

Оглавление

Об авторе	17
Предисловие от Джо Армстронга, одного из создателей языка	19
Предисловие	21
Благодарности	23
Вступление	25
Об этом уроке	25
Что такое Erlang?	25
Что вам потребуется, чтобы начать	29
Где получить помощь	31
1 Давайте начнём	33
Интерактивная консоль	33
Ввод команд интерпретатора	34
Выход из интерпретатора	34
Некоторые основы Erlang	35
Числа	36
Неизменные переменные	37
Атомы	39
Булева алгебра и сравнение	40
Кортежи	43
Списки	45
Генераторы списков	48
Работа с двоичными данными	51
Двоичные строки	56
Двоичные генераторы	56
2 Модули	59
Что такое модули?	59
Объявление модуля	60
Компилируем код	64
Параметры компилятора	66

Макросы	67
Больше о модулях	68
Метаданные	68
Циклические зависимости	70
3 Синтаксис функций	71
Сопоставление с образцом	71
Более сложные образцы	73
Переменные в связке	74
Охрана! Охрана!	76
Что за Если?	79
В случае... если (case ...of)	82
Что же лучше?	83
4 Типы (вернее, их отсутствие)	85
Типизация сильная, как динамит	85
Преобразование типов	87
Охрана типов данных	88
Для типозависимых	90
5 Привет, рекурсия	91
Длина списка	92
Длина хвостовой рекурсии	94
Больше рекурсивных функций	95
Функция дублирования duplicate	96
Функция переворота reverse	97
Функция отрезания sublist	98
Функция склеивания пар zip	99
Быстро! Сортируй!	101
Думаем рекурсивно	106
6 Функции высшего порядка	109
Становимся функциональными	109
Анонимные функции	111
Больше анонимных функций	112
Область видимости функции и замыкания	113
Отображения, фильтры, свёртки и так далее	115
Фильтры	116
Сворачиваемся (fold)	117
Больше абстракций	119

7	Ошибки и исключения	121
	Коллекция ошибок	121
	Ошибки компиляции	122
	Нет, ТВОЯ логика ошибочна!	124
	Ошибки времени выполнения	124
	Создание исключений	128
	Исключения-ошибки <code>error</code>	128
	Выходы процессов	129
	Броски исключений <code>throw</code>	130
	Обработка исключений	131
	Обработка разных типов исключений	132
	После <code>catch</code>	135
	Попытка выполнить несколько выражений	135
	Обождите, это ещё не всё!	136
	Попробуйте <code>try</code> в дереве	139
8	Функциональный подход к решению проблем	143
	Калькулятор в обратной польской записи	143
	Как работают RPN-калькуляторы	144
	Создаём RPN-калькулятор	145
	Тестируем код	148
	Из Хитроу в Лондон	150
	Рекурсивное решение проблемы	151
	Пишем код (из Хитроу в Лондон)	153
	Запуск программы без интерпретатора Erlang	158
9	Короткий экскурс в структуры данных	161
	Записи	161
	Объявление записей	162
	Чтение значений из записей	164
	Совместное использование записей	166
	Хранилища данных ключ/значение	167
	Для небольших объёмов данных	167
	Большие хранилища: словари и общие сбалансированные деревья	169
	Множество множеств	171
	Упорядоченные множества <code>ordsets</code>	172
	Множества <code>sets</code>	172
	Общие сбалансированные множества <code>gb_sets</code>	172
	Множества множеств <code>sofs</code>	172
	Ориентированные графы	173
	Очереди	173
	Конец недолгой прогулки	175

10 Автостопом по параллельным вычислениям	177
Не паникуйте	178
Концепции конкурентности	179
Масштабируемость	180
Устойчивость к сбоям	180
Реализация конкурентности	182
Не совсем непохоже на линейный рост	183
Всего хорошего, и спасибо за рыбу!	185
Порождение процессов	185
Отправка сообщений	188
Получение сообщений	189
11 Ещё о параллельной обработке	193
Утверждайте ваше состояние	193
Мы обожаем сообщения, но держим их в секрете	195
Тайм-аут	197
Избирательное получение	200
Подводные камни выборочного приёма сообщений	201
Больше подводных граблей	203
12 Ошибки и процессы	205
Связи	205
Это ловушка!	208
Старые исключения, новые идеи	209
Всё меняет exit/2	211
Убивая меня (не очень) нежно...	213
Мониторы	213
Именованые процессы	215
13 Проектирование параллельного приложения	221
Понимание проблемы	221
Определяем протокол	224
Построим фундамент	226
Модуль событий	227
События и циклы	228
Добавляем интерфейс	231
Сервер событий	233
Обработка сообщений	235
Горячая любовь к коду	238
Я сказал, спрячьте ваши сообщения	240
Пробный запуск	242
Добавляем надзор	243

