O'REILLY®

Real World OCaml（中文版）

真实世界的OCaml

Yaron Minsky, Anil Madhavapeddy,
Jason Hickey 著
苏金国 彭小姣 译

中国电力出版社

Real World OCaml（中文版）

Yaron Minsky, Anil Madhavapeddy, Jason Hickey 著

苏金国 彭小姣 译

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo O'REILLY®

O'Reilly Media, Inc.授权中国电力出版社出版

中国电力出版社

图书在版编目（CIP）数据

真实世界的OCaml/（美）闵斯基（Minsky, Y.），（美）麦哈瓦佩迪（Madhavapeddy, A.），（美）希基（Hickey, J.）著；苏金国，彭小姣译.
—北京：中国电力出版社，2015.8

书名原文：Real World OCaml

ISBN 978-7-5123-7637-3

I. ①真… II. ①闵… ②麦… ③希… ④苏… ⑤彭… III. ①程序语言－程序设计 IV. ①TP312

中国版本图书馆CIP数据核字（2015）第086168号

北京市版权局著作权合同登记

图字：01-2015-1875号

Copyright © 2014 Yaron Minsky, Anil Madhavapeddy, Jason Hickey.

Simplified Chinese Edition, jointly published by O'Reilly Media, Inc. and China Electric Power Press, 2015. Authorized translation of the English edition, © 2014 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由O'Reilly Media, Inc.出版2014

简体中文版由中国电力出版社出版2015。英文原版的翻译得到O'Reilly Media, Inc.的授权。此简体中文版的出版和销售得到出版权和销售权的所有者——O'Reilly Media, Inc.的许可。

版权所有，未得书面许可，本书的任何部分和全部不得以任何形式重制。

安徽大学图书馆 藏书 www.lib.ahu.edu.cn

封面设计/ Randy Comer，张健

出版发行/ 中国电力出版社（http://www.cepp.sgcc.com.cn）

地　　址/ 北京市东城区北京站西街19号（邮政编码100005）

经　　销/ 全国新华书店

印　　刷/ 北京丰源印刷厂印刷

开　　本/ 787毫米×980毫米　16开本　29.25印张　561千字

版　　次/ 2015年8月第一版　2015年8月第一次印刷

印　　数/ 0001－3000册

定　　价/ 78.00元（册）

敬 告 读 者

本书封底贴有防伪标签，刮开涂层可查询真伪

本书如有印装质量问题，我社发行部负责退换

版 权 专 有　翻 印 必 究

O'Reilly Media, Inc.介绍

O'Reilly Media通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自1978年开始，O'Reilly一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了Make杂志，从而成为DIY革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版，在线服务或者面授课程，每一项O'Reilly的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar博客有口皆碑。”

——Wired

“O'Reilly凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——Business 2.0

“O'Reilly Conference是聚集关键思想领袖的绝对典范。”

——CRN

“一本O'Reilly的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim是位特立独行的商人，他不光放眼于最长远、最广阔的视野并且切实地按照Yogi Berra的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去Tim似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——Linux Journal

献给Lisa，是你坚定地相信语言的力量，帮助我找到了真正的自己。——Yaron

献给妈妈和爸爸，感谢你们带我去图书馆，让我张开想象的翅膀。——Anil

献给Nobu，每天都带我踏上新的旅程。——Jason

此为试读，需要完整PDF请访问：www.ertongbook.com

目录

前言 1

第一部分 语言概念

第1章 导览旅行 11

1.1 OCaml作为计算器 11
1.2 函数和类型推断 13
1.3 元组、列表、选项和模式匹配 17
1.4 记录和变体 25
1.5 命令式编程 27
1.6 一个完整的程序 32
1.7 下一章的内容 33

第2章 变量和函数 34

2.1 变量 34
2.2 函数 38

第3章 列表和模式 55

3.1 列表基础 55
3.2 使用模式从列表抽取数据 56
3.3 模式匹配的局限性（和好处） 58
3.4 有效地使用List模块 61
3.5 尾递归 67

3.6 更简洁更快速的模式 69

第4章 文件、模块和程序 73

4.1 单文件程序 73
4.2 多文件程序和模块 76
4.3 签名和抽象类型 77
4.4 签名中的具体类型 80
4.5 嵌套模块 81
4.6 打开模块 82
4.7 包含模块 84
4.8 模块的常见错误 86
4.9 基于模块的设计 88

