

Содержание

Предисловие	19
Для кого предназначена эта книга.....	20
Как использовать эту книгу.....	21
Дополнительные источники информации	22
Соглашения, используемые в данной книге.....	23
Учебные примеры	24
Авторские благодарности	25
 I Контекст.....	 27
1 Философские вопросы.....	29
1.1 Культура? Какая культура?	29
1.2 Долговечность Unix	30
1.3 Доводы против изучения культуры Unix	31
1.4 Что в Unix делается неверно	32
1.5 Что в Unix делается верно.....	33
1.5.1 Программное обеспечение с открытым исходным кодом.....	33
1.5.2 Кроссплатформенная переносимость и открытые стандарты	34
1.5.3 Internet и World Wide Web	34
1.5.4 Сообщество открытого исходного кода	35
1.5.5 Гибкость на всех уровнях.....	36
1.5.6 Особый интерес исследования Unix.....	36
1.5.7 Уроки Unix применимы в других операционных системах	37
1.6 Основы философии Unix	37
1.6.1 Правило модульности: следует писать простые части, связанные ясными интерфейсами.....	40
1.6.2 Правило ясности: ясность лучше, чем мастерство	41
1.6.3 Правило композиции: следует разрабатывать программы, которые будут взаимодействовать с другими программами	41
1.6.4 Правило разделения: следует отделять политику от механизма и интерфейсы от основных модулей.....	42
1.6.5 Правило простоты: необходимо проектировать простые программы и “добавлять сложность” только там, где это необходимо	43
1.6.6 Правило расчетливости: пишите большие программы, только если после демонстрации становится ясно, что ничего другого не остается	44

1.6.7	Правило прозрачности: для того чтобы упростить проверку и отладку программы, ее конструкция должна быть обозримой.....	44
1.6.8	Правило устойчивости: устойчивость— следствие прозрачности и простоты.....	45
1.6.9	Правило представления: знания следует оставлять в данных, чтобы логика программы могла быть примитивной и устойчивой.....	46
1.6.10	Правило наименьшей неожиданности: при проектировании интерфейсов всегда следует использовать наименее неожиданные элементы	46
1.6.11	Правило тишины: если программа не может “сказать” что-либо неожиданное, то ей вообще не следует “говорить”	47
1.6.12	Правило исправности: когда программа завершается аварийно, это должно происходить явно и по возможности быстро	47
1.6.13	Правило экономии: время программиста стоит дорого; поэтому экономия его времени более приоритетна по сравнению с экономией машинного времени.....	48
1.6.14	Правило генерации: избегайте кодирования вручную; если есть возможность, пишите программы для создания программ.....	49
1.6.15	Правило оптимизации: создайте опытные образцы, заставьте их работать, прежде чем перейти к оптимизации	49
1.6.16	Правило разнообразия: не следует доверять утверждениям о “единственно верном пути”	51
1.6.17	Правило расширяемости: проектируйте с учетом изменений в будущем, поскольку будущее придет скорее, чем кажется.....	51
1.7	Философия Unix в одном уроке.....	52
1.8	Применение философии Unix	52
1.9	Подход также имеет значение	53
2	История: слияние двух культур.....	55
2.1	Истоки и история Unix, 1969—1995 гг.	55
2.1.1	Начало: 1969—1971 гг.	56
2.1.2	Исход: 1971—1980 гг.	58
2.1.3	TCP/IP и Unix-войны: 1980—1990 гг.	61
2.1.4	Бои против империи: 1991—1995 гг.....	67
2.2	Истоки и история хакерской культуры, 1961—1995 гг.....	69
2.2.1	Академические игры: 1961—1980 гг.....	69
2.2.2	Internet и движение свободного программного обеспечения: 1981—1991 гг.....	71

