The Pragmatic Programmers

TURING 图灵程序设计丛书

Erlang之父权威著作

带你领先一步，精通下一代主流编程语言

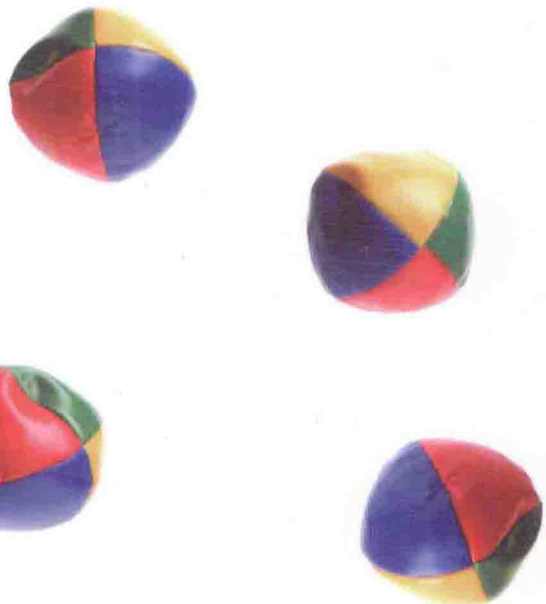
Erlang程序设计

（第2版）

Programming Erlang

Software for a Concurrent World

Second Edition



[瑞典] Joe Armstrong 著

牛化成 译

人民邮电出版社

POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Erlang程序设计

（第2版）

【瑞典】Joe Armstrong 著
牛化成 译

Programming Erlang

Software for a Concurrent World
Second Edition

人民邮电出版社
北　京

图书在版编目（CIP）数据

Erlang程序设计 : 第2版 / (瑞典) 阿姆斯特朗(Armstrong, J.) 著 ; 牛化成译. -- 北京 : 人民邮电出版社, 2014.6
(图灵程序设计丛书)
ISBN 978-7-115-35457-0

Ⅰ. ①E… Ⅱ. ①阿… ②牛… Ⅲ. ①程序语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2014)第097736号

内 容 提 要

本书由 Erlang 之父 Joe Armstrong 编写，是毋庸置疑的经典著作。书中兼顾了顺序编程、并发编程和分布式编程，深入讨论了开发 Erlang 应用中至关重要的文件和网络编程、OTP、ETS 和 DETS 等主题。新版针对入门级程序员增加了相关内容。

本书适合 Erlang 初学者和中级水平 Erlang 程序员学习参考。

◆ 著 [瑞典] Joe Armstrong
译 牛化成
责任编辑 朱 巍
执行编辑 陈婷婷
责任印制 焦志炜

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：28
字数：662千字 2014年6月第1版
印数：1－4 000册 2014年6月北京第1次印刷
著作权合同登记号 图字：01-2013-8803号

定价：89.00元

读者服务热线：(010)51095186转600 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

版权声明

这个世界是并行的。

如果希望将程序的行为设计得与真实世界物体的行为相一致，那么程序就应该具有并发结构。

使用专门为并发应用设计的语言，开发将变得极为简便。

Erlang程序模拟了人类如何思考，如何交互。

——Joe Armstrong

第1版推荐序

Erlang算不上是一种“大众流行”的程序设计语言，而且即使是Erlang的支持者，大多数也对于Erlang成为“主流语言”并不持乐观态度。然而，自从2006年以来，Erlang语言确实在国内外一批精英程序员中暗流涌动，光我所认识和听说的，就有不少于一打技术高手像着了魔一样迷上了这种已经有二十多年历史的老牌语言。这是一件相当奇怪的事情。因为就年龄而言，Erlang大约与Perl同年，比C++年轻四岁，长Java差不多十岁，但Java早已经是工业主流语言，C++和Perl甚至已经进入其生命周期的下降阶段。照理说，一个被扔在角落里二十多载无人理睬的老家伙合理的命运就是坐以待毙，没想到Erlang却像是突然吃了返老还童丹似的在二十多岁的“高龄”又火了一把，不但对它感兴趣的人数量激增，而且还成立了一些组织，开发实施了一些非常有影响力的软件项目。这是怎么回事呢？