第5章 记录 91

5.1 模式和完备性 93
5.2 字段双关 95
5.3 重用字段名 96
5.4 功能更新 99
5.5 可变字段 101
5.6 首类字段 102

第6章 变体 106

6.1 Catch-All情况和重构 108
6.2 结合记录和变体 110
6.3 变体和递归数据结构 114
6.4 多态变体 117

第7章 错误处理 125

7.1 错误感知返回类型 125
7.2 异常 130
7.3 选择错误处理策略 139

第8章 命令式编程 141
8.1 示例：命令式字典 141
8.2 基本可变数据 145
8.3 for和while循环 148
8.4 示例：双向链表 149
8.5 懒惰和其他良性影响 153
8.6 输入和输出 160
8.7 计算顺序 166
8.8 副作用和弱多态 168
8.9 小结 173

第9章 仿函数 175
9.1 一个简单例子 176
9.2 一个更大的例子：间隔计算 177
9.3 扩展模块 189

第10章 首类模块 193
10.1 使用首类模块 193
10.2 示例：队列处理框架 199
10.3 如果没有首类模块 208

第11章 对象 210
11.1 OCaml对象 211
11.2 对象多态 212
11.3 不可变对象 214
11.4 如何使用对象 215
11.5 子类型化 216

第12章 类 225
12.1 OCaml类 225
12.2 类参数和多态 226
12.3 对象类型作为接口 228
12.4 继承 231

12.5 类类型 232
12.6 开放递归 233
12.7 私有方法 235
12.8 二值化方法 236
12.9 虚类和方法 239
12.10 初始化方法 242
12.11 多重继承 243

第二部分 工具和技术

第13章 映射和散列表 251

13.1 映射 252
13.2 散列表 261
13.3 映射和散列表之间的选择 264

第14章 命令行解析 268

14.1 基本命令行解析 269
14.2 参数类型 272
14.3 为命令行增加标签标志 277
14.4 组合子命令 278
14.5 对解析的高级控制 281
14.6 使用bash实现命令行自动完成 286
14.7 其他命令行解析器 288

第15章 处理JSON数据 289

15.1 JSON基础 289
15.2 用Yojson解析JSON 290
15.3 从JSON结构选择值 292
15.4 构造JSON值 296
15.5 使用非标准JSON扩展 298
15.6 JSON自动映射到OCaml类型 299

第16章 用OCamllex和Menhir完成解析 306
16.1 词法分析和解析 307
16.2 定义解析器 309
16.3 定义词法分析器 312
16.4 集成 316

第17章 利用S-表达式实现数据串行化 319
17.1 基本用法 320
17.2 Sexp格式 323
17.3 保持不变式 324
17.4 得到合适的错误消息 327
17.5 S-表达式转换指令 329

第18章 利用Async实现并发编程 334
18.1 Async基础 335
18.2 示例：回显服务器 340
18.3 示例：用DuckDuckGo搜索定义 345
18.4 异常处理 349
18.5 超时、撤销和选择 356
18.6 处理系统线程 358

第三部分 运行时系统

第19章 外部函数接口 365
19.1 示例：终端界面 366
19.2 基本标量C类型 370
19.3 指针和数组 371
19.4 结构和联合 374
19.5 向C传递函数 380
19.6 关于C绑定 383

第20章 值的内存表示.....385
20.1 OCaml块和值.....386
20.2 块和值.....387
20.3 元组、记录和数组.....389
20.4 变体和列表.....390
20.5 多态变体.....392
20.6 字符串值.....392
20.7 定制堆块.....393
第21章 了解垃圾回收器.....395
21.1 标记和清扫垃圾回收.....395
21.2 世代垃圾回收.....396
21.3 快速次堆.....396
21.4 长生存期的主堆.....398
21.5 为值关联最终化函数.....405
第22章 编译器前端：解析和类型检查.....407
22.1 工具链概览.....407
22.2 解析源代码.....410
22.3 预处理源代码.....414
22.4 静态类型检查.....419
22.5 类型化语法树.....431
第23章 编译器后端：字节码和原生代码.....435
23.1 无类型Lambda形式.....435
23.2 生成可移植的字节码.....439
23.3 编译快速原生代码.....444
23.4 文件扩展名小结.....453

此为试读，需要完整PDF请访问：www.ertongbook.com

前言

为什么选择OCaml？

编程语言确实很重要，会影响你编写的代码的可靠性、安全性和效率，还决定了阅读、重构和扩展这些代码的难易。你了解的语言可能还会改变你的思维方式，即使没有具体使用这些语言，它们也会对你如何设计软件产生影响。