Пространства имён (вернее, их отсутствие)	245
14 Представляем ОТР	247
Общий вид процесса, абстрактно	248
Простейший сервер	249
Представляем сервер котят	249
Обобщаем вызовы	252
Обобщаем внутренний цикл сервера	253
Функции для старта	254
Обобщаем сервер котят	255
Конкретная реализация против обобщения	257
Обратный вызов в будущее	258
Функция <code>init</code>	259
Функция <code>handle_call</code>	260
Функция <code>handle_cast</code>	260
Функция <code>handle_info</code>	261
Функция <code>terminate</code>	261
.BEAM Me Up, Scotty!	262
15 Ярость против конечных автоматов	267
Что такое конечный автомат?	267
Обобщённые конечные автоматы	271
Функция инициализации	271
Функция <code>ИмяСостояния</code>	272
Функция <code>handle_event</code>	273
Функция <code>handle_sync_event</code>	273
Функции <code>code_change</code> и <code>terminate</code>	274
Спецификация торговой системы	274
Покажи мне свои движения	274
Определяем диаграммы состояний и переходы	276
Игровой обмен между двумя игроками	283
Общедоступный интерфейс	283
Функции общения между КА	284
Функции обратного вызова <code>gen_fsm</code>	286
Это было что-то	296
Готовы к реальному миру?	297
16 Обработчики событий	299
Справься-ка с этим! *перезаряжает ружьё*	299
Обобщённые обработчики событий	300
Функции <code>init</code> и <code>terminate</code>	302
Функция <code>handle_event</code>	302

Функция <code>handle_call</code>	303
Функция <code>handle_info</code>	303
Функция <code>code_change</code>	303
Пришло время кёрлинга!	303
Табло со счётом	304
Игровые события	305
Уведомите прессу!	309
17 Кто присмотрит за наблюдателями?	315
Принципы работы наблюдателей	316
Использование наблюдателей	318
Стратегии перезапуска	319
Ограничения перезапуска	321
Спецификации на детей	321
Репетиция музыкальной группы	324
Музыканты	324
Наблюдатель за группой	328
Динамические процессы-наблюдатели	331
Динамическое использование стандартных наблюдателей	331
Использование наблюдателя <code>simple_one_for_one</code>	333
18 Строим приложение	335
Пул процессов	335
Теория луковых слоёв	337
Дерево для пула	339
Реализация наблюдателей	341
Работаем с работниками	345
Пишем рабочий процесс	352
Беги, пул, беги	354
Чистим бассейн	356
19 Строим приложение в стиле ОТР	359
Пул – мой второй автомобиль	360
Файл ресурсов приложения	361
Преобразование пула	364
Поведение приложения	365
Из хаоса к приложению	367
Библиотечные приложения	371

20	Счетовод для приложений	373
	От ОТР-приложения к настоящему	373
	Файл приложения	374
	Модуль обратного вызова приложения и наблюдатель	376
	Диспетчер	377
	Возврат результатов в стиле продолжений (CPS)	378
	Режимы диспетчера и приёма	381
	Счётчик	388
	Беги, программа, беги	390
	Вложенные приложения	392
	Сложные завершения работы	393
21	Релиз — наше слово	395
	Чиним текущие трубы	395
	Завершение работы виртуальной машины	396
	Обновление файлов приложения	396
	Компилируем приложения	397
	Релизы с помощью Systools	398
	Создание загрузочного файла	399
	Упаковка релиза	400
	Релизы с помощью Reltool	402
	Опции Reltool	407
	Рецепты для Reltool	410
	Отпустите меня, релизы	415
22	Квест по прокачке процессов	417
	Проблемы обновления приложений и релизов	417
	Девятый круг Erl	420
	Process Quest	421
	Приложение regis-1.0.0	422
	Приложение processquest-1.0.0	423
	Приложение sockserv-1.0.0	424
	Создаём релиз	424
	Делаем Process Quest лучше	427
	Обновляем функции code_change	428
	Файлы обновления приложений (appup)	430
	Обновляем релиз	434
	Обзор обновления релиза (relup)	438

