

Оглавление

| | |
|--|-----------|
| Предисловие к русскому изданию | 14 |
| Вступительное слово | 15 |
| Предисловие | 16 |
| Франческо: Почему Erlang? | 16 |
| Саймон: Почему Erlang? | 17 |
| Для кого эта книга? | 17 |
| Как читать эту книгу | 17 |
| Условные обозначения | 19 |
| Использование кода из примеров | 20 |
| Safari Books Online | 20 |
| Как связаться с нами | 20 |
| Благодарности | 21 |
| Благодарности от российского издательства | 22 |
| 1 Введение | 23 |
| 1.1 Почему Erlang? | 23 |
| 1.2 История Erlang | 25 |
| 1.3 Особенности Erlang | 26 |
| Высокоуровневые конструкции | 26 |
| Параллельные вычисления и передача сообщений | 27 |
| Масштабируемые, безопасные и эффективные параллельные вычисления | 28 |
| Применение в системах реального времени | 29 |
| Надёжность | 29 |
| Распределённые вычисления | 30 |
| Интеграция и открытость | 31 |
| 1.4 Erlang и многоядерные процессоры | 31 |
| 1.5 Erlang на практике | 32 |
| Коммутатор ATM AXD301 | 33 |
| CouchDB | 34 |
| Erlang и C++ | 35 |
| 1.6 Как применять Erlang? | 36 |

| | |
|--|-----------|
| 2 Основы Erlang | 38 |
| 2.1 Целые числа | 38 |
| 2.2 Интерпретатор Erlang | 39 |
| 2.3 Действительные числа | 39 |
| Арифметические операции | 40 |
| 2.4 Атомы | 41 |
| 2.5 Логические значения | 43 |
| 2.6 Кортежи | 44 |
| 2.7 Списки | 45 |
| Символы и строки | 46 |
| Атомы и строки | 46 |
| Создание и обработка списков | 47 |
| Функции и операции, определённые на списках | 48 |
| 2.8 Сравнение термов | 51 |
| 2.9 Переменные | 53 |
| 2.10 Сложные структуры данных | 55 |
| 2.11 Сопоставление с образцом | 57 |
| 2.12 Функции | 62 |
| 2.13 Модули | 65 |
| Компиляция и виртуальная машина Erlang | 66 |
| Директивы модулей | 66 |
| 2.14 Упражнения | 68 |
| 3 Последовательное программирование в Erlang | 70 |
| 3.1 Условные выражения | 71 |
| case-выражение | 71 |
| Область видимости переменных | 74 |
| if-выражение | 75 |
| 3.2 Охранные выражения | 76 |
| 3.3 Встроенные функции | 78 |
| Извлечение элементов объекта и формирование запросов к объектам | 79 |
| Преобразование типов | 80 |
| Словарь процессов | 81 |
| Метапрограммирование | 81 |
| Процессы, порты, распределённые программы и системная информация | 82 |
| Ввод-вывод | 83 |
| 3.4 Рекурсия | 85 |
| Хвостовая рекурсия | 90 |
| Оптимизация хвостовой рекурсии | 93 |
| Итерационные функции против рекурсивных | 94 |

| | | |
|----------|---|------------|
| 3.5 | Ошибки времени выполнения | 95 |
| 3.6 | Обработка ошибок | 98 |
| | С помощью try ... catch | 98 |
| | С помощью catch | 102 |
| 3.7 | Библиотечные модули | 106 |
| | Документация | 106 |
| | Полезные модули | 107 |
| 3.8 | Отладчик | 109 |
| 3.9 | Упражнения | 111 |
| 4 | Параллельное программирование | 117 |
| 4.1 | Создание процессов | 118 |
| 4.2 | Передача сообщений | 120 |
| 4.3 | Приём сообщений | 122 |
| 4.4 | Выборочный приём сообщений | 124 |
| 4.5 | Пример эхо-процесса | 128 |
| 4.6 | Зарегистрированные процессы | 130 |
| 4.7 | Время задержки | 133 |
| 4.8 | Анализ производительности | 135 |
| 4.9 | Каркас процессов | 136 |
| 4.10 | Хвостовая рекурсия и утечки памяти | 137 |
| 4.11 | Один случай из практики параллельного программирования | 138 |
| 4.12 | Состояние гонки, взаимная блокировка, голодание процессов | 140 |
| 4.13 | Менеджер процессов | 142 |
| 4.14 | Упражнения | 143 |
| 5 | Шаблоны проектирования процессов | 145 |
| 5.1 | Модель клиент/сервер | 146 |
| | Пример модели клиент/сервер | 147 |
| 5.2 | Пример шаблона процессов | 152 |
| 5.3 | Конечный автомат | 154 |
| | Пример конечного автомата | 154 |
| | Одноместный семафор | 156 |
| 5.4 | Менеджер и обработчик событий | 159 |
| | Пример обобщённого менеджера событий | 160 |
| | Обработчики событий | 163 |
| 5.5 | Упражнения | 165 |
| 6 | Обработка ошибок в процессах | 167 |
| 6.1 | Соединение процессов и сигналы выхода | 167 |
| | Перехват сигналов выхода | 170 |
| | Функции наблюдения за процессами | 172 |