之所以写这本书，这是因为我们对编程语言的重要性深信不疑，特别是相信OCaml是一个值得学习的重要语言。在我们三个人的学术和职业生涯中，使用OCaml都已经超过了15年，在这些年间，我们见证了它逐步成为构建复杂软件系统的一个秘密武器。这本书就是要让更多的读者认识和使用这个秘密武器。我们会给出一个简明的指南，如果你想在真实世界里有效地使用OCaml，可以从这本书中了解你需要知道的知识。

OCaml与众不同的是，它拥有编程语言设计领域中最有利的一点：集效率、有效性和实用性于一身，这是任何其他语言无法比拟的。很大程度上，这是因为OCaml是40多年来开发的很多关键语言特性的完美组合。包括：

- 垃圾回收（Garbage collection），用于实现自动内存管理，现在这已经成为几乎所有现代高级语言的一个特性。
- 首类函数（First-class functions），首类函数可以像常规的值一样传递，JavaScript、Common Lisp和C#中都提供了首类函数。
- 静态类型检查（Static type-checking），可以提高性能，并减少运行时错误，Java和C#都有这个特性。

- 参数化多态（Parametric polymorphism），允许构造抽象涵盖不同数据类型，这与Java和C#中的泛型和C++中的模板很类似。
- 充分支持不变性编程（immutable programming），也就是说，编程不会对数据结构带来破坏性的更新。这个特性常见于传统函数式语言（如Scheme），另外一些分布式大数据框架（如Hadoop）也提供了这个特性。
- 自动类型推断（Automatic type inference），不必在程序中辛苦地定义每一个变量的类型，可以根据所用的值来推断变量的类型。C#允许使用有隐式类型的局部变量，这就是自动类型推断的一种限定形式，另外C++中也利用auto关键字支持自动类型推断。
- 代数数据类型（Algebraic data types）和模式匹配（pattern matching），可以用来定义和管理复杂数据结构。Scala和F#对这个特性提供了支持。

有些人可能知道并且喜欢所有这些特性，但对另外一些人来说，这些特性都是头一次听说，完全是陌生的，不过大多数人之前可能用过其他一些语言，其中可能也有类似的特性。从这本书可以了解到，通过一定的变形可以将所有这些特性集成在一种语言中。尽管这些特性都是外来的，但这些思想与主流语言的交叉很有限，在主流语言中体现很少。即使主流语言确实有其中的一些特性，比如C#中的首类函数或Java中的参数化多态，但往往很受限，而且通常很难用。能够完全涵盖所有这些思想的语言只有使用静态类型的函数式编程语言，如OCaml、F#、Haskell、Scala和Standard ML。

在这一大堆语言中，OCaml可谓独树一帜，因为它既具备强大的能力，还保持着高度的实用性。编译器提供了一种简单直接的编译策略，可以生成高性能的代码而不需要繁杂的优化，也没有动态、即时（just-in-time，JIT）编译的复杂性。基于这一点，再加上OCaml严格的评价模型，使得运行时行为不再难以预测。垃圾回收器是增量式的（incremental），这样可以避免大规模垃圾回收（garbage collection，GC）可能带来的短暂停顿，而且非常精准（precise），这意味着它会回收所有不再引用的数据（这与很多引用计数型垃圾回收器不同），另外运行时库很简单，而且有高度的可移植性。

如果程序员想要转向一个更好的编程语言，同时还希望能很好地完成实际工作，对这些程序员来说，上述优点使得OCaml成为一个非常棒的选择。

OCaml简史

OCaml是由法国国家信息与自动化研究所（INRIA）的Xavier Leroy、Jerome Vouillon、Damien Doligez和Didier Remy在1996年编写的。他们对从20世纪60年代开始出现的ML做了长期研究，并受到启发，之后OCaml得以问世，并在学术界继续得到了大量支持。

ML原先是作为LCF（可计算函数逻辑，Logic for Computable Functions)的元语言设计的，LCF是Robin Milner（当时在斯坦福，后来在剑桥）于1972年发布的一个辅助验证工具。后来ML变成一个编译器，以便在不同的机器上使用LCF，到了20世纪80年代，ML本身已经逐步成为一个完备的系统。

Caml的第一个实现于1987年问世。它由Ascander Suarez创建，后来得到Pierre Weis和Michel Mauny的进一步开发。1990年，Xavier Leroy和Damien Doligez构建了一个新实现，取名为Caml Light，它基于一个字节码解释器，并且提供了一个快速的顺序垃圾回收器。接下来的几年中，不断有一些有用的库出现，如Michel Mauny的语法管理工具，这大大促进了Caml在教育和研究领域中的应用。