23 Ведро сокетов	441
Списки ввода-вывода	441
UDP и TCP: бро-токолы	443
UDP-сокеты	445
TCP-сокеты	448
Больше контроля с помощью inet	451
Возвращаемся к Sockserv	454
Куда дальше?	465
24 Совет Организации модульных наций	467
EUnit – а что такое EUnit?	468
Генераторы тестов	472
Заготовки окружения тестов	474
Больше контроля над тестами	477
Документация для тестов	478
Тестируем приложение regis	479
Тот, кто вяжет EUnit'ы	489
25 Медведи, ETS и корешки: poSQL-база данных в памяти бесплатно	491
Почему ETS	491
Концепция в основе ETS	493
Операции над ETS	495
Создание и удаление таблиц	496
Вставка и поиск данных	498
Встретить свою половинку	500
Вас выбрали!	502
DETS	507
A Little Less Conversation, a Little More Action, Please	508
Интерфейс	509
Подробности реализации	510
26 Распреденомикон	515
Это моя громовая палка	516
Заблуждения о распределённых вычислениях	519
Сеть надёжна	519
Сетевые задержки незначительны	520
Пропускная способность сети бесконечна	521
Сеть хорошо защищена	522
Топология неизменна	523
Сеть администрирует один человек	523
Передача данных бесплатна	524
Сеть однородна	525

В двух словах о заблуждениях	526
Жив, или Живой мертвец	526
Моя вторая кепка — теорема CAP	528
Согласованность	528
Доступность	529
Устойчивость к разделению	529
Выжившие среди зомби и CAP	530
Настройка Erlang-кластера	534
Сквозь пустыню на узле без имени	534
Соединение узлов	535
Ещё инструменты	536
Печеньки (куки)	539
Консоль на удалённом узле	540
Скрытые узлы	541
Стены сделаны из огня, а очки не работают	543
Зов из запределья	544
Модуль net_kernel	544
Модуль global	544
Модуль grpc	546
Закапывая распредемоникон	548
27 Распределённые приложения OTP	551
Добавляем больше к OTP	551
Аварийное переключение и возврат управления	552
Волшебный восьмой шар	555
Строим приложение	555
Модуль наблюдателя	556
Модуль сервера	556
Делаем приложение распределённым	559
28 Common Test для необычных тестов	565
Что такое Common Test?	565
Структура Common Test	566
Создаём простую коллекцию тестов	568
Выполняем тесты	569
Тестирование с состоянием	572
Группы тестов	574
Определение групп тестов	575
Свойства группы тестов	576
Комната совещаний	577
Возвращение коллекций тестов	582
Спецификации тестов	583

Содержимое файла спецификации	583
Создаём файл спецификации	585
Запуск тестов с файлом спецификации	585
Тестирование в больших масштабах	586
Файл спецификации для распределённых тестов	589
Запуск распределённых тестов	590
Интеграция EUnit внутри Common Test	591
Есть ещё?	592
29 Mnesia и искусство помнить	593
Что такое Mnesia?	594
Что хранит хранилище?	595
Данные для сохранения	595
Структура таблицы	596
От записи к таблице	598
О схемах и таблицах Mnesia	599
Создание таблиц	601
Установка базы данных	602
Запуск нашего приложения	605
Доступ и контекст	606
Чтение, запись и даже больше	608
Реализуем первые запросы	610
Тест для добавления услуг	610
Тесты для поиска в базе	613
Учётные записи и новые потребности	617
Встреча с боссом	619
Удаление записей, наглядно	622
Запросы с генераторами списков	625
Помните Mnesia	626
30 Спецификации типов и Dialyzer	629
PLT — это лучший бутерброд	629
Успешная типизация	631
Выведение типов и несовпадения	634
Типа про типы типов	637
Одиночные типы	638
Объединённые и встроенные типы	638
Определение новых типов	642
Типы в записях	643
Типизируем функции	644
Практикуемся в типизации	649
Экспортное определение типов	653

Типизированные поведения	655
Полиморфические типы	657
Мы купили зоопарк	658
Некоторые опасности	660
Ты — мой тип	661
31 Карты	663
Об этой главе	663
EER, EER!	664
Какими будут карты	665
Модуль <code>maps</code>	665
Синтаксис	665
Неприукрашенные подробности	668
Коротенькие ножки для ранних релизов	669
Мексиканское противостояние	670
Карты против записей против словарей	670
Карты против списков свойств	674
Как бы я исправил эту книгу для добавления карт	675
Вот и всё, ребята	675
Послесловие	677
Другие применения Erlang	677
Библиотеки при участии сообщества	679
Ваши идеи меня заинтриговали...	680
Это конец?	680
Приложение: синтаксис Erlang	681
Шаблон	681
Предложение на английском языке	683
И, Или, Готово	684
В качестве вывода	684

Об авторе

Фред Хеберт (Fred Hébert) — программист-самоучка с опытом в разработке веб-сайтов, веб-сервисов и общей разработки серверного программного обеспечения на различных языках. Его онлайн-учебник под названием «Изучай Erlang во имя добра!» («Learn You Some Erlang for Great Good!») считается одним из лучших способов изучить Erlang. Во время его работы в Erlang Solutions Ltd. он занимался созданием учебных материалов и преподавал курсы Erlang во многих странах Запада. В данный момент он работает над платформой ставок в реальном времени на показ рекламы с помощью Erlang (AdGear) и был признан Лучшим пользователем Erlang 2012 года (Erlang User of the Year).