| | |
|--|------------|
| Функции останова | 174 |
| Встроенные функции и терминология | 174 |
| Семантика распространения сигналов выхода | 176 |
| 6.2 Построение надёжных систем | 177 |
| Наблюдение за клиентами | 178 |
| Пример процесса-наблюдателя | 181 |
| 6.3 Упражнения | 183 |
| 7 Записи и макросы | 186 |
| 7.1 Записи | 186 |
| Записи, первое знакомство | 187 |
| Применение записей | 188 |
| Функции и сопоставление с образцом | 189 |
| Записи в интерпретаторе | 190 |
| Реализация записей | 191 |
| Встроенные функции для работы с записями | 192 |
| 7.2 Макросы | 193 |
| Простые макросы | 193 |
| Параметризованные макросы | 194 |
| Отладка и макросы | 195 |
| Заголовочные файлы | 196 |
| 7.3 Упражнения | 197 |
| 8 Обновление приложений | 200 |
| 8.1 Обновление модулей | 200 |
| 8.2 За кулисами | 203 |
| Загрузка кода | 206 |
| Сервер кода | 207 |
| Очищение модулей | 209 |
| 8.3 Обновление процессов | 209 |
| 8.4 Файл <code>.erlang</code> | 214 |
| 8.5 Упражнения | 214 |
| 9 Новые типы данных и высокоуровневые выражения | 216 |
| 9.1 Функциональное программирование | 216 |
| 9.2 Тип <code>fun</code> и функции высшего порядка | 217 |
| Функция как аргумент | 217 |
| Определение функций на лету: <code>fun</code> -выражения | 219 |
| Функция как результат | 220 |
| Использование определённых функций | 221 |
| Функции и переменные | 222 |
| Стандартные функции высшего порядка | 223 |

| | |
|--|------------|
| Ленивые вычисления и списки | 224 |
| 9.3 Генераторы списков | 225 |
| Первый пример | 225 |
| Обобщённые генераторы списков | 225 |
| Несколько генераторов | 227 |
| Стандартные функции | 227 |
| 9.4 Двоичные данные и сериализация | 228 |
| Двоичные данные | 228 |
| Битовый синтаксис | 230 |
| Битовое сопоставление с образцом | 232 |
| Генераторы двоичных данных | 233 |
| Пример: декодирование сегментов TCP | 234 |
| Битовые операции | 235 |
| Сериализация | 236 |
| 9.5 Ссылки | 238 |
| 9.6 Упражнения | 239 |
| 10 ETS- и Dets-таблицы | 241 |
| 10.1 ETS-таблицы | 241 |
| Реализация и оптимальный выбор типа таблиц | 242 |
| Создание таблиц | 244 |
| Работа с таблицей | 245 |
| Пример: построение предметного указателя, первая часть | 246 |
| Обход таблицы | 249 |
| Пример: построение предметного указателя, вторая часть | 249 |
| Извлечение данных: match | 251 |
| Извлечение данных: select | 253 |
| Другие операции с таблицами | 254 |
| Записи и ETS-таблицы | 255 |
| Визуализация таблиц | 256 |
| 10.2 Dets-таблицы | 256 |
| 10.3 Пример: база данных абонентов мобильной связи | 259 |
| Интерфейс базы данных | 260 |
| Сервер базы данных | 266 |
| 10.4 Упражнения | 271 |
| 11 Распределённое программирование | 273 |
| 11.1 Распределённые приложения в Erlang | 273 |
| 11.2 Распределённые приложения в Erlang: основы | 275 |
| Имена узлов и область видимости | 277 |
| Взаимодействие и безопасность | 278 |
| Взаимодействие и сообщения | 280 |

| | |
|---|------------|
| Соединение узлов | 281 |
| Удалённый вызов процедур | 284 |
| Модуль grc | 286 |
| Ключевые модули для распределённого программирования | 287 |
| 11.3 Процесс erpmd | 288 |
| Распределённый Erlang за межсетевым экраном | 289 |
| 11.4 Упражнения | 290 |
| 12 Поведения OTP | 291 |
| 12.1 Введение в поведения OTP | 291 |
| 12.2 Обобщённый сервер | 294 |
| Запуск сервера | 294 |
| Передача сообщений | 296 |
| Завершение сервера | 298 |
| Полный текст примера | 299 |
| Тестирование gen_server | 302 |
| 12.3 Наблюдатель | 305 |
| Спецификация наблюдателя | 306 |
| Спецификация дочерних процессов | 306 |
| Пример | 308 |
| Динамические дочерние процессы | 309 |
| 12.4 Приложения | 309 |
| Структура директорий | 310 |
| Resource-файл | 312 |
| Запуск и завершение приложений | 313 |
| Менеджер приложений | 316 |
| 12.5 Управление релизами | 316 |
| 12.6 Другие поведения и источники для самостоятельного изучения | 319 |
| 12.7 Упражнения | 320 |
| 13 Начала работы с Mnesia | 322 |
| 13.1 Для чего подходит Mnesia | 322 |
| 13.2 Настройка Mnesia | 324 |
| Задание схемы | 324 |
| Запуск Mnesia | 325 |
| Таблицы Mnesia | 325 |
| 13.3 Транзакции | 328 |
| Запись | 329 |
| Чтение и удаление | 329 |
| Индексация | 330 |
| Грязные операции | 332 |
| 13.4 Разрыв соединения | 334 |