Xavier Leroy继续为Caml Light扩展了新的特性，终于在1995年发布了Caml Special Light。这个实现增加了一个快速原生代码编译器，大大提高了可执行程序的效率，使Caml Special Light的性能可与C++等主流语言媲美。受Standard ML的启发，Caml Special Light还实现了一个模块系统，提供了强大的抽象工具，从而可以更容易地构造大规模程序。

现代OCaml于1996年诞生，Didier Remy和Jerome Vouillon实现了一个强大而精巧的对象系统。这个对象系统以一种静态的、类型安全的方式支持多种常见的面向对象特性，并因此而闻名，在类似C++或Java等语言中，同样的这些面向对象特性都需要完成运行时检查。2000年，Jacques Garrigue进一步扩展了OCaml，增加了很多新特性，如多态方法、变体，以及标签参数和可选参数。

近十年来，OCaml逐步吸引了大量的用户，并不断加入新的语言改进，以支持越来越庞大的商业和学术代码基。首类模块、通用代数数据类型（Generalized Algebraic Data Types，GADTs）和动态链接大大提升了语言的灵活性。另外还为x86_64、ARM、PowerPC和Sparc提供了快速原生代码支持，对于重视资源使用、可预测性和性能的系统来说，这些特性使得OCaml成为一个理想的选择。

Core标准库

只有语言本身还不够。你还需要一组丰富的库作为应用的基础。学习OCaml时，大多数困惑都源于编译器附带的标准库是受限的，只涵盖了你期望的通用标准库功能中很小的一个子集。这是因为，标准库并不是一个通用工具。开发标准库只是为了在编译器自引导时使用，所以有意做得很小、很简单。

还好，在开源软件的世界里，已经无法阻挡人们编写替代库的脚步，这些替代库将作为编译器附带标准库的补充，这正是Core标准库所要做的。

Jane Street是一家使用OCaml十余年的公司，它们开发了Core供内部使用，不过从一开始，Core的设计就着眼于要成为一个通用的标准库。类似OCaml语言本身，Core的实现充分考虑了正确性、可靠性和性能。

Core包括一些语法扩展，可以为OCaml提供有用的新功能，另外还有一些库（如Async网络通信库）进一步扩展了Core的“疆域”，允许构建复杂的分布式系统。所有这些库都是按照Apache 2自由软件许可证发布的，允许个人出于爱好免费使用，也允许在学术和商业领域免费使用。

OCaml平台

Core是一个丰富而有效的标准库，不过除此以外，还有一个与OCaml关联更紧密的软件。很多程序员从1996年首次发布OCaml以来就一直在使用这个语言，已经生成了很多有用的库和工具。在本书给出的例子中，我们将介绍其中一些库和工具。

利用一个名为OPAM（http://opam.ocaml.org/）的包管理工具，这些第三方库的安装和管理会容易得多。随着本书内容的展开，我们会更深入地解释OPAM，不过现在起码要知道它构成了OCaml平台的基础，这是一组工具和库，结合OCaml编译器使用时，可以让你更快、更有效地构建实际应用。

我们还会使用OPAM来安装utop命令行界面。这是一个现代交互式工具，支持命令历史、宏扩展、模块完成以及其他一些优秀的特性，利用这个工具可以更方便地使用OCaml语言。我们将在整本书中使用utop，让你能采用交互方式完成每一个示例。

关于本书

本书面向那些对传统编程语言有一定经验但对静态类型函数式编程了解不多的程序员。取决于你的背景，我们介绍的一些概念对你来说可能是全新的，包括传统的函数式编程技术，如高阶函数和不可变的数据类型，以及OCaml强大的类型和模块系统的很多方面。

如果你对OCaml已经有所了解，这本书可能会给你一个惊喜。Core重新定义了标准命名空间的大部分内容，以便更好地使用OCaml模块系统，而且默认提供了大量功能强大的可重用数据结构。尽管原先的OCaml代码仍能与Core互操作，不过为了充分发挥OCaml的能力，还是应该转而使用Core。我们编写的所有新代码都使用了Core，而且我们相信Core模型确实值得学习。它在很多数百万行的大规模代码基中得到了成功使用，用OCaml构建复杂的应用时，它将帮助你扫除很大的障碍。