Предисловие от Джо Армстронга, одного из создателей языка

Учиться программированию — весело или, как минимум, должно быть весело. Если для вас это не весело, вам не понравится заниматься этим. Во время моей карьеры программиста я изучил несколько разных языков программирования, и это не всегда было весело. Является ли изучение нового языка весёлым, зависит в большой мере от того, как вам был представлен этот язык.

Когда вы начинаете работать с новым языком программирования, поначалу кажется, что всё, что вы делаете, — это изучаете новый язык. Но если посмотреть глубже, вы занимаетесь кое-чем более основательным — вы изучаете новый способ мышления. Именно этот новый способ мышления и является восхитительным, а не те незначительные подробности пунктуации в языке или как он выглядит в сравнении с вашим любимым языком.

Функциональное программирование — это одна из тех областей программирования, которые приобрели репутацию «сложных» (параллельные вычисления даже ещё сложнее), и таким образом написание книги об Erlang, которая бы освещала идеи функционального программирования *плюс* конкурентного программирования, — это проект, внушающий страх. Не ошибитесь здесь: введение в функциональное программирование не очень лёгкое, и введение в конкурентное программирование тоже имеет свои сложности. Чтобы сделать и то, и другое с юмором и лёгкостью, надо иметь особенный талант.

Фред Хеберт показал, что у него имеется как раз такой талант. Он объясняет сложные идеи так, что они выглядят простыми.

Одним из величайших препятствий на пути изучения Erlang является не столько то, что идеям его свойственна сложность, но и то, что они сильно отличаются от идей большинства других языков, которые вам встречались. Для изучения Erlang вам следует на время забыть то, что вы изучили в других языках. Переменные в Erlang не переменны. Вам не следует программировать в ожидании ошибки. Процессы действительно очень дешёвы, и вы можете иметь тысячи их одновременно, даже миллионы, если вам так захочется. Ох, и потом этот странный синтаксис. Erlang совершенно не похож на Java; нет ни методов, ни классов, ни объектов. И, обождите... знак равенства вовсе не означает «равно», вместо этого он означает «сопоставь с этим образцом».

Фред вовсе не устрашает этих проблем; он работает над темой с деликатным сухим юмором и объясняет сложные темы так просто, что мы забываем о сложности.

Это четвёртая большая книга об Erlang, и она является прекрасным дополнением к библиотеке Erlang. Но она не только про Erlang. Многие из идей в книге Фреда одинаково хорошо подходят и к Haskell, и к OCaml или F#.

Я надеюсь, что многие из вас получат удовольствие, читая книгу Фреда, так же как и я, и что вы сочтёте изучение Erlang процессом приятным и наводящим на мысли. Если вы будете вводить программы, написанные в этой книге, и запускать их по мере чтения, вы узнаете ещё больше. Написание программ намного сложнее, чем их чтение, и первым шагом будет дать вашим пальцам привыкнуть к набору программ и избавиться от мелких ошибок синтаксиса, которые неизбежно происходят. По мере углубления в книгу вы будете писать программы, которые довольно сложно написать на других языках, но надеюсь, что вы не заметите этой сложности в Erlang. Вскоре вы будете писать распределённые программы.

А потом становится весело...

Спасибо, Фред, за прекрасную книгу.
Джо Армстронг (Joe Armstrong)
Стокгольм, Швеция
6 ноября, 2012 г.

Предисловие

Эта книга изначально была написана в виде веб-сайта, который до сих пор доступен по адресу <http://learnyousomeerlang.com/> (спасибо отзывчивости издательства No Starch Press в отношении всех вещей, касающихся публикации и технических материалов).

Поскольку первые главы увидели свет ещё в 2009 году, книга «Изучай Erlang» выросла из небольшого урока на три главы в одну из книг, рекомендованных официальной документацией по изучению Erlang, и стала большим достижением в моей жизни. Я озадачен этим и благодарен за всё, что это мне принесло, начиная от друзей и заканчивая работой и титулом Лучший пользователь Erlang 2012 года.