根本原因在于Erlang天赋异禀恰好适应了计算环境变革的大趋势：CPU的多核化与云计算。

自2005年C++标准委员会主席Herb Sutter在*Dr. Dobb's Journal*上发表《免费午餐已经结束》一门以来，人们已经确凿无疑地认识到，如果未来不能有效地以并行化的软件充分利用并行化的硬件资源，我们的计算效率就会永远停滞在仅仅略高于当前的水平上，而不得动弹。因此，未来的计算必然是并行的。Herb Sutter本人曾表示，如果一门语言不能够以优雅可靠的方式处理并行计算的问题，那它就失去了在21世纪的生存权。“主流语言”当然不想真的丧失掉这个生存权，于是纷纷以不同的方式解决并行计算的问题。就C/C++而言，除了标准委员会致力于以标准库的方式来提供并行计算库之外，标准化的OpenMP和MPI，以及Intel的Threading Building Blocks库也都是可信赖的解决方案；Java在5.0版中引入了意义重大的concurrency库，得到Java社区的一致推崇；而微软更是采用了多种手段来应对这一问题：先是在.NET中引入APM，随后又在Robotics Studio中提供了CCR库，最近又发布了Parrallel FX和MPI.NET，可谓不遗余力。然而，这些手法都可以视为亡羊补牢，因为这些语言和基础设施在创造时都没有把并行化的问题放到优先的位置来考虑。与它们相反，Erlang从其构思的时候起，就把“并行”放到了中心位置，其语言机制和细节的设计无不从并行角度出发和考虑，并且在长达二十年的发展完善中不断成熟。今天，Erlang可以说是为数不多的天然适应多核的可靠计算环境，这不能不说是一种历史的机缘。

另一个可能更加迫切的变革，就是云计算。Google的实践表明，用廉价服务器组成的服务器集群，在计算能力、可靠性等方面能够达到价格昂贵的大型计算机的水准，毫无疑问，这是大型、超大型网站和网络应用梦寐以求的境界。然而，要到达这个境界并不容易。目前一般的网站为了达成较好的可延展性和运行效率，需要聘请有经验的架构师和系统管理人员，手工配置网络服务

端架构，并且常备一个高水准的系统运维部门，随时准备处理各种意外情况。可以说，虽然大多数Web企业只不过是想在这些基础设施上运行应用而已，但仅仅为了让基础设施正常运转，企业就必须投入巨大的资源和精力。现在甚至可以说，这方面的能力成了大型和超大型网站的核心竞争力。这与操作系统成熟之前人们自己动手设置硬件并且编写驱动程序的情形类似——做应用的人要精通底层细节。这种格局的不合理性一望便知，而解决的思路也是一目了然——建立网络服务端计算的操作系统，也就是类似Google已经建立起来的“云计算”那样的平台。所谓“云计算”，指的是结果，而当前的关键不是这个结果，而是作为手段的“计算云”。计算云实际上就是控制大型网络服务器集群计算资源的操作系统，它不但可以自动将计算任务并行化，充分调动大型服务器集群的计算能力，而且还可以自动应对大多数系统故障，实现高水平的自主管理。计算云技术是网络计算时代的操作系统，是绝对的核心技术，也正因此，很多赫赫有名的中外大型IT企业都在不惜投入巨资研发计算云。包括我在内的很多人都相信，云计算将不仅从根本上改变我们的计算环境，而且将从根本上改变IT产业的盈利模式，是真正几十年一遇的重大变革，对于一些企业和技术人员来说是重大的历史机遇。恰恰在这个主题上，Erlang又具有先天的优势，这当然也是归结于其与生俱来的并行计算能力，使得开发计算云系统对于Erlang来说格外轻松容易。现在Erlang社区已经开发了一些在实践中被证明非常有效的云计算系统，学习Erlang和这些系统是迅速进入这个领域并且提高水平的捷径。

由此可见，Erlang虽然目前还不是主流语言，但是有可能会在未来一段时间发挥重要的作用，因此，对于那些愿意领略技术前沿风景的“先锋派”程序员来说，了解和学习Erlang可能是非常有价值的投资。即使你未来不打算使用Erlang，也非常有可能从Erlang的设计和Erlang社区的智慧中得到启发，从而能够在其他语言的项目中更好地完成并行计算和云计算相关的设计和实现任务。再退一步说，就算只是从开启思路、全面认识计算本质和并行计算特性的角度出发，Erlang也值得了解。所以，我很希望这本书在中国程序员社区中不要遭到冷遇。

