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 13

В погоне за счастьем 13
Что обновилось в этом издании 18
Стандартный дистрибутив 19
Электронная документация 21
Печатная документация 25
Дополнительные источники 27
Соглашения, принятые в этой книге 28
Благодарности 29
Хотим услышать ваши отзывы 30

Часть I. Общий обзор 31

1. Обзор Perl 33

Введение 33
Естественные и искусственные языки 34
Стандартный пример 48
Дескрипторы файлов 52
Операторы 54
Управляющие структуры 61
Регулярные выражения 68
Обработка списков 74
Чего вы не знаете, то вам не навредит (сильно) 76

Часть II. Анатомия Perl 79

2. Всякая всячина 81

Атомы 82
Молекулы 83
Встроенные типы данных 85
Переменные 87
Имена 88
Скалярные значения 94
Контекст 105

Списочные значения и массивы 108
Хеши 113
Таблицы имен и дескрипторы файлов 115
Операторы ввода 116

3. Унарные и бинарные операторы 123

Термы и списковые операторы (слева) 126
Оператор «стрелка» 127
Автоинкрементирование и автодекрементирование 128
Возведение в степень 129
Идеографические унарные операторы 129
Операторы связывания 130
Мультипликативные операторы 131
Аддитивные операторы 132
Операторы сдвига 133
Именованные унарные операторы и операторы проверки файлов 133
Операторы сравнения 138
Операторы равенства 138
Операторы поразрядного действия 139
Логические операторы в стиле C (короткого действия) 140
Оператор диапазона 141
Условный оператор 142
Операторы присваивания 144
Оператор запятой 146
Списковые операторы (справа) 147
Логические and, or, not и xor 147
Операторы C, отсутствующие в Perl 148

4. Операторы и объявления 149

Простые операторы 150
Составные операторы 151
Операторы if и unless 153
Операторы Loop 154
Голые блоки 162
goto 165
Глобальные объявления 166
Объявления с областью видимости 168
Прагмы 176

5. Поиск по шаблону 179

Бестиарий регулярных выражений 180
Операторы поиска по шаблону 183
Метасимволы и метазнаки 199
Классы символов 208

Квантификаторы 219
Позиции 222
Захват и кластеризация 225
Чередование 230
Управление процессом 232
Сложные шаблоны 247

6. Подпрограммы 262

Синтаксис 262
Семантика 264
Передача ссылок 269
Прототипы 271
Атрибуты подпрограмм 277

7. Форматы 280

Переменные форматов 284
Нижние колонтитулы 286

8. Ссылки 288

Что такое ссылка? 289
Создание ссылок 291
Использование жестких ссылок 297
Символические ссылки 310
Фигурные скобки, квадратные скобки и кавычки 311

9. Структуры данных 315

Массивы массивов 316
Хеши массивов 323
Массивы хешей 325
Хеши хешей 327
Хеши функций 330
Более сложные записи 331
Сохранение структур данных 334

10. Пакеты 335

Таблицы имен 340
Автозагрузка 344

11. Модули 346

Использование модулей 346
Создание модулей 349
Замещение встроенных функций 353

12. Объекты 355

Краткая памятка по объектно-ориентированному жаргону 355
Система объектов Perl 357
Вызов методов 358
Создание объектов 364
Наследование классов 369
Деструкторы экземпляров 378
Управление данными экземпляров 380
Управление данными класса 391
Резюме 394

13. Перегрузка 395

Прагма overload 396
Обработчики перегрузки 397
Перегружаемые операторы 398
Конструктор копий (=) 405
Когда обработчик перегрузки отсутствует (nomethod и fallback) 406
Константы перегрузки 407
Открытые функции перегрузки 409
Наследование и перегрузка 409
Перегрузка на этапе исполнения 410
Диагностика перегрузки 410

14. Связанные переменные 411

Связывание скаляров 413
Связывание массивов 421
Связывание хешей 427
Связывание указателей файлов 433
Тонкая ловушка при отвязывании 444
Модули для связывания в CPAN 447

Часть III. Perl как технология 449

15. Unicode 451

Байты и символы 453
Действие символьной семантики 455
Осторожно, работают 人 459

16. Межпроцессное взаимодействие 461

Сигналы 462
Файлы 469
Каналы 477
System V IPC 485
Сокеты 489

17. Потоки 497

Модель процессов 498
Модель потоков 499

18. Компиляция 515

Жизненный цикл программ на Perl 516
Компилирование кода 518
Выполнение кода 524
Серверы компиляторов 527
Генераторы кода 528
Средства разработки кода 530
Компилятор и интерпретатор: авангардизм и ретро 532

