Programming Perl

Third Edition

Larry Wall, Tom Christiansen
& Jon Orwant

O'REILLY®

Программирование на Perl

Третье издание

Ларри Уолл, Том Кристиансен и Джон Орвант

Санкт-Петербург
2002

Ларри Уолл, Том Кристиансен и Джон Орвант

Программирование на Perl

Перевод С. Маккавеева

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научные редакторы	*К. Иванов, В. Рижий*
Редактор	*В. Овчинников*
Корректура	*С. Беляева*
Верстка	*А. Дорошенко*

Уолл Л., Кристиансен Т., Орвант Д.
Программирование на Perl. – Пер. с англ. – СПб: Символ-Плюс, 2002. – 1152 с., ил.
ISBN 5-93286-020-0

Первое издание «Программирование на Perl», ставшее непререкаемой библией языка, вышло в 1991 году. Сейчас Perl завоевал широкую популярность, что и потребовало нового, уже третьего издания, содержащего как введение в язык Perl для новичков в программировании, так и отличный справочник по языку.

Ларри Уолл – создатель Perl и один из авторов этой книги. По образованию он еще и лингвист. Возможно, поэтому Perl стал необычайно гибким языком, где одного и того же можно достичь многими путями, как это прекрасно демонстрирует автор. Уже написано много книг, в которых рассматриваются многочисленные возможности Perl, однако только в этой книге рассказывается, зачем эти возможности были созданы и как их использовать в полную силу. Особенно полезен Perl в системном администрировании и веб-программировании.

Что нового в третьем издании? Практически все. Оно не просто расширено в соответствии с релизом Perl 5.6, но и полностью реорганизовано и усилено множеством примеров. Большая часть разделов радикально переработана, например, разделы, посвященные объектно-ориентированному программированию и регулярным выражениям. Кроме того, добавлено много новых глав, рассматривающих работу с профилями, Unicode, потоки, компилирование, документацию pod и внутреннюю организацию Perl.

ISBN 5-93286-020-0
ISBN 0-596-00027-8 (англ)

Издательство «Символ-Плюс». 193148, Санкт-Петербург, ул. Пинегина, 4,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.
Подписано в печать 28.02.2002. Формат 70x100 1/16 . Печать офсетная.
Объем 72 печ. л. Тираж 3000 экз. Заказ N
Отпечатано с диапозитивов в Академической типографии «Наука» РАН
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Предисловие

В погоне за счастьем

Perl – язык, с помощью которого вы сделаете свою работу.

Конечно, если эта работа – программирование, то теоретически ее можно сделать с помощью любого «полного» компьютерного языка. Но опыт показывает, что компьютерные языки различаются не столько *возможностью* что-либо сделать, сколько *легкостью*, с которой это достигается. На одном полюсе находятся так называемые языки четвертого поколения, с помощью которых можно легко делать одни вещи и почти невозможно другие. На другом полюсе – так называемые языки с промышленными возможностями (industrial-strength languages), посредством которых одинаково трудно делать почти все.

Perl не таков. Если сказать кратко, то он разработан так, чтобы легко решать простые задачи, сохраняя возможность решать трудные.

Что это за «простые задачи», которые должны решаться легко? Разумеется, те, которые мы решаем изо дня в день. Нам нужен язык, с помощью которого легко работать с числами и текстом, файлами и каталогами, компьютерами и сетями, а в особенности – с программами. Он должен позволять легко запускать внешние программы и просматривать результаты их работы в поиске интересных данных. Он должен позволять легко отправлять эти интересные данные другим программам, способным обрабатывать их особым образом. Он должен также позволять нам легко разрабатывать собственные программы, изменять их и производить отладку. И, конечно, наши собственные программы должны легко компилироваться и запускаться, а также быть переносимыми на любую современную операционную систему.

Все это, а также многое другое, делает Perl.

Первоначально разработанный как интегрирующий язык для Unix, Perl давно распространился на большинство других операционных систем. Поскольку Perl выполняется почти везде, он является одной из наиболее переносимых сред программирования, имеющихся на сегодняшний день. Чтобы написать переносимые программы на C или C++, необходимо расставить все эти странные пометки `#ifdef` для каждой операционной системы. Для обеспечения переносимости программ на Java нужно разбираться в индивидуальных особенностях всех новых реализаций. Для создания переносимых

сценариев командного процессора нужно помнить синтаксис всех команд каждой версии операционной системы и пытаться найти общий знаменатель, благодаря которому они, как можно надеяться, будут работать всюду. А чтобы создавать переносимые программы на Visual Basic потребуется дать более гибкое определение понятию «переносимость». :-)

В Perl удается избежать таких проблем, сохраняя при этом многие преимущества других языков и добавляя собственные чудеса. Чудеса связаны с рядом причин: практичностью набора его функций, изобретательностью сообщества разработчиков Perl и богатством, которое предоставляет движение open source в целом. Однако в значительной мере чудеса обусловлены гибридной природой Perl. У Perl смешанное происхождение, и многообразие средств всегда рассматривалось в нем как сила, а не слабость. Perl – это язык, говорящий: «Дайте мне вашу усталость, вашу бедность»[1]. Если вы чувствуете себя словно стиснутым в толпе и стремитесь «дышать свободой», то Perl – для вас.

Perl охватывает различные культуры. Его взрывное распространение в значительной мере питалось стремлением бывших системных Unix-программистов взять с собой как можно больше из «старого мира». Для них Perl является квинтэссенцией культуры Unix, оазисом в пустыне «невозможности перейти из одного места в другое». Существует, однако, и движение в обратном направлении: веб-дизайнеры, работающие под Windows с удовольствием обнаруживают возможность запустить свои Perl-программы на Unix-сервере своей компании без их дополнительной переработки.

Хотя Perl особенно популярен среди системных программистов и разработчиков для Интернета, это связано лишь с тем, что они первыми его открыли; аудитория Perl значительно шире. Получивший при создании скромный статус языка обработки текста, Perl превратился в сложный язык программирования общего назначения с богатой средой разработки программ, укомплектованной отладчиками, профайлерами, формировщиками перекрестных ссылок, компиляторами, библиотеками, синтаксически-ориентированными редакторами и всеми остальными атрибутами «настоящего» языка программирования – если они вам требуются. Но все они относятся к поддержке возможностей решения сложных задач, с чем справляются многие другие языки. Уникальность Perl в том, что он никогда не терял из виду возможность легко решать задачи простые.

Поскольку Perl является одновременно мощным и доступным средством, он постоянно используется во всех мыслимых сферах – от аэрокосмической техники до молекулярной биологии, от математики до лингвистики, от графики до обработки документов, от обработки баз данных до сетевого администрирования. Perl используется теми, кому необходимо быстро проанализировать или преобразовать большие объемы данных, будь то последова-

[1] «Give me your tired, your poor, Your huddled masses yearning to breathe free...» – слова, которыми начинается надпись на основании Статуи Свободы. – *Примеч. ред.*

тельность генов ДНК, веб-страницы или фьючерсы на свиные внутренности. В самом деле в сообществе Perl бытует шутка, что очередной крупный кризис на рынке ценных бумаг может разразиться из-за ошибки, сделанной кем-нибудь в своем сценарии Perl. (Светлой стороной окажется то, что безработные рыночные аналитики окажутся, так сказать, востребованными.)

Существует много слагаемых успеха. Perl как проект open source достиг его еще до того, как это движение получило свое название. Perl свободно распространяется, и так будет всегда. Каждый может работать с Perl так, как сочтет удобным, и на основе очень либеральной политики лицензирования. Если вы занимаетесь коммерческой деятельностью и хотите воспользоваться Perl, можете смело идти вперед. Разрешается применять Perl в разрабатываемых коммерческих приложениях бесплатно и без ограничений. А для тех, у кого возникнет проблема, которую сообщество Perl не сможет решить, существует последняя инстанция: сам исходный код. Сообщество Perl не занимается продажей своих профессиональных тайн под видом «обновлений». Сообщество Perl никогда не «выйдет из дела» и не оставит вас с брошенным на произвол судьбы продуктом.

Безусловно, популярности Perl способствует его бесплатное распространение. Но этого недостаточно для объяснения феномена Perl, поскольку многие бесплатно распространяемые пакеты не преуспели. Дело не в том, что он бесплатен; он доставляет удовольствие. Люди чувствуют желание творить на Perl, поскольку он дает свободу самовыражения: можно выбирать между целями оптимизации – скоростью работы компьютера или скоростью программирования, между многословием и выразительностью, между «читабельностью» и возможностью поддержки или повторного использования, или переносимости, или обучаемости, или поучительности. Можно оптимизировать даже непонятность, если принять участие в конкурсе на самую непонятную программу – Obfuscated Perl Contest.

