

OpenResty开源项目创始人力荐

Broadview[®]
www.broadview.com.cn

Nginx

罗剑锋 著

完全开发指南

使用C、C++和OpenResty



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Nginx

完全开发指南

使用C、C++和OpenResty

罗剑锋 著

电子工业出版社

Publishing House of Electronics Industry

北京•BEIJING

内 容 简 介

Nginx 是著名的 Web 服务器，性能优异，运行效率远超传统的 Apache、Tomcat，广泛应用于国内外诸多顶级互联网公司。

Nginx 的一个突出特点是其灵活优秀的模块化架构，可以在不修改核心的前提下增加任意功能，自 2004 年发布至今，已经拥有百余个官方及非官方的功能模块（如 proxy、mysql、redis、rtmp、lua 等），使得 Nginx 成长为了一个近乎“全能”的服务器软件。

Nginx 功能强大，架构复杂，学习、维护和开发的门槛较高。为了帮助读者跨越这一障碍，本书深入最新的 Nginx 源码（Stable 1.12.0），详细剖析了模块体系、动态插件、功能框架、进程模型、事件驱动、线程池、TCP/UDP/HTTP 处理等 Nginx 核心运行机制，在此基础上讲解如何使用 C、C++、Lua、nginScript 等语言来增强扩展 Nginx，让任何人都能够便捷、轻松地开发和定制 Nginx，进而应用到自己的实际工作中，创造出更多的价值。

本书结构严谨、脉络清晰、论述精确、详略得当、图文并茂，值得广大软件开发工程师、系统运维工程师和编程爱好者拥有。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

图书在版编目（CIP）数据

Nginx 完全开发指南：使用 C、C++和 OpenResty / 罗剑锋著. —北京：电子工业出版社，2017.6
ISBN 978-7-121-31457-5

I. ①N… II. ①罗… III. ①互联网络—网络服务器—程序设计—指南②C 语言—程序设计—指南
IV. ①TP368.5-62②TP312.8-62

中国版本图书馆 CIP 数据核字(2017)第 094580 号

策划编辑：孙学瑛

责任编辑：安 娜

印 刷：北京京科印刷有限公司

装 订：北京京科印刷有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编：100036

开 本：787×980 1/16 印张：39.75 字数：901 千字

版 次：2017 年 6 月第 1 版

印 次：2017 年 6 月第 1 次印刷

定 价：99.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888，88258888。

质量投诉请发邮件至 zltts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

前言

缘起

最早接触 Nginx 大概是在 2011 年，面对着一个全新的 Web 服务器，和大多数人一样最初我也是一片茫然，能找到的参考资料十分有限，安装、配置、运行几乎都是“摸着石头过河”，犯过许多低级错误。

随着对 Nginx 逐渐熟悉，它的高并发处理能力给我留下了深刻的印象，作为一个开源软件的爱好者，很自然地想要探究一下它的内部工作原理。我由此开始了对 Nginx 源码的钻研之路，中间经过了很多的艰辛曲折，走过不少的弯路。

我最常用的工作语言是 C++，所以在阅读 Nginx 源码时也总以 C++ 的面向对象方式来思考和理解，以对象作为切入点记笔记、画 UML：从最简单的 `ngx_str_t`、`ngx_array_t` 入手，然后到 `ngx_request_t`、`ngx_upstream_t` 等复杂的结构，再围绕着这些对象研究相关的功能函数和处理流程，梳理代码逻辑的同时也摸索着使用 C++ 编写 Nginx 模块的方法，逐渐积累了一些用起来颇为顺手的小工具——当然还是比较初级的形式。

三年多前，我被调到了新的工作岗位，需要重度使用 Nginx 开发，这让我以前的零散积累终于有了用武之地。那段时间里使用 C/C++ 陆续做了很多东西，也借着机会重新优化了原有的工具代码。

繁忙的工作之余，我有了种进一步整理经验的迫切感，因为只有系统完整地分享这些知识，才能让更多的人基于 Nginx 二次开发，让 Nginx 更好地为网络世界服务。

同一时间，市面上也出现了一些 Nginx 开发相关的资料、书籍，但在我看来却有“粗制滥造”之嫌：行文混乱，“车轱辘话”“口头禅”满天飞，甚至大段照抄指令说明，还有对源码

