Оглавление

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 дёшевы, 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 устрашается 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.

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:

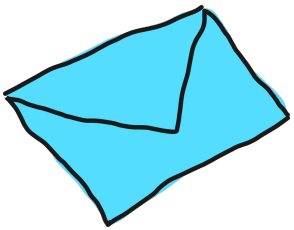
```
x = add_two_to(3) = 5
таким образом, x = 5
```

Когда функции всегда возвращают одинаковый результат для одних и тех же входных значений, это называется ссылочной 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 актор — это отдельный процесс внутри виртуальной машины. Простыми словами, если бы вы были актором 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

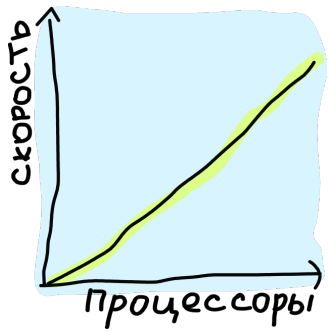
Далее по тексту книги вам встретится множество секций в рамке, названных, как эта (вы узнаете их, когда увидите). Erlang в настоящее время набирает популярность из-за яростных обсуждений, которые могут заставить людей поверить, что Erlang является чем-то большим, чем он есть. Эти текстовые вставки будут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

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 Eurobot[3] в 2009 году. Другой пример — Wings 3D, программа 3D-моделирования с открытым исходным кодом[4], 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 закачки[5].

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 закачки: *http://erlang.org/download.html.*

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 закачки Erlang Solutions[6].

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 Homebrew[7]:

```
$ brew install erlang
```

или с помощью MacPorts[8]:

[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-книге[9] (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 рассылка[10] (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/mailinglist.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 Documentation[12].

[12] *https://erlangcentral.org/*.