本书是由Erlang创造者Joe Armstrong亲自执笔撰写的Erlang语言权威参考书，原作以轻松引导的方式帮助读者在实践中理解Erlang的深刻设计思路，并掌握以Erlang开发并行程序的技术，在技术图书中属于难得的佳作。两位译者我都认识，他们都是技术精湛而思想深刻的“先锋派”，对Erlang有着极高的热情，因此翻译质量相当高，阅读起来流畅通顺，为此书中译本添色不少。有兴趣的读者集中一段时间按图索骥，完全有可能就此踏上理解Erlang、应用Erlang的大路。

孟岩

IBM中国公司媒体关系主管

前CSDN首席分析师兼《程序员》杂志技术主编

前　言

越来越多的新硬件是并行的，所以新的编程语言必须支持并发性，否则它们将会灭亡。

“处理器行业的发展方向是不断添加更多的核心，但是没人知道如何给它们编程。我的意思是：双核？可以。四核？很难。八核？算了吧。”

——史蒂夫·乔布斯[①]

乔布斯说得不对，我们其实知道如何给多核编程。我们在Erlang里就这么做，而且核心越多，许多程序就跑得越快。

Erlang从一开始就被设计用于自下而上地编写并发式、分布式、容错、可扩展和软实时（soft real-time）系统的程序。软实时系统是指电话交换机和银行业务系统这样的系统，对它们而言，快速的响应时间很重要，但偶尔错过了时限也不是什么灾难性的。Erlang系统已经被大规模部署，并且控制了全世界许多重要的移动通信网络。

如果你的问题是并发的，或者正在组建多用户的系统，或者你组建的系统需要随时间而改变，那么使用Erlang也许会为你节省大量的工作，因为Erlang就是特别为组建这些系统而设计的。

“问题在于可变状态，傻瓜。”

——摘自Brian Goetz所著《Java并发编程实战》

Erlang是函数式编程语言。函数式编程禁止代码存在副作用。副作用和并发性不能共存。在Erlang里，改变单个进程内部的状态是允许的，但一个进程改动另一个的状态则不行。Erlang里没有互斥，没有同步方法，也没有内存共享式编程的各种设备。

各个进程能且只能用一种方法进行交互，那就是交换消息。进程之间不共享任何数据。这就是我们可以把Erlang程序轻松部署到多核或网络的原因。

编写Erlang程序时，实现的方式不是让单个进程执行所有任务，而是生成大量只做简单事情的小进程，并让它们互相通信。

本书主要内容

本书介绍并发性、分布、容错和函数式编程，阐述如何编写一个没有锁与互斥、只是纯粹使

① http://bits.blogs.nytimes.com/2008/06/10/apple-in-parallel-turning-the-pc-world-upside-down/

用消息传递的分布式并发系统，如何在多核CPU上自动加速程序，如何编写让人们互相交流的分布式应用程序。还介绍了如何编写容错和分布式系统的设计模式，如何给并发性建模、再把这些模型映射到计算机程序上（我把这一过程称为面向并发性的程序设计）。

目标读者

本书的目标读者上至经验丰富但想要了解更多Erlang内部细节和背后哲学的Erlang程序员，下至不折不扣的初学者。书中的内容已经被从专家到初学者的各级程序员审读过。第2版与第1版的一个主要区别是，第2版添加了大量针对初学者的解释性材料。高级Erlang程序员可以跳过这些介绍材料。

本书还想阐明函数式编程、并发编程与分布式编程，并将它们以合适的方式呈现给之前不了解并发或函数式编程的读者。编写函数式程序和并行程序长久以来都被当成是一种“巫术”，希望本书能让人们对此有新的认识。