Perl может предоставить все эти степени свободы, поскольку это язык с раздвоением личности. Это одновременно очень простой язык и очень богатый язык. Perl собрал отовсюду хорошие идеи и поместил их в простую в использовании логическую структуру. Для тех, кому он просто нравится, Perl – это *Practical Extraction and Report Language* (практический язык извлечения данных и создания отчетов). Для тех, кто любит его, Perl – это *Pathologically Eclectic Rubbish Lister* (паталогически эклектичный язык для распечатки чепухи). А для многочисленных минималистов Perl кажется проявлением бесцельной избыточности. Но это хорошо. Редукционисты должны существовать (в основном среди физиков). Редукционисты стремятся разъять целое на части. Мы, все остальные, пытаемся собрать их вместе.

Во многих отношениях Perl is a simple language (простой язык). Не требуется знать множества особых заклинаний, чтобы скомпилировать программу на Perl – ее можно просто выполнить как пакетный файл или сценарий оболочки. Типы и структуры Perl просты в использовании и понимании. Perl не налагает свои произвольные ограничения на данные – строки и массивы мо-

гут быть сколь угодно велики, лишь бы хватило оперативной памяти, и их организация позволяет им легко увеличиваться по мере надобности. Perl не требует изучения новых синтаксиса и семантики, в значительной мере заимствуя их из других языков, с которыми вы можете быть знакомы (например, C, *awk*, BASIC, Python, английский и греческий). На практике почти любой программист может прочесть хорошо написанный код Perl и составить себе представление о том, что он делает.

Очень важно, что нет необходимости изучить Perl полностью, чтобы начать писать полезные программы. Можно начать изучение Perl с «тонкого конца». Вы можете программировать на языке «детский лепет Perl», и мы обещаем не смеяться над этим. Точнее, мы обещаем смеяться не более, чем над первыми творческими попытками ребенка. Многие идеи Perl заимствованы из естественного языка, и одна из лучших состоит в том, что можно использовать лишь подмножество языка, если его достаточно для того, чтобы передать мысль. В культуре Perl приемлема любая степень владения языком. Полицию по охране языка мы к вам не пришлем. Сценарий Perl будет «правильным», если выполнит задачу прежде, чем начальник вас уволит.

Будучи во многих отношениях простым, Perl является и богатым языком, который нужно долго изучать. Это расплата за возможность решать сложные задачи. Хотя понадобится некоторое время на освоение всех средств Perl, вы будете рады иметь в своем распоряжении расширенные возможности, когда настанет момент и они вам понадобятся.

Благодаря своему происхождению Perl был богатым языком даже тогда, когда служил «просто» языком преобразования данных, предназначенным для перемещения по файлам, просмотра больших объемов текста, создания и получения динамических данных и вывода легко форматируемых отчетов, основанных на этих данных. Но в какой-то момент начался расцвет Perl. Он стал также и языком для работы с файловой системой, управления процессами, администрирования баз данных, программирования в архитектуре клиент-сервер, создания безопасных программ, управления данными в Сети и даже для объектно-ориентированного и функционального программирования. Эти возможности не были просто механически присоединены к Perl – каждая новая синергически работает с остальными, поскольку с самого начала Perl проектировался как интегрирующий язык.

Но Perl может объединять в единое целое не только свои собственные функции. Он разработан как модульно расширяемый язык. Perl позволяет быстро разрабатывать, программировать, отлаживать и разворачивать приложения, а также при необходимости без труда расширять функциональные возможности этих приложений. Perl можно встраивать в другие языки, а также встраивать другие языки в Perl. С помощью механизма импорта модулей можно использовать эти внешние определения, как если бы они были встроенными функциями Perl. Объектно-ориентированные внешние библиотеки сохраняют свою объектную ориентированность в Perl.

Perl полезен и в других отношениях. В отличие от строго интерпретируемых языков, таких как командные файлы и сценарии оболочки, которые одно-

временно компилируют и выполняют одну команду, Perl сначала быстро компилирует программу в промежуточный формат. Подобно любому другому компилятору, он осуществляет различного вида оптимизации и мгновенно реагирует на любые ошибки – от синтаксических и семантических до неудачи при связывании с библиотеками. Когда компилирующий интерфейс Perl оказывается удовлетворен вашей программой, он передает промежуточный код на выполнение интерпретатору (либо какому-либо из нескольких модулей серверов, способных создавать текст на C в виде байт-кодов). Все это выглядит сложным, однако компилятор и интерпретатор работают весьма эффективно, и обычный цикл компиляции-прогона-исправления занимает считанные секунды. В совокупности с большим количеством предоставляемых Perl функций амортизации отказов эта возможность короткого времени оборота делает Perl языком, на котором действительно возможно быстрое прототипирование. Позже, по мере совершенствования программы вы можете повысить требовательность к себе и заставить себя программировать, применяя меньше искусства, основанного на интуиции, и больше дисциплины. Perl и в этом окажет содействие, если его хорошо об этом попросить.

Perl также способствует созданию более защищенных программ. Помимо всех обычных интерфейсов защиты, предоставляемых другими языками, Perl защищает от случайных ошибок в системе безопасности посредством уникального механизма трассировки данных, автоматически определяющего данные, которые поступили из ненадежного источника, и предотвращает выполнение опасных операций. Наконец, Perl позволяет создавать специальные защищенные отсеки, в которых можно безопасно выполнять код сомнительного происхождения с запрещением опасных операций.

Парадоксально, но самая большая помощь, которую Perl может оказать программисту, связана не столько с самим Perl, сколько с людьми, которые с ним работают. Сообщество Perl составляют люди, которые более чем кто-либо другой готовы прийти на помощь. Если считать, что в движении Perl есть что-то благочестивое, то оно лежит в самой его сердцевине. Ларри хотел, чтобы сообщество было чем-то вроде рая, и в целом его желание пока осуществляется. Внесите и свой вклад в то, чтобы оно таким и оставалось.

Возможно, вы изучаете Perl, поскольку хотите спасти мир либо просто из любопытства, либо вам приказал делать это ваш начальник – в любом случае этот учебник позволит познакомиться как с основами, так и со сложными вопросами. И хотя мы не намереваемся учить вас программированию, проницательный читатель что-то приобретет как от искусства, так и от науки программирования. Мы подстрекаем вас развивать в себе три великие добродетели программиста: *лень*, *нетерпение* и *самомнение* (*laziness*, *impatience*, *hubris*). Мы надеемся, что, читая эту книгу, вы найдете ее местами довольно занимательной (и местами очень занимательной). Если этого окажется недостаточно для того, чтобы вы не заснули, то постоянно напоминайте себе, что изучение Perl повысит ценность вашего резюме. Так что читайте дальше.

Что обновилось в этом издании

Пожалуй, почти все.

Даже там, где мы сохранили удачные места из предыдущего издания (а полагаем, что таких было достаточно много), мы существенно пересмотрели и переработали материал, стремясь достичь нескольких целей. Во-первых, мы хотели повысить доступность книги для тех, кто не получил подготовки в области вычислительной техники. Мы уменьшили предполагаемый объем знаний читателя. В то же время мы старались оживить изложение в надежде, что те, кто уже частично знаком с обсуждаемым предметом, не заснут во время чтения.

Во-вторых, мы хотели представить последние достижения в развитии самого Perl. С этой целью мы не стеснялись сообщать о текущем состоянии разработок, даже сознавая, что они все еще находятся в экспериментальной стадии. Несмотря на то что ядро Perl стоит как скала в течение ряда лет, скорость разработки некоторых его экспериментальных расширений подчас весьма впечатляет. Мы честно сообщаем о тех случаях, когда, по нашему мнению, электронная документация может оказаться более надежной, чем сведения в нашей книге. Perl – это язык рабочего человека, и мы не боимся называть вещи своими именами.

В-третьих, мы хотели облегчить поиск нужного материала, поэтому уменьшили объем глав, сделали их более вразумительными, а также объединили главы в части по смыслу. Вот структура нового издания:

Часть 1 «Общий обзор»

Начать всегда труднее всего. В этой части базовые идеи Perl излагаются в неформальном виде – устройтесь поудобнее в вашем любимом кресле. Не будучи полным учебным руководством, эта часть просто предлагает быстрое начало, которое может потребоваться не каждому читателю. Поищите в разделе «Печатная документация» книги, которые могут лучше соответствовать вашему стилю учебы.