| | |
|---|------------|
| 13.5 Дополнительные источники информации | 335 |
| 13.6 Упражнения | 335 |
| 14 Создание GUI-приложений средствами wxErlang | 337 |
| 14.1 wxWidgets | 337 |
| 14.2 wxErlang. Порт wxWidgets для Erlang | 338 |
| Объекты и типы | 339 |
| Обработка событий, идентификаторы объектов и типы событий | 339 |
| Соберём всё вместе | 341 |
| 14.3 Первый пример: микроблог | 342 |
| 14.4 Мини-блог | 344 |
| 14.5 Установка и запуск wxErlang | 348 |
| 14.6 Упражнения | 350 |
| 15 Работа с сокетами | 351 |
| 15.1 User Datagram Protocol | 351 |
| 15.2 Transmission Control Protocol | 355 |
| Пример обмена данными через TCP | 356 |
| 15.3 Модуль inet | 360 |
| 15.4 Дополнительные источники информации | 361 |
| 15.5 Упражнения | 362 |
| 16 Взаимодействие с другими языками программирования | 364 |
| 16.1 Обзор средств взаимодействия | 364 |
| 16.2 Взаимодействие с Java | 366 |
| Узлы и почтовые ящики | 366 |
| Представление типов Erlang | 367 |
| Обмен сообщениями | 367 |
| Соберём всё вместе: и вновь RPC | 368 |
| Взаимодействие | 369 |
| Мелким шрифтом | 370 |
| Но и это ещё не всё | 370 |
| 16.3 Узлы C | 371 |
| Дополнительные возможности | 375 |
| 16.4 Вызов Erlang из командной строки UNIX: erl_call | 375 |
| 16.5 Порты | 375 |
| Команды управления | 377 |
| Обмен данными | 378 |
| 16.6 Библиотечная поддержка обмена данными | 379 |
| Работаем с Ruby: electricity | 380 |
| 16.7 Подключаемые драйверы и FFI | 381 |
| 16.8 Упражнения | 382 |

| | |
|--|------------|
| 17 Отладка приложений в Erlang | 383 |
| 17.1 Введение | 383 |
| 17.2 Трассировочные встроенные функции | 385 |
| Трассировочные флаги для процессов | 386 |
| Флаги наследования | 388 |
| Сборка мусора и отсчёты времени | 389 |
| 17.3 Трассировка вызовов с помощью функции trace_pattern | 391 |
| 17.4 Трассировщик dbg | 394 |
| Первые шаги | 394 |
| Трассировка и профилирование функций | 397 |
| Трассировка локальных и глобальных вызовов функций | 398 |
| Распределённая трассировка | 399 |
| Перенаправление вывода | 400 |
| 17.5 Спецификация сопоставления: fun-синтаксис | 404 |
| Создание спецификаций функцией fun2ms | 404 |
| Разница между спецификациями для ets и dbg | 412 |
| 17.6 Устройство спецификаций сопоставления | 412 |
| Голова | 413 |
| Условные выражения | 414 |
| Тело спецификации | 417 |
| Сохранение спецификаций сопоставления | 421 |
| 17.7 Дополнительная литература | 422 |
| 17.8 Упражнения | 422 |
| 18 Типы и документация | 424 |
| 18.1 Типы в Erlang | 424 |
| Пример: записи с типизированными полями | 424 |
| Нотация типов в Erlang | 425 |
| 18.2 TypeEr | 427 |
| Dialyzer: утилита для статической проверки кода Erlang | 430 |
| 18.3 Создание документации с помощью EDoc | 431 |
| Документация к модулю usr_db | 432 |
| Запуск EDoc | 434 |
| Типы в EDoc | 436 |
| Дополнительные возможности EDoc | 437 |
| 18.4 Упражнения | 439 |
| 19 EUnit и разработка через тестирование | 440 |
| 19.1 Разработка через тестирование | 440 |
| 19.2 EUnit | 441 |
| Как пользоваться EUnit | 442 |
| Функциональное тестирование: сериализация дерева | 442 |