2.2.3	Linux и реакция прагматиков: 1991—1998 гг.....	73
2.3	Движение открытого исходного кода: с 1998 года до настоящего времени	75
2.4	Уроки истории Unix.....	77
3	Контраст: сравнение философии Unix и других операционных систем.....	79
3.1	Составляющие стиля операционной системы	79
3.1.1	Унифицирующая идея операционной системы	80
3.1.2	Поддержка многозадачности	80
3.1.3	Взаимодействующие процессы.....	81
3.1.4	Внутренние границы	82
3.1.5	Атрибуты файлов и структуры записи	83
3.1.6	Двоичные форматы файлов.....	84
3.1.7	Предпочтительный стиль пользовательского интерфейса	84
3.1.8	Предполагаемый потребитель	85
3.1.9	Входные барьеры для разработчика	86
3.2	Сравнение операционных систем.....	87
3.2.1	VMS.....	87
3.2.2	MacOS.....	90
3.2.3	OS/2.....	91
3.2.4	Windows NT	94
3.2.5	BeOS	97
3.2.6	MVS.....	99
3.2.7	VM/CMS	101
3.2.8	Linux.....	103
3.3	Все повторяется	105
II	Проектирование.....	109
4	Модульность: четкость и простота	111
4.1	Инкапсуляция и оптимальный размер модуля.....	113
4.2	Компактность и ортогональность	115
4.2.1	Компактность.....	115
4.2.2	Ортогональность	117
4.2.3	Правило SPOT	118
4.2.4	Компактность и единый жесткий центр.....	120
4.2.5	Значение освобождения	122
4.3	Иерархичность программного обеспечения.....	122
4.3.1	Сравнение нисходящего и восходящего программирования.....	122
4.3.2	Связующие уровни.....	125
4.3.3	Учебный пример: язык C считается тонким связующим уровнем.....	126

4.4	Библиотеки	127
4.4.1	Учебный пример: подключаемые подпрограммы GIMP	129
4.5	Unix и объектно-ориентированные языки	130
4.6	Создание модульного кода	132
5	Текстовое представление данных: ясные протоколы лежат в основе хорошей практики	135
5.1	Важность текстовой формы представления	137
5.1.1	Учебный пример: формат файлов паролей в Unix	139
5.1.2	Учебный пример: формат файлов .newsrc	140
5.1.3	Учебный пример: PNG — формат графических файлов	142
5.2	Метаформаты файлов данных	142
5.2.1	DSV-стиль	143
5.2.2	Формат RFC 822	144
5.2.3	Формат Cookie-Jar	145
5.2.4	Формат record-jar	146
5.2.5	XML	147
5.2.6	Формат Windows INI	149
5.2.7	Unix-соглашения по текстовым файловым форматам	150
5.2.8	Аргументы “за” и “против” сжатия файлов	152
5.3	Проектирование протоколов прикладного уровня	153
5.3.1	Учебный пример: SMTP, простой протокол передачи почты	154
5.3.2	Учебный пример: POP3, почтовый протокол 3-й версии	155
5.3.3	Учебный пример: IMAP, протокол доступа к почтовым сообщениям	156
5.4	Метаформаты протоколов прикладного уровня	158
5.4.1	Классический метапротокол прикладного уровня в Internet	158
5.4.2	HTTP как универсальный протокол прикладного уровня	159
5.4.3	BEEP: Blocks Extensible Exchange Protocol	161
5.4.4	XML-RPC, SOAP и Jabber	162
6	Прозрачность: да будет свет	163
6.1	Учебные примеры	165
6.1.1	Учебный пример: <i>audacity</i>	165
6.1.2	Учебный пример: параметр -v программы <i>fetchmail</i>	166
6.1.3	Учебный пример: GCC	169
6.1.4	Учебный пример: <i>kmwail</i>	170
6.1.5	Учебный пример: SNG	172
6.1.6	Учебный пример: база данных Terminfo	174
6.1.7	Учебный пример: файлы данных Freeciv	177
6.2	Проектирование, обеспечивающее прозрачность и воспринимаемость	179
6.2.1	Дзэн прозрачности	179