Часть 2 «Анатомия Perl»

В этой части проводится глубокое и ничем не сдерживаемое обсуждение внутреннего устройства языка на всех уровнях абстракции – от типов данных, переменных и регулярных выражений до подпрограмм, модулей и объектов. Читатель получит хорошее представление о том, как работает язык, а также несколько советов по правильному проектированию программ. (А тех, кто никогда не использовал язык с поиском по шаблону, ждет особое удовольствие.)

Часть 3 «Perl как технология»

Многое можно делать с помощью одного только Perl, но в этой части вы достигнете более высокого уровня мастерства. Узнаете о том, как заставить Perl пройти через все препятствия, которые поставит перед ним ваш компьютер, – от обработки, взаимодействия процессов и многопоточнос-

ти до компилирования, вызова, отладки и профилирования, а также создания собственных внешних расширений на C или C++ или интерфейсов к имеющимся API. Perl будет счастлив побеседовать с любым интерфейсом на вашем компьютере да, пожалуй, и любом другом компьютере в Интернете, если позволят погодные условия.

Часть 4 «Perl как культура»

Каждому ясно, что у культуры должен быть свой язык, но сообществу Perl всегда было ясно, что у языка должна быть культура. В этой части мы рассматриваем программирование на Perl как человеческую деятельность, являющуюся частью реального мира людей. Мы даем также много советов относительно того, как можно заниматься самосовершенствованием и как сделать, чтобы ваши программы приносили больше пользы людям.

Часть 5 Справочный материал

Здесь мы собрали главы, в которых читатель сможет найти что-либо в алфавитном порядке – от специальных переменных и функций до стандартных модулей и прагм. Глоссарий окажется особенно полезен тем, кто не знаком с жаргоном, используемым в вычислительной технике. Например, те, кто не знает, что такое «прагма», могут прямо сейчас посмотреть значение этого слова. (А тем, кто не знает значение слова «такое», мы не можем помочь ничем.)

Стандартный дистрибутив

В настоящее время большинство поставщиков операционных систем включают Perl в качестве стандартной составляющей своей системы. На момент написания данной книги Perl входит в стандартные дистрибутивы AIX, BeOS, BSDI, Debian, DG/UX, DYNIX/ptx, FreeBSD, IRIX, LynxOS, Mac OS X, OpenBSD, RedHat, SINIX, Slackware, Solaris, SuSE и Tru64. Некоторые компании поставляют Perl на отдельных CD с бесплатным программным обеспечением или через группы обслуживания клиентов. Сторонние поставщики, такие как ActiveState, предоставляют откомпилированные дистрибутивы для ряда операционных систем, в том числе производимых Microsoft.

Даже если поставщик включил Perl в стандартный дистрибутив, в конечном итоге, возможно, понадобится откомпилировать и установить Perl самостоятельно. В результате вы будете знать, что ваша версия является самой свежей, и сможете сами выбрать, куда установить библиотеки и документацию. Также можно будет решить, следует ли скомпилировать Perl с поддержкой дополнительных расширений, таких как многопоточность, большие файлы или множество низкоуровневых опций отладки, доступ к которым осуществляется через ключ командной строки `<option>-D</option>`. (Отладчик уровня пользователя поддерживается всегда.)

Проще всего загрузить комплект исходного кода Perl, указав броузеру домашнюю страницу на *www.perl.com*, где находятся также ссылки на компи-

лированные двоичные модули для платформ, компиляторы C для которых затерялись.

Можно также направиться прямо в CPAN (Comprehensive Perl Archive Network, архив Perl, описанный в главе 22 «CPAN») по адресу *http://www.perl.com/CPAN* или *http://www.cpan.org*. Если работа с ними окажется слишком медленной (а это может случиться, поскольку они очень популярны), следует найти зеркальный сервер CPAN поблизости от себя. Ниже приведены URL лишь некоторых из зеркал, разбросанных по всему свету и числом превышающих сотню:

http://www.funet.fi/pub/languages/perl/CPAN
ftp://ftp.funet.fi/pub/languages/perl/CPAN/
ftp://ftp.cs.colorado.edu/pub/perl/CPAN/
ftp://ftp.cise.ufl.edu/pub/perl/CPAN/
ftp://ftp.perl.org/pub/perl/CPAN/
http://www.perl.com/CPAN-local
http://www.cpan.org/
http://www.perl.org/CPAN/
http://www.cs.uu.nl/mirror/CPAN/
http://CPAN.pacific.net.hk/

Первая парочка из этого списка на сайте *funet.fi* указывает на основное хранилище CPAN. Список всех остальных сайтов CPAN находится в файле *MIRRORED.BY*, поэтому лучше сначала скачать этот файл, а затем выбрать понравившееся зеркало. Доступ к одним осуществляется через FTP, к другим – через HTTP (если вы спрятались за корпоративным брандмауэром, это может иметь значение). Редиректор *http://www.perl.com/CPAN* пытается сделать этот выбор вместо вас. При желании можно изменить свой выбор позже.

Получив исходный код и распаковав его в каталог, нужно прочесть файлы *README* и *INSTALL*, чтобы узнать, как скомпилировать и скомпоновать Perl. Может иметься и файл *INSTALL.platform*, где *platform* представляет платформу вашей операционной системы.

Если данная платформа является разновидностью Unix, то команды, необходимые для получения, конфигурирования, сборки и установки Perl, могут быть примерно следующие. Во-первых, необходимо выбрать команду, с помощью которой будет получен исходный код. Можно воспользоваться *ftp* :

```
% ftp ftp://ftp.funet.fi/pub/languages/perl/CPAN/src/latest.tar.gz
```

(Не бойтесь заменить адрес на ближайшее зеркало CPAN. Конечно, если вы живете в Финляндии, то это и есть ваше ближайшее зеркало CPAN.) При невозможности воспользоваться *ftp* можно осуществить загрузку через Интернет с помощью броузера или средства командной строки:

```
% wget http://www.funet.fi/pub/languages/perl/CPAN/src/latest.tar.gz
```

Теперь нужно распаковать, сконфигурировать, собрать и установить:

```
% tar zxf latest.tar.gz          Или сначала gunzip, tar xf.
% cd perl-5.6.0                  Или 5.* для других версий.
% sh Configure -des              Принимает ответы по умолчанию.
% make test && make install      Установка обычно производится
                                 суперпользователем.
```

При этом используется обычная среда разработки на C, так что если компилятора C на машине нет, скомпилировать Perl не удастся. Посмотрите в каталоге CPAN *ports* самые свежие данные по каждой платформе относительно поставки Perl вместе с дистрибутивом (и если так, то какой версии), возможности обойтись стандартным комплектом исходного кода или необходимости специального переноса. Ссылки для загрузки даются для тех систем, которые обычно требуют специальных переносов, или систем от поставщиков, для которых отсутствие компилятора C – нормальная практика (правильнее было бы сказать, «ненормальная» практика).

Электронная документация

Обширная электронная документация по Perl входит в состав его стандартного дистрибутива. (О бумажной документации говорится в следующем разделе.) Дополнительная документация появляется, как только устанавливается новый модуль из CPAN.

Упоминая в этой книге «страницу руководства Perl», мы имеем в виду комплект электронных страниц руководства по Perl, который находится на вашем компьютере. Под *страницей электронного руководства* (*manpage*) будем понимать просто файл с документацией, для чтения которого не обязательно иметь Unix-программу *man*. Страницы руководства Perl могут быть установлены даже как страницы HTML, особенно на системах, отличных от Unix.

Электронные страницы руководства по Perl разделены на несколько секций, поэтому можно легко найти нужное, не продираясь через сотни страниц текста. Поскольку страница верхнего уровня называется просто *perl*, то в Unix команда *man perl* должна привести именно на нее.[1] Эта страница, в свою очередь, ведет на более специальные страницы. Например, *man perlre* выведет страницу руководства по регулярным выражениям Perl. Команда *perldoc* часто работает в тех системах, в которых не работает команда *man*. На компьютерах Macintosh следует выполнить команду *Shuck*. Конкретный перенос может также предоставлять страницы руководства по Perl в формате HTML или родном для системы формате подсказки. Уточните этот вопрос у своего системного администратора – если, конечно, сами не являетесь им.