虽然本书不假定读者拥有特定的函数式或并发编程知识，但是需要读者熟悉一两种编程语言。

当你接触一门新的编程语言时，往往难以想到“适合用新语言解决的问题”。书中的练习会给你一些提示。这些问题很适合在Erlang里解决。

新版说明

首先，书中的内容已经做了更新，以反映出自第1版面世以来Erlang历经的所有变化。现在我们已经涵盖了所有的官方语言改动，并介绍了Erlang的R17版。

第2版把焦点转移到了满足初学者的需求上，相比第1版有了更多的解释性文字。那些针对高级用户或可能迅速变化的材料已经被转移到了在线资源区。

事实证明，第1版里的编程练习非常受读者欢迎，因此现在每一章的最后都附上了练习。这些练习复杂程度各异，所以初学者和高级用户总能找到适合自己的。

在新增的一些章节中，你将学到Erlang的类型系统和Dialyzer、映射组（在Erlang的R17版里新增）、WebSocket、编程习惯用语及如何集成第三方代码。还新增了一篇附录，介绍如何组建一个独立的最小化Erlang系统。

本书的最后一章“福尔摩斯的最后一案”是全新的，里面提供了一项练习：处理大批量的文本并从中提取意义。这一章是开放式的，希望它后面的练习能激发更进一步的工作。

路线图

要学会跑，必须先学会走路。Erlang程序是由许多同时运行的小型顺序程序组成的。在编写并发代码之前，我们需要能够编写顺序代码。这就意味着在第11章之前，我们不会深入到编写并发程序的细节当中。

- 第一部分简单介绍了并发编程的核心概念，并带你快速游历Erlang。

- 第二部分详细介绍了Erlang的顺序编程，还讨论了构建Erlang应用程序所需的类型和方法。
- 第三部分是本书的核心，我们将学习如何编写并发和分布式的Erlang程序。
- 第四部分讨论了主要的Erlang库、跟踪和调试的技巧，以及组织Erlang代码的技巧。
- 第五部分涉及应用程序。你会学到如何将外部软件与Erlang的核心库集成，以及如何转换你自己的代码来把它奉献给开源事业。我们将讨论编程习惯用语和如何为多核CPU编程。最后，“夏洛克·福尔摩斯”会分析我们的思路。

在每一章的最后，你都会看到一组精心挑选的编程练习。它们的目的是测试你对本章知识的掌握情况，同时向你发起挑战。这些问题难度各异，最大的难题是合适的研究项目。即使你并不打算尝试解决所有问题，单单思考问题本身以及如何解决问题也会增强你对书中内容的理解。

本书里的代码

大多数代码片段都来源于可下载的完整运行范例[①]。为了方便读者查询，如果书中的某个代码清单可供下载，代码片段上方就会有一个提示条（就像此处所显示的）：

shop1.erl

```
-module(shop1).
-export([total/1]).

total([{What, N}|T]) -> shop:cost(What) * N + total(T);
total([])            -> 0.
```

这个提示条内含代码在下载区里的路径。如果你正在阅读本书的电子版，并且你的电子书阅读器支持超链接，就可以点击提示条，代码应该会出现在浏览器窗口中。

帮帮我！出问题了

学习新东西并不容易，你会遇到棘手的难题。当你遇到难题时，原则一是不要轻易放弃。原则二是寻求帮助。原则三是询问“福尔摩斯”。

原则一很重要。有些人尝试了Erlang，遇到难题后选择了放弃，没有告诉任何人。如果我们不知道某个问题存在，我们就无法修复它。

寻求帮助的最佳方式是首先用谷歌找找看，如果谷歌帮不上忙，你可以给Erlang的邮件列表[②]发送邮件。如果想要更快速的响应，还可以试试`irc.freenode.net`上的`#erlounge`或`#erlang`。

有时候，问题的答案也许在Erlang邮件列表的某篇旧帖子上，但你就是找不到。在第27章有一个可以本地运行的程序，它能对Erlang邮件列表里的所有旧帖子执行复杂的搜索。

言归正传，下面感谢那些帮助我编写本书（以及本书第1版）的人们。你可以跳过去这一内容直奔第1章，在那里我将带你快速游历Erlang。

① http://www.pragprog.com/titles/jaerlang2/source_code

② erlang-questions@erlang.org

致谢

第 1 版

许多人对本书的准备工作提供了帮助，我想在这里对他们表示感谢。

首先，感谢编辑Dave Thomas。Dave指导我写作，提出的问题无穷无尽。为什么要这样？为什么要那样？当我开始编写这本书时，Dave说我的写作风格就像是“站在岩石上说教”。他说：“我想要你和人们对话，而不是说教。”这本书因此变得更好了。谢谢你，Dave!

