

深入新版Nginx源码 详细剖析Nginx核心运行机制
便捷、轻松地开发和定制Nginx

Broadview[®]
www.broadview.com.cn

Nginx

罗剑锋 著

完全开发指南

使用C、C++、JavaScript和Lua

 中国工信出版集团

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

Nginx

完全开发指南

使用C、C++、JavaScript和Lua

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 1.16.0), 详细剖析了模块体系、动态插件、功能框架、内存分配、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++、JavaScript、Lua 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++、JavaScript 和 Lua / 罗剑锋著. —北京: 电子工业出版社, 2019.5
ISBN 978-7-121-36436-5

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
③JAVA 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 (2019) 第 083148 号

策划编辑: 孙学瑛

责任编辑: 牛 勇

印 刷: 山东华立印务有限公司

装 订: 山东华立印务有限公司

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 字数: 871 千字

版 次: 2019 年 5 月第 1 版

印 次: 2019 年 5 月第 1 次印刷

定 价: 109.00 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，走过了不少的弯路。

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。

五年多前，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 企业内部。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

Nginx 完全开发指南：使用 C、C++、JavaScript 和 Lua

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，而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，越来越多的开发者开始转向了 OpenResty，使用 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，大约在五年左右。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。

“大事件”

2019 年，Nginx 身上发生了一件了不得的“大事件”：在独立运营了 8 年之后，Nginx.Inc 公司正式被它的“竞争对手”F5 Networks 收购。

收购的价格并不算高，只有 6.7 亿美元。可以比较一下去年被微软收购的源码托管网站 GitHub，出价是 75 亿。对于 Nginx 这个几乎占据了互联网一半份额的顶尖 Web 服务器来说，感觉实在是有点低。

这也算得上是开源软件商业化的又一个标志性案例了。再往前，还有 IBM 以 340 亿美元收购 RedHat、Sun 以 10 亿美元收购 MySQL（但后来又被 Oracle 收购）。昔日的一个个免费开源明星软件，怀着使命和愿景陆续走向商业化，但又在这条道路上最终倒下。理想终究败给了现实，令人不得不感慨这个“美丽而又残酷”的世界。^①

目前“硕果仅存”的同级知名开源软件应该只有 Apache 和 Linux 了。值得注意的是它们并没有成立商业公司，而是以基金会的形式运作，通过赞助和会员制等形式来筹措资金。另一方面也确实是因为它们的“体量”足够大，能够把“免费”“开源”的大旗继续打下去。

当然，Nginx 和 F5 都发表了官方声明，表示 Nginx 会继续保持开源，共同维护开源社区的活力。在这一点上我选择相信他们，相信被收购后的 Nginx 会有更加光明的未来。

再补上一句私心话：“还等着在 Nginx 上跑 HTTP/3 呢。”

① 出自《进击的巨人 Season 1》片尾曲“美しき残酷な世界”。

新版的变化

我一直是 C++ 语言的坚定拥护者，很早就在积极推动使用 C++ 开发定制 Nginx，但这几年下来感觉“收效甚微”：一是 C 语言开发的惯性太大，二是操作系统对 C++ 标准支持不力，无法用上 11/14 的特性。几番折腾下来，有点“心灰意冷”了。

所以这次的新版就改以 C 作为主力开发语言，同时“忍痛割爱”，大幅度删减了各章节的 C++ 代码，C++ 的全部内容都压缩到一个章节里（第 20 章），不再详细介绍 C++ 封装类的具体实现和内部工作原理，也许这样会更贴近目前 Nginx 开发的现状。

上一版编写时比较匆忙，有些重要知识没来得及讲，这次“清退”C++ 后腾出了大量的页码，就新增了两章（第 12、14 章），仔细分析 Nginx 的内存池和进程间通信机制，补上了“短板”，可以说离“完全”又近了一步。

此外还有一些其他的改动，比如重新梳理子请求的内部实现（第 11 章），JavaScript 由附录升级为正文（第 21 章），专门用两章（第 23、24 章）论述调试、测试与性能优化（非常有用的火焰图）等。

简而言之，“本店全面翻新升级，欢迎光临”。

学习 HTTP

Nginx 是一个 Web 服务器，主要用的是 HTTP 协议，但在实际工作中我经常看到很多人对 HTTP 了解有限，知识掌握的不全面，学习 Nginx 开发时难免“磕磕绊绊”，不明白 Nginx 为什么要这么做、那么做，即使有源码也是不明就里。

所以我打算近期在极客时间 (<https://time.geekbang.org>) 开设一个专栏课程，名字还没想好，暂且叫“透视 HTTP 协议”，以 RFC 为准，精读透讲 HTTP 还有 HTTPS。读者如果感兴趣可以去网站上看看，肯定不会让你失望的。

这也是我第一次自己给自己打广告，但愿不会影响你的心情。

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。

那么, 祝您阅读愉快!