[1] Если при этом открывается нечто необозримое, то, вероятно, вы обращаетесь к древнему руководству версии 4. Проверьте, не указывает ли MANPATH на места, где можно производить археологические раскопки. (Введите *perldoc perl*, чтобы узнать, как настроить MANPATH соответственно выдаче команды *perl –V:man.dir*.)

Навигация по стандартным страницам руководства

В незапамятные времена (если говорить о Perl, то имеется в виду 1987 год) страница руководства *perl* была кратким документом объемом около 24 печатных страниц. Например, раздел по регулярным выражениям занимал всего два абзаца. (Этого было достаточно при условии знакомства с *egrep*.) В некоторых отношениях с тех пор изменилось почти все. Если считать стандартную документацию, различные утилиты, сведения о переносах на различные платформы и множество стандартных модулей, то наберется свыше 1500 печатных страниц документации, разбросанных по многим отдельным страницам электронного руководства. (И это без учета модулей из CPAN, которые могут быть у вас установлены, и в немалом количестве.)

Но в других отношениях не изменилось ничего: по-прежнему жива страница руководства *perl* и по-прежнему это лучшее место для начала, когда не известно, откуда начать. Разница в том, что, попав туда, нельзя остановиться. Документация – это больше не кустарное производство, это торговый пассаж с сотнями магазинов. Войдя в дверь, необходимо найти плакат, который поможет определить, где находится лавка или универсальный магазин, продающие то, зачем вы пришли. Конечно, освоившись в торговом центре, как правило, заранее знаешь, куда направиться.

Вот некоторые вывески:

Страница руководства	Что освещает
perl	Какие страницы руководства по Perl имеются
perldata	Типы данных
perlsyn	Синтаксис
perlop	Операторы и их приоритеты
perlre	Регулярные выражения
perlvar	Предопределенные переменные
perlsub	Подпрограммы
perlfunc	Встроенные функции
perlmod	Как использовать модули Perl
perlref	Ссылки
perlobj	Объекты
perlipc	Межпроцессное взаимодействие
perlrun	Как выполнять команды Perl плюс ключи
perldebug	Отладка
perldiag	Диагностические сообщения

Это лишь небольшой отрывок, но он состоит из важных частей. Как видно, если нужны сведения об операторе, то для этого подойдет *perlop*. А если нуж-

но что-то выяснить о предопределенных переменных, следует искать в *perlvar*. Если получено непонятное диагностическое сообщение, идите на *perldiag*. И так далее.

В стандартное руководство по Perl входит список часто задаваемых вопросов (FAQ). Он разбит на следующие девять разных страниц:

Страница руководства	Что освещает
perlfaq1	Общие вопросы по Perl
perlfaq2	Получение Perl и сведения о нем
perlfaq3	Инструменты программирования
perlfaq4	Обработка данных
perlfaq5	Файлы и форматы
perlfaq6	Регулярные выражения
perlfaq7	Общие вопросы языка Perl
perlfaq8	Взаимодействие с системой
perlfaq9	Сетевое взаимодействие

Некоторые страницы руководства содержат замечания по специфике платформ:

Страница руководства	Что освещает
perlamiga	Перенос на Amiga
perlcygwin	Перенос на Cygwin
perldos	Перенос на MS-DOS
perlhpux	Перенос на HP-UX
perlmachten	Перенос на Power MachTen
perlos2	Перенос на OS/2
perlos390	Перенос на OS/390
perlvms	Перенос на DEC VMS
perlwin32	Перенос на MS-Windows

(Относительно переносов см. также главу 25 «Переносимость программ Perl» и каталог CPAN *ports*, описанный выше.)

Поиск на страницах электронного руководства

Нет необходимости читать все 1500 печатных страниц, чтобы найти иголку в стоге сена. Старая поговорка гласит, что нельзя «грепнуть» (*grep*) мертвое дерево.[1] Кроме обычных средств поиска, присутствующих в большинстве

[1] Не забудьте, что при необходимости можно заглянуть в глоссарий.

программ просмотра документов, в версии 5.6.1 каждая основная страница руководства имеет собственные средства поиска и отображения. Можно осуществлять поиск в отдельных страницах, используя имя страницы руководства в качестве команды и передавая регулярное выражение Perl (см. главу 5 «Поиск по шаблону») в качестве шаблона для поиска:

```
% perlop comma
% perlfunc split
% perlvar ARGV
% perldiag 'assigned to typeglob'
```

Если не известно точно, в каком месте документации находятся требующиеся сведения, можно расширить поиск. Например, для поиска во всех FAQ воспользуйтесь командой *perlfaq* (которая тоже является страницей руководства):

```
% perlfaq round
```

Команда *perltoc* (которая тоже является страницей руководства) осуществляет поиск в общем для всех страниц руководства оглавлении:

```
% perltoc typeglob
perl5005delta: Undefined value assigned to typeglob
perldata: Typeglobs and Filehandles
perldiag: Undefined value assigned to typeglob
```

Можно также проводить поиск во всем электронном руководстве по Perl, включая заголовки, описания и примеры, используя команду *perlhelp*:

```
% perlhelp CORE::GLOBAL
```

Подробности см. в странице руководства по *perldoc*.

Страницы руководства, не принадлежащие Perl

Когда мы ссылаемся на документацию, не принадлежащую Perl, как в *getitimer*(2), ссылка указывает на страницу руководства *getitimer* из раздела 2 *Unix Programmer's Manual*.[1] Страницы руководства для системных вызовов, таких как *getitimer*, могут отсутствовать на системах, отличных от Unix, но это может быть и правильно, потому что системные вызовы на них все равно нельзя использовать. Для тех, кому действительно требуется документация

[1] В разделе 2 должны содержаться данные только о прямых вызовах операционной системы. (Они часто называются системными обращениями («system calls»), но мы неуклонно называем их в этой книге системными вызовами (*syscalls*), чтобы не спутать с функцией system, не имеющей отношения к системным вызовам. Однако между системами есть некоторые различия в том, какие вызовы реализованы как системные, а какие – как обращения к библиотекам C, поэтому есть вероятность, что *getitimer*(2) будет обнаружена в разделе 3.

по команде, системному вызову или библиотечной функции Unix, скажем, что многие организации поместили свои страницы руководства в Интернете. Задав поиск «+crypt(3)+manual» на AltaVista, можно найти много адресов.

Хотя страницы руководства по Perl верхнего уровня обычно устанавливаются в секции 1 стандартных каталогов *man*, мы не будем добавлять (1) к названиям таких страниц руководства в нашей книге. Их легко узнать, поскольку они имеют вид «`perl`*`бубубу`*».

Печатная документация

Тем, кто хочет больше узнать про Perl, рекомендуем некоторые издания:

- *Perl 5 Pocket Reference*, 3d ed., by Johan Vromans, O'Reilly, 2000 (Карманный справочник по Perl, 3-е изд., Йохан Вроманс). Эта маленькая брошюра служит удобным справочником по Perl.
- *Perl Cookbook*, by Tom Christiansen and Nathan Torkington, O'Reilly, 1998 («Perl: библиотека программиста», Т. Кристиансен, Н. Торкингтон, изд-во «Питер», 2000). Эта книга дополняет ту, которую вы сейчас держите в руках.
- *Elements of Programming with Perl*, by Andrew L. Johnson Manning, 1999 (Программирование на Perl, Эндрью Л. Джонсон). Эта книга предназначена для обучения с азов непрограммистов тому, как нужно программировать вообще и делать это с помощью Perl в частности.
- *Learning Perl*, 2d ed., by Randal Schwartz and Tom Christiansen, O'Reilly, 1997 («Изучаем Perl», Р. Шварц, Т. Кристиансен, BHV-Киев). Эта книга научит системных администраторов и программистов для Unix 30 основным процентам Perl, которые им понадобятся в 70 процентах случаев. Эрик Олсон (Erik Olson) переработал эту книгу, создав вариант для программистов на системах Microsoft под названием *Learning Perl for Win32 Systems*.
- *Perl: The Programmer's Companion*, by Nigel Chapman, Wiley, 1997 (Perl: спутник программиста, Найджел Чапмен). Эта чудесная книга предназначена для профессионалов в вычислительной технике и программировании безотносительно к используемой платформе. Perl освещается кратко, но полно.
- *Mastering Regular Expressions*, by Jeffrey Friedl, O'Reilly, 1997 («Регулярные выражения. Библиотека программиста», Д. Фридл, изд-во «Питер», 2001). Хотя в книге не освещены последние нововведения в регулярных выражениях Perl, она является ценным справочником для всех, кто хочет знать, как в действительности работают регулярные выражения.
- *Object Oriented Perl*, by Damian Conway, Manning, 1999 (Объектно-ориентированный Perl, Дамиан Конвей). Для начинающих и опытных разработчиков объектно-ориентированных программ. Эта удивительная книга

излагает обычные и тайные приемы создания мощных объектных систем на Perl.

