

Erlang — Lurkmore

«* komar поехал на конференцию по erlang

<kran_> На конференцию по erlang можно ехать одновременно разными видами транспорта.

»

— <http://ipfw.ru/quotes/9806>

Эрланг (пинд. *Error Language*, также *Enlarg*, *Телебейсик*) — **небыдлокодерский** язык программирования, придуманный расовой шведской компанией Ericsson (без Sony) с целью **захватить мир**. Выгодно отличается от **Хаскелла** тем, что на нём реально можно писать полезные программы (и даже зарабатывать кровавые доллары). Язык лексически еурсивен, на 13.7% более компактен, функционален, многопоточен и необычайно полон **матана**. На нём можно делать **динамические опердени**, не приседая со штангой! Официально заявлено, что он назван в честь **математика Агнера Краупа Эрланга**, но всем понятно, что это аббревиатура от Ericsson Language.

Кратко →

Эрланг позиционируется как язык всея всего, **чудесная таблетка** от запоров, **быдлокодерства** и школоты. Говорят, что на нём легко наваять чудо-мега-пуперсистему, книгу бытия и ответить на **главный вопрос о смысле жизни**. На практике же скатывается в обыкновенное **говно**, вконец запутывающее новобранцев и иже с ними. Содержит много **лямбда-ямбда** и **синхрофозантрон-конструкций**.

Такое положение вещей вынуждает **приверженцев** старого мира активно бурлить, попутно вставляя новые **костыли** в старый-добрый **C**. Особо прыткие **уже так навставляли**, что мама не горюй.

Сам эрланг ничего не делает, а только **даёт указания**.

Кошерен, ибо **не хранит данные вместе с кодом**.

Любители **выебнуться** писать на эрланге хвастаются тем, что могут наплодить кучу потоков, работающих быстрее света, но мы-то знаем, что за все это время они наплодили чуть менее, чем нихуя программ.

Кошерности →

- Может менять куски кода один на другой прямо в продакшне, что невозбранно создаёт возможности для кровавого патчинга. До этого подобная фича применялась разве что в **Лиспе**. Ну и в **РНР** используется чуть менее, чем везде.
- Может с лёгкостью создавать **тысячи тредов**. Треды в эрланге — вовсе и не треды даже, а акторы (*Actors*), общающиеся между собой записочками.
- Может одинаково легко взаимодействовать с процессами как на своем, так и на других хостах.
- В аццкий набор стандартных библиотек входит написанная на нем же распределенная по-sql **СУБД** — mnesia (плюс плюшки вроде ETS таблиц для хранения данных в ОЗУ и DETS — для хранения того же, но на диске).
- Будучи практически **Ъ-функциональным** языком, основательно (не как **Haskell**, но заметно) препятствует использованию изменяемых данных. Это вызывает **баттхёрт**, но почти исключает взаимную блокировку потоков исполнения во время записи данных (кто пытался отлаживать такое в распределённых средах, тот оценит).
- Уникальный подход к обработке ошибок: ты не епошишь весь код в try/catch как последний школий, а строишь так называемые деревья наблюдения (supervision tree), правда, если ты не крайнее днище и всё-таки используешь **ОТР**.
- Честная вытесняющая многозадачность. В отличие от прочих языков, в которых переключение между "зелёными" потоками обычно происходит только при вызове функций из рантайма языка, Enlarg люто-бешено вытесняет потоки, истратившие свой лимит редукций, буквально на **полусло**
- Содержит развитый до фанатичной ебанутости фреймворк для работы с SNMP.