19. Интерфейс командной строки 537

Обработка команд 537
Переменные окружения 554

20. Отладчик Perl 557

Использование отладчика 558
Команды отладчика 560
Настройка отладчика 569
Автоматическое выполнение 573
Поддержка отладчика 575
Профайлер Perl 578

21. Внутри и снаружи 582

Как работает Perl 583
Внутренние типы данных 583
Расширение Perl (использование C из Perl) 584
Вызов Perl из C 591
Мораль «сей басни» 596

Часть IV. Perl как культура 597

22. CPAN 599

Каталог modules архива CPAN 601
Использование модулей CPAN 604
Создание модулей CPAN 606

23. Защита данных 609

Обработка ненадежных данных 610
Обработка ошибок синхронизации 622
Работа с ненадежным кодом 629

24. Распространенные приемы программирования 639

Обычные промахи новичков 639
Эффективность 648
Стиль программирования 658
Свободный разговор на Perl 662
Генерирование программ 672

25. Переносимость программ Perl 677

Перевод строки 678
Остроконечники, тупоконечники и ширина чисел 680
Файлы и файловые системы 681
Взаимодействие с системой 682
Межпроцессное взаимодействие (IPC) 683
Внешние подпрограммы (XS) 683
Стандартные модули 684
Дата и время 684
Многоязычность 685
Стиль 685

26. Документация в формате POD 686

О pod в двух словах 686
Трансляторы и модули Pod 695
Создание собственных инструментов для работы с pod 697
Ловушки pod 700
Документирование программ Perl 701

27. Культура Perl 703

История развития 703
Поэзия Perl 706

Часть V. Справочный материал 709

28. Специальные имена 711

Специальные имена, сгруппированные по типам 712
Специальные переменные в алфавитном порядке 714

29. Функции 736

Функции Perl по категориям 739
Функции Perl в алфавитном порядке 741

30. Стандартная библиотека Perl 895

Библиотековедение 895
Обзор библиотеки Perl 897

31. Модули прагм 900

use attributes 901
use autouse 902
use base 903
use blib 904
use bytes 904
use charnames 905
use constant 906
use diagnostics 908
use fields 910
use filetest 912
use integer 913
use less 914
use lib 914
use locale 916
use open 917
use overload 917
use re 918
use sigtrap 919
use strict 922
use subs 925
use vars 925
use warnings 926

32. Стандартные модули 930

Перечень по типу 931
Benchmark 941
Carp 943
CGI 944
CGI::Carp 944
Class::Struct 945
Config 946
CPAN 947
Cwd 947
Data::Dumper 948
DB_File 948
Dumpvalue 950
English 950
Errno 951
Exporter 951
Fatal 952
Fcntl 953
File::Basename 953
File::Compare 954
File::Copy 955
File::Find 955

File::Glob 956
File::Spec 959
File::stat 960
File::Temp 960
FileHandle 961
Getopt::Long 964
Getopt::Std 965
IO::Socket 966
IPC::Open2 967
IPC::Open3 967
Math::BigInt 968
Math::Complex 969
Math::Trig 969
Net::hostent 969
POSIX 970
Safe 972
Socket 973
Symbol 974
Sys::Hostname 975
Sys::Syslog 976
Term::Cap 977
Text::Wrap 978
Time::Local 978
Time::localtime 979
User::grent 980
User::pwent 980

33. Диагностические сообщения 982

Глоссарий 1045

Алфавитный указатель 1084

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.

I

Общий обзор

1

Обзор Perl

Введение

Мы полагаем, что Perl является языком, который легко изучить и использовать, и надеемся убедить вас в своей правоте. Одним из обстоятельств, делающих Perl простым, является то, что не нужно произносить много предварительных слов, прежде чем сказать то, что вы собираетесь сказать. Во многих языках программирования необходимо объявить типы, переменные и подпрограммы, которые предполагается использовать, прежде чем написать хотя бы одну строчку исполняемого кода. И это правильно, что необходимы объявления, если речь идет о сложных задачах, требующих сложных структур данных. Но для многих более простых, повседневных задач хотелось бы иметь язык программирования, на котором можно просто сказать:

```
print "Howdy, world!\n";
```

и программа это сделает.