只使用传统编译器标准库的代码还会存在，不过可以参考另外一些在线资源来学习这些

代码如何工作。本书强调的是本书作者们在构造健壮的可扩展软件系统时个人亲身使用并推崇的一些技术。

本书内容

本书分为3大部分：

- 第1部分介绍语言本身，首先是一个导览旅行，提供了OCaml语言的一个概要全貌。不要指望能完全理解这个旅行中的方方面面。这只是为了让你对语言的多个不同方面有所认识，相关的内容将在后面的各章中进一步介绍。

  介绍了核心语言之后，第1部分将转向更高级的一些特性，如模块、仿函数和对象，消化这些内容需要花一定的时间。不过，理解这些概念非常重要。即使你不使用OCaml而要使用其他现代语言，这些概念对你也很有帮助，其中很多概念都受到了ML的启发。

- 第2部分建立在这些基础知识之上，将利用一些有用的工具和技术来建立实际应用，从命令行解析到异步网络编程都有涵盖。在这个过程中，你会看到如何将第1部分中的概念集成在一起，构成实际的库和工具，从而结合这个语言的不同特性达到最好的效果。

- 第3部分讨论了OCaml的运行时系统和编译器工具链。与其他一些语言实现相比（如Java或.NET的CLR），OCaml的运行时系统相当简单。学习这一部分将帮助你构建非常高效的系统，或者可以与C库交互。我们还将在这一部分讨论使用诸如GNU *gdb*等工具实现的性能测试和调试技术。

安装说明

本书中使用了我们在写这本书的过程中开发的一些工具。其中一些工具对OCaml编译器有所改进，这意味着你要确保有最新的开发环境（使用编译器的4.01版本）。通过使用OPAM包管理器，安装过程很大程度上都是自动化的。关于如何安装以及要安装哪些包，有关的说明见Real World OCaml安装页面（*http://realworldocaml.org/install*）。

本书出版时，Core还不支持Windows操作系统，所以仅Mac OS X、Linux、FreeBSD和OpenBSD能保证Core的正常使用。请检查在线安装说明，了解关于Windows的更新情况，或者可以安装一个Linux虚拟机来学习本书。

本书并不作为一个参考手册。我们的目的是教你学习这种语言，了解库工具和技术，这些将帮助你成为一名非常出色的OCaml程序员。不过，这本书并不能取代API文档或

OCaml手册和手册页，可以在*http://realworldocaml.org/doc*找到本书引用的所有库和工具的文档。

代码示例

本书中的所有代码示例都可以按照一个类公共域许可证在线免费获取。如果需要，完全可以复制你认为合适的任何代码段并在你自己的代码中使用，对于这些代码段的使用没有任何要求和限制。

代码库的地址是：*https://github.com/realworldocaml/examples*。本书中的每个代码段都有一个标题（电子版中这些标题是可单击的），指出代码库中该代码的文件名，以便找到这个代码段相关的源代码、shell脚本或辅助数据文件。

如果你认为使用这些代码示例时超出了合理使用范围，或者不在以上所述的权限许可之内，可以与我们联系：*permissions@oreilly.com*。

Safari®图书在线

Safari®图书在线（*www.safaribooksonline.com*）是一个应需而变的数字图书馆，通过图书和视频方式提供世界顶尖作者在技术和商业领域积累的专家经验（*http://www.safaribooksonline.com/content*）。技术专家、软件开发人员、Web设计人员和企业以及有创意的专业人员都使用Safari图书在线作为其主要资源来完成研究、解决问题、深入学习和资质培训。

Safari图书在线为机构（*http://www.safaribooksonline.com/organizations-teams*）、政府部门（*http://www.safaribooksonline.com/government*）和个人（*http://www.safaribooksonline.com/individuals*）提供了多种产品组合（*http://www.safaribooksonline.com/subscriptions*）和定价程序。订阅者可以在一个可以快捷搜索的数据库中访问多家出版社提供的成千上万种图书、培训视频和正式出版前手稿，如O'Reilly Media、Prentice Hall Professional、Addison-Wesley Professional、Microsoft Press、Sams、Que、Peachpit Press、Focal Press、Cisco Press、John Wiley & Sons、Syngress、Morgan Kaufmann、IBM Redbooks、Packt、Adobe Press、FT Press、Apress、Manning、New Riders、McGraw-Hill、Jones & Bartlett、Course Technology以及其他数十家出版公司（*http://www.safaribooksonline.com/publishers*）。关于Safari图书在线的更多信息，请访问我们的在线网站（*http://www.safaribooksonline.com/*）。

此为试读，需要完整PDF请访问：www.ertongbook.com