您的朋友 罗剑锋

2019年5月19日 于 北京 798 园区

目录

第0章 导读.....	1	1.3.1 语法格式.....	20
0.1 关于本书.....	1	1.3.2 进程管理.....	21
0.2 读者对象.....	3	1.3.3 动态模块.....	23
0.3 读者要求.....	4	1.3.4 运行日志.....	23
0.4 运行环境.....	5	1.3.5 事件机制.....	23
0.5 本书的结构.....	5	1.4 HTTP 服务.....	24
0.6 如何阅读本书.....	6	1.4.1 基本配置.....	25
0.7 本书的源码.....	7	1.4.2 server 配置.....	26
第1章 Nginx 入门.....	9	1.4.3 location 配置.....	26
1.1 关于 Nginx.....	9	1.4.4 file 配置.....	28
1.1.1 历史.....	10	1.5 TCP/UDP 服务.....	28
1.1.2 特点.....	10	1.6 反向代理.....	29
1.1.3 进程模型.....	11	1.6.1 上游集群.....	29
1.1.4 版本.....	13	1.6.2 负载均衡.....	30
1.2 安装 Nginx.....	13	1.6.3 代理转发.....	31
1.2.1 准备工作.....	14	1.7 变量.....	31
1.2.2 快速安装.....	14	1.8 总结.....	33
1.2.3 运行命令.....	15	第2章 Nginx 开发准备.....	35
1.2.4 验证安装.....	16	2.1 源码结构.....	35
1.2.5 定制安装.....	17	2.2 源码特点.....	36
1.3 配置 Nginx.....	19	2.2.1 代码风格.....	36
		2.2.2 代码优化.....	37

2.2.3 面向对象思想	37	3.8 摘要算法	56
2.3 头文件	38	3.8.1 Times33	57
2.4 总结	38	3.8.2 CRC	57
第3章 Nginx 基础设施	39	3.8.3 MurmurHash	58
3.1 常数	39	3.8.4 MD5	59
3.1.1 环境信息	39	3.8.5 SHA-1	59
3.1.2 版本信息	40	3.9 数据编码	60
3.1.3 错误码	40	3.9.1 Base64	60
3.2 整数类型	41	3.9.2 HTML/JSON	61
3.2.1 标准整数类型	41	3.10 总结	62
3.2.2 自用整数类型	42	第4章 Nginx 高级数据结构	63
3.2.3 无效值	42	4.1 动态数组	63
3.3 内存池	44	4.1.1 结构定义	64
3.3.1 结构定义	44	4.1.2 操作函数	65
3.3.2 操作函数	45	4.1.3 用法示例	66
3.3.3 用法示例	46	4.2 单向链表	67
3.4 字符串	46	4.2.1 结构定义	67
3.4.1 结构定义	46	4.2.2 操作函数	68
3.4.2 操作函数	47	4.2.3 用法示例	68
3.4.3 用法示例	50	4.3 双端队列	70
3.5 时间	51	4.3.1 结构定义	70
3.5.1 结构定义	51	4.3.2 操作函数	71
3.5.2 操作函数	51	4.3.3 用法示例	73
3.5.3 用法示例	52	4.4 红黑树	74
3.6 日期	52	4.4.1 结构定义	75
3.6.1 结构定义	52	4.4.2 操作函数	77
3.6.2 操作函数	53	4.4.3 用法示例	78
3.6.3 用法示例	54	4.5 缓冲区	80
3.7 运行日志	54	4.5.1 结构定义	80
3.7.1 结构定义	54	4.5.2 操作函数	82
3.7.2 操作函数	55	4.5.3 用法示例	83
3.7.3 用法示例	56	4.6 数据块链	84