- *Mastering Algorithms with Perl*, by Jon Orwant, Jarkko Hietaniemi, and John Macdonald, O'Reilly, 1999 (Perl: алгоритмы, Джон Орвант, Йаркко Хьетаниеми и Джон Макдональд). Все полезные приемы из курса вычислительных алгоритмов, но без тягостных доказательств. Книга освещает фундаментальные и полезные алгоритмы, относящиеся к графам, текстам, множествам и ряду других областей.
- *Writing Apache Modules with Perl and C*, by Lincoln Stein and Doug MacEachern, O'Reilly, 1999 (Разработка модулей Apache на Perl и C, Л. Штайн). Это руководство по веб-программированию учит тому, как расширить возможности веб-сервера, особенно с использованием модуля `mod_perl` для создания быстро выполняющихся сценариев CGI и доступа из Perl к Apache API.
- *The Perl Journal*, редактируемый Джоном Орвантом (Jon Orwant). Этот ежеквартальный журнал, издаваемый программистами и для программистов, регулярно публикует глубокие материалы по программированию, технические приемы, последние новости и прочее.

Существует много других книг и публикаций по Perl, и по старческой дряхлости мы наверняка позабыли упомянуть некоторые хорошие. (Из милосердия мы пренебрегли обязанностью отметить некоторые плохие издания.)

Помимо перечисленных публикаций, относящихся к Perl, мы рекомендуем следующие книги. Они не посвящены непосредственно Perl, но их удобно иметь под рукой для получения справки, совета или вдохновения.

- *The Art of Computer Programming*, by Donald Knuth, vol. 1, *Fundamental Algorithms*; vol. 2, *Seminumerical Algorithms*; vol. 3, *Sorting and Searching*, Addison-Wesley, 1998.[1]
- *Introduction to Algorithms*, by Cormen, Leiserson, and Rivest MIT Press and McGraw-Hill, 1990 (Введение в алгоритмы, Кормен, Лейзерсон и Райвест).
- *Algorithms in C: Fundamental Data Structures, Sorting, Searching*, 3d ed., by Robert Sedgewick, Addison-Wesley, 1997 («Фундаментальные алгоритмы на C++. Анализ, структуры данных, сортировка, поиск», 3-е издание, Р. Седжвик, изд-во «Диасофт»).
- *The Elements of Programming Style*, by Kernighan and Plauger, Prentice-Hall, 1988 (Элементы стиля программирования, Керниган и Плаугер).
- *The Unix Programming Environment*, by Kernighan and Pike, Prentice-Hall, 1984 (Среда разработки Unix, Керниган и Пайк).

---

[1] Д. Кнут «Искусство программирования», том 1 «Основные алгоритмы»; том 2 «Получисленные алгоритмы»; том 3 «Сортировка и поиск»; 3-е издание, изд-во «Вильямс», 2000.

- *POSIX Programmer's Guide*, by Donald Lewine, O'Reilly, 1991 (Posix: руководство программиста, Д. Левин).
- *Advanced Programming in the UNIX Environment*, by W. Richard Stevens, Addison-Wesley, 1992 (Профессиональное программирование в среде Unix, У. Сивенс).
- *TCP/IP Illustrated*, vols. 1–3, by W. Richard Stevens, Addison-Wesley, 1994–1996 (Иллюстрированный TCP/IP, У. Стивенс).
- *The Lord of the Rings*, by J. R. R. Tolkien (последнее издание: Houghton Mifflin, 1999).

Дополнительные источники

Интернет – чудесное изобретение, и все мы продолжаем открывать возможности его наиболее полного использования. (Конечно, некоторые предпочитают «открывать» Интернет так, как Толкиен открывал Центр Земли.)

Perl в Сети

Посетите домашнюю страницу Perl *http://www.perl.com/*. На ней рассказывается, что нового есть в мире Perl, содержатся исходные коды и переносы, тематические статьи, документация, расписания конференций и многое другое.

Посетите также веб-страницу Perl Mongers *http://www.perl.org*, чтобы взглянуть на Perl с точки зрения широких масс, так сказать, на «корни», которые бурно разрастаются во всех частях света, за исключением Южного полюса, где их приходится держать в закрытом помещении. Местные группы PM проводят регулярные небольшие встречи, на которых можно обменяться профессиональным опытом с другими хакерами Perl, живущими в той же части света.

Телеконференции (группы новостей Usenet)

Телеконференции Perl служат огромным, хотя подчас беспорядочным, источником сведений о Perl. Начать можно с *comp.lang.perl.moderated* – модерируемой телеконференции с невысоким потоком сообщений, содержащей объявления и технические дискуссии. Благодаря модерированию телеконференция вполне читаема.

Группа *comp.lang.perl.misc* с интенсивным потоком сообщений обсуждает все – от технических вопросов до философии Perl, игр Perl и поэзии Perl. Как и сам Perl, *comp.lang.perl.misc* предназначена для пользы, и в ней можно задать любой глупый вопрос.[1]

[1] Конечно, некоторые вопросы такие глупые, что на них и отвечать не стоит (особенно если на них уже есть ответы в страницах руководства и FAQ. Зачем просить помощи в телеконференции, если можно самому найти ответ, потратив меньше времени, чем занимает ввод запроса?)

В группе *comp.lang.perl.tk* обсуждается, как применять в Perl популярный комплект Tk. Группа *comp.lang.perl.modules* обсуждает разработку и использование модулей Perl, которые служат лучшим способом создания повторно используемого кода. Когда вы будете читать эту книгу, могут появиться новые телеконференции *comp.lang.perl.нечто*, поэтому поищите их.

Если для доступа в Usenet используется не обычная программа чтения новостей, а веб-броузер, введите `"news:"` и имя телеконференции, чтобы получить доступ к одной из указанных здесь телеконференций. (Это возможно, только если у вас есть доступ к серверу новостей.) В другом варианте, если для поиска в Usenet используются такие службы, как AltaVista или Deja, укажите `"*perl*"` в качестве телеконференций, которые нужно найти.

Другой телеконференцией, материалы которой могут понадобиться, по крайней мере, тем, кто занимается CGI-программированием для Интернета, является *comp.infosystems.www.authoring.cgi*. Хотя, строго говоря, это не группа Perl, большинство обсуждаемых в ней программ написаны на Perl. К ней полезно обратиться по вопросам Perl, связанным с Сетью, если только вы не используете `mod_perl` под Apache; тогда можно подписаться на *comp.infosystems.www.servers.unix*.

Сообщения об ошибках

В том маловероятном случае, если ошибка обнаружена в самом Perl, а не в вашей программе, нужно постараться воспроизвести ее в минимальном по объему контрольном примере и сообщить о ней с помощью программы *perlbug*, поставляемой с Perl. Дополнительные сведения можно найти на *http://bugs.perl.org*.

Соглашения, принятые в этой книге

Некоторые из принятых нами соглашений более подробно обсуждаются в соответствующих местах. Соглашения по коду обсуждаются в разделе «Стиль программирования» главы 24 «Распространенные приемы программирования». В некотором смысле наши лексические соглашения представлены в глоссарии (наш словарь).

В книге приняты следующие типографские соглашения:

Курсив

Предназначен для URL, страниц руководства, маршрутов и программ. Новые термины тоже выделяются курсивом при первом появлении в тексте. Для многих из этих терминов можно найти альтернативные определения в глоссарии, если недостаточно тех, которые имеются в тексте.

`Моноширинный`

Используется в примерах и в обычном тексте для вывода кода. Данные обозначаются моноширинным шрифтом и заключаются в кавычки (""), не являющиеся частью значения.

`Моноширинный полужирный`

Моноширинный полужирный шрифт применяется для выделения ключей командной строки. Это позволяет различать, скажем, ключ вывода предупредительных сообщений `-w` и оператор проверки файла `-w`. Этот шрифт используется также в примерах для обозначения текста, вводимого буквально.

`Моноширинный курсив`

Служит для обозначения общих элементов кода, вместо которых необходимо подставить конкретные значения.