| | |
|---|------------|
| 19.3 Инфраструктура EUnit | 446 |
| Макрос assert | 446 |
| Функции для генерации тестов | 446 |
| Представление тестов в EUnit | 447 |
| 19.4 Тестирование приложений с состоянием | 448 |
| Окружения: инициализация и очистка | 448 |
| 19.5 Тестирование параллельных программ | 449 |
| 19.6 Упражнения | 450 |
| 20 Стиль и эффективность | 451 |
| 20.1 Приложения и модули | 451 |
| Библиотеки | 452 |
| Грязный код | 452 |
| Интерфейсы | 453 |
| Возвращаемые значения | 453 |
| Внутренние структуры данных | 455 |
| 20.2 Процессы и параллельные вычисления | 456 |
| 20.3 Стилевые соглашения | 460 |
| 20.4 Стратегии разработки | 465 |
| 20.5 Эффективность | 467 |
| Последовательное программирование | 467 |
| Списки | 469 |
| Хвостовая и нехвостовая рекурсия | 471 |
| Параллельное программирование | 471 |
| 20.6 И наконец... | 473 |
| Приложение | 475 |
| Основы работы с Erlang | 475 |
| Установка | 475 |
| Запуск интерпретатора | 475 |
| Средства разработки | 477 |
| Редакторы | 477 |
| Другие средства | 478 |
| Дополнительные источники информации | 479 |
| Об авторах | 481 |

Предисловие к русскому изданию

За два года, прошедших со времени выхода английской версии "Erlang Programming", мы увидели множество положительных изменений в Erlang-сообществе. Значительно выросло количество обращений к erlang.org. Репозиторий с исходным кодом Erlang был перемещён на github, что привело к увеличению количества исправлений и улучшений, которые реализованы силами сообщества. Также вырос интерес к изучению Erlang - сотни людей принимают участие в конференциях Erlang Factory в Лондоне и Сан-Франциско, а количество участников Erlang User Conference в Стокгольме увеличилось более чем в два раза. По всему миру были организованы более короткие встречи Erlang Factory Lite, включая США, Австралию и Европу. Увеличивается использование Erlang в промышленности, от встраиваемых систем до суперкомпьютеров. Каждый день мы узнаем о новых, интересных внедрениях Erlang. Его называют языком облака, он приводит в движение инфраструктуры облачных вычислений всех форм и размеров. Он также стал одним из наиболее известных языков для многоядерного программирования. Кроме того, он сильно повлиял на множество новых языков программирования, которые были созданы в последние годы.

Хотя книга и была закончена, мы продолжили работать вместе над другими проектами. Наиболее известным является совместный проект между Erlang Solutions и университетом Кент, (Великобритания) по электронному обучению и сертификации по темам, относящимся к Erlang и Erlang/OTP. Мы успешно выполнили проект (финансируемый ЕС) на тему "Erlang и Property Based-тестирование", который изменил способы тестирования и разработки программ на Erlang. Мы начинаем новый проект, также финансируемый ЕС, который будет направлен на улучшение масштабирования Erlang на крупных многоядерных системах. Есть в планах и новая книга о масштабировании Erlang с помощью OTP.

Хотя русскоговорящее сообщество относительно неизвестно нам, мы знаем, что оно значительно выросло за последние несколько лет. Мы слышали, что Erlang является повторяющейся темой на большинстве конференций для программистов. Большие компании и стартапы построили свой бизнес вокруг Erlang и ищут разработчиков на этом языке. Новые статьи, обзоры и примеры кода на Erlang ежедневно публикуются в различных сообществах программистов. Отличная работа! Хотя книги по Erlang выходят на многих языках, нам доставляет удовольствие видеть публикуемый русский перевод "Erlang Programming", и мы надеемся, что он поможет сообществу и коммерческим пользователям.

Приятного чтения!

Франческо Чезарини, Саймон Томпсон,

Вступительное слово

Erlang предоставляет решение трёх проблем, связанных с разработкой параллельных, распределённых приложений, работающих в режиме реального времени:

- возможность быстрой и эффективной разработки приложений;
- разработка приложений, устойчивые как к программным, так и к аппаратным неполадкам;
- возможность обновления приложений "на лету", без необходимости остановки выполнения.

При создании Erlang мы думали в первую очередь о телекоммуникационных приложениях, но задачи, решаемые в этой области, оказались настолько общими, что теперь Erlang применяется в самых разнообразных областях, среди них распределённые базы данных, финансовые приложения, серверы обмена мгновенными сообщениями. Недавний вслеск интереса к Erlang был вызван успешным применением Erlang при работе на многоядерных процессорах. В то время как в других языках переход на многоядерные процессоры связан с серьёзными проблемами, в приложениях, написанных на Erlang, для этого почти ничего менять не приходится.

Изначально Erlang распространялся очень медленно, возможно потому, что разработчикам казалось слишком рискованным использование языка, содержащего столько необычных технологий: функциональное программирование, легковесные параллельные вычисления, асинхронная передача сообщений и уникальный метод обработки падений приложений. Так быстро ставший популярным Java лишь немногим отличается от C++, и людям было гораздо проще переключиться на него. Однако для достижения тех целей, о которых я только что упомянул, мы чувствуем, что наш подход выдержал испытание временем. Сейчас его популярность быстро растёт.