的曲解，未免有点儿“误人子弟”，读起来实在是难受。终于，在“忍无可忍”的心态之下，我动起了写作本书的念头。

经过近一年的努力，现在这本书终于呈现在了读者面前，结构上基本反映了我学习研究 Nginx 时的心路历程，从最初的“一无所知”起步，逐渐深入到定制开发的层次，希望能与读者“心有戚戚焉”。

Nginx 随感

毫无疑问，Nginx 是目前这个地球上所能获得的最强劲的 Web 服务器（没有之一），同时也是目前最成熟、最优秀的 TCP/HTTP 服务器开发框架。

Nginx 资源消耗低，并发处理性能高，配置灵活，能够连接 CGI、PHP、MySQL、Memcached 等多种后端，还有着出色的负载均衡能力，可以整合封装各种 service，构建稳定高效的服务。如今 Nginx 已经成为了网站架构里不可或缺的关键组件，广泛应用于国内外许多大型 IT 企业。每一个繁忙的网站背后，可能都有 Nginx 默默工作的身影。

在 Nginx 出现之前，使用 C/C++ 开发 Web 服务器是项比较“痛苦”的工作，虽然有很多网络程序库可以使用（例如 asio、libevent、thrift 等），但它们通常只关注较底层的基础功能实现，离成熟的“框架”相距甚远，不仅开发过程烦琐低效，而且程序员还必须要处理配置管理、进程间通信、协议解析等许多 Web 服务之外的其他事情，才能开发出一个较为完善的服务器程序。但即使开发出了这样的服务器，通常性能上也很难得到保证，会受到程序库和开发者水平等因素的限制——很长一段时间里，C/C++ 在 Web 服务器领域都没有大展拳脚的机会。

Nginx 的横空出世为 Web 服务器开辟了一个崭新的天地，它搭建了一个高性能的服务器开发框架，而且是一个完整的、全功能的服务器。模块化的架构设计很好地分离了底层支撑模块和上层逻辑模块，底层模块处理了配置、并发等服务器的外围功能，核心支撑模块定义了主体的 TCP/HTTP 处理框架。开发者可以把大部分精力集中在上层的业务功能实现上，再也不必去为其他杂事而分心，提高了软件的开发效率。

在 Nginx 框架里，C/C++ 程序员可以尽情发挥自己的专长，充分利用 Nginx 无阻塞处理的优势，打造出高质量的 Web 应用服务器，与其他系统一较高下。

Nginx 和 C/C++

Igor Sysoev 选择用 C 语言（准确地说是 ANSI C）来实现 Nginx 肯定是经过了认真

的考虑。

作为与 UNIX 一同诞生的编程语言，C 语言一直是系统级编程的首选。和其他高级语言相比，它简单可靠，更接近计算机底层硬件，运行效率更高。指针更是 C 语言的一大特色，善用指针能够完成许多其他语言无法完成的工作。

以 C 语言实现的 Nginx 没有“虚拟机”的成本，省略了不必要的中间环节，直接操纵计算机硬件，从根本上提高了 Web 服务器的处理能力。虽然 C 语言不直接支持面向对象，但 Nginx 灵活运用了指针，采用结构体+函数指针的形式，达到了同样的效果，从而使软件拥有了良好的结构。

C++是仅次于 C 的系统级编程语言，在兼容 C 的同时又增加了类、异常、模板等新特性，还支持面向对象、泛型、函数式、模板元等多种编程范式，可以说是计算机语言里的一个“庞然大物”。C++的特性很多，有的也很好用，但总体上的确比较复杂，易学难精，容易被误用和滥用，导致低效、难维护的代码，我想这可能是 Igor Sysoev 放弃使用 C++的一个重要原因。

另一个可能的原因是 C 语言本身已经非常稳定，几十年来没有太大的变动，在各个系统里都支持得非常好。而 C++在 1998 年才有了第一个标准，并且现在还在发展之中，语言特性还不够稳定（例如 `export`、`register` 等曾经的关键字在 C++11 里就已经被废弃），许多编译器对 C++的支持程度也有差异，这与 Nginx 的高可移植性目标明显不符。

但 C++毕竟还是有很多的优点，类可以更好地封装信息、异常简化了错误处理、模板能够在编译期执行类型计算。在 C++11 标准颁布之后，C++更是几乎变成了一门“全新”的语言，`auto`/`decltype`/`nullptr`/`noexcept` 等新关键字增强了语言的描述能力，标准库也扩充了相当多的组件，易用性和稳定性都大大提升。