6.2.2	Программирование, обеспечивающее прозрачность и воспринимаемость	180
6.2.3	Прозрачность и предотвращение избыточной защищенности	181
6.2.4	Прозрачность и редактируемые формы представления	182
6.2.5	Прозрачность, диагностика и восстановление после сбоев.....	184
6.3	Проектирование, обеспечивающее удобство сопровождения	185
7	Мультипрограммирование: разделение процессов для разделения функций.....	187
7.1	Отделение контроля сложности от настройки производительности	189
7.2	Классификация IPC-методов в Unix	190
7.2.1	Передача задач специализированным программам.....	190
7.2.2	Каналы, перенаправление и фильтры	192
7.2.3	Упаковщики	196
7.2.4	Оболочки безопасности и цепи Бернштайна.....	197
7.2.5	Подчиненные процессы	199
7.2.6	Равноправный межпроцессный обмен данными	200
7.3	Проблемы и методы, которых следует избегать	207
7.3.1	Устаревшие IPC-методы в Unix	207
7.3.2	Методы удаленного вызова процедур	209
7.3.3	Опасны ли параллельные процессы?	211
7.4	Разделение процессов на уровне проектирования	212
8	Мини-языки: поиск выразительной нотации	215
8.1	Классификация языков.....	217
8.2	Применение мини-языков	219
8.2.1	Учебный пример: <i>sng</i>	219
8.2.2	Учебный пример: регулярные выражения.....	220
8.2.3	Учебный пример: Glade	223
8.2.4	Учебный пример: <i>m4</i>	225
8.2.5	Учебный пример: XSLT	225
8.2.6	Учебный пример: инструментарий Documenter's Workbench	227
8.2.7	Учебный пример: синтаксис конфигурационного файла <i>fetchmail</i>	231
8.2.8	Учебный пример: <i>awk</i>	232
8.2.9	Учебный пример: PostScript	234
8.2.10	Учебный пример: утилиты <i>bc</i> и <i>dc</i>	235
8.2.11	Учебный пример: Emacs Lisp.....	236
8.2.12	Учебный пример: JavaScript.....	237
8.3	Проектирование мини-языков	238
8.3.1	Определение соответствующего уровня сложности.....	238
8.3.2	Расширение и встраивание языков	240

8.3.3	Написание специальной грамматики.....	241
8.3.4	Проблемы макросов	242
8.3.5	Язык или протокол прикладного уровня.....	243
9	Генерация кода: повышение уровня спецификации.....	245
9.1	Создание программ, управляемых данными.....	246
9.1.1	Учебный пример: <i>ascii</i>	247
9.1.2	Учебный пример: статистическая фильтрация спама	248
9.1.3	Учебный пример: программирование метаклассов в <i>fetchmail</i>	249
9.2	Генерация специального кода.....	254
9.2.1	Учебный пример: генерация кода для <i>ascii</i> -дисплеев	255
9.2.2	Учебный пример: генерация HTML-кода для табличного списка	257
10	Конфигурация: правильное начало	261
10.1	Конфигурируемые параметры	261
10.2	Месторасположение конфигурационной информации.....	263
10.3	Файлы конфигурации	264
10.3.1	Учебный пример: файл <i>.netrc</i>	266
10.3.2	Переносимость на другие операционные системы.....	267
10.4	Переменные окружения.....	268
10.4.1	Системные переменные окружения	268
10.4.2	Пользовательские переменные окружения.....	269
10.4.3	Когда использовать переменные окружения	270
10.4.4	Переносимость на другие операционные системы.....	272
10.5	Параметры командной строки	272
10.5.1	Параметры командной строки от -a до -z	273
10.5.2	Переносимость на другие операционные системы.....	278
10.6	Выбор метода	279
10.6.1	Учебный пример: <i>fetchmail</i>	279
10.6.2	Учебный пример: сервер XFree86	281
10.7	Нарушение правил	283
11	Интерфейсы: модели проектирования пользовательских интерфейсов в среде Unix	285
11.1	Применение правила наименьшей неожиданности	286
11.2	История проектирования интерфейсов в Unix	288
11.3	Оценка конструкций интерфейсов	290
11.4	Компромиссы между CLI- и визуальными интерфейсами	292
11.4.1	Учебный пример: два способа написания программы калькулятора	295
11.5	Прозрачность, выразительность и возможность конфигурирования.....	297
11.6	Модели проектирования интерфейсов в Unix	299

