочку для управления процессами, обеспечивая выполнение команд в заданном порядке, зависящем от успешного завершения каждого этапа выполнения.

Хотя внешне командная оболочка очень похожа на режим командной строки в ОС Windows, она гораздо мощнее и способна выполнять самостоятельно очень сложные программы. Вы можете не только выполнять команды и вызывать утилиты ОС Linux, но и разрабатывать их. Командная оболочка выполняет программы оболочки, часто называемые *сценариями* или *скриптами*, которые интерпретируются во время выполнения. Такой подход облегчает отладку, потому что вы легко можете выполнять программу построчно и не тратить время на перекомпиляцию. Но для задач, которым важно время выполнения или необходимо интенсивное использование процессора, командная оболочка оказывается неподходящей средой.

Немного теории

Вот мы и добрались до теоретических основ UNIX и, конечно, Linux. ОС UNIX основана на интенсивном многократном применении кода и зависит от него. Вы разработали маленькую простую утилиту, и пользователи применяют ее как одну из ссылок в строке, формирующей команду. Одно из наслаждений, доставляемых ОС Linux, — разнообразие имеющихся отличных средств. Примером может служить следующая команда:

```
$ ls -al | more
```