在 Nginx 里使用 C++时要对 C++的长处和不足有清醒的认识，避免多层次继承、虚函数等影响效率的编程范式，只使用经过充分验证的、能够切实提高开发效率和性能的语言特性和库，避免华而不实的技术炫耀，尽量做到像 Nginx 源码那样质朴踏实。只有这样，才能够发挥出 $1+1>2$ 的作用，让 Nginx 从 C++中得到更进一步的发展动力。

Nginx 和 OpenResty

多年以前 Nginx 开发使用的语言只能是 C 和 C++，而现在，越来越多的开发者逐渐转向了 OpenResty，使 Lua 搭建高并发、高性能、高扩展性的 Web Server。

我接触 OpenResty 的时间并不算很长，大约在四年左右。由于 C/C++程序员“天生的傲

慢”，一开始对 OpenResty 确实有点儿“抵触情绪”，总觉得脚本程序比不上 C/C++ 实现。然而随着使用的增多，特别是在研究了它的源码之后，我不得不感慨 OpenResty 的精致、完美和强大，简直是所有 Nginx 开发者“梦寐以求的至宝”。

由于 agentzh 对 Nginx 的运行机制了如指掌，OpenResty 的核心部分——ngx_lua 一个模块就涵盖了 access/rewrite/content/log 等多个处理阶段，再搭配上小巧灵活的 Lua 和高效的 LuaJIT，我们就能够在更高级的业务层次上使用“胶水”代码来调用组合 Nginx 底层功能，轻松开发出丰富 Web 服务，极大地节约了宝贵的时间和精力。

当然，OpenResty 并不只有 ngx_lua，围绕着 ngx_lua 还有众多的库和辅助工具，构成了一个相当完善的生态环境，这些组件相互支撑，利用得当可以更好地提高生产效率。

OpenResty 现在正处于蓬勃发展的阶段，今后的 OpenResty 也许不仅限于 Nginx 和 Web Server，而将成为一个更通用的开发平台，工作语言也不仅限于 Lua，可能还会有其他新的语言（例如 agentzh 正在做的 edgelang 和 fanlang），让我们拭目以待。

致谢

首先当然要感谢 Nginx 的作者 Igor Sysoev，没有他就不会有如此优秀的 Web 服务器，也就不会有本书的诞生。

OpenResty 创始人章亦春（agentzh）是一位非常亲切随和的人，在 Nginx、DSL、Dynamic Tracing 等领域造诣极高，本书部分章节有幸经他审阅，在此表示最诚挚的谢意。

亲情永远是人生命中最值得珍惜的部分，我要感谢父母多年来的养育之恩和“后勤”工作，感谢妻子在生活中的陪伴，感谢两个可爱的女儿，愿你们能够永远幸福快乐。

最后，我也要感谢读者选择本书，希望读者能够在阅读过程中有所收获，在 Nginx 开发过程中获得乐趣。

您的朋友 罗剑锋

2017 年 4 月 28 日 于 北京 亚运村

目录

第 0 章 导读.....	1	1.3.2 进程配置.....	20
0.1 关于本书	1	1.3.3 动态模块配置.....	22
0.2 读者对象	2	1.3.4 运行日志配置.....	22
0.3 读者要求	3	1.3.5 events 配置	23
0.4 运行环境	4	1.3.6 http 配置	23
0.5 本书的结构	4	1.3.7 server 配置.....	25
0.6 如何阅读本书	7	1.3.8 location 配置.....	26
0.7 本书的源码	8	1.3.9 file 配置	27
第 1 章 Nginx 入门.....	9	1.3.10 upstream 配置.....	27
1.1 关于 Nginx	9	1.3.11 变量.....	28
1.1.1 历史.....	10	1.4 总结	30
1.1.2 特点.....	10	第 2 章 Nginx 开发准备	31
1.1.3 进程模型.....	11	2.1 开发环境	31
1.1.4 版本.....	12	2.1.1 C++标准	31
1.2 安装 Nginx	13	2.1.2 Boost 程序库	32
1.2.1 准备工作.....	13	2.2 目录结构	32
1.2.2 快速安装.....	14	2.3 源码特点	34
1.2.3 运行命令.....	14	2.3.1 代码风格.....	34
1.2.4 验证安装.....	16	2.3.2 代码优化.....	34
1.2.5 定制安装.....	16	2.3.3 面向对象思想.....	34
1.3 配置 Nginx	19	2.4 使用 C++	35
1.3.1 配置文件格式	19	2.4.1 实现原则.....	35