Таким языком и является Perl. На практике этот пример представляет собой законченную программу[1], и если послать ее интерпретатору Perl, тот выведет на экран `"Howdy, world"`. (`\n` в этом примере обеспечивает перевод строки в конце вывода.)

---

[1] Или «сценарий» (script), «приложение» (application), «исполняемый объект» (executable), «штуковина» (doohickey). Кому что больше нравится.

Вот и все. *После* того как вы сказали то, что хотели, тоже не требуется произносить много слов. В отличие от многих языков программирования, Perl считает, что «вывалиться» из программы – нормальный способ ее завершения. Конечно, при желании вы *можете* явно вызвать функцию `exit`, как *можете* и объявить некоторые переменные, либо даже сделать *обязательным* объявление всех переменных. Но право выбора принадлежит вам. В Perl можно поступать «так, как должно», как бы вы это ни определили.

Есть много других причин, по которым Perl легко использовать, но перечислять сейчас их все бессмысленно, поскольку этому и посвящена наша книга. Говорят, что дьявол проявляет себя в частностях, но Perl старается облегчить вашу жизнь и в трудных ситуациях. На любом уровне Perl заботится о том, чтобы доставить вас туда, куда нужно, с минимумом суеты и максимумом удовольствия. Вот почему многие программисты на Perl ходят с глупыми ухмылками на физиономиях.

Эта глава представляет собой обзор, поэтому мы не пытаемся представить Perl с рациональной стороны. Равно как не претендуем на полноту изложения или логичность. Для этого предназначены следующие главы. Вулканическим натурам, андроидам и другим аналогично настроенным лицам следует пропустить этот обзор и перейти сразу к главе 2 «Всякая всячина», где плотность информации выше. С другой стороны, тем, кому нужен учебник с тщательно выстроенным продвижением вперед, возможно, следует обратиться к прекрасной книге Рэндала Шварца (Randal Schwartz) «Learning Perl» (опубликованную O'Reilly & Associates). Но и данную книгу пока не выбрасывайте.

Эта глава представляет Perl *другому* полушарию вашего мозга, тому, которое называют ассоциативным, артистическим, управляющим эмоциями или просто впитывающим. Для этого Perl рассматривается с разных точек зрения, чтобы дать читателю о нем такое же ясное представление, какое слепцы могли получить о слоне (ощупывая его). Возможно, нам удастся больше: мы все-таки имеем дело с верблюдом (см. обложку). Надеемся, что хотя бы одно из этих представлений Perl поможет вам меньше горбатиться.

Естественные и искусственные языки

Языки были изобретены людьми и для блага людей. В анналах компьютерных наук этот факт случайно оказался позабытым.[1] Поскольку (допуская вольность речи) можно сказать, что создателем Perl оказался лингвист, предполагалось, что этот язык будет работать так же гладко, как естественные языки. На деле это охватывает множество аспектов, поскольку естественные языки хорошо работают одновременно на нескольких уровнях. Можно было бы перечислить многие из этих лингвистических принципов, но самым важным принципом архитектуры языка является тот, что прос-

[1] Точнее, об этом факте вспоминают от случая к случаю.

тые вещи должны делаться просто, а сложные вещи должны быть реализуемы. (На самом деле, здесь два принципа.) Они могут показаться вам очевидными, однако многие языки программирования не удовлетворяют хотя бы одному из этих условий.

Естественные языки удовлетворяют обоим условиям, поскольку люди непрерывно пытаются выражать как простые, так и сложные вещи, и поэтому язык развивается так, чтобы справляться с обеими задачами. Возможность развития была заложена в Perl на этапе его конструирования, и он действительно эволюционировал со времени своего создания. Вклад в эволюцию Perl вносили многие люди на протяжении ряда лет. Мы часто шутим, что верблюд – это лошадь, которую проектировала комиссия, но если вдуматься, то верблюд достаточно хорошо приспособлен для жизни в пустыне. Эволюция привела верблюда к относительной самодостаточности. (С другой стороны, в результате эволюции верблюд не стал лучше пахнуть. Как и Perl.) Это одна из многих странных причин, по которым мы сделали верблюда талисманом для Perl, но к лингвистике это имеет отдаленное отношение.