Когда я начал написание этой книги, одной из моих целей стало сделать её как можно более доступной для как можно большего количества разработчиков. Вот почему английская версия появилась в виде бесплатного веб-сайта, и продолжает оставаться бесплатной по сей день. Одна проблема, которую я не смог решить, это языковой барьер: книга была написана на английском и это сильно ограничило аудиторию её читателей.

Русскоязычное сообщество Erlang пыталось перевести книгу множество раз, часто даже во время её написания и размещения на сайте. Несмотря на эти усилия, ни одна из этих инициатив не принесла плодов. Дмитрий Литовченко нашёл меня в IRC-чате и спросил разрешения выполнить перевод.

То, что вы держите сейчас в руках, является результатом его усилий и настойчивости вместе с поддержкой, которую предоставила команда издательства ДМК Пресс. Большое им спасибо и, надеюсь, вам понравится результат их работы.

Новичку

Когда вы смотрите на Erlang-программистов издалека, как посторонний, они могут показаться странным небольшим сообществом людей, которые верят в принципы, которым больше никто не должен или не хочет следовать. Их принципы выглядят непрактичными, ограниченными в том, как их можно применить. Хуже того, граждане страны Erlang могут выглядеть подобно членам религиозной секты, абсолютно уверенным в том, что им известен один истинный путь к сердцу программного обеспечения. Это тот же «один истинный путь», который недавно проповедовали фанатики языков программирования из семейств Lisp и Haskell, гордые ученики школы мышления в стиле формальных доказательств, программисты на Smalltalk, поклонники стеков из мира Forth и так далее. Всё так же, всё те же; они обещают огромный успех и доставляют обещанное разными способами, но программы, которые

пишем мы, простые программисты, продолжают содержать ошибки, слишком дороги, и их невозможно обслуживать.

Вероятно, вас привело к Erlang обещание конкурентности или параллелизма. Возможно, это аспект языка, касающийся распределённых вычислений, а может быть, необычный подход к устойчивости против сбоев. Конечно, подход к Erlang с долей скептицизма — это хорошая мысль. Erlang не решит всех ваших проблем, в конце концов, это не его, а ваша работа. Erlang — это всего лишь стильный ящик с инструментами, которые помогут вам в этом.

Тем, кто уже знаком с Erlang

Вы уже знаете Erlang, и возможно, даже очень хорошо. В таком случае я надеюсь, что эта книга будет интересной, может, станет справочником, или несколько её глав помогут вам узнать больше о тонкостях языка и его окружения, с которыми вы не были знакомы ранее.

Также, возможно, вы знаете Erlang лучше, чем я, во всех смыслах. В таком случае я надеюсь, что эта книга станет замечательным пресс-папье на вашем столе или заполнит лишнее пространство в вашей библиотеке.

Тем, кто прочитал эту книгу на сайте

Спасибо за вашу поддержку, и надеюсь, что вам понравился профессионально отредактированный оригинальный текст вместе с обновлением материала до версии Erlang R15B+, а также новая глава 31 «Карты», относящаяся к Erlang 17.0.

Вступление

Об этом уроке

Это — начало книги «Изучай Erlang во имя добра!». Прочтение данного урока должно стать одним из ваших первых шагов в изучении Erlang, давайте немного об этом поговорим.

У меня появилась идея написать эту книгу по прочтении бесплатного урока Мирана Липовачи (Miran Lipovača) «Изучай Хаскель во имя добра!» (LYAH)¹; Я посчитал, что он провёл отличную работу, чтобы сделать язык привлекательным и изучение языка дружественным. Поскольку я уже знал его лично, я спросил его, что он думает насчёт того, что я напишу книгу, подобную его книге, но об Erlang. Ему понравилась идея, к тому же он немного интересовался и Erlang.



И вот я печатаю этот текст.

Конечно, у моей мотивации были и другие источники. Когда я начал, я считал введение в язык трудным (Интернет содержит редкую документацию, а книги стоят ощутимо дорого), и я думал, что сообщество только выиграет от уроков в стиле «Изучай Haskell». Также я видел, как люди переоценивали или недооценивали достоинства Erlang на основании бездумных обобщений.

Эта книга, таким образом, — это способ изучить Erlang для людей, которые имеют базовые знания программирования на императивных языках (таких как C/C++, Java, Python, Ruby и так далее) и могут иметь или не иметь опыта в функциональном программировании (таких как Haskell, Scala, Clojure, OCaml...). Я также хочу написать эту книгу в честной манере, «продавая» читателю Erlang таким, какой он есть, освещая его слабые и сильные стороны.

Что такое Erlang?