2.4.2 代码风格.....	36	3.6.4 日期操作函数.....	66
2.4.3 编译脚本.....	36	3.6.5 C++封装时间.....	67
2.5 C++包装类.....	38	3.6.6 C++封装日期.....	68
2.5.1 类定义.....	38	3.7 运行日志.....	70
2.5.2 构造和析构.....	39	3.7.1 结构定义.....	71
2.5.3 成员函数.....	40	3.7.2 操作函数.....	71
2.6 总结.....	40	3.7.3 C++封装.....	72
第3章 Nginx 基础设施.....	41	3.8 总结.....	74
3.1 头文件.....	41	第4章 Nginx 高级数据结构.....	77
3.1.1 Nginx 头文件.....	41	4.1 动态数组.....	77
3.1.2 C++头文件.....	42	4.1.1 结构定义.....	77
3.2 整数类型.....	42	4.1.2 操作函数.....	79
3.2.1 标准整数类型.....	43	4.1.3 C++动态数组.....	79
3.2.2 自定义整数类型.....	43	4.2 单向链表.....	83
3.2.3 无效值.....	44	4.2.1 结构定义.....	83
3.2.4 C++封装.....	44	4.2.2 操作函数.....	84
3.3 错误处理.....	47	4.2.3 C++迭代器.....	85
3.3.1 错误码定义.....	48	4.2.4 C++单向链表.....	87
3.3.2 C++异常.....	48	4.3 双端队列.....	90
3.4 内存池.....	50	4.3.1 结构定义.....	90
3.4.1 结构定义.....	51	4.3.2 操作函数.....	91
3.4.2 操作函数.....	51	4.3.3 C++节点.....	93
3.4.3 C++封装.....	52	4.3.4 C++迭代器.....	95
3.4.4 清理机制.....	54	4.3.5 C++双端队列.....	97
3.4.5 C++内存分配器.....	57	4.4 红黑树.....	101
3.5 字符串.....	58	4.4.1 节点结构定义.....	101
3.5.1 结构定义.....	59	4.4.2 树结构定义.....	102
3.5.2 操作函数.....	59	4.4.3 操作函数.....	103
3.5.3 C++封装.....	61	4.4.4 C++红黑树.....	104
3.6 时间与日期.....	64	4.5 缓冲区.....	108
3.6.1 时间结构定义.....	64	4.5.1 结构定义.....	108
3.6.2 时间操作函数.....	64	4.5.2 操作函数.....	110
3.6.3 日期结构定义.....	65	4.5.3 C++缓冲区.....	111

4.6 数据块链	113	第 6 章 Nginx 模块体系	139
4.6.1 结构定义	114	6.1 模块架构	139
4.6.2 操作函数	114	6.1.1 结构定义	139
4.6.3 C++节点	115	6.1.2 模块的签名	141
4.6.4 C++迭代器	117	6.1.3 模块的种类	142
4.6.5 C++数据块链	118	6.1.4 模块的函数指针表	143
4.7 键值对	120	6.1.5 模块的类图	144
4.7.1 简单键值对	120	6.1.6 模块的组织形式	145
4.7.2 散列表键值对	121	6.1.7 模块的初始化	147
4.8 总结	121	6.1.8 模块的动态加载	150
第 5 章 Nginx 开发综述	123	6.2 配置解析	152
5.1 最简单的模块	123	6.2.1 结构定义	152
5.1.1 模块设计	124	6.2.2 配置解析的基本流程	156
5.1.2 配置解析	124	6.2.3 配置数据的存储模型	157
5.1.3 处理函数	126	6.2.4 访问配置数据	163
5.1.4 模块集成	128	6.2.5 确定配置数据的位置	163
5.1.5 编译脚本和命令	129	6.2.6 配置解析函数	165
5.1.6 测试验证	130	6.2.7 配置数据的合并	166
5.2 开发基本流程	131	6.2.8 配置指令的类型	167
5.2.1 设计	131	6.3 源码分析	168
5.2.2 开发	132	6.3.1 ngx_core_module	168
5.2.3 编译	133	6.3.2 ngx_errlog_module	171
5.2.4 测试验证	133	6.4 C++封装	172
5.2.5 调优	133	6.4.1 NgxModuleConfig	172
5.2.6 流程图	133	6.4.2 NgxModule	176
5.3 编译脚本	134	6.4.3 NgxTake	178
5.3.1 运行机制	134	6.4.4 NGX_MODULE_NULL	180
5.3.2 使用的变量	135	6.5 C++开发模块	180
5.3.3 模块脚本	135	6.5.1 模块的基本组成	180
5.3.4 两种脚本格式	136	6.5.2 配置信息类	181
5.3.5 旧式编译脚本	136	6.5.3 业务逻辑类	183
5.4 总结	137	6.5.4 模块集成类	184
		6.5.5 实现源文件	186