Книга представляет собой отличное практическое введение в Erlang. В ней вы найдёте также и много интересных историй, объясняющих идеи, которые легли в основание Erlang.

Счастливого и, уверен, полезного прочтения!

Майк Вильямс,
директор Traffic and Feature Software
Product Development Unit, WCDMA Ericsson AB,
один из создателей Erlang

Предисловие

Как появилась эта книга? Мы решили написать книгу о Erlang прежде всего потому, что нам очень нравится этот язык, нам захотелось вернуть сообществу Erlang крупицу того, что мы получили. Хотя к Erlang нас привели разные дороги, но результат оказался одним и тем же: мы интересно проводим время, решая увлекательные и сложные задачи, прилагая лишь малую долю усилий, которые бы потребовались на решение с помощью других языков. И мы используем Erlang в реальных проектах, а не только в свободное от работы время.

Франческо: Почему Erlang?

Шёл 1994 год, я учился в Уппсальском университете, один из предметов был посвящён параллельному программированию. Как-то раз лектор показал нам первое издание книги "Concurrent programming in Erlang" (Prentice Hall) и сказал: "Прочтите эту книгу", — затем он показал распечатку и добавил: "Вот упражнения, выполните их". Это были все слова для Erlang, после этого мы говорили о теории потоков вычислений, разделяемой памяти, семафорах и взаимных блокировках.

Главное упражнение курса заключалось в создании симулятора мира, населённого морковками, зайцами и волками. Зайцы бегают по миру в поисках морковки, которая появляется в разных местах случайным образом. Наевшись морковки, зайцы толстеют и делятся пополам. Волки питаются зайцами; съев достаточное количество зайцев, волк также толстееет и делится пополам. Зайцы и волки могут общаться с сородичами, если те находятся не слишком далеко от них. Так они распространяют информацию о пище и хищниках. Если заяц найдёт поле с морковкой, другие зайцы поспешат к нему, а если волк увидит зайца, в погоне будет участвовать целая стая.

Было интересно понаблюдать за результатами. Блуждающий заяц мог попасть в окружение стаи волков, в то время как остальные зайцы бросались в рассыпную, иногда останавливаясь, чтобы схватить морковку по дороге. Все морковки, зайцы и волки были представлены в Erlang отдельными процессами, которые взаимодействовали между собой с помощью передачи сообщений.

На решение этого задания у меня ушло примерно 40 часов. Я был удивлён простотой модели параллельных вычислений и отсутствием системных потоков вычислений, но, несмотря на то что мне понравилось программировать на Erlang, в то время я не придал этому языку особого значения. Erlang был одним из десятка языков, которые мне преподавали. В курсе функционального программирования

мы изучали ML, а в курсе приложений реального времени – ADA. Тогда Erlang был для меня просто ещё одним языком. Но всё изменилось, когда мы приступили к курсу объектно-ориентированного программирования (ООП).

В курсе ООП нужно было решить ту же задачу с симуляцией мира морковок, зайцев и волков, но на этот раз нам пришлось решать её с помощью ООП-языка Eiffel. Лектор настаивал на том, что этот язык идеально подходит для создания симуляторов. У нас не было выбора. Несмотря на то что я уже решил эту задачу и мог пользоваться большой частью алгоритмов, в этот раз на решение у меня и моего напарника ушло по 120 человеко-часов.

Этот случай открыл мне глаза. Я понял, что будущее — за декларативным подходом и моделью параллельных вычислений Erlang. К тому времени я не был уверен в том, что Erlang займёт ведущее место при переоценке ценностей в программировании, но у меня не было сомнений в том, что Erlang и его предшественники окажут сильное влияние на этот язык. Я позвонил Джо Армстронгу — одному из создателей Erlang. Неделю спустя я пришёл в компьютерную лабораторию Ericsson. Я никогда не жалел об этом.

Саймон: Почему Erlang?

Я занимаюсь функциональным программированием с 1980-х, Erlang известен мне с момента его появления, это было 20 лет назад. Больше всего меня привлекает в Erlang то, что он изначально создавался для решения практических и сложных задач, причём решения должны были быть элегантными и эффективными. Именно поэтому с недавних пор всё чаще и чаще стали появляться приложения, написанные на Erlang.

Также Erlang — небольшой язык, на нём гораздо проще разрабатывать приложения, чем на Java, C++ или даже Haskell. Благодаря этому и качеству написанных на Erlang библиотек, команда функционального программирования Кентского университета смогла разработать Wrangler — приложение для улучшения кода программ (refactoring), написанных на Erlang.

Для кого эта книга?

Наша книга познакомит вас с Erlang, предварительные знания Erlang или функционального программирования не обязательны.