Erlang — функциональный язык программирования. Если вы когда-либо работали с императивными языками, операторы языка, такие как `i++`, могут казаться вам обычными, но в функциональном программировании они не разрешаются. Фактически менять

¹ Издание на русском языке: <http://dmkpress.com/catalog/computer/programming/functional/978-5-97060-038-5/>.
Онлайн-версия на английском языке: <http://learnyouahaskell.com/>.

значение любой переменной строго запрещается! Это ограничение может звучать поначалу необычно, но если вспомнить школьные уроки математики, переменные ведут себя именно так, как вас учили на уроках:

```
y = 2
x = y + 3
x = 2 + 3
x = 5
```

Если бы я добавил следующее, вы бы оказались сбиты с толку:

```
x = 5 + 1
x = x
таким образом, 5 = 6
```

Функциональное программирование признаёт эту проблему: Если я напишу, что x равно 5, то логически я не могу утверждать, что оно к тому же равно и 6! Это было бы нечестно. А также поэтому функция, вызванная с одним и тем же параметром, должна всегда возвращать один и тот же результат:

```
x = add_two_to(3) = 5
таким образом, x = 5
```

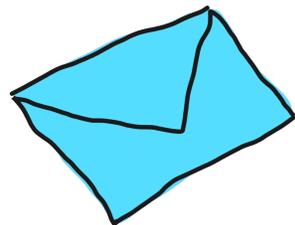
Когда функции всегда возвращают одинаковый результат для одних и тех же входных значений, это называется ссылкой прозрачностью (*referential transparency*). Это позволяет нам заменить однажды вычисленное значение `add_two_to(3)` с результатом 5, поскольку результат выражения $3+2$ будет всегда равен 5. Это означает, что мы можем склеить десятки функций вместе, чтобы решить более сложные проблемы и при этом ничего не сломать в нашем алгоритме. Выглядит логично и чисто, не так ли? Но есть одна проблема:

```
x = today() = 2009/10/22
-- ждём один день --
x = today() = 2009/10/23
x = x
таким образом, 2009/10/22 = 2009/10/23
```

О нет! Мои прекрасные уравнения! Они внезапно все стали неправильными! Как так получилось, что моя функция возвращает разный результат каждый день?

Очевидно, в некоторых случаях следует нарушить ссылочную прозрачность. Erlang подходит к функциональному программированию очень прагматично: подчиняется его чистейшим принципам (ссылочная прозрачность, избегание изменяемых данных и так далее), но позволяет себе нарушить их, когда появляются проблемы реального мира.

Хотя Erlang является функциональным языком программирования, авторы языка также сделали акцент на одновременном исполнении программ и высокой надёжности. Чтобы иметь возможность запускать десятки задач одновременно, Erlang использует так называемую *модель акторов*², где каждый актер — это отдельный процесс внутри виртуальной машины. Простыми словами, если бы вы были актером в мире Erlang, вы были бы одиноким человеком, сидящим в тёмной комнате без окон и ожидающим сообщение из почтового ящика. Как только вы получили сообщение, вы реагируете на него каким-то особым способом: вы платите по счетам, если пришли счета, вы отвечаете на пожелания ко дню рождения письмами благодарности и игнорируете письма, содержание которых вам непонятно.



Модель акторов Erlang можно представить себе как мир, в котором все сидят одни в своих комнатах и могут делать всего несколько различных задач. Все общаются со всеми строго только посредством написания писем, и никак иначе. В то время как это может показаться очень скучной жизнью (и веком расцвета почтовых услуг), это означает, что вы можете попросить множество людей исполнить определённые задачи для вас, и никто из них никогда не ошибётся и не повлияет на работу других. Они даже могут не знать о существовании других людей, кроме вас (и это замечательно).

На практике Erlang заставляет вас писать акторов (процессы), которые не делятся информацией с другими частями кода никак иначе, кроме как передачей сообщений друг другу. Все коммуникации являются явными, безопасными, их можно перехватить и отследить в целях отладки.

Erlang — это не просто язык, но также и среда разработки в целом. Код компилируется в байт-код и работает внутри виртуальной машины. Таким образом, Erlang, так же как Java и как дети-непоседы с шилом в попе, будет бегать где угодно. Вот некоторые из компонентов стандартного пакета поставки:

- инструменты для разработки (компилятор, отладчик, профайлер, система для тестирования, необязательный анализатор типов);
- библиотека Открытой телекоммуникационной платформы (*Open Telecommunication Platform*, или OTP);
- веб-сервер;
- богатые возможности и инструменты трассировки;

² http://ru.wikipedia.org/wiki/Модель_акторов.