6.5.6 增加更多功能	187	第 8 章 Nginx HTTP 请求处理	221
6.6 总结	187	8.1 状态码	221
第 7 章 Nginx HTTP 框架综述	191	8.2 请求结构体	222
7.1 框架简介	191	8.3 请求行	223
7.1.1 模块分类	191	8.3.1 请求方法	223
7.1.2 处理流程	192	8.3.2 协议版本号	224
7.1.3 请求结构体	194	8.3.3 资源标识符	224
7.1.4 请求的处理阶段	195	8.4 请求头	225
7.1.5 请求的环境数据	197	8.5 请求体	226
7.2 处理引擎	198	8.5.1 结构定义	226
7.2.1 函数原型	198	8.5.2 操作函数	227
7.2.2 处理函数的存储方式	198	8.6 响应头	227
7.2.3 内容处理函数	199	8.6.1 结构定义	228
7.2.4 引擎的数据结构	200	8.6.2 操作函数	228
7.2.5 引擎的初始化	201	8.7 响应体	229
7.2.6 引擎的运行机制	202	8.8 源码分析	229
7.2.7 日志阶段的处理	205	8.8.1 ngx_http_static_module	230
7.3 过滤引擎	205	8.8.2 ngx_http_not_modified_filter_ module	231
7.3.1 函数原型	206	8.9 C++封装	232
7.3.2 过滤函数链表	206	8.9.1 ngxHeaders	232
7.3.3 过滤函数的顺序	207	8.9.2 ngxRequestBody	235
7.3.4 过滤链表的运行机制	209	8.9.3 ngxRequest	236
7.3.5 请求体过滤	210	8.9.4 ngxResponse	238
7.4 源码分析	211	8.10 开发 handler 模块	241
7.4.1 ngx_http_static_module	211	8.10.1 模块设计	241
7.4.2 ngx_http_not_modified_filter_ module	212	8.10.2 配置信息类	241
7.5 C++封装	213	8.10.3 业务逻辑类	242
7.5.1 ngxModuleCtx	213	8.10.4 模块集成类	243
7.5.2 ngxHttpCoreModule	215	8.10.5 实现源文件	245
7.5.3 ngxFilter	217	8.10.6 编译脚本	245
7.6 总结	219	8.10.7 测试验证	246
		8.11 开发 filter 模块	246

8.11.1 模块设计	246	9.5.2 NgxUpstreamHelper	283
8.11.2 配置信息类	246	9.5.3 NgxHttpUpstreamModule	285
8.11.3 环境数据类	247	9.5.4 NgxLoadBalance	287
8.11.4 业务逻辑类	248	9.6 开发 upstream 模块	288
8.11.5 模块集成类	251	9.6.1 模块设计	288
8.11.6 实现源文件	252	9.6.2 配置信息类	288
8.11.7 编译脚本	253	9.6.3 业务逻辑类	289
8.11.8 测试验证	253	9.6.4 模块集成类	292
8.12 总结	253	9.6.5 实现源文件	293
第 9 章 Nginx HTTP 请求转发	255	9.6.6 编译脚本	293
9.1 框架简介	255	9.6.7 测试验证	294
9.1.1 工作原理	256	9.7 开发 load-balance 模块	294
9.1.2 请求结构体	257	9.7.1 模块设计	294
9.1.3 上游结构体	258	9.7.2 配置信息类	294
9.1.4 上游配置参数	260	9.7.3 业务逻辑类	295
9.2 请求转发机制	261	9.7.4 模块集成类	297
9.2.1 回调函数	261	9.7.5 实现源文件	298
9.2.2 初始化	263	9.7.6 编译脚本	299
9.2.3 设置连接参数	264	9.7.7 测试验证	299
9.2.4 启动连接	265	9.8 总结	299
9.2.5 处理数据	265	第 10 章 Nginx HTTP 子请求	301
9.3 负载均衡机制	266	10.1 子请求简介	301
9.3.1 结构定义	267	10.1.1 工作原理	302
9.3.2 初始化模块入口	271	10.1.2 请求结构体	303
9.3.3 初始化地址列表	272	10.1.3 回调函数	304
9.3.4 初始化算法	274	10.1.4 待处理请求链表	306
9.3.5 执行算法	274	10.1.5 子请求存储结构	306
9.4 源码分析	275	10.2 子请求运行机制	307
9.4.1 ngx_http_memcached_module	275	10.2.1 创建子请求	307
9.4.2 ngx_http_upstream_ip_hash_		10.2.2 处理引擎	311
module	278	10.2.3 数据整理	312
9.5 C++封装	281	10.3 C++封装	314
9.5.1 NgxUpstream	281	10.3.1 NgxSubRequestHandler	314