4.6.1 结构定义	84	6.1.4 模块的函数指针表	107
4.6.2 操作函数	85	6.1.5 模块的类图	108
4.6.3 用法示例	85	6.1.6 模块的组织形式	109
4.7 总结	86	6.1.7 模块的静态加载	111
第 5 章 Nginx 开发概述	87	6.1.8 模块的动态加载	113
5.1 开发示例	87	6.2 配置解析	116
5.1.1 模块设计	87	6.2.1 结构定义	116
5.1.2 配置解析	88	6.2.2 基本流程	119
5.1.3 处理函数	90	6.2.3 存储模型	120
5.1.4 模块集成	92	6.2.4 访问配置数据	125
5.1.5 编译脚本	93	6.2.5 配置数据的位置	126
5.1.6 测试验证	94	6.2.6 配置数据的解析	127
5.2 开发流程	94	6.2.7 配置数据的合并	129
5.2.1 设计	95	6.2.8 配置指令的类型	130
5.2.2 开发	95	6.3 源码分析	131
5.2.3 编译	96	6.3.1 ngx_core_module	131
5.2.4 测试验证	96	6.3.2 ngx_errlog_module	133
5.2.5 调优	96	6.4 总结	135
5.2.6 流程图	97	第 7 章 Nginx 功能框架	137
5.3 编译脚本	97	7.1 框架简介	137
5.3.1 运行机制	98	7.1.1 模块分类	137
5.3.2 脚本变量	98	7.1.2 处理流程	138
5.3.3 添加模块	99	7.1.3 请求的处理阶段	140
5.3.4 脚本格式	99	7.1.4 请求结构体	141
5.3.5 旧式脚本	100	7.1.5 请求的环境数据	143
5.4 总结	101	7.2 处理引擎	144
第 6 章 Nginx 模块体系	103	7.2.1 函数原型	144
6.1 模块架构	103	7.2.2 处理函数的存储方式	144
6.1.1 结构定义	103	7.2.3 内容处理函数	145
6.1.2 模块的签名	105	7.2.4 引擎的数据结构	146
6.1.3 模块的种类	106	7.2.5 引擎的初始化	147
		7.2.6 引擎的运行机制	148

7.2.7 日志阶段的处理.....	151	8.9.3 处理函数.....	173
7.3 过滤引擎.....	151	8.9.4 注册函数.....	174
7.3.1 函数原型.....	151	8.9.5 模块集成.....	175
7.3.2 过滤函数链表.....	152	8.9.6 编译脚本.....	176
7.3.3 过滤函数的顺序.....	153	8.9.7 测试验证.....	176
7.3.4 过滤链表的运行机制.....	155	8.10 开发示例: filter.....	176
7.3.5 请求体过滤.....	156	8.10.1 模块设计.....	176
7.4 源码分析.....	156	8.10.2 配置数据.....	177
7.4.1 ngx_http_static_module.....	157	8.10.3 环境数据.....	177
7.4.2 ngx_http_not_modified_filter_module.....	158	8.10.4 注册过滤函数.....	178
7.5 总结.....	159	8.10.5 过滤响应头.....	178
第 8 章 Nginx 请求处理.....	161	8.10.6 过滤响应体.....	179
8.1 状态码.....	161	8.10.7 模块集成.....	181
8.2 请求结构体.....	162	8.10.8 编译脚本.....	182
8.3 请求行.....	163	8.10.9 测试验证.....	182
8.3.1 请求方法.....	163	8.11 总结.....	183
8.3.2 协议版本号.....	164	第 9 章 Nginx 请求转发.....	185
8.3.3 资源标识符.....	164	9.1 框架简介.....	185
8.4 请求头.....	165	9.1.1 工作原理.....	186
8.5 请求体.....	166	9.1.2 请求结构体.....	187
8.5.1 结构定义.....	166	9.1.3 上游结构体.....	188
8.5.2 操作函数.....	167	9.1.4 上游配置参数.....	189
8.6 响应头.....	167	9.2 请求转发.....	190
8.6.1 结构定义.....	167	9.2.1 回调函数.....	190
8.6.2 操作函数.....	168	9.2.2 初始化.....	192
8.7 响应体.....	169	9.2.3 设置参数.....	193
8.8 源码分析.....	169	9.2.4 启动连接.....	194
8.8.1 ngx_http_static_module.....	169	9.2.5 处理响应头.....	194
8.8.2 ngx_http_not_modified_filter_module.....	171	9.2.6 处理响应体.....	195
8.9 开发示例: content handler.....	172	9.3 负载均衡.....	196
8.9.1 模块设计.....	172	9.3.1 结构定义.....	196
8.9.2 配置数据.....	172	9.3.2 初始化模块入口.....	200