11.6.1	Модель фильтра	299
11.6.2	Модель заклинаний.....	301
11.6.3	Модель источника.....	302
11.6.4	Модель приемника.....	302
11.6.5	Модель компилятора	302
11.6.6	Модель редактора <i>ed</i>	303
11.6.7	Rogue-подобная модель	304
11.6.8	Модель “разделения ядра и интерфейса”	307
11.6.9	Модель CLI-сервера.....	312
11.6.10	Модель интерфейсов на основе языков.....	312
11.7	Применение Unix-моделей проектирования интерфейсов.....	313
11.7.1	Модель многопараметрических программ.....	314
11.8	Использование Web-браузера в качестве универсального клиента	315
11.9	Молчание — золото.....	318
12	Оптимизация.....	321
12.1	Отказ от оптимизации	321
12.2	Измерения перед оптимизацией	322
12.3	Размер кода.....	324
12.4	Пропускная способность и задержка	325
12.4.1	Пакетные операции.....	326
12.4.2	Совмещение операций	327
12.4.3	Кэширование результатов операций.....	327
13	Сложность: просто, как только возможно, но не проще	329
13.1	Сложность	329
13.1.1	Три источника сложности	330
13.1.2	Компромиссы между сложностью интерфейса и реализации.....	332
13.1.3	Необходимая, необязательная и случайная сложность	333
13.1.4	Диаграмма видов сложности	334
13.1.5	Когда простоты не достаточно	336
13.2	Редакторы	336
13.2.1	<i>ed</i>	337
13.2.2	<i>vi</i>	339
13.2.3	<i>Sam</i>	340
13.2.4	<i>Emacs</i>	341
13.2.5	<i>Wily</i>	342
13.3	Необходимый и достаточный размер редактора	343
13.3.1	Идентификация проблем сложности	343
13.3.2	Компромиссы не действуют	346
13.3.3	Является ли <i>Emacs</i> доводом против Unix-традиции?	348
13.4	Необходимый размер программы.....	350

III Реализация..... 353

14	Языки программирования: С или не С?	355
14.1	Многообразие языков в Unix	355
14.2	Доводы против С.....	356
14.3	Интерпретируемые языки и смешанные стратегии.....	358
14.4	Сравнение языков программирования.....	359
14.4.1	С.....	359
14.4.2	C++	361
14.4.3	Shell	363
14.4.4	Perl.....	366
14.4.5	Tcl	368
14.4.6	Python	370
14.4.8	Emacs Lisp.....	376
14.5	Тенденции будущего	378
14.6	Выбор X-инструментария.....	379
15	Инструментальные средства: тактические приемы разработчика	383
15.1	Операционная система, дружественная к разработчику.....	383
15.2	Выбор редактора.....	384
15.2.1	Полезные сведения о <i>vi</i>	385
15.2.2	Полезные сведения о Emacs	385
15.2.3	“Антирелигиозный” выбор: использование обоих редакторов	386
15.3	Генераторы специализированного кода	387
15.3.1	<i>yacc</i> и <i>lex</i>	387
15.3.2	Учебный пример: <i>Glade</i>	391
15.4	Утилита <i>make</i> : автоматизация процедур.....	391
15.4.1	Базовая теория <i>make</i>	391
15.4.2	Утилита <i>make</i> в разработке не на C/C++	393
15.4.3	Правила <i>make</i>	394
15.4.4	Генерация <i>make</i> -файлов	396
15.5	Системы контроля версий.....	398
15.5.1	Для чего используется контроль версий	399
15.5.2	Контроль версий вручную	399
15.5.3	Автоматизированный контроль версий.....	400
15.5.4	Unix-инструменты для контроля версий.....	401
15.6	Отладка времени выполнения	403
15.7	Профилирование	404
15.8	Комбинирование инструментов с Emacs	405
15.8.1	Emacs и <i>make</i>	405
15.8.2	Emacs и отладка во время выполнения	406
15.8.3	Emacs и контроль версий	406
15.8.4	Emacs и профилирование	407

15.8.5	Лучше, чем IDE	407
16	Повторное использование кода: не изобретая колесо	409
16.1	История случайного новичка	410
16.2	Прозрачность — ключ к повторному использованию кода	413
16.3	От повторного использования к открытому исходному коду.....	414
16.4	Оценка проектов с открытым исходным кодом	415
16.5	Поиск открытого исходного кода	417
16.6	Вопросы использования программ с открытым исходным кодом ..	419
16.7	Вопросы лицензирования	420
16.7.1	Что определяется как открытый исходный код	420
16.7.2	Стандартные лицензии на открытый исходный код	422
16.7.3	Когда потребуется адвокат	424
IV	Сообщество	425
17	Переносимость: переносимость программ и соблюдение стандартов	427
17.1	Эволюция C.....	428
17.1.1	Ранняя история C	429
17.1.2	Стандарты C.....	430
17.2	Стандарты Unix	432
17.2.1	Стандарты и Unix-войны	432
17.2.2	Влияние новых Unix-систем	435
17.2.3	Стандарты Unix в мире открытого исходного кода.....	436
17.3	IETF и процесс RFC-стандартизации.....	437
17.4	Спецификации — ДНК, код — РНК	439
17.5	Программирование, обеспечивающее переносимость	442
17.5.1	Переносимость и выбор языка.....	443
17.5.2	Обход системных зависимостей	446
17.5.3	Инструменты, обеспечивающие переносимость	447
17.6	Интернационализация.....	447
17.7	Переносимость, открытые стандарты и открытый исходный код.....	448
18	Документация: объяснение кода в Web-сообществе	451
18.1	Концепции документации.....	452
18.2	Стиль Unix	454
18.2.1	Склонность к большим документам	454
18.2.2	Культурный стиль.....	455
18.3	Многообразие форматов документации в Unix.....	456
18.3.1	<i>troff</i> и инструментарий Documenter's Workbench	456
18.3.2	$\TeX$	458
18.3.3	Texinfo	459