其次，感谢我身后由语言专家组成的委员会。他们帮助我决定该省去哪些内容，还帮助我阐明了某些难以解释的部分。在这里感谢（排名不分先后）Björn Gustavsson、Robert Virding、Kostis Sagonas、Kenneth Lundin、Richard Carlsson和Ulf Wiger。

感谢Claes Vikström提供了有关Mnesia的宝贵建议，感谢Rickard Green提供了有关SMP Erlang的信息，也感谢Hans Nilsson提供了用于文本索引程序的单词归一算法（stemming algorithm）。

Sean Hinde和Ulf Wiger帮助我理解了如何使用各种OTP内部细节。Serge Aleynikov向我解释了活动套接字以帮助我理解。

感谢妻子Helen Taylor，她校对了好几章，并在我需要提神醒脑时端来热茶。不仅如此，她还容忍了我在7个月的时间里表现出的相当沉迷的行为。另外还要感谢Thomas和Claire，感谢Bach和Handel，感谢我的猫Zorro和Daisy，以及我的“卫星定位”Doris，它帮助我保持理智，在被抚摸时会呜呜叫，还会把我带往正确的地址。

最后，感谢所有填写了勘误请求的书稿读者：我咒骂过你们，也赞美过你们。当第一稿面世时，出乎我的意料，整本书在两天内就被读完，你们的评论把每一页都撕成了碎片。但是，这个过程促成了一本好书的诞生，它的质量远超我的想象。当许多人说“我看不懂这一页”时（这发生过好多次），我就被迫重新思考并改写相关内容。谢谢你们每个人的帮助。

第 2 版

首先，我的新编辑Susannah Pfalzer关于调整本书组织方式和重心的建议很有用。和你一起工作很棒，你教会了我很多。

Kenneth Lundin和OTP小组的成员们努力工作，实现了第2版里描述的那些新的语言特性。

第1版的许多读者反馈了他们所不理解的内容，我希望现在这些已经被修正了。

映射组（map）的设计灵感来源于Richard A. O'Keefe的工作（他把它们称为“框架”，即frame）。Richard多年来一直在Erlang邮件列表里捍卫框架的价值。谢谢你的评论和建议，Richard。

Kostis Sagonas在处理类型系统方面提供了许多有用的反馈。

我还想感谢Loïc Hoguin允许我们使用Nine Nines公司Cowboy Web服务器的一些代码作为例子，以及那些来自Basho、为BitLocker编写代码的人们。我还想感谢Dave Smith对rebar所做的工作。

许多人帮助我审读了不同批次的第2版草稿。我想感谢他们所有人，因为他们让这本书变得更好：Erik Abefelt、Paul Butcher、Mark Chu-Carroll、Ian Dees、Henning Diedrich、Jeremy Frens Loïc Hoguin、Andy Hunt、Kurt Landrus、Kenneth Lundin、Evan Miller、Patrik Nyblom、Tim Ottinger、Kim Shrier和Bruce Tate。

感谢Helen Taylor（Twitter @mrsjoeerl）为我提供无数杯热茶，在我几欲失去创作动力时给我鼓劲。

感谢古斯塔夫·马勒、拉赫曼尼诺夫、理查德·瓦格纳和乔治·弗里德里希·亨德尔（以及鲍勃·迪伦和其他一些人）创作了我编写本书大部分内容时播放的背景音乐。

目　录

第一部分　为何用 Erlang

第 1 章　什么是并发……2
1.1　给并发建模……2
1.1.1　开始模拟……3
1.1.2　发送消息……4
1.1.3　接收消息……4
1.2　并发的益处……4
1.3　并发程序和并行计算机……5
1.4　顺序和并发编程语言……6
1.5　小结……6

第 2 章　Erlang 速览……7
2.1　Shell……7
2.1.1　=操作符……8
2.1.2　变量和原子的语法……8
2.2　进程、模块和编译……9
2.2.1　在 shell 里编译并运行 Hello World……9
2.2.2　在 Erlang shell 外编译……9
2.3　你好，并发……10
2.3.1　文件服务器进程……10
2.3.2　客户端代码……13
2.3.3　改进文件服务器……14
2.4　练习……14