Некошерности →

- Динамическая типизация. Большую часть времени настоящий Эрланг-программист проводит за размышлениями на тему «откуда прилетела эта [НЭХ](#), обрушившая мой уютный gen_server».
- Производительность. Во многом следствие первого пункта — эрланг столь же стремителен, как [известный розовый покемон](#).
- Records. Нормальных записей в Эрланг не завезли, приходится обходиться костыльной надстройкой над кортежами.
- Guards. В отличие от того же Haskell, где в guard-expression можно запихнуть хоть Аллаха, в Эрланге есть только унылые match-expressions.
- Отсутствие чистоты. Что ни говори, а Эрланг всё-таки не чисто-функциональный язык.
- Сложность анализа кода и отладки. По возможности устроить спагетти из кучи процессов, ожидающих невесть что, невесть откуда...
- Нельзя перегружать инфиксные операторы и, как следствие, нет возможности управлять control flow с помощью продолжений, iteratees и прочих [монад](#). Отчасти пофикшено в Elixir.
- Излишнее упование на принцип Let it crash приводит к тому, что чаще всего супервизоры нужны только для того, чтобы превратить быструю смерть приложения в более долгую и мучительную.
- Мелкие ошибки в коде, помноженные на косяки в виртуальной машине, могут с течением времени приводить к поистине термоядерным проблемам. Утечки атомов, переполненные мейлбоксы — всё это медленно, но неизбежно отправляет VM в Адъ.

«Hello, Joe! Hello, Mike!» →

[Erlang: The Movie](#)

Вот что случается, если программистам дать видеокамеру

«— Hello, Joe.

— Hello, Mike. — Hello, Mike. — Hello, Robert. — Hello, Joe.

»

— разработчики эрланга об основах языка

Интернет-общественности язык известен прежде всего как буквы на экране компьютера в знаменитом видео, где инженеры Эрикссон показывали в камеру свои бледные лица и названивали друг другу по телефону с непонятными целями. Для непосвящённого человека естественна единственная реакция: [«what?»](#)

Пример кода на эрланге →

Первая функция сработает, если передать сортировке пустой список ;
 Вторая функция разделит список на заголовок «Pivot» и хвост «Rest», после чего →
 дважды войдёт в рекурсию, выполняя операторы ++ и [перемешивая || элементы ← списков] .

```
%% quicksort:qsort(List)
%% Sort a list of items
-module(quicksort).
-export([qsort/1]).
```

```
qsort([]) -> [];
qsort([Pivot|Rest]) ->
qsort([ X || X <- Rest, X < Pivot]) ++ [Pivot] ++ qsort([ Y || Y <- Rest, Y >= Pivot]).
```

Что характерно, этот недо-QuickSort будет практически одинаково выглядеть на 95% функциональных языков, исключая, разве что, LISP.

Алсо, это чуть ли не единственная на планете реализация быстрой сортировки, требующая $O(n^2)$ дополнительной памяти в худшем случае из-за врождённой ущербности языка (на любом нормальном языке дополнительной памяти не требуется вовсе). Про скорость работы лучше просто промолчать. Но любителей эрланга не останавливают такие мелочи, как реальность, практическая применимость и здравый смысл. Какая разница, что quicksort перестал быть quick (и существенный пласт алгоритмов в

принципе не реализуем на эрланге с сохранением асимптотической сложности)? Зато строчек мало, и можно прикинуться умным перед коллегами!

Что за _ →

IRL, это был пример взлетевшего велосипеда, когда компании потребовалось написать один-единственный продукт на тему телефонии.

Огромное количество нужных программистам вещей в язык не вошли просто потому, что они не понадобились в том единственном проекте в самом начале. Массивы? **Не надо**. Циклы? **Не знаем**. Строки? **Не, не слышали**. Когда петух клюнул их в задницу, строки пришлось приматывать **изоляцией** как **односвязный список чисел**. Отличать список чисел от строк язык не может до сих пор и перед выводом строки на экран просто пробегается по всем числам. Все маленькие? Кажется, это буквы. **100500**? Наверное, это не буквы...

Перерабатывать язык никто не стал, потому что на сыром прототипе уже был запущен проект Эрикссона — и весь код у них оказался write only. Разумеется, пришлось городить дополнительные костыли для структурирования кода. Разумеется, они плохо вписались в базовый язык.