18.3.4	POD.....	459
18.3.5	HTML.....	460
18.3.6	DocBook.....	460
18.4	Современный хаос и возможный выход из положения.....	460
18.5	DocBook.....	461
18.5.1	Определения типов документов.....	461
18.5.2	Другие DTD-определения.....	463
18.5.3	Инструментальная связка DocBook	463
18.5.4	Средства преобразования	466
18.5.5	Инструменты редактирования.....	467
18.5.6	Связанные стандарты и практические приемы.....	467
18.5.7	SGML	468
18.5.8	Справочные ресурсы по XML-DocBook.....	468
18.6	Лучшие практические приемы написания Unix-документации	468
19	Открытый исходный код: программирование в новом Unix-сообществе.....	471
19.2	Unix и открытый исходный код	472
19.2	Лучшие практические приемы при взаимодействии с разработчиками открытого исходного кода.....	474
19.2.1	Хорошая практика обмена исправлениями.....	474
19.2.2	Хорошая практика наименования проектов и архивов	478
19.2.3	Хорошая практика разработки	481
19.2.4	Хорошая практика создания дистрибутивов.....	484
19.2.5	Практические приемы хорошей коммуникации	488
19.3	Логика лицензирования: как выбрать лицензию	490
19.4	Почему следует использовать стандартную лицензию	491
19.5	Многообразие лицензий на открытый исходный код.....	491
19.5.1	Лицензия MIT или Консорциума X.....	491
19.5.2	Классическая BSD-лицензия.....	492
19.5.3	Артистическая лицензия	492
19.5.4	General Public License	493
19.5.5	Mozilla Public License	493
20	Будущее: опасности и перспективы.....	495
20.1	Сущность и случайность в традиции Unix.....	495
20.2	Plan 9: каким представлялось будущее Unix	498
20.3	Проблемы в конструкции Unix	500
20.3.1	Unix-файл представляет собой только большой блок байтов	500
20.3.2	Слабая поддержка GUI-интерфейсов в Unix	502
20.3.3	Удаление файлов в Unix необратимо.....	502
20.3.4	Unix предполагает статичную файловую систему	503

20.3.5	Конструкция системы управления задачами была плохо реализована.....	503
20.3.6	В Unix API не используются исключительные ситуации	504
20.3.7	Вызовы <i>ioctl(2)</i> и <i>fcntl(2)</i> являются препятствиями.....	505
20.3.8	Модель безопасности Unix, возможно, слишком примитивна	506
20.3.9	Unix имеет слишком много различных видов имен	506
20.3.10	Файловые системы могут считаться вредными	506
20.3.11	На пути к глобальному адресному пространству Internet.....	507
20.4	Проблемы в окружении Unix.....	507
20.5	Проблемы в культуре Unix	509
20.6.	Причины верить.....	512
А	Глоссарий аббревиатур.....	513
Б	Список литературы.....	517
В	Персональный вклад	525
Г	Корни без корней: Unix-коаны Мастера Фу	529
	Предисловие редактора	529
	Мастер Фу и десять тысяч строк	530
	Мастер Фу и Скрипт Кидди	531
	Мастер Фу рассуждает о двух дорогах.....	531
	Мастер Фу и консультант по методологии.....	532
	Мастер Фу рассуждает о графическом пользовательском интерфейсе	533
	Мастер Фу и фанатик Unix	533
	Мастер Фу рассуждает о природе Unix.....	534
	Мастер Фу и конечный пользователь.....	534
	Дополнительная информация.....	535
	Предметный указатель.....	536