Когда кто-то произносит слово «лингвистика», у многих возникает одно из двух представлений: либо они начинают думать о словах, либо начинают думать о предложениях. Но слова и предложения являются лишь двумя удобными способами членения речи. Те и другие можно разбить на более мелкие смысловые единицы или объединить в более крупные. А смысл каждой единицы существенно зависит от синтаксического, семантического и практического контекста, в котором находится эта единица. В естественном языке есть слова разных видов: существительные, глаголы и т. д. Если произнести слово «dog» (собака) изолированно, в отрыве от контекста, представляется существительное, хотя можно использовать это слово и в другом смысле. Это значит, что существительное может выполнять функции глагола, прилагательного или наречия, в зависимости от контекста. If you dog a dog during the dog days of summer, you'll be a dog tired dogcatcher.[1]

Perl также по-разному оценивает слова в зависимости от контекста. Далее мы увидим, как он это делает. Помните просто, что Perl пытается понять, о чем идет речь, как это делает любой хороший слушатель. Perl усиленно старается соблюсти со своей стороны условия сделки. Скажите просто, чего вы хотите, и Perl обычно «сообразит». (Если только вы не говорите чепухи – анализатор Perl понимает Perl значительно лучше, чем английский или суахили.)

Но вернемся к существительным. Существительное может быть именем конкретного объекта или родового класса объектов без указания того, на который из них производится ссылка в данный момент. Это различие проводится в большинстве языков программирования, когда конкретная величина называется значением, а общая – переменной. Значение где-то существует,

---

[1] Пример многозначности dog и зависимости от контекста. – *Примеч. перев.*

Возможно, читатель устал, как собака, от всей этой лингвистической трескотни. Но мы хотим, чтобы было понятно, чем Perl отличается от обычных компьютерных языков!

а вот переменная в течение срока своей жизни связывается с одним или более значениями. Поэтому, кто бы ни осуществлял интерпретацию, переменная должна отслеживать эту связь. Интерпретатор может находиться в мозгу человека или в компьютере.

Синтаксис переменных

Переменная является просто удобным способом хранения чего-либо, именованным местонахождением, в котором программист может найти нечто требующееся ему, когда оно позднее понадобится. Как и в реальной жизни, есть разного типа места для хранения, одни из которых достаточно скрыты, а другие открыты для всех. Некоторые хранилища временные, а другие – более постоянные. Ученые в области вычислительной техники любят говорить об «области видимости» переменных, подразумевая под этим выражением именно то, что в нем сказано. В Perl имеются достаточно удобные средства решения проблем области видимости, с которыми читатель в свое время будет иметь удовольствие познакомиться. Оно еще не наступило. (Те, кто испытывает любопытство, могут поискать прилагательные `local`, `my` и `our` в главе 29 «Функции», либо заглянуть в раздел «Объявления, ограниченные областью видимости» главы 4 «Операторы и объявления».)

В данный момент полезнее провести классификацию переменных по типу данных, которые они могут содержать. Как и в английском языке, в Perl основное различие проводится между данными в единственном и во множественном числе. Строки и числа являются данными в единственном числе, а списки строк или чисел – во множественном. (А когда мы доберемся до объектно-ориентированного программирования, вы обнаружите, что типичный объект выглядит единичным снаружи, но множественным изнутри – как группа студентов.) Переменную в единственном числе мы называем *скаляром*, а во множественном – *массивом*. Поскольку строка может храниться в скалярной переменной, можно написать такую несколько более длинную (и снабженную комментариями) версию нашего первого примера:

```
$phrase = "Howdy, world!\n";          # Присвоить значение переменной.
print $phrase;                        # Вывести переменную
```

Обратите внимание, что нам не пришлось предварительно определять, какой тип имеет переменная `$phrase`. Символ `$` указывает Perl на то, что `phrase` представляет собой скалярную переменную, т. е. содержит единственное значение. Напротив, массив должен начинаться с символа `@`. (Запоминание облегчается тем, что `$` является стилизованным «s», от «scalar», тогда как `@` – это стилизованное «a», от «array».)