Мы приводим массу примеров, большинство из которых представляют собой фрагменты более крупных программ. Некоторые примеры являются завершенными программами, что можно увидеть по начальной строке `#!`. Почти все наши более длинные программы начинаются со строки:

```
#!/usr/bin/perl
```

В других примерах приводится текст, который следует ввести в командной строке. Мы используем `%` для обозначения приглашения оболочки:

```
% perl -e 'print "Hello, world.\n"'
Hello, world.
```

Этот стиль – типичный образец стандартной командной строки Unix, в которой одиночные кавычки представляют «наиболее закавыченный» формат. На других системах соглашения по использованию кавычек и символов заполнителей могут быть иными. Например, многие интерпретаторы команд в MS-DOS и VMS требуют двойные, а не одиночные кавычки при задании аргументов, содержащих пробелы и символы-заполнители.

Благодарности

Здесь мы публично приносим благодарность нашим рецензентам, что должно компенсировать все грубости, сказанные им в частном порядке. Это: Тод Миллер (Todd Miller), Шэрон Хопкинс Рауэзан (Sharon Hopkins Rauenzahn), Рич Рауэнзан (Rich Rauenzahn), Пол Маркес (Paul Marquess), Пол Грасси (Paul Grassie), Натан Торкингтон (Nathan Torkington), Йохан Вроманс (Johan Vromans), Джефф Хемер (Jeff Haemer), Гурусами Сарати (Gurusamy Sarathy), Глория Уолл (Gloria Wall), Дэн Сугальский (Dan Sugalski) и Эбигайл (Abigail).

Хотим выразить особую благодарность Тиму О'Рейли (Tim O'Reilly) и его Associates за побуждение авторов к написанию книг, которые читателям приятно читать.

Хотим услышать ваши отзывы

Мы протестировали и выверили все сведения, представленные в этой книге с возможно большей тщательностью, но читатели могут обнаружить, что некоторые функции претерпели изменения (или даже что мы допустили ошибки!). Просим сообщить обо всех найденных ошибках, а также пожеланиях для последующих изданий по адресу:

O'Reilly & Associates, Inc.

101 Morris Street Sebastopol, CA 95472

1-800-998-9938 (в США и Канаде)

1-707-829-0515 (международный/местный)

1-707-829-0104 (факс)

Можно также посылать сообщения электронной почтой. Чтобы быть включенным в наш лист почтовой рассылки или запросить каталог, посылайте письма по адресу *info@oreilly.com*. Чтобы задать технический вопрос или сообщить свои комментарии по поводу этой книги, шлите почту по адресу *bookquestions@oreilly.com*.

Для этой книги существует сайт, на котором мы публикуем список замеченных ошибок и прочие сведения, относящиеся к верблюду:

http://www.oreilly.com/catalog/pperl3

Там вы найдете также тексты всех примеров из этой книги, которые можно загрузить, чтобы не вводить все самим, как это пришлось делать нам.

III

Perl как технология

15

Unicode

Если вы еще не знаете, что такое Unicode, то скоро узнаете – даже если пропустите эту главу, – потому что работа с Unicode становится неизбежностью. (Некоторые рассматривают ее как неизбежное зло, но скорее это неизбежное добро. Во всяком случае это неизбежное бремя.)

Исторически, люди создавали наборы символов, отражающие их потребности в контексте собственной культуры. Поскольку во всех культурах людям свойственна естественная лень, они стремились включать только необходимые им символы и исключать ненужные. Все было прекрасно, пока мы общались только с людьми, принадлежащими нашей собственной культуре, но теперь, когда мы начинаем использовать Интернет для обмена информацией между разными культурами, такой исключающий подход служит источником проблем. Не так просто разобраться, как вводить символы с акцентами с помощью американской клавиатуры. И что на свете (в буквальном смысле) нужно, чтобы создать многоязычную веб-страницу?

Ответ или по крайней мере часть его, дает Unicode (а также XML). Unicode является включающим, а не исключающим набором символов. Хотя можно и нужно торговаться по поводу отдельных деталей работы с Unicode (а таких деталей очень много), общее желание заключается в том, чтобы все были в достаточной мере удовлетворены[1] Unicode и стремились к его использованию в качестве интернационального носителя для обмена текстовыми данными. Никто не заставляет вас применять Unicode, как никто не заставляет

[1] Или, в некоторых случаях, недостаточно несчастны.

вас читать эту главу (мы надеемся). Люди всегда смогут использовать старые исключающие наборы символов внутри своей собственной культуры. Однако в таком случае страдает переносимость.

Закон сохранения страдания гласит, что если уменьшить страдания в одном месте, они возрастут в некотором другом месте. В случае Unicode мы должны пострадать при переходе от байтовой семантики к символьной семантике. Поскольку по прихоти истории Perl был изобретен американцем, он исторически путает понятия байта и символа. При переходе на Unicode Perl должен каким-то образом их распутать.

Парадоксально, но, заставляя Perl самостоятельно распутывать байты и символы, мы можем позволить программирующему на Perl путать их и полагаться на то, что Perl разберется, – так же, как мы позволяем программистам путать числа и строки и полагаться на Perl в проведении необходимых преобразований. В допустимых пределах подход Perl к Unicode такой же, как ко всему остальному: Делай Правильную Вещь. В идеале хотелось бы достичь следующих четырех целей:

Цель №1:

> Старые байт-ориентированные программы не должны случайно спотыкаться на старых байт-ориентированных данных, с которыми привыкли работать.

Цель №2:

> Старые байт-ориентированные программы должны чудесным образом начать работать с новыми символьно-ориентированными данными, когда это потребуется.

Цель №3:

> Программы должны выполняться с одинаковой скоростью как в новом символьно-ориентированном режиме, так и в старом байт-ориентированном режиме.

Цель №4:

> Perl должен сохраниться как единый язык, а не разветвиться на байт-ориентированный Perl и символьно-ориентированный Perl.

В совокупности достичь этих целей практически невозможно. Но мы весьма приблизились к ним. Или, скорее, мы находимся в процессе значительного приближения, т. к. работа продолжается. По мере развития Unicode то же происходит и с Perl. Но наш всеобъемлющий план нацелен на обеспечение безопасного маршрута перехода, который приведет нас туда, куда нужно, с минимальными потерями в пути. Как мы этого добиваемся, рассказывается в следующем разделе.

Байты и символы

В версиях Perl до 5.6 все строки рассматривались как последовательности байт.[1] Однако в версиях начиная с 5.6 строка может содержать символы, которые шире одного байта. Мы теперь рассматриваем строки не как последовательности байт, а как последовательности чисел в диапазоне `0 .. 2**32-1` (или, в случае 64-разрядных компьютеров, `0 .. 2**64-1`). Эти числа представляют абстрактные символы, и чем больше число, тем, в некотором смысле, «шире» символ; но, в отличие от многих других языков, Perl не привязан к какой-либо конкретной ширине представления символов. Perl использует кодировку переменной длины (на основе UTF-8), поэтому числа для этих абстрактных символов не обязательно упаковываются по одному в байт. Очевидно, что если некоторому символу соответствует число `18 446 744 073 709 551 615` (т. е. `\x{ffff_ffff_ffff_ffff}`), то оно не поместится в одном байте (на самом деле оно занимает 13 байт), но если все символы строки находятся в диапазоне десятичных чисел `0..127`, то, конечно, они упаковываются по одному в байт, т. к. UTF-8 совпадает с ASCII в младших семи разрядах.

Perl использует UTF-8, только когда считает это выгодным, поэтому если все символы вашей строки находятся в диапазоне `0..255`, есть большая вероятность, что все символы упакованы в байты, но при отсутствии других сведений в этом нельзя быть уверенным, поскольку внутренне Perl при необходимости осуществляет преобразование между символами фиксированной 8-разрядной длины и символами переменной длины UTF-8. Суть в том, что обычно вам не приходится об этом беспокоиться, т. к. на абстрактном уровне семантика символов сохраняется независимо от представления.

В любом случае, если в строке есть символы, числовое значение которых больше `255` в десятичной системе, строка обязательно хранится в UTF-8. Точнее, она хранится в расширенной Perl-версии UTF-8, которую мы называем *utf8* в честь прагмы с таким именем, но главным образом потому, что это легче вводить. (И потому, что в «настоящей» UTF-8 допускаются только численные значения символов, утвержденные Консорциумом Unicode. В utf8 разрешается иметь любые численные значения символов, которые вам нужны для работы. Perl не волнует, являются ли значения ваших символов официально корректными или просто корректными.)