Мы предполагаем, что у вас есть опыт разработки на Java, C++, Ruby или каком-либо другом популярном языке программирования. По ходу изложения материала мы будем акцентировать внимание на отличиях Erlang от того, к чему вы привыкли.

Как читать эту книгу

Книга состоит из двух частей, главы первой необходимо читать последовательно, а главы второй — можно читать в любом порядке, они не зависят друг от друга.

Первые 11 глав познакомят вас с ключевыми моментами Erlang:

- В главе 1 приведено общее описание возможностей языка. Из неё вы узнаете, почему Erlang подходит для построения надёжных, распараллеленных приложений. Также мы расскажем об истории развития языка и рассмотрим несколько случаев успешного применения, из которых вы узнаете, почему Erlang может оказаться достойным кандидатом для реализации ваших проектов.
- Главы 2 и 3 описывают основы последовательного программирования в Erlang. Мы расскажем вам о *рекурсии* — центральной схеме построения функций в Erlang и об особенностях *единичного присваивания*. Присваивание в Erlang сильно отличается от присваивания из языков Java и С.
- По ходу изучения последовательного программирования мы также поговорим о *базовых типах данных*: числах, атомах, строках, списках, кортежах — и сравним их с аналогичными конструкциями из других языков. В главе 7 мы поговорим о записях, и в главе 9 — о функциональных типах и двоичных данных, а в главе 10 — о ETS-таблицах, предназначенных для хранения большого числа данных.
- В главах 4–6 мы поговорим о том, что отличает Erlang от остальных языков программирования, мы поговорим о параллельном программировании. Параллельное программирование в Erlang основано на *процессах*, которые выполняются в отдельных областях памяти и могут обмениваться данными лишь с помощью *передачи сообщений*.
- Глава 8 посвящена *обновлению приложений* "на лету", то есть без остановки приложения.
- В последней главе первой части мы поговорим о распределённом программировании. Распределённые приложения состоят из нескольких вычислителей (или узлов), которые, работая на одном или нескольких компьютерах, могут совместно решать некоторую задачу.

Оставшиеся главы книги посвящены различным аспектам языка. Среди них:

- *Open Telecom Platform* (OTP) — набор библиотек и шаблонов проектирования, предназначенных для построения надёжных, масштабируемых приложений. О них мы поговорим в главе 12.
- В стандартную поставку Erlang входят несколько полезных приложений. В главе 13 мы поговорим о *базе данных Mnesia*, в главе 14 — о библиотеке разработки *графических интерфейсов wxErlang*.
- Глава 15 посвящена обмену данными через *сокеты*. В главе 16 вы узнаете о том, как Erlang может взаимодействовать с другими языками программирования: с Java, C, Ruby и многими другими.
- В стандартную поставку Erlang входит несколько очень полезных служебных программ, о них мы и поговорим в следующих главах. В главе 17 мы

подробно обсудим отладку в Erlang. В Erlang можно проводить отладку, не снижая производительности приложения. Из главы 18 вы узнаете о средствах проверки корректности программ и о создании документации. Глава 19 посвящена проведению модульного тестирования в Erlang.

- В главе 20 мы расскажем о том, как писать элегантные, наглядные и эффективные программы. В этой главе собран опыт сообщества Erlang, в ней вы найдёте общие соображения по написанию программ.

Из приложения вы узнаете, как установить и запустить Erlang, основы работы с интерпретатором, там мы коснёмся некоторых популярных средств разработки и приведём список дополнительных источников информации о Erlang.

Каждая глава заканчивается набором упражнений. Все примеры из книги могут быть загружены с сайта:

<http://www.erlangprogramming.com>

Также на сайте книги вы можете найти ссылки на дополнительные источники и на сайты сообщества Erlang.

Эта книга совместима с выпуском Erlang R13 (R13-B). Большая часть примеров будет работать и в более ранних версиях, известные нам случаи несовместимости с более поздними версиями приведены на сайте книги.

Условные обозначения

В книге используются следующие условные обозначения:

Курсив

Обозначает новые термины, URL, имена и расширения файлов, интонационное ударение и ключевые фразы.

Моноширинный шрифт

Обозначает компьютерный код, команды, параметры, переменные, атрибуты, ключи, запросы, функции, методы, типы, классы, модули, свойства, флаги, значения, объекты, события, обработчики событий, теги XML и XHTML, макросы и ключевые слова.

Курсив в коде

Обозначает текст, вместо которого подставляются другие выражения, зависящие от контекста.



Картинка обозначает совет, подсказку или общие замечания.



Картинка обозначает предупреждение или предостережение.

Использование кода из примеров

Цель этой книги заключается в том, чтобы вы научились писать программы и приложения на Erlang. Вы можете использовать код из этой книги в ваших программах и документации.

Вам нужно получить разрешение от издательства лишь в том случае, если вы используете значительную часть из примера. Так, если вы взяли несколько кусков из разных примеров и вставили в свою программу, вам не нужно разрешение. Если вы отвечаете на вопрос цитатой из книги, вам также не нужно разрешение.