В Perl имеются некоторые другие типы переменных с такими непривлекательными именами, как «хеш» (hash), «дескриптор» (handle) и «запись таблицы имен» (typeglob). Подобно скалярам и массивам, переменным этих типов также предшествуют разыменовывающие префиксы (funny characters – забавные символы). Вот полный список разыменовывающих префиксов, которые могут вам встретиться:

Тип	Символ	Пример	Что обозначает
Скаляр	$	$cents	Отдельное значение (число или строку)
Массив	@	@large	Список значений с числовым ключом
Хеш	%	%interest	Группа значений, ключом доступа к которым служит строка
Подпрограмма	&	&how	Фрагмент кода Perl, который может быть вызван
Typeglob	*	*struck	Все, что имеет имя struck

Некоторые пуристы языка указывают на эти разыменовывающие префиксы как на причину отвращения к Perl. Это поверхностный взгляд. У этих символов много достоинств, не последним из которых является возможность подстановки, или интерполяции (interpolation), этих символов в строки без всякого дополнительного синтаксиса. Кроме того, сценарии Perl легко читать (тем, кто взял на себя труд изучить Perl!), поскольку существительные отличаются от глаголов. В язык могут добавляться новые глаголы без ущерба для старых сценариев. (Мы уже говорили, что в Perl на этапе проектирования была заложена возможность развития.) И аналогия с существительными приведена не зря: есть масса примеров в английском и других языках, когда требуются грамматические маркеры существительных. Вот так мы думаем! (Мы думаем.)

Единственное число

В приведенном выше примере мы видели, что скаляру можно присваивать новое значение с помощью оператора = подобно тому, как это делается во многих других языках программирования. Скалярным переменным можно присваивать скалярные значения любого вида: целые числа, числа с плавающей запятой, строки и даже такие таинственные вещи, как ссылки на другие переменные или объекты. Есть много способов создания этих значений, которые должны присваиваться.

Как и в оболочке Unix[1], можно использовать различные механизмы расстановки кавычек для создания значений разного вида. Двойные кавычки (double quotes) производят *интерполяцию переменных*[2] и *интерполяцию обратной косой черты* (например, превращение \n в символ перевода строки), тогда как одиночные кавычки подавляют интерполяцию. А обратные

[1] Здесь и всюду, говоря «Unix», мы подразумеваем любую операционную систему, похожую на Unix, в том числе BSD, Linux и, конечно, Unix.

[2] Иногда называемую программистами для оболочки «подстановкой», но мы предпочитаем сохранить это слово для использования в Perl в других целях. Поэтому, пожалуйста, называйте это интерполяцией. Мы используем этот термин в текстологическом смысле («этот отрывок является гностической интерполяцией»), а не в математическом («данная точка на графике получена интерполяцией между двумя другими точками»).

кавычки (наклоненные влево) вызывают выполнение внешней программы и возврат ее выдачи, которая перехватывается как одна строка, содержащая все строки выдачи.

```
$answer = 42;                # целое число
$pi = 3.14159265;            # "действительное" число
$avocados = 6.02e23;         # научное обозначение
$pet = "Camel";              # строка
$sign = "I love my $pet";    # строка с интерполяцией
$cost = 'It costs $100';     # строка без интерполяции
$thence = $whence;           # значение другой переменной
$salsa = $moles * $avocados; # гастрохимическое выражение
$exit = system("vi $file");  # числовой статус выполнения команды
$cwd = `pwd`;                # строка выдачи команды
```

И хотя мы еще не рассказали о необычных значениях, следует указать, что скаляры могут содержать ссылки на другие структуры данных, в том числе подпрограммы и объекты.

```
$ary = \@myarray;            # ссылка на именованный массив
$hsh = \%myhash;             # ссылка на именованный хеш
$sub = \&mysub;              # ссылка на именованную подпрограмму

$ary = [1,2,3,4,5];          # ссылка на анонимный массив
$hsh = {Na => 19, Cl => 35}; # ссылка на анонимный хеш
$sub = sub { print $state }; # ссылка на анонимную подпрограмму

$fido = new Camel "Amelia";  # ссылка на объект
```

Если используется переменная, которой еще ни разу не присваивалось значение, то неинициализированная переменная автоматически начинает существовать, когда в ней возникает необходимость. В соответствии с принципом минимизации неожиданности переменная создается с нулевым значением: "" или 0. В зависимости от контекста использования, переменные автоматически интерпретируются как строки, числа или значения «true» (истина) или «false» (ложь), обычно называемые булевыми значениями. Вспомните, как важен контекст в естественном языке. В Perl разные операторы предполагают получение в качестве параметров различного вида одиночных значений, и мы будем говорить, что данные операторы предоставляют для этих параметров скалярный контекст. Иногда мы будем более конкретно говорить, что оператор предоставляет числовой, строковый или булев контекст для этих параметров. (Ниже мы расскажем о списочном контексте, который противопоставляется скалярному.) Perl автоматически преобразует данные к виду, требуемому текущим контекстом, в соответствии со здравым смыслом. Предположим, например, что вы сказали следующее:

```
$camels = '123';
print $camels + 1, "\n";
```