10.3.2	NgxSubRequest.....	316	11.5.1	添加变量.....	341
10.4	数据回传模块.....	317	11.5.2	读写变量.....	343
10.4.1	模块设计.....	317	11.6	在模块里使用复杂变量.....	343
10.4.2	环境数据类.....	317	11.6.1	配置信息类.....	344
10.4.3	业务逻辑类.....	319	11.6.2	业务逻辑类.....	344
10.4.4	模块集成类.....	321	11.6.3	模块集成类.....	344
10.4.5	编译脚本.....	322	11.6.4	测试验证.....	344
10.5	在模块里使用子请求.....	323	11.7	总结.....	345
10.5.1	模块设计.....	323	第 12 章	Nginx 辅助设施.....	347
10.5.2	配置信息类.....	323	12.1	摘要算法.....	347
10.5.3	业务逻辑类.....	324	12.1.1	MD5.....	347
10.5.4	测试验证.....	327	12.1.2	SHA-1.....	348
10.6	总结.....	328	12.1.3	MurmurHash.....	349
第 11 章	Nginx 变量.....	329	12.1.4	C++封装.....	349
11.1	结构定义.....	329	12.2	编码和解码.....	352
11.1.1	变量值.....	329	12.2.1	CRC 校验.....	352
11.1.2	变量访问对象.....	330	12.2.2	Base64 编码解码.....	353
11.1.3	变量的存储.....	331	12.2.3	URI 编码解码.....	354
11.1.4	请求结构体.....	331	12.2.4	HTML 和 JSON 编码.....	355
11.2	运行机制.....	332	12.3	正则表达式.....	356
11.2.1	注册变量.....	333	12.4	共享内存.....	356
11.2.2	获取变量.....	333	12.4.1	结构定义.....	357
11.2.3	修改变量.....	334	12.4.2	操作函数.....	357
11.3	复杂变量.....	334	12.4.3	C++共享内存.....	358
11.3.1	结构定义.....	334	12.5	总结.....	359
11.3.2	运行机制.....	335	第 13 章	Nginx 进程机制.....	361
11.4	C++封装.....	335	13.1	基本系统调用.....	361
11.4.1	NgxVariableValue.....	336	13.1.1	errno.....	361
11.4.2	NgxVariable.....	337	13.1.2	getrlimit.....	362
11.4.3	NgxVarManager.....	339	13.2	进程系统调用.....	362
11.4.4	NgxVariables.....	340	13.2.1	getpid.....	362
11.4.5	NgxComplexValue.....	340	13.2.2	fork.....	363
11.5	在模块里使用变量.....	341			

13.2.3	waitpid.....	363
13.3	信号系统调用	364
13.3.1	kill	364
13.3.2	sigaction	365
13.3.3	sigsuspend	365
13.4	结构定义	365
13.4.1	ngx_cycle_t	365
13.4.2	ngx_core_conf_t	366
13.4.3	ngx_process_t	367
13.5	全局变量	368
13.5.1	命令行相关	368
13.5.2	操作系统相关	369
13.5.3	进程功能相关	369
13.5.4	信号功能相关	370
13.6	启动过程	370
13.6.1	基本流程	370
13.6.2	解析命令行	371
13.6.3	版本和帮助信息	372
13.6.4	初始化 cycle	372
13.6.5	测试配置	374
13.6.6	发送信号	374
13.6.7	守护进程化	374
13.6.8	启动工作进程	375
13.6.9	流程图	376
13.7	信号处理	377
13.7.1	信号处理函数	377
13.7.2	发送信号	378
13.7.3	处理信号	378
13.8	单进程模式	379
13.8.1	single 进程	379
13.8.2	single 进程流程图	381
13.9	多进程模式	382
13.9.1	产生子进程	382
13.9.2	master 进程	383