第二部分　顺序编程

第 3 章　基本概念……16
3.1　启动和停止 Erlang shell……16
3.1.1　在 shell 里执行命令……17
3.1.2　可能出错的地方……17
3.1.3　在 Erlang shell 里编辑命令……18
3.2　简单的整数运算……18
3.3　变量……19
3.3.1　Erlang 的变量不会变……20
3.3.2　变量绑定和模式匹配……20
3.3.3　为什么一次性赋值让程序变得更好……21
3.4　浮点数……22
3.5　原子……22
3.6　元组……23
3.6.1　创建元组……24
3.6.2　提取元组的值……25
3.7　列表……26
3.7.1　专用术语……26
3.7.2　定义列表……27
3.7.3　提取列表元素……27
3.8　字符串……27
3.9　模式匹配再探……29
3.10　练习……30

第 4 章　模块与函数……31
4.1　模块是存放代码的地方……31
4.1.1　常见错误……33
4.1.2　目录和代码路径……33
4.1.3　给代码添加测试……33
4.1.4　扩展程序……34
4.1.5　分号放哪里……36
4.2　继续购物……36
4.3　fun：基本的抽象单元……39

4.3.1 以 fun 作为参数的函数……40
4.3.2 返回 fun 的函数……41
4.3.3 定义你自己的控制抽象……42
4.4 简单列表处理……42
4.5 列表推导……45
4.5.1 Quicksort……46
4.5.2 毕达哥拉斯三元数组……47
4.5.3 回文构词……48
4.6 内置函数……48
4.7 关卡……49
4.7.1 关卡序列……49
4.7.2 关卡示例……50
4.7.3 true 关卡的作用……51
4.8 case 和 if 表达式……52
4.8.1 case 表达式……52
4.8.2 if 表达式……53
4.9 构建自然顺序的列表……54
4.10 归集器……55
4.11 练习……56

第 5 章 记录与映射组……57

5.1 何时使用映射组或记录……57
5.2 通过记录命名元组里的项……58
5.2.1 创建和更新记录……59
5.2.2 提取记录字段……59
5.2.3 在函数里模式匹配记录……59
5.2.4 记录是元组的另一种形式……60
5.3 映射组：关联式键–值存储……60
5.3.1 映射组语法……60
5.3.2 模式匹配映射组字段……62
5.3.3 操作映射组的内置函数……63
5.3.4 映射组排序……64
5.3.5 以 JSON 为桥梁……64
5.4 练习……66

第 6 章 顺序程序的错误处理……67

6.1 处理顺序代码里的错误……67
6.2 用 try...catch 捕捉异常错误……69
6.2.1 try...catch 具有一个值……69
6.2.2 简写法……70
6.2.3 try...catch 编程样例……71
6.3 用 catch 捕捉异常错误……72
6.4 针对异常错误的编程样式……72
6.4.1 改进错误消息……72
6.4.2 经常返回错误时的代码……73
6.4.3 错误可能有但罕见时的代码……73
6.4.4 捕捉一切可能的异常错误……74
6.5 栈跟踪……74
6.6 抛错要快而明显，也要文明……75
6.7 练习……75

第 7 章 二进制型与位语法……76

7.1 二进制型……76
7.2 位语法……78
7.2.1 打包和解包 16 位颜色……78
7.2.2 位语法表达式……79
7.2.3 位语法的真实例子……81
7.3 位串：处理位级数据……85
7.4 练习……87

第 8 章 Erlang 顺序编程补遗……88

8.1 apply……89
8.2 算术表达式……90
8.3 元数……91
8.4 属性……91
8.4.1 预定义的模块属性……91
8.4.2 用户定义的模块属性……93
8.5 块表达式……94
8.6 布尔值……94
8.7 布尔表达式……95
8.8 字符集……95
8.9 注释……95
8.10 动态代码载入……96
8.11 Erlang 的预处理器……99
8.12 转义序列……99
8.13 表达式和表达式序列……100
8.14 函数引用……101
8.15 包含文件……101
8.16 列表操作：++和--……102
8.17 宏……102