Вставка существенной части примера в документацию вашего приложения *требует разрешения*. Продажа и распространение CD-ROM с примерами также *требует разрешения*.

Нам будет приятно, если при упоминании цитаты из книги вы будете ссылаться на источник, хотя это и не обязательно. Обычно ссылка включает заголовок, автора, издательство и ISBN. К примеру, "Erlang programming, Франческо Чезарини и Саймон Томпсон. © 2009 Франческо Чезарини и Саймон Томпсон, 978-0-596-51818-9".

Если вы чувствуете, что выходите за указанные выше рамки использования кода, вы можете спокойно связаться с нами по адресу permission@oreilly.com

Safari Books Online

Если вы видите значок Safari ® Books Online на обложке книги, это означает, что книга доступна в Интернете на виртуальной книжной полке O'Reilly.

Safari предлагает возможность посещения виртуальной библиотеки, здесь вы можете просматривать тысячи лучших технических книг, копировать куски кода, загружать главы и искать самые свежие ответы на ваши вопросы. Посетить библиотеку можно бесплатно на сайте <http://my.safaribooksonline.com>.

Как связаться с нами

Комментарии и вопросы по книге могут быть отправлены в издательство O'Reilly по адресу:

O'Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

800-998-9938 (в США или Канаде)

707-829-0515 (международный или местный)

707-829-0104 (факс)

На сайте книги вы можете найти исправления, примеры и другую дополнительную информацию:

<http://www.oreilly.com/catalog/9780596518189>

или на:

<http://www.erlangprogramming.org>

Комментарии, касающиеся технических аспектов книги, отправляйте на почту:

bookquestions@oreilly.com

Узнать о наших книгах, конференциях, филиалах издательства O'Reilly, веб-сети O'Reilly можно на нашем сайте:

<http://www.oreilly.com>

Благодарности

Мы хотели бы поблагодарить всех тех, без кого написание этой книги было бы невозможным. Начнём с Яна "просто Генри" Нистрёма (Jan Nyström), который помог нам начать этот проект.

Мы хотели бы поблагодарить команду O'Reilly за безграничную поддержку, в особенности редактора Майка Лукидеса (Mike Loukides), который терпеливо направлял нас и поддерживал на написание новых глав. Мы хотели бы поблагодарить литературного редактора Одри Дойл (Audrey Doyle), а также Рэйчел Монаган (Rachel Monaghan), Марлоу Шеффера (Marlow Shaeffer), Люси Хаскинс (Lucie Haskins), Самита Махерджи (Sumita Mukherji) и всех тех, кто участвовал в изда-нии этой книги.

Мы хотим поблагодарить команду OTP, в особенности Бьорна Густавсона (Bjorn Gustavsson), Сверкера Эрикссона (Sverker Eriksson), Дэна Гудмундсона (Dan Gudmundsson), Кенет Лундина (Keneth Lundin), Хакана Матсона (Håkan Matsson), Раймо Нисканена (Raimo Niskanen), и Патрика Ниблома (Patrik Nyblom) не только за то, что вводили нас в курс дела самых передовых возможностей Erlang/OTP, но и за точность и корректность комментариев.

Также мы хотели бы отметить тех, кто вносил в книгу ценные замечания и исправления: Тома Артса (Thomas Arts), Зви Аврахама (Zvi Avraham), Франса Божича (Franc Bozic), Ричарда Карлсона (Richard Carlsson), Дейла Харви (Dale Harvey), Оскара Хельстрёма (Oscar Hellström), Стива Кирша (Steve Kirsch), Чарльза Мак Найта (Charles McKnight), Пола Оливера (Paul Oliver), Пьера Омидьяра (Pierre Omidyar), Октавио Ороэзю (Octavio Orozio), Рекса Педжа (Rex Page), Михала Пташека (Michal Ptaszek), Корадо Санторо (Corrado Santoro), Стива Виноского (Steve Vinoski), Дэвида Велтона (David Welton), Ульфа Вигера (Ulf Wiger) и Майка Вильямса (Mike Williams), – мы не будем вдаваться в детали, кто за что отвечает, просто скажем, что с вашей помощью эта книга стала гораздо лучше.

Франческо хотел бы поблагодарить Элисон за терпение и поддержку. Я не догадывался, на что иду, соглашаясь взяться за это дело, да и ты тоже. До следующей книги обещаю тебе выходные без ноутбука и мобильного телефона. Я хотел бы поблагодарить за поддержку команду из Erlang Training and Consulting, а также Саймона — с ним было здорово писать эту книгу. Мне бы хотелось написать с ним ещё одну книгу, результат стоил того, но пока мы отдохнём!

Саймон благодарит Джейн, Алису и Рори за терпение и поддержку в эти тяжёлые месяцы, без вас этого бы никогда не случилось. Спасибо Франческо за приглашение, мне очень понравилось работать вместе с ним. Надеюсь, мы напишем что-нибудь вместе, но не слишком скоро...