Первоначальным значением $camels является строка, но она преобразуется в число, чтобы прибавить к ней единицу, а затем обратно в строку, чтобы вы-

вести ее как 124. Символ перевода строки, представленный "\n", тоже находится в строковом контексте, но поскольку он изначально является строкой, преобразование не требуется. Обратите, однако, внимание на двойные кавычки: использование одиночных кавычек, т. е. '\n', привело бы к появлению строки из двух символов, а именно косой черты и «n», которая никак не может представлять перевод строки.

Итак, в некотором смысле двойные и одиночные кавычки предоставляют еще один способ задания контекста. Интерпретация содержимого закавыченной строки зависит от того, какого вида кавычки используются. (Далее мы рассмотрим некоторые другие операторы, синтаксически работающие как кавычки, но использующие строку некоторым особым способом, например, для поиска по шаблону или подстановки. Все они тоже действуют как строки в двойных кавычках. Контекст *двойных кавычек* является в Perl «интерполирующим» и предоставляется многими операторами, которые не напоминают двойные кавычки.)

Аналогично ссылка ведет себя как ссылка в «разыменовывающем» контексте и как обычная скалярная величина в других случаях. Например, можно сказать так:

```
$fido = new Camel "Amelia";
if (not $fido) { die "dead camel"; }
$fido->saddle();
```

Здесь создается ссылка на объект Camel, которая помещается в переменную $fido. В следующей строке мы проверяем значение $fido как булевой величины на равенство «true» и возбуждаем исключительную ситуацию (т. е. жалуемся, что она не истинна); в данном случае это означает, что конструктор new Camel не смог создать требуемый объект Camel. Однако в последней строке мы обращаемся с $fido как со ссылкой, запрашивая метод saddle() для объекта, содержащегося в $fido и представляющего собой Camel, поэтому Perl ищет метод saddle() для объектов Camel. Подробнее об этом сказано ниже. Сейчас просто запомните, что в Perl важен контекст, поскольку благодаря ему Perl узнает, что вам требуется, без ваших явных указаний, необходимых во многих других языках программирования.

Множественное число

В некоторых типах переменных содержатся множественные значения, связанные вместе логически. В Perl есть два типа многозначных переменных: массивы и хеши. Во многих случаях они ведут себя как скаляры: они начинают существовать, когда в них возникает надобность, ничего не содержа в себе. Но в отличие от скаляров, при присвоении им значений они предоставляют *списочный* контекст в правой части оператора присваивания, а не скалярный.

Массивы и хеши также различаются между собой. Массив следует использовать, когда нужно найти что-либо по номеру. Хеш применяется для того, чтобы найти что-либо по имени. Эти два понятия дополняют друг друга.

Часто можно наблюдать использование массива для перевода номера месяца в название и соответствующего хеша для перевода названия месяца обратно в номер. (Однако хеши могут содержать не только числа. К примеру, можно создать хеш, который будет переводить названия месяцев в названия камней, соответствующих месяцам.)

Массивы. *Массив* – это упорядоченный список скаляров, к которым обращаются[1] по их местоположению в списке. Список может содержать числа, строки или смесь того и другого. (Он может также содержать ссылки на вложенные массивы (субмассивы) и вложенные хеши (субхеши).) Чтобы присвоить массиву значение в виде списка, нужно просто объединить вместе значения с помощью круглых скобок:

```
@home = ("кушетка", "стул", "стол", "печка");
```

И наоборот, используя `@home` в списочном контексте, например, с правой стороны присвоения списку, мы получим на выходе тот самый список, который ввели. Например, присвоить значения, взятые из массива, четырем скалярным переменным можно так:

```
($potato, $lift, $tennis, $pipe) = @home;
```

Это называется присвоением списка. Логически оно происходит параллельно, поэтому можно обменять значениями две переменные с помощью:

```
($alpha,$omega) = ($omega,$alpha);
```

Как и в языке C, нумерация массивов начинается с нуля (zero-based), поэтому, работая с элементами массива с первого по четвертый, следует обращаться к ним по индексам от 0 до 3.[2] Индексы массивов заключаются в квадратные скобки [вот такие], поэтому при выборе отдельного элемента массива нужно ссылаться на него так – `$home[n]`, где `n` является значением индекса (на единицу меньшим номера элемента), который нам нужен. Посмотрите на следующий пример. Поскольку мы имеем дело со скалярным элементом, ему обязательно должен предшествовать символ `$`.