- база данных Mnesia (система хранения данных вида ключ/значение, способная реплицироваться на множество серверов, с поддержкой вложенных транзакций и хранящая любые Erlang-значения.

Виртуальная машина и библиотеки также позволяют вам обновлять ваш код прямо во время работы системы без остановки любой программы, распространять ваш код с лёгкостью на множество компьютеров и управлять ошибками и сбоями в простой, но очень мощной манере.

В этой книге мы увидим, как использовать почти все эти инструменты и достичь безопасности.

Говоря о безопасности, вам следует знать об общем правиле в Erlang: позвольте коду упасть, в отличие от падающего самолёта, где могут погибнуть десятки пассажиров, но скорее подобно канатоходцу со страховочной сеткой под ним. В то время как вам следует избегать делать ошибки, вам не нужно проверять возникновение каждой возможной ошибки в большинстве случаев.

Умение Erlang восстанавливаться после ошибок, организация кода в виде акторов и масштабирование вашей системы с помощью распределения и параллельного исполнения звучат отлично, и это подводит нас к следующей части разговора...



Не принимайте всё на веру

Далее по тексту книги вам встретится множество секций в рамке, названных, как эта (вы узнаете их, когда увидите). Erlang в настоящее время набирает популярность из-за яростных обсуждений, которые могут заставить людей поверить, что Erlang является чем-то большим, чем он есть. Эти текстовые вставки будут напоминать вам, что в реальности не всё так радужно, особенно если вы очень впечатлительны.

Первое — это разговоры о массивных возможностях масштабирования Erlang благодаря его легковесным процессам. Это правда, что процессы Erlang очень лёгкие: вы легко можете запустить сотни тысяч их на недорогом ноутбуке, но это не означает, что *следует* их использовать везде просто потому, что вы *можете*. Например, создавать игру-стрелялку, где всё, включая пули, является актором, — это безумие. Единственная вещь, которую вы прострелите в такой игре, — это собственная нога. Передача сообщений между акторами хоть и быстрая, но всё же имеет цену. Если разделять задачи на слишком мелкие, ваша программа получится заметно медленней!

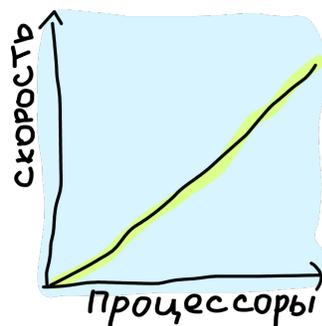
Я распишу это всё ещё раз более подробно, когда мы изучим достаточно материала, чтобы начать беспокоиться об этом. Просто запомните, что для быстрого решения

проблемы не следует применять бездумно параллелизм. Но и не огорчайтесь; во многих проблемах использование сотен процессов не только возможно, но и полезно! Это просто не подходит для всех проблем.

Erlang так же, по слухам, масштабируется прямо пропорционально количеству ядер процессора в вашем компьютере. На практике это не совсем правда: да, это возможно, но очень многие проблемы не позволяют легко разделить решение на множество одновременно работающих задач.

Есть ещё кое-что, о чём следует помнить: в то время как Erlang делает некоторые вещи очень хорошо, технически возможно получить те же результаты с помощью других языков. Также верно и обратное; оценивайте каждую проблему внимательно и выбирайте подходящий для проблемы инструмент. Erlang не является панацеей и будет особенно плохо выглядеть в таких задачах, как обработка графики и звука, драйверы операционных систем и тому подобные низкоуровневые задачи либо задачи, связанные с обработкой массивов данных. В то же время Erlang блестяще справится с задачами больших серверов (очереди заданий и сообщений, веб-серверы, ставки на бирже в реальном времени, реализация распределённых баз данных), с выполнением работы совместно с другими языками, с реализацией сетевых протоколов. Что находится между успехом и полным провалом, зависит только от вас.

Работая с Erlang, не стоит обязательно ограничиваться только серверными приложениями: есть примеры неожиданно успешных и удивительных проектов на Erlang. Например, IANO — робот, созданный командой Eurobot Team университета Катании (University of Catania) и использующий Erlang для искусственного интеллекта. IANO выиграл серебряную медаль на соревновании Eurobot³ в 2009 году. Другой пример — Wings 3D, программа 3D-моделирования с открытым исходным кодом⁴, написанная на Erlang и, таким образом, кросс-платформенная.



Что вам потребуется, чтобы начать

Всё, что вам потребуется, — это удобный для вас текстовый редактор и окружение Erlang. Исходные коды Erlang и установщик для Windows можно найти на официальной странице загрузки⁵.