Благодарности от российского издательства

Издательство ДМК Пресс выражает благодарность Кириллу Заборскому и Алексею Отту за редактирование текста и ценные замечания.

1 Введение

Почему нам так хочется познакомить вас с Erlang? Чем он так хорош? Легковесная модель параллельных вычислений Erlang остаётся непревзойдённой. Она не зависит от операционной системы и способна справляться с огромным числом одновременно запущенных процессов. В отличие от многих общепринятых языков, Erlang избегает использования разделяемой памяти (*shared memory*), благодаря чему Erlang прекрасно работает на многоядерных процессорах. Отсутствие разделяемой памяти решает проблемы, связанные с синхронизацией и возникновением узких мест. Декларативная природа языка делает программы более краткими и элегантными. Встроенные средства языка позволяют создавать надёжные, отказоустойчивые приложения для работы в режиме реального времени. Также в Erlang предусмотрены мощные средства интеграции, так что приложения, написанные на нём, могут быть легко встроены в большие системы, делая возможным постепенный переход на Erlang.

Несмотря на то что Erlang существует уже давно, сам язык, виртуальная машина и библиотеки шагают в ногу с быстро изменяющимися запросами индустрии программного обеспечения. Команда разработчиков, увлечённых и преданных своему делу профессионалов вместе с учёными из разных стран делают это возможным.

В этой главе мы дадим общий обзор средств языка, посмотрим, в чём причина его успеха, а также расскажем о некоторых фактах из истории языка, оказавших влияние на его развитие. Мы приведём примеры применения Erlang для разработки как коммерческого, так и свободно распространяемого программного обеспечения и проведения научных исследований. Посмотрим, в чём Erlang превосходит другие языки программирования. В заключение главы мы поделимся нашим опытом разработки приложений на Erlang.

1.1 Почему Erlang?

Для каких приложений Erlang подходит лучше всего? Если вы хотите написать приложение, загруженное численными алгоритмами или графикой, или клиентское приложение для мобильных телефонов, извините, вы ошиблись книгой. Но если вы разрабатываете высокоуровневое, распараллеленное, но в то же время надёжное и масштабируемое приложение, предназначенное для работы в режиме реального времени¹, способное выжать максимум из многопроцессорности и

¹Принято различать системы реального времени на hard real-time systems и soft real-time systems: в первом случае временные рамки выполнения операций жёстко установлены, система не имеет

встраиваться в системы, написанные на других языках, Erlang вам подойдёт. Как говорил *Тим Брэй* (Tim Bray), директор веб-технологий в Sun Microsystems на съезде OSCON в июле 2008 года:

”Если бы кто-нибудь попросил меня за крупную сумму написать приложение, связанное с обработкой сообщений, которое должно было бы работать без сбоев годы, я бы, не мешкая, выбрал Erlang”.

Многие компании уже пользуются Erlang:

- Amazon в приложении SimpleDB, базе данных для Amazon EC2 (Amazon Elastic Compute Cloud);
- Yahoo! для Delicious, сервера хранения закладок на веб-страницы, которым пользуются более 5 миллионов пользователей, а число зарегистрированных URL достигает 150 миллионов;
- Facebook для обмена сообщениями между пользователями, этой службой пользуются более 100 миллионов человек;
- T-Mobile в службах SMS и аутентификации;
- Motorola для обработки звонков служб общественной безопасности;
- Ericsson в узлах поддержки для сетей GPRS и 3G по всему миру.

Список наиболее популярных свободно распространяемых приложений, написанных на Erlang:

- Wings 3D — программа 3D-моделирования для создания текстурированных моделей;
- Ejabberd — XMPP-сервер, основанный на IM-сервере;
- CouchDB — документо-ориентированная база данных, не требующая описания схемы данных, позволяющая проводить распределение вычислений между несколькими процессорами или серверами;
- MochiWeb — библиотека, предназначенная для построения легковесных HTTP-серверов. На ней основаны такие службы, как MochiBot и MochiAds, динамически генерирующие содержание веб-страниц для миллионов пользователей ежедневно;
- RabbitMQ — реализация протокола AMQP. AMPQ — развивающийся стандарт для высокопроизводительного обмена сообщениями.

На протяжении многих лет Уппсальский университет возглавляет исследования, относящиеся к Erlang. В рамках проекта HiPE (High Performance Erlang Project) там проводятся исследования, направленные на повышение эффективности Erlang. Не отстают и другие университеты. В Кентском университете (Великобритания) и Университете Итвош Лоран (Венгрия) работают над созданием

права их нарушать, а во втором случае мы стремимся к выполнению заданных требований, но не можем их гарантировать. Далее, если это не оговорено отдельно, мы будем подразумевать под системами реального времени soft real-time systems, поэтому иногда этот термин переводят как системы псевдореального или квазиреального времени.

Конец ознакомительного фрагмента.

Приобрести книгу можно
в интернет-магазине
«Электронный универс»

e-Univers.ru