Если вы хотите присваивать значения элементам массива по одному, предшествующее присвоение можно записать так:

```
$home[0] = "кушетка";
$home[1] = "стул";
$home[2] = "стол";
$home[3] = "печка";
```

---

[1] Или ключом (индексом, нижним индексом) для доступа к которым является местоположение. Выберите то, что вам нравится.

[2] Если это кажется вам непривычным, считайте индекс смещением, т. е. числом предшествующих элементов массива. Очевидно, перед первым элементом нет других элементов, поэтому его смещение равно 0. Так уж думают компьютеры. (Мы так думаем.)

Поскольку массивы упорядочены, над ними можно выполнять полезные действия, такие как операции со стеком push и pop. Стек, в сущности, является просто упорядоченным списком с началом и концом. С концом в особенности. Perl рассматривает конец массива как вершину стека. (Хотя большинство программистов на Perl представляют себе массивы горизонтальными, с вершиной стека, находящейся справа.)

Хеши. *Хеш* представляет собой неупорядоченный набор скаляров, к которым обращаются[1] по некоторому строковому значению, связанному с каждым скаляром. По этой причине хеши часто называют *ассоциативными массивами*. Но это слишком длинное название для ввода с клавиатуры, а говорится о них так часто, что мы решили дать им какое-нибудь краткое и броское название. Другой причиной, по которой мы выбрали название «хеш», является желание подчеркнуть тот факт, что они неупорядочены. (Так совпало, что в их внутренней реализации используется поиск по хеш-таблице, в результате чего работа с ними происходит столь быстро вне зависимости от числа помещенных в них значений.) Однако применить к хешу push или pop нельзя: это не имеет смысла. У хеша нет начала и конца. Тем не менее, хеши являются очень мощным и полезным средством. Пока вы не начали думать на языке хешей, вы, в сущности, не начали думать на Perl. На рис. 1.1 показаны упорядоченные элементы массива и неупорядоченные (но поименованные) элементы хеша.

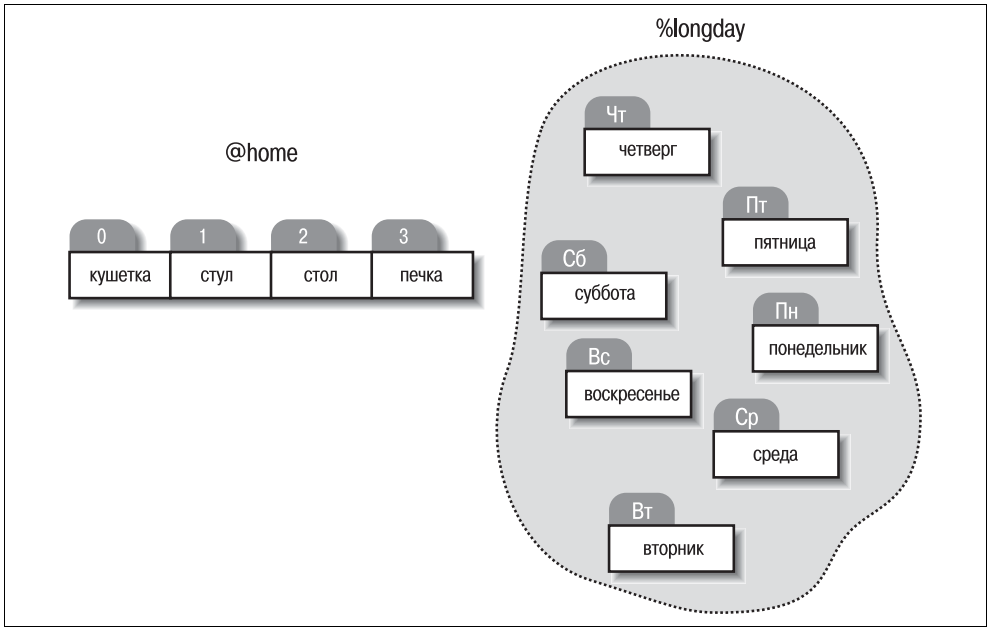

Рис. 1.1. Массив и хеш

[1] Или ключом (индексом, нижним индексом) для доступа к которым является местоположение. Выберите то, что вам нравится.