8.18 模式的匹配操作符……104
8.19 数字……105
8.19.1 整数……105
8.19.2 浮点数……105
8.20 操作符优先级……106
8.21 进程字典……106
8.22 引用……108
8.23 短路布尔表达式……108
8.24 比较数据类型……108
8.25 元组模块……109
8.26 下划线变量……109
8.27 练习……110
第 9 章 类型……111
9.1 指定数据和函数类型……111
9.2 Erlang 的类型表示法……113
9.2.1 类型的语法……113
9.2.2 预定义类型……114
9.2.3 指定函数的输入输出类型……114
9.2.4 导出类型和本地类型……116
9.2.5 不透明类型……116
9.3 dialyzer 教程……117
9.3.1 错误使用内置函数的返回值……118
9.3.2 内置函数的错误参数……119
9.3.3 错误的程序逻辑……119
9.3.4 使用 dialyzer……120
9.3.5 干扰 dialyzer 的事物……120
9.4 类型推断与成功分型……121
9.5 类型系统的局限性……123
9.6 练习……125
第 10 章 编译和运行程序……126
10.1 改变开发环境……126
10.1.1 设置载入代码的搜索路径……126
10.1.2 在系统启动时执行一组命令……127
10.2 运行程序的不同方式……128
10.2.1 在 Erlang shell 里编译和运行……128
10.2.2 在命令提示符界面里编译和运行……129
10.2.3 作为 Escript 运行……130
10.2.4 带命令行参数的程序……131
10.3 用 makefile 使编译自动化……132
10.4 当坏事发生……135
10.4.1 停止 Erlang……135
10.4.2 未定义（缺失）的代码……135
10.4.3 shell 没有反应……136
10.4.4 我的 makefile 不工作……137
10.4.5 Erlang 崩溃而你想阅读故障转储文件……137
10.5 获取帮助……138
10.6 调节运行环境……138
10.7 练习……139

第三部分　并发和分布式程序

第 11 章 现实世界中的并发……142
第 12 章 并发编程……145
12.1 基本并发函数……145
12.2 客户端–服务器介绍……147
12.3 进程很轻巧……151
12.4 带超时的接收……153
12.4.1 只带超时的接收……154
12.4.2 超时值为 0 的接收……154
12.4.3 超时值为无穷大的接收……155
12.4.4 实现一个定时器……155
12.5 选择性接收……156
12.6 注册进程……157
12.7 关于尾递归的说明……158
12.8 用 MFA 或 Fun 进行分裂……160
12.9 练习……160
第 13 章 并发程序中的错误……161
13.1 错误处理的理念……161
13.1.1 让其他进程修复错误……162
13.1.2 任其崩溃……162
13.1.3 为何要崩溃……162
13.2 错误处理的术语含义……163
13.3 创建连接……164
13.4 同步终止的进程组……164

13.5 设立防火墙……165
13.6 监视……166
13.7 基本错误处理函数……166
13.8 容错式编程……167
13.8.1 在进程终止时执行操作……167
13.8.2 让一组进程共同终止……168
13.8.3 生成一个永不终止的进程……169
13.9 练习……170

第 14 章 分布式编程……171

14.1 两种分布式模型……171
14.2 编写一个分布式程序……172
14.3 创建名称服务器……173
14.3.1 第 1 阶段：一个简单的名称服务器……173
14.3.2 第 2 阶段：客户端在一个节点，服务器在相同主机的另一个节点……174
14.3.3 第 3 阶段：同一局域网内不同机器上的客户端和服务器……175
14.3.4 第 4 阶段：跨互联网不同主机上的客户端和服务器……176
14.4 分布式编程的库和内置函数……177
14.4.1 远程分裂示例……178
14.4.2 文件服务器再探……180
14.5 cookie 保护系统……181
14.6 基于套接字的分布式模型……182
14.6.1 用 lib_chan 控制进程……182
14.6.2 服务器代码……183
14.7 练习……185

第四部分 编程库与框架

第 15 章 接口技术……188

15.1 Erlang 如何与外部程序通信……188
15.2 用端口建立外部 C 程序接口……190
15.2.1 C 程序……191
15.2.2 Erlang 程序……193
15.2.3 编译和链接端口程序……195
15.2.4 运行程序……195
15.3 在 Erlang 里调用 shell 脚本……196
15.4 高级接口技术……196
15.5 练习……197