Мы сказали, что по большей части вам не приходится об этом беспокоиться, но людям все равно хочется беспокоиться. Допустим, вы используете v-строку для представления адреса IPv4:

```
$locaddr = v127.0.0.1;      # Наверняка хранится как байты.
$oreilly = v204.148.40.9;   # Может храниться как байты или utf8.
$badaddr = v2004.148.40.9; # Наверняка хранится как utf8.
```

[1] Можете называть их «октетами», но мы считаем, что в наше время эти слова стали почти синонимами, поэтому будем придерживаться слова, используемого «синими воротничками».

Каждый может сообразить, что эта $badaddr не будет действовать в качестве IP-адреса. Поэтому можно подумать, что если сетевой адрес O'Reilly's перевести в представление UTF-8, он больше не сможет работать. Но символы в строке – это абстрактные числа, а не байты. Всюду, где используются адреса IPv4, например в функции gethostbyaddr, числа абстрактных символов должны автоматически принудительно преобразовываться обратно в байтовое представление (а для $badaddr должна возникать ошибка).

Интерфейсы между Perl и реальным миром должны учитывать детали представления. В допустимых пределах существующие интерфейсы стараются поступать правильно без дополнительных указаний. Но иногда приходится давать инструкции некоторым интерфейсам, например функции open), а если вы пишете собственный интерфейс к реальному миру, он должен быть настолько интеллектуален, чтобы разбираться в обстановке самостоятельно, или хотя бы достаточно интеллектуален, чтобы выполнять команды, задающие режим работы, отличный от установленного по умолчанию.[1]

Поскольку Perl старается поддерживать прозрачную семантику символов внутри самого языка, единственное место, где программиста должна беспокоить символьная и байтовая семантика, – это его интерфейсы. По умолчанию все старые интерфейсы Perl с окружающим миром являются байт-ориентированными, поэтому они создают и принимают байт-ориентированные данные. Это значит, что на абстрактном уровне все строки являются последовательностями чисел в диапазоне 0..255, поэтому, если ваша программа не переводит их принудительно в представление utf8, то ваша прежняя программа продолжает работать с байт-ориентированными данными, как и ранее. Так что можете пометить галочкой указанную выше Цель №1.

Если вы хотите, чтобы старая программа работала с новыми символьно-ориентированными данными, пометьте свои символьно-ориентированные интерфейсы, чтобы Perl рассчитывал на получение символьно-ориентированных данных от этих интерфейсов. После того как вы это сделаете, Perl должен автоматически произвести преобразования, необходимые для сохранения символьной абстракции. Единственное отличие в том, что в программу вводятся строки, помеченные как потенциально содержащие символы, бо́льшие чем 255, поэтому при выполнении операции над строкой байт и строкой utf8 Perl внутренне преобразует байтовую строку в строку utf8, прежде чем выполнить операцию. Обычно строки utf8 преобразуются обратно в байтовые, только если вы посылаете их на байтовый интерфейс, и тогда, если строка содержит символы, бо́льшие чем 255, возникает проблема, кото-

[1] В некоторых системах могут присутствовать средства для одновременного переключения всех ваших интерфейсов. Если указывается ключ командной строки -C (или глобальная переменная ${^WIDE_SYSTEM_CALLS} установлена равной 1), то все системные вызовы будут использовать соответствующие API расширенных символов. (В настоящее время это реализовано только в Microsoft Windows.) В текущие планы сообщества Linux входит, чтобы все интерфейсы переключались в режим UTF-8, когда $ENV{LC_CTYPE} имеет значение «UTF-8». В других сообществах может быть принят другой подход. Пройденные нами пути могут отличаться.

рую можно решать разными способами в зависимости от конкретного интерфейса. Так что можете пометить галочкой указанную выше Цель №2.

Иногда требуется смешать код, понимающий символьную семантику, с кодом, который должен выполняться с байтовой семантикой, например в коде ввода/вывода, который читает или пишет блоки фиксированного размера. В таком случае можно поместить объявление `use bytes` вокруг байт-ориентированного кода и заставить его использовать байтовую семантику даже со строками, помеченными как строки utf8. Ответственность за все необходимые преобразования лежит в этом случае на вас. Но это способ более строгого локального прочтения Цели №1 за счет более свободного глобального прочтения Цели №2.

Цель №3 в значительной мере достигнута, частично путем откладываемых преобразований между представлениями в байтах и utf8, а частично путем увиливания от реализации потенциально замедляющих функций Unicode, таких как поиск свойств символов в гигантских таблицах.

Цель №4 достигнута в результате пожертвования небольшой частью совместимости интерфейсов ради достижения остальных Целей. Можно считать, что мы не разветвили Perl надвое, но можно посмотреть и так, что версия Perl 5.6 *является* ответвившейся версией Perl по отношению к более ранним версиям, и мы не думаем, что люди станут переходить на нее с более ранних версий, прежде чем убедятся, что новая версия делает то, что им нужно. Но так всегда бывает с новыми версиями, поэтому позволим себе также поставить галочку и у Цели №4.

Действие символьной семантики

Вывод из всего этого такой, что обычный встроенный оператор будет действовать над символами, если только не окажется в области действия прагмы `use bytes`. Однако даже вне области действия `use bytes`, если все операнды оператора хранятся как 8-разрядные символы (т. е. ни один операнд не хранится в utf8), символьная семантика неотличима от байтовой, а результат оператора внутренне хранится в 8-разрядном формате. В результате сохраняется обратная совместимость, если только не подавать в программу символы, находящиеся за пределами Latin-1.

Прагма `utf8` является, по преимуществу, средством обеспечения совместимости, которое активирует распознавание UTF-8 в литералах и идентификаторах, встречающихся синтаксическому анализатору. Она может также применяться для активизации некоторых более экспериментальных функций поддержки Unicode. Наша долгосрочная цель состоит в превращении прагмы `utf8` в код пустой операции.

Прагма `use bytes` никогда не станет кодом пустой операции. Она не только необходима для байт-ориентированного кода, но и оказывает побочный эффект, определяя байт-ориентированные оболочки для некоторых функций, применяемые вне области действия `use bytes`. На момент написания

данной книги единственная такая оболочка определена для `length`, но со временем, вероятно, появятся другие. Для использования такой оболочки нужно сказать:

```
use bytes ();   # Загрузить оболочки без импорта байтовой семантики.
...
$charlen =        length("\x{ffff_ffff}");   # Возвращает 1.
$bytelen = bytes::length("\x{ffff_ffff}");   # Возвращает 7.
```

Вне области видимости объявления `use bytes` Perl версии 5.6 работает (или, по крайней мере, должен работать) следующим образом:

- В строках и шаблонах теперь могут содержаться символы с порядковым значением больше 255:

  ```
  use utf8;
  $convergence = "☞ ☜";
  ```

  В предположении, что вы пишете программу в редакторе с поддержкой Unicode, такие символы обычно встречаются непосредственно в литеральных строках в виде символов UTF-8. В настоящее время для того, чтобы иметь возможность использовать UTF-8 в литералах, необходимо объявить `use utf8` в начале программы.

  Если редактора с поддержкой Unicode нет, всегда можно задать конкретный символ в ASCII, используя расширение в виде нотации \x. Символ в диапазоне Latin-1 можно записать в виде \x{ab} или \xab, но если число содержит больше двух шестнадцатеричных цифр, то его следует заключить в фигурные скобки. Символы Unicode задаются с помощью шестнадцатеричного кода в фигурных скобках после \x. Например, смайлик ☺ в Unicode обозначается как \x{263A}. В Perl нет синтаксических конструкций, предполагающих, что символы Unicode состоят ровно из 16 бит, поэтому нельзя, как в других языках, написать \u263A; ближайшим эквивалентом служит \x{263A}.

  Как вставлять именованные символы с помощью `\N{`*`CHARNAME`*`}`, см. в описании прагмы `use charnames` в главе 31 «Модули прагм».

- Идентификаторы в сценарии Perl могут содержать буквенно-цифровые символы Unicode, в том числе идеографические символы (идеограммы):

  ```
  use utf8;
  $人++;        # Ребенок родился.[1]
  ```

  И снова для распознавания UTF-8 в сценарии требуется (пока) `use utf8`. В данное время вы можете самостоятельно принимать решение об использовании канонических форм символов – Perl не пытается (пока) канонизировать имена переменных вместо вас. Мы рекомендуем применять в программах канонизацию согласно нормализованной форме C, поскольку

[1] Иероглиф 人 означает «человек». – *Примеч. науч. ред.*

ку когда-нибудь Perl будет делать это по умолчанию. См. на *www.unicode.org* последний технический доклад по канонизации.