13.9.3	master 进程流程图	387
13.9.4	worker 进程	388
13.9.5	worker 进程流程图	389
13.10	总结	390

第 14 章 Nginx 事件机制 393

14.1	基本系统调用	393
14.1.1	errno	394
14.1.2	ioctl	394
14.1.3	setitimer	394
14.1.4	gettimeofday	395
14.2	socket 系统调用	395
14.2.1	socket	396
14.2.2	bind	396
14.2.3	listen	396
14.2.4	accept	396
14.2.5	connect	397
14.2.6	recv	397
14.2.7	send	397
14.2.8	setsockopt	398
14.2.9	close	398
14.2.10	函数关系图	398
14.3	epoll 系统调用	399
14.3.1	epoll_create	400
14.3.2	epoll_ctl	400
14.3.3	epoll_wait	401
14.3.4	LT 和 ET	401
14.3.5	函数关系图	402
14.4	结构定义	403
14.4.1	ngx_event_t	403
14.4.2	ngx_connection_t	404
14.4.3	ngx_listening_t	405
14.4.4	ngx_cycle_t	407
14.4.5	ngx_os_io_t	408

14.4.6 ngx_event_actions_t.....	411	第 15 章 Nginx 多线程机制.....	455
14.4.7 ngx_posted_events	413	15.1 eventfd 系统调用	455
14.4.8 结构关系图.....	415	15.2 pthread 系统调用	456
14.5 定时器	415	15.2.1 pthread_create	456
14.5.1 红黑树.....	415	15.2.2 pthread_exit.....	457
14.5.2 操作函数.....	416	15.3 结构定义	457
14.5.3 超时处理.....	416	15.3.1 ngx_thread_task_t	457
14.6 模块体系	419	15.3.2 ngx_thread_pool_queue_t.....	458
14.6.1 函数指针表.....	419	15.3.3 ngx_thread_pool_t.....	458
14.6.2 模块的组织形式.....	420	15.3.4 结构关系图.....	459
14.6.3 核心配置.....	422	15.4 事件通知	460
14.6.4 epoll 模块.....	423	15.4.1 函数接口.....	460
14.7 全局变量	425	15.4.2 初始化	460
14.7.1 更新时间相关	425	15.4.3 发送通知.....	461
14.7.2 事件机制相关	426	15.4.4 处理通知.....	462
14.7.3 负载均衡相关	426	15.5 运行机制	463
14.7.4 统计相关	427	15.5.1 完成任务队列.....	463
14.8 运行机制	427	15.5.2 创建线程池.....	463
14.8.1 模块初始化.....	427	15.5.3 创建任务.....	464
14.8.2 进程初始化.....	429	15.5.4 投递任务.....	465
14.8.3 基本参数初始化.....	429	15.5.5 执行任务.....	466
14.8.4 epoll 初始化.....	430	15.5.6 任务完成回调.....	468
14.8.5 连接池初始化.....	431	15.5.7 销毁线程池.....	468
14.8.6 监听端口初始化.....	433	15.6 在模块里使用多线程	469
14.8.7 初始化流程图.....	434	15.6.1 模块设计.....	470
14.8.8 添加事件.....	435	15.6.2 配置信息类.....	470
14.8.9 删除事件.....	439	15.6.3 业务逻辑类.....	470
14.8.10 处理事件.....	440	15.6.4 测试验证.....	474
14.8.11 接受连接.....	444	15.7 总结	474
14.8.12 负载均衡.....	447	第 16 章 Nginx Stream 机制	477
14.8.13 避免阻塞.....	452	16.1 模块体系	477
14.9 总结	452		