9.3.3 初始化地址列表.....	201	10.1.5 子请求存储结构.....	228
9.3.4 初始化算法.....	203	10.2 运行机制.....	228
9.3.5 执行算法.....	204	10.2.1 创建子请求.....	229
9.4 源码分析.....	204	10.2.2 处理引擎.....	233
9.4.1 ngx_http_memcached_module.....	205	10.2.3 数据整理.....	234
9.4.2 ngx_http_upstream_ip_hash_module.....	207	10.3 开发示例.....	235
9.5 开发示例: upstream.....	210	10.3.1 模块设计.....	236
9.5.1 模块设计.....	210	10.3.2 配置数据.....	236
9.5.2 配置数据.....	210	10.3.3 环境数据.....	236
9.5.3 上行数据.....	212	10.3.4 回调函数.....	236
9.5.4 下行数据.....	212	10.3.5 处理函数.....	237
9.5.5 启动转发.....	213	10.3.6 注册函数.....	238
9.5.6 注册函数.....	214	10.3.7 测试验证.....	239
9.5.7 模块集成.....	214	10.4 总结.....	239
9.5.8 编译脚本.....	215		
9.5.9 测试验证.....	216	第 11 章 Nginx 变量.....	241
9.6 开发示例: balance.....	216	11.1 结构定义.....	241
9.6.1 模块设计.....	216	11.1.1 变量.....	242
9.6.2 配置数据.....	216	11.1.2 复杂变量.....	243
9.6.3 算法数据结构.....	217	11.1.3 变量的存储.....	244
9.6.4 模块入口.....	217	11.1.4 请求结构体.....	244
9.6.5 算法实现.....	218	11.2 操作变量.....	245
9.6.6 模块集成.....	219	11.2.1 添加变量.....	245
9.6.7 编译脚本.....	220	11.2.2 获取变量.....	246
9.6.8 测试验证.....	220	11.2.3 修改变量.....	247
9.7 总结.....	220	11.2.4 编译复杂变量.....	247
		11.2.5 获取复杂变量.....	247
第 10 章 Nginx 子请求.....	223	11.3 开发示例: 变量.....	248
10.1 框架简介.....	223	11.3.1 模块设计.....	248
10.1.1 工作原理.....	224	11.3.2 定义变量.....	248
10.1.2 请求结构体.....	225	11.3.3 添加变量.....	249
10.1.3 回调函数.....	226	11.3.4 获取变量.....	249
10.1.4 待处理请求链表.....	228	11.3.5 测试验证.....	250

11.4 开发示例：复杂变量.....	251	第 13 章 Nginx 进程机制.....	291
11.4.1 模块设计.....	251	13.1 基本系统调用.....	291
11.4.2 定义复杂变量.....	251	13.1.1 errno.....	291
11.4.3 编译复杂变量.....	251	13.1.2 getrlimit.....	292
11.4.4 获取复杂变量.....	252	13.2 进程系统调用.....	292
11.4.5 测试验证.....	252	13.2.1 getpid.....	292
11.5 总结.....	252	13.2.2 fork.....	293
第 12 章 Nginx 内存管理机制.....	255	13.2.3 waitpid.....	293
12.1 基本系统调用.....	256	13.3 信号系统调用.....	294
12.1.1 malloc.....	256	13.3.1 kill.....	294
12.1.2 posix_memalign.....	257	13.3.2 sigaction.....	295
12.1.3 free.....	257	13.3.3 sigsuspend.....	295
12.2 块式内存池.....	258	13.4 结构定义.....	295
12.2.1 结构定义.....	258	13.4.1 ngx_cycle_t.....	295
12.2.2 常量定义.....	261	13.4.2 ngx_core_conf_t.....	296
12.2.3 创建内存池.....	261	13.4.3 ngx_process_t.....	297
12.2.4 分配内存.....	263	13.5 全局变量.....	298
12.2.5 分配大块内存.....	264	13.5.1 命令行相关.....	298
12.2.6 分配小块内存.....	265	13.5.2 操作系统相关.....	299
12.2.7 释放内存.....	270	13.5.3 进程功能相关.....	299
12.2.8 清理机制.....	270	13.5.4 信号功能相关.....	300
12.2.9 清空内存池.....	271	13.6 启动过程.....	300
12.2.10 销毁内存池.....	272	13.6.1 基本流程.....	300
12.3 页式内存池.....	273	13.6.2 解析命令行.....	301
12.3.1 结构定义.....	274	13.6.3 版本和帮助信息.....	301
12.3.2 常量定义.....	276	13.6.4 初始化 cycle.....	301
12.3.3 初始化内存池.....	277	13.6.5 测试配置.....	303
12.3.4 分配内存.....	279	13.6.6 发送信号.....	304
12.3.5 分配大块内存.....	281	13.6.7 守护进程化.....	304
12.3.6 分配小块内存.....	283	13.6.8 启动工作进程.....	305
12.3.7 释放内存.....	286	13.6.9 流程图.....	305
12.4 总结.....	288	13.7 信号处理.....	306
		13.7.1 信号处理函数.....	307