Всё это разительно отличается от других языков для небудда — **лиспа** и **хаскеля**. Первый прошел бесконечно долгую и болезненную эволюцию на реальных задачах, породил множество диалектов, и, спустя полвека, наконец-то принял современный вид (**точнее два**). Второй годами вылизывался яйцеголовыми теоретиками (больше философами, чем программистами) и стал как минимум целостным. Вся история эрланга вкратце — **завелось и поехало**.

Но →

Почему взлетел?...

Лучше всех это знает Джо Армстронг, автор языка. Вот что он пишет в своей книге «Programming Erlang»:

В начальной школе учительница объясняла нам: «Если в разных частях выражения стоит переменная X, то все они имеют одно и то же значение». И мы решали уравнения: так как мы знали, что $X+Y=10$ и что $X-Y=2$, то мы делали вывод, что $X=6$ и что $Y=4$ в обоих выражениях.

Затем мы перешли к изучению программирования и нам показали строку: $X = X + 1$

Мы были против и нам казалось, что так делать нельзя. Но учитель объяснил нам, что мы не правы и что нам надо разучиться и забыть то, что мы знали раньше. «Икс», объяснил он, это такая коробочка, где лежат данные, и её содержимое можно менять.

Эрланг не позволяет менять переменные: они выглядят как обозначения в математических выражениях. Когда вы связываете переменную со значением, вы утверждаете некий факт. Теперь переменная имеет это значение — и теперь так и будет.

А раз нет изменяемого состояния, то программу можно разделить надвое в любой точке — и она будет выполняться на двух ядрах или на двух процессорах, без семафоров и блокировок. Встречные блокировки (когда один поток захватывает семафоры A,B,C,D, а другой захватывает D,C,A и они останавливаются и ждут друг друга до бесконечности) тоже не возникают. **Так всё ли ясно?**

Подводный кот — такой он один.

Конкуренентов в занимаемой нише у него не было и нет.

Алсо →

- Джо Армстронг — **байкер**.
- На Erlange написан православный **джаббер**-сервер ejabberd, который используют на самых высоконагруженных джаббер-серверах в мире. Автор сего винрара, выходец из Севастополя в забугорье через **Дефолт-сити**, Алекс Щепин^[1], не иначе как за это самое был удостоен звания лучшего программера на эрланге в 2006 году [1].
- Первая версия Эрланга была написана на **Prolog**.
- Эрланг — единица измерения загрузки телефонных линий.
- На Эрланге написан RabbitMQ, Вики: [2].
- Эрлангер — персонаж inferнального опуса **нашего всего** «Замок».
- **Эрлан** (Erlang) — китайский бог с третьим глазом во лбу.

Примечания →

1. ↑ Он же соавтор джаббер-клиента Tkabber, но уже на тикле

См. также →

- [Haskell](#)
- [Нерд](#)
- [Жаббер](#)

Ссылки →

- [Wings3D](#) — редактор для [низкополигонального моделирования](#). Написан на сабже.



Языки программирования

++i + ++i 1C AJAX BrainFuck C Sharp C++ Dummy mode Erlang Forth FUBAR
God is real, unless explicitly declared as integer GOTO Haskell Ifconfig Java JavaScript LISP
My other car Oracle Pascal Perl PHP Prolog Pure C Python RegExp Reverse Engineering
Ruby SAP SICP Tcl TeX Xyzy Анти-паттерн Ассемблер Быдлокодер
Выстрелить себе в ногу Грязный хак Дискета ЕГГОГ Индусский код Инжалид дежице
Капча КОИ-8 Костыль Лог Метод научного тыка Очередь Помолясь Проблема 2000
Программист Процент эс Рекурсия Свистелки и перделки Спортивное программирование
СУБД Тестировщик Умение разбираться в чужом коде Фаза Луны Фортран Хакер
Языки программирования

[ae:Erlang en:w:Erlang w:Erlang](#)