- Регулярные выражения ищут символы, а не байты. К примеру, точка соответствует символу, а не байту. Если Консорциум Unicode когда-нибудь одобрит алфавит Tengwar, то (несмотря на то, что такие символы представляются четырьмя байтами UTF-8) он будет соответствовать поиску:

```
"\N{TENGWAR LETTER SILME NUQUERNA}" =~ /^.$/[1]
```

Шаблон \C служит для обеспечения поиска одного байта («char» в C, отсюда и \C). Применяйте \C с осторожностью, поскольку он может вызвать рассинхронизацию с границами символов в строке и будет получена ошибка «Malformed UTF-8 character» (неправильный символ UTF-8). Нельзя использовать \C в квадратных скобках, поскольку он не представляет какой-либо конкретный символ или группу символов.

Классы символов в регулярных выражениях сравниваются с символами, а не байтами, и сравниваются со свойствами символов, заданными в базе данных свойств Unicode. Поэтому \w может применяться для поиска идеографического символа (идеограммы):

```
"人" =~ /\w/  </para>
```

- Можно использовать именованные свойства и диапазоны блоков Unicode как классы символов посредством новых конструкций \p (соответствует свойству) и \P (не соответствует свойству). Например, \p{Lu} соответствует любому символу в верхнем регистре в смысле Unicode, а \p{M} соответствует любому символу маркировки. Для однобуквенных свойств скобки могут быть опущены, поэтому символы маркировки можно искать также с помощью \pM. Есть много предопределенных классов символов, например \p{IsMirrored} и \p{InTibetan}:

```
"\N{greek:Iota}" =~ /\p{Lu}/
```

Можно также использовать \p и \P в квадратных скобках классов символов. (В версии Perl 5.6.0 требуется объявить use utf8, чтобы свойства символов правильно работали. В дальнейшем это ограничение будет снято.) О подробностях, связанных с поиском свойств Unicode, см. главу 5 «Поиск по шаблону».

- Особый шаблон \X ищет расширенные последовательности Unicode (на языке Стандарта – «комбинированная символьная последовательность» (combining character sequence)), в которых первый символ является базовым, а последующие символы – маркерами, примененными к базовому символу. Это эквивалентно (?:\PM\pM*):

```
"o\N{COMBINING TILDE BELOW}" =~ /\X/
```

[1] Tengwar letter silme nuquerna – буква («тенгва») «сильме перевернутая» ϼ из алфавита Tengwar. – *Примеч. ред.*

Не разрешается использовать \X в квадратных скобках, поскольку он может найти несколько символов и не соответствует какому-либо конкретному символу или набору символов.

- Оператор tr/// транслитерирует символы вместо байтов. Чтобы преобразовать все символы вне диапазона Latin-1 в вопросительный знак, можно сказать:

```
tr/\0-\x{10ffff}/\0-\xff?/;       # utf8 в символы latin1
```

- Операторы перевода регистра, получая на входе символы, обращаются к таблицам перевода регистра Unicode. Обратите внимание, что uc переводит в верхний регистр, а ucfirst – в заглавный регистр (для языков, в которых такое различие есть). Естественно, что соответствующие последовательности с обратной косой чертой имеют ту же семантику:

```
$x = "\u$word";       # перевести в заглавный регистр первую букву $word
$x = "\U$word";       # перевести в верхний регистр $word
$x = "\l$word";       # перевести в нижний регистр первую букву $word
$x = "\L$word";       # перевести в нижний регистр  $word
```

Будьте осторожны, поскольку эти таблицы не пытаются обеспечить полное соответствие прямого и обратного преобразований для каждого случая, особенно в языках, использующих различное число символов в заглавном или верхнем регистрах и эквивалентной букве нижнего регистра. Как сказано в стандарте, хотя сами свойства регистра являются нормативными, соответствия регистров несут только информационную нагрузку.

- Большинство операторов, работающих с позицией в строке или ее длиной, автоматически переходят на использование позиций символов, в том числе chop, substr, pos, index, rindex, sprintf, write и length. В число операторов, которые умышленно не переключаются, входят vec, pack и unpack. В число операторов, которым это безразлично, входят chomp, а также все операторы, рассматривающие строку как битовую корзину, например, sort по умолчанию и операторы, обрабатывающие имена файлов.

```
use bytes;
$bytelen = length("I do &合氣道.");   # 15 байт
no bytes;
$charlen = length("I do 合氣道.");   # но 9 символов
```

- Буквы "c" и "C" в pack/unpack *не* меняются, поскольку часто используются для байт-ориентированных форматов. (И снова представьте себе "char" в языке C.) Однако есть новый спецификатор "U", осуществляющий преобразование между символами UTF-8 и целыми числами:

```
pack("U*", 1, 20, 300, 4000) eq v1.20.300.4000
```

- Функции chr и ord работают с символами:

```
chr(1).chr(20).chr(300).chr(4000) eq v1.20.300.4000
```

Иными словами, `chr` и `ord` действуют как `pack("U")` и `unpack("U")`, а не как `pack("C")` и `unpack("C")`. На самом деле, две последние осуществляют теперь эмуляцию байт-ориентированных `chr` и `ord`, если вам слишком лень написать `use bytes`.

- И наконец `scalar reverse` обращает посимвольно, а не побайтно:

```
"☞ ☜" eq reverse "☜ ☞" </para>
```

Если посмотреть в каталог *ПУТЬ_К_PERLLIB/unicode*, то можно найти ряд файлов, имеющих отношение к определению вышеописанной семантики. База данных свойств Unicode Консорциума Unicode находится в файле с именем *Unicode.300* (для Unicode 3.0). Этот файл уже преобразован *mktables.PL* в кучу маленьких файлов *.pl* в том же самом каталоге (и подкаталогах *Is/*, *In/* и *To/*), некоторые из которых Perl автоматически проглатывает, чтобы реализовывать такие вещи, как `\p` (см. каталоги *Is/* и *In/*) и `uc` (см. каталог *To/*). Другие файлы заглатываются такими модулями, как прагма `use charnames` (см. *Name.pl*). Но во время написания этой книги там было еще много файлов, ожидавших, когда вы создадите модуль для доступа к ним:

ArabLink.pl
ArabLnkGrp.pl
Bidirectional.pl
Block.pl
Category.pl
CombiningClass.pl
Decomposition.pl
JamoShort.pl
Number.pl
To/Digit.pl

Значительно более удобочитаемая сводка по Unicode с многочисленными ссылками находится в *ПУТЬ_К_PERLLIB/unicode/Unicode3.html*.

Заметьте, что когда консорциум Unicode выпускает новую версию, некоторые из этих имен могут изменяться, поэтому следует быть осмотрительным. Найти *ПУТЬ_К_PERLLIB* можно с помощью следующего заклинания:

```
% perl -MConfig -le 'print $Config{privlib}'
```

Практически обо всем, что относится к Unicode, можно узнать, выписав *The Unicode Standard, Version 3.0* (ISBN 0-201-61633-5).

Осторожно, работают 人

Во время написания этой книги (т. е. речь идет о версии Perl 5.6.0) сохранялся ряд предостережений по использованию Unicode. (Проверьте возможные обновления в своей электронной документации.)

- Существующий компилятор регулярных выражений не создает полиморфных кодов операций. Это означает, что некоторый шаблон определяется как способный искать символы Unicode во время его компиляции (на основании присутствия в шаблоне символов Unicode), а не во время поиска на этапе исполнения. Это нужно изменить, чтобы настраиваться на поиск Unicode, когда строка для поиска представлена в Unicode.
- В настоящее время нет простого способа пометки данных, читаемых из файла или другого внешнего источника, как utf8. На этом в ближайшем будущем будут сосредоточены большие усилия, и положение, возможно, уже изменится, когда вы будете читать эту книгу.
- Нет метода для автоматического перевода входных или выходных данных в кодировку, отличную от UTF-8. Однако это планируется сделать в ближайшее время, поэтому сверьтесь со своей электронной документацией.
- Применение национальных установок с utf8 может привести к неожиданным результатам. В настоящее время делаются некоторые попытки использования 8-разрядной информации о национальных настройках с символами в диапазоне `0..255`, но оно с очевидностью некорректно для национальных установок, затрагивающих символы за пределами этого диапазона (при отображении в Unicode). При этом также происходит замедление работы. Настоятельно рекомендуется избегать национальных установок.

Unicode – это забавно, нужно только корректно определить, что такое забавно.