第 16 章 文件编程……198

16.1 操作文件的模块……198
16.2 读取文件的几种方法……199
16.2.1 读取文件里的所有数据类型……199
16.2.2 分次读取文件里的数据类型……200
16.2.3 分次读取文件里的行……202
16.2.4 读取整个文件到二进制型中……202
16.2.5 通过随机访问读取文件……203
16.3 写入文件的各种方式……205
16.3.1 把数据列表写入文件……206
16.3.2 把各行写入文件……207
16.3.3 一次性写入整个文件……207
16.3.4 写入随机访问文件……209
16.4 目录和文件操作……209
16.4.1 查找文件信息……210
16.4.2 复制和删除文件……211
16.5 其他信息……211
16.6 一个查找工具函数……212
16.7 练习……214

第 17 章 套接字编程……216

17.1 使用 TCP……216
17.1.1 从服务器获取数据……216
17.1.2 一个简单的 TCP 服务器……219
17.1.3 顺序和并行服务器……222
17.1.4 注意事项……223
17.2 主动和被动套接字……224
17.2.1 主动消息接收（非阻塞式）……224
17.2.2 被动消息接收（阻塞式）……225
17.2.3 混合消息接收（部分阻塞式）……225

17.3 套接字错误处理 ……226
17.4 UDP ……227
17.4.1 最简单的 UDP 服务器与客户端 ……227
17.4.2 一个 UDP 阶乘服务器 ……228
17.4.3 UDP 数据包须知 ……230
17.5 对多台机器广播 ……230
17.6 一个 SHOUTcast 服务器 ……231
17.6.1 SHOUTcast 协议 ……232
17.6.2 SHOUTcast 服务器的工作原理 ……232
17.6.3 SHOUTcast 服务器的伪代码 ……233
17.6.4 运行 SHOUTcast 服务器 ……234
17.7 练习 ……235

第 18 章 用 WebSocket 和 Erlang 进行浏览 ……236

18.1 创建一个数字时钟 ……237
18.2 基本交互 ……239
18.3 浏览器里的 Erlang shell ……240
18.4 创建一个聊天小部件 ……241
18.5 简化版 IRC ……244
18.6 浏览器里的图形 ……247
18.7 浏览器–服务器协议 ……249
18.7.1 从 Erlang 发送消息到浏览器 ……249
18.7.2 从浏览器到 Erlang 的消息 ……250
18.8 练习 ……251

第 19 章 用 ETS 和 DETS 存储数据 ……252

19.1 表的类型 ……252
19.2 影响 ETS 表效率的因素 ……254
19.3 创建一个 ETS 表 ……255
19.4 ETS 示例程序 ……255
19.4.1 三字母组合迭代函数 ……256
19.4.2 创建一些表 ……257
19.4.3 创建表所需的时间 ……258
19.4.4 访问表所需的时间 ……258
19.4.5 获胜者是…… ……259
19.5 保存元组到磁盘 ……260
19.6 其余操作 ……262
19.7 练习 ……263

第 20 章 Mnesia：Erlang 数据库 ……264

20.1 创建初始数据库 ……264
20.2 数据库查询 ……265
20.2.1 选择表里的所有数据 ……266
20.2.2 从表里选择数据 ……267
20.2.3 从表里有条件选择数据 ……268
20.2.4 从两个表里选择数据（联接） ……268
20.3 添加和移除数据库里的数据 ……269
20.3.1 添加行 ……269
20.3.2 移除行 ……270
20.4 Mnesia 事务 ……270
20.4.1 中止事务 ……271
20.4.2 载入测试数据 ……273
20.4.3 do()函数 ……273
20.5 在表里保存复杂数据 ……274
20.6 表的类型和位置 ……275
20.6.1 创建表 ……276
20.6.2 常用的表属性组合 ……277
20.6.3 表的行为 ……278
20.7 表查看器 ……278
20.8 深入挖掘 ……279
20.9 练习 ……279

第 21 章 性能分析、调试与跟踪 ……280

21.1 Erlang 代码的性能分析工具 ……281
21.2 测试代码覆盖 ……281
21.3 生成交叉引用 ……283
21.4 编译器诊断信息 ……283
21.4.1 头部不匹配 ……284
21.4.2 未绑定变量 ……284
21.4.3 未结束字符串 ……284
21.4.4 不安全变量 ……284
21.4.5 影子变量 ……285
21.5 运行时诊断 ……286
21.6 调试方法 ……287