Для Windows систем просто скачайте и запустите файл установки. Не забудьте добавить папку, куда установился Erlang, в системную переменную PATH, чтобы Erlang стал виден из командной строки.

³ <http://eurobot.dmi.unict.it/>.

⁴ <http://www.wings3d.com>.

⁵ Официальная страница загрузки: <http://erlang.org/download.html>.

ЧТО ВАМ ПОТРЕБУЕТСЯ, ЧТОБЫ НАЧАТЬ

В дистрибутивах Linux, основанных на Debian (в том числе Ubuntu и Mint), вы сможете установить пакет командой:

```
$ sudo apt-get install erlang
```

На Fedora/Redhat (при наличии системной утилиты yum) можно добиться того же эффекта, набрав команду:

```
$ sudo yum install erlang
```

Однако эти репозитории часто содержат устаревшие версии Erlang; использование устаревшей версии приведёт к некоторым отличиям с данной книгой, а также, вероятно, к более медленной работе некоторых приложений. Таким образом, я рекомендую собирать Erlang из исходного кода. Прочтите *README* внутри архива и просмотрите результаты поиска Google для необходимых шагов. Готовые пакеты любых версий под множество дистрибутивов можно найти на сайте загрузки Erlang Solutions⁶.

На FreeBSD для вас доступно множество возможностей. Если вы используете portmaster, то можно исполнить команду:

```
$ portmaster lang/erlang
```

Для обычных портов команда выглядит так:

```
$ cd /usr/ports/lang/erlang; make install clean
```

И наконец, если вы хотите использовать пакеты, исполните команду

```
$ pkg_add -rv erlang
```

Если вы используете Mac OS X, то вы можете установить Erlang с помощью Homebrew⁷:

```
$ brew install erlang
```

или с помощью MacPorts⁸:

⁶ Erlang/OTP: <https://www.erlang-solutions.com/downloads/download-erlang-otp>.

⁷ Mac OS X Homebrew: <http://mxcl.github.com/homebrew/>.

⁸ MacPorts: <http://macports.org>.

```
$ port install erlang
```

ПРИМЕЧАНИЕ. Во время написания этой книги я использовал Erlang версии R15B, таким образом, для наилучших результатов используйте как минимум такую или более новую версию. Однако почти всё содержимое этой книги подходит и для более старых версий R13B и выше.

Вместе с Erlang вам также следует скачать и набор файлов, доступных для этой книги. Они содержат проверенные копии любой программы или модуля, написанных на этих страницах, и могут оказаться полезными для исправления ваших собственных программ. Также они могут стать основой для более поздних глав книги, если вы пожелаете пропустить начальные главы. Все файлы упакованы в ZIP-архив и доступны по ссылке <http://learnyousomeerlang.com/static/erlang/learn-you-some-erlang.zip>. Кроме этого, приведённые в книге примеры не зависят больше ни от каких других программ или библиотек.

Где получить помощь

Существует несколько мест, где можно попросить и получить помощь. Если вы используете Linux, то вы можете открыть man-страницы и получить локально аналог официальной HTML-документации. Например, в Erlang есть модуль `lists` (который мы скоро попробуем): чтобы получить документацию по модулю, просто напечатайте в консоли:

```
$ erl -man lists
```

На Windows установка должна содержать HTML-документацию. Вы можете скачать её в любое время с сайта официальной документации Erlang <http://erlang.org/doc/> или проконсультироваться с неофициальной HTML-документацией по Erlang <http://erldocs.com/>, которая несколько лучше организована.

Рекомендации по лучшим практикам программирования можно найти в онлайн-книге⁹ (на английском), как только вы захотите сделать свой код лучше и чище. Код в этой книге также по возможности будет следовать указаниям упомянутой книги.

И наконец, бывают моменты, когда просто документации совсем недостаточно. Когда такое случается, я обращаюсь к двум основным источникам: официальная почтовая рассылка¹⁰ (достаточно даже просто читать её, чтобы многое узнать и понять) а также каналы чата `#erlang`¹¹ (английский) и каналы `erlang` и `erlang-talks` в сети XMPP-чатов Jabber.ru (русский).

⁹ http://www.erlang.se/doc/programming_rules.shtml

¹⁰ <http://www.erlang.org/static/doc/maillinglist.html>

¹¹ <irc://irc.freenode.net/erlang>

Да, и если вы предпочитаете кормиться с готовых кулинарных книг и списков решений и рецептов, большая коллекция собрана здесь: «Erlang Central: Cookbooks и Tutorials» в разделе Documentation¹².

¹² <https://erlangcentral.org/>.