13.7.2	发送信号	307
13.7.3	处理信号	308
13.8	单进程模式.....	309
13.8.1	single 进程.....	309
13.8.2	single 进程流程图	311
13.9	多进程模式.....	311
13.9.1	产生子进程	311
13.9.2	master 进程.....	313
13.9.3	master 进程流程图	316
13.9.4	worker 进程	317
13.9.5	worker 进程流程图	319
13.10	总结	320

第 14 章 Nginx 进程间通信机制..... 323

14.1	基本系统调用.....	323
14.1.1	atomic	323
14.1.2	sched_yield	324
14.1.3	semaphore	324
14.1.4	mmap	325
14.2	共享内存（I）	325
14.2.1	结构定义	325
14.2.2	创建共享内存	325
14.2.3	使用共享内存	326
14.3	自旋锁	326
14.3.1	自旋锁定	327
14.3.2	解除锁定	328
14.3.3	使用自旋锁	328
14.4	互斥锁	328
14.4.1	结构定义	328
14.4.2	创建互斥锁	329
14.4.3	互斥锁定	330
14.4.4	解除锁定	331
14.4.5	销毁互斥锁	332

14.4.6	使用互斥锁	332
14.5	读写锁	333
14.5.1	写锁定	333
14.5.2	读锁定	333
14.5.3	解除锁定	334
14.5.4	降级锁定	334
14.5.5	使用读写锁	335
14.6	共享内存（II）	335
14.6.1	结构定义	335
14.6.2	添加共享内存	336
14.6.3	创建共享内存	337
14.6.4	使用共享内存	338
14.7	总结	339

第 15 章 Nginx 事件机制..... 341

15.1	基本系统调用.....	341
15.1.1	errno.....	342
15.1.2	ioctl.....	342
15.1.3	setitimer	342
15.1.4	gettimeofday	342
15.2	socket 系统调用	343
15.2.1	socket.....	343
15.2.2	bind.....	343
15.2.3	listen	344
15.2.4	accept.....	344
15.2.5	connect.....	344
15.2.6	recv	344
15.2.7	send.....	345
15.2.8	setsockopt	345
15.2.9	close.....	345
15.2.10	函数关系图	346
15.3	epoll 系统调用.....	346
15.3.1	epoll_create.....	347

15.3.2	epoll_ctl	347	15.9	运行机制	381
15.3.3	epoll_wait	348	15.9.1	添加事件	382
15.3.4	LT 和 ET	349	15.9.2	删除事件	385
15.3.5	函数关系图	350	15.9.3	处理事件	386
15.4	结构定义	350	15.9.4	接受连接	390
15.4.1	ngx_event_t	350	15.9.5	负载均衡	392
15.4.2	ngx_connection_t	351	15.10	避免阻塞	397
15.4.3	ngx_listening_t	353	15.11	总结	398
15.4.4	ngx_cycle_t	354			
15.4.5	ngx_os_io_t	355	第 16 章	Nginx 多线程机制	401
15.4.6	ngx_event_actions_t	359	16.1	eventfd 系统调用	401
15.4.7	ngx_posted_events	361	16.2	pthread 系统调用	402
15.4.8	关系图	362	16.3	结构定义	402
15.5	定时器	362	16.3.1	ngx_thread_task_t	403
15.5.1	红黑树	362	16.3.2	ngx_thread_pool_queue_t	403
15.5.2	操作函数	363	16.3.3	ngx_thread_pool_t	404
15.5.3	超时处理	363	16.3.4	结构关系图	405
15.6	模块体系	366	16.4	事件通知	405
15.6.1	函数指针表	366	16.4.1	函数接口	405
15.6.2	模块的组织形式	367	16.4.2	初始化	406
15.6.3	核心配置	369	16.4.3	发送通知	407
15.6.4	epoll 模块	370	16.4.4	处理通知	407
15.7	全局变量	371	16.5	运行机制	408
15.7.1	更新时间相关	371	16.5.1	完成任务队列	408
15.7.2	事件机制相关	372	16.5.2	创建线程池	408
15.7.3	负载均衡相关	373	16.5.3	创建任务	409
15.7.4	统计相关	373	16.5.4	投递任务	410
15.8	进程初始化	374	16.5.5	执行任务	411
15.8.1	初始化函数	374	16.5.6	任务完成回调	413
15.8.2	基本参数初始化	376	16.5.7	销毁线程池	414
15.8.3	事件机制初始化	377	16.6	开发示例	415
15.8.4	连接池初始化	378	16.6.1	模块设计	415
15.8.5	监听端口初始化	379	16.6.2	配置数据	416
15.8.6	初始化流程图	381			