16.1.1	函数指针表.....	477
16.1.2	基础模块.....	478
16.1.3	核心模块.....	480
16.1.4	结构关系图.....	481
16.1.5	存储模型.....	482
16.2	监听端口.....	483
16.2.1	结构定义.....	483
16.2.2	解析配置.....	485
16.2.3	启动监听.....	489
16.3	处理引擎.....	491
16.3.1	阶段定义.....	491
16.3.2	函数原型.....	491
16.3.3	处理函数的存储方式.....	492
16.3.4	引擎数据结构.....	492
16.3.5	结构关系图.....	493
16.3.6	引擎的初始化.....	493
16.4	过滤引擎.....	495
16.4.1	函数原型.....	495
16.4.2	过滤函数链表.....	495
16.5	运行机制.....	496
16.5.1	会话结构体.....	496
16.5.2	创建会话.....	497
16.5.3	执行引擎.....	500
16.5.4	通用阶段处理.....	501
16.5.5	预读数据.....	502
16.5.6	产生响应数据.....	506
16.5.7	过滤数据.....	506
16.5.8	结束会话.....	506
16.6	开发 stream 模块.....	507
16.6.1	C++封装.....	507
16.6.2	discard 协议.....	508
16.6.3	time 协议.....	510
16.6.4	changen 协议.....	512
16.7	总结.....	514

第 17 章 Nginx HTTP 机制..... 517

17.1	结构定义.....	517
17.1.1	ngx_http_state_e.....	517
17.1.2	ngx_http_connection_t.....	518
17.1.3	ngx_http_request_t.....	518
17.2	初始化连接.....	519
17.2.1	建立连接.....	520
17.2.2	等待数据.....	521
17.2.3	读取请求头.....	524
17.3	执行引擎.....	528
17.3.1	初始化引擎.....	528
17.3.2	通用阶段.....	530
17.3.3	改写阶段.....	531
17.3.4	访问控制阶段.....	532
17.3.5	内容产生阶段.....	533
17.4	处理请求体.....	534
17.4.1	丢弃缓冲区数据.....	535
17.4.2	读取并丢弃数据.....	536
17.4.3	读事件处理函数.....	537
17.4.4	启动丢弃处理.....	538
17.5	发送数据.....	540
17.5.1	发送初始化.....	540
17.5.2	事件处理函数.....	541
17.6	结束请求.....	543
17.6.1	释放请求资源.....	543
17.6.2	检查引用计数结束请求.....	544
17.6.3	检查状态结束请求.....	545
17.6.4	综合处理结束请求.....	546
17.7	总结.....	548

第 18 章 Nginx 与设计模式..... 551

18.1	设计模式简介.....	551
18.2	框架级别的模式.....	551
18.3	业务级别的模式.....	553

18.4 代码级别的模式	554	19.4.5 应用开发流程.....	584
18.5 总结	556	19.5 功能接口	585
第 19 章 OpenResty 开发	557	19.5.1 运行日志.....	585
19.1 简介	557	19.5.2 时间与日期.....	586
19.1.1 历史.....	558	19.5.3 变量	587
19.1.2 版本.....	559	19.5.4 正则表达式.....	587
19.1.3 组成.....	559	19.5.5 请求处理.....	588
19.1.4 性能.....	561	19.5.6 请求转发.....	590
19.1.5 安装.....	562	19.5.7 子请求	592
19.1.6 目录结构.....	563	19.5.8 定时器	592
19.1.7 命令行工具	564	19.5.9 共享内存.....	593
19.1.8 参考手册.....	565	19.6 应用实例	594
19.2 Lua 语言	566	19.6.1 处理请求.....	594
19.2.1 注释.....	566	19.6.2 过滤请求.....	595
19.2.2 数据类型.....	567	19.6.3 转发请求.....	596
19.2.3 变量.....	568	19.6.4 子请求	597
19.2.4 运算.....	569	19.7 Stream Lua 模块	598
19.2.5 语句.....	570	19.7.1 功能接口.....	598
19.2.6 函数.....	572	19.7.2 discard.....	599
19.2.7 表.....	574	19.7.3 time	599
19.2.8 标准库.....	575	19.7.4 chargin.....	600
19.2.9 模块.....	576	19.7.5 echo.....	600
19.3 LuaJIT.....	577	19.8 lua-resty 库	601
19.3.1 continue.....	578	19.8.1 core.....	601
19.3.2 bit	578	19.8.2 cJSON.....	602
19.3.3 ffi.....	579	19.8.3 redis.....	603
19.4 Lua 模块	581	19.9 总结	603
19.4.1 指令简介.....	581	第 20 章 结束语	605
19.4.2 配置指令.....	581	20.1 本书的遗憾	605
19.4.3 功能指令.....	582	20.2 下一步	605
19.4.4 指令关系图.....	584	20.3 临别赠言	606