

实战派架构师倾力之作
配套赠送书中案例源代码

C语言 从新手到高手

关松元◎著

The C Programming Language
From Novice To Master

一本带你轻松入门、直接上岗的实操手册
分享给更多有志于通过技术改变生活的人

中国铁道出版社有限公司
CHINA RAILWAY PUBLISHING HOUSE CO., LTD.

前言

FOREWORD

编写初衷

编者 (CIP) 数据

C语言 从新手到高手

关淞元◎著

关淞元◎著

GYUAN FONG XING

关淞元◎著

关淞元◎著

关淞元◎著

关淞元◎著

关淞元◎著

关淞元◎著

关淞元◎著

关淞元◎著

关淞元◎著

中国铁道出版社有限公司
CHINA RAILWAY PUBLISHING HOUSE CO., LTD.

图书在版编目 (CIP) 数据

C 语言从新手到高手/关淞元著. —北京: 中国铁道出版社有限公司, 2020. 1

ISBN 978-7-113-26318-8

I. ①C… II. ①关… III. ①C 语言—程序设计 IV. ①TP312. 8

中国版本图书馆 CIP 数据核字 (2019) 第 229258 号

书 名: C 语言从新手到高手

C YUYAN CONG XINSHOU DAO GAOSHOU

作 者: 关淞元

责任编辑: 王 佩

读者热线电话: 010-63560056

责任印制: 赵星辰

封面设计: 仙境

出版发行: 中国铁道出版社有限公司 (100054, 北京市西城区右安门西街 8 号)

印 刷: 三河市宏盛印务有限公司

版 次: 2020 年 1 月第 1 版 2020 年 1 月第 1 次印刷

开 本: 700mm×1 000mm 1/16 印张: 19.75 字数: 387 千

书 号: ISBN 978-7-113-26318-8

定 价: 79.00 元

版权所有 侵权必究

凡购买铁道版图书, 如有印制质量问题, 请与本社读者服务部联系调换。电话: (010) 51873174

打击盗版举报电话: (010) 51873659

一、编写初衷

目前各种编程语言百花齐放,但是C语言仍然是程序执行效率很高的语言之一,也是贴近底层硬件的语言之一,笔者希望通过本书能让大家了解C语言编程的本质以及设计精髓,而不是简简单单只将C语言当成一种工具。

书中内容来源于笔者从业多年的学习和实战经验,希望分享给更多有志于通过技术改变生活的人。

通过本书的讲解帮助更多的程序开发者提升自身的计算机思维,并明确自己后续的技术发展路线。

二、内容亮点

和市面上同类图书相比,本书的主要特色是根据作者多年的开发架构经验编写,凝聚了一个菜鸟程序员慢慢成长为架构师的程序设计之路,书中包含很多开发设计实例,以及作者对于C语言和设计模式的独特见解,以及如何运用Cache和NUMA等技术来优化程序执行效率,随着多核技术的蓬勃发展,本书也涉及C语言在多核开发下的优势和劣势。

本书共分四大部分。

第一部分新手篇:第1~6章,着重介绍C语言基础语法。

第二部分进阶篇:主要介绍C语言的数据类型、预处理器、编译原理,通过实例阐述C语言的魅力。

第三部分实战篇:包含C语言的经典数据结构和通用设计模式等,以及Linux下程序调试的方法。

第四部分高手篇:介绍如何进行程序优化、Cache利用、NUMA技术、多核技术等。

三、读者对象

本书将 C 语言拟人化，通过一个人的慢慢成长过程来阐述 C 语言学习的进阶之路。

本书既适用于没有经验的编程爱好者，同样适用于对 C 语言有一定经验的使用者，更适合刚刚走出大学校门的读者使用，希望通过对本书的学习完成由学生到一个职业编程人员的快速思维转换。

通过本书，读者可以在短期内对 C 语言有全面的了解，书中包含了大量可执行的实例供读者学习，帮助读者具备实际上手编码能力，以及一定的程序设计和优化能力。

四、致谢

在本书的编写中，首先感谢公司的领导和业内的朋友，他们对本书的策划提出了大量的宝贵意见和建议。

感谢我的妻子孙苗苗，她在我漫长的写作过程中对儿子无微不至的照顾，使我免除了后顾之忧。

感谢我的儿子果果，在本书的构思过程中迎来了这个小生命，他使得我有了更多写作的灵感和动力（本书拟人化的写作手法即来源于此）。

感谢我的父母，他们长久以来对我的培养、鼓励及支持，才使得我有能力以及精力投入到写作过程中。

感谢中国铁道出版社有限公司，因为有他们，才使得本书最终面世。

由于作者水平有限，书中错误之处在所难免，恳请专家和读者批评指正，联系邮箱：boyteam@163.com。

最后以乔布斯的经典名句结尾：“Stay hungry, Stay foolish.”

编者

2019 年 10 月

提示：扫描下方二维码或输入链接地址：<http://www.m.crpdm.com/2019/1021/14191.shtml>，即可获得本书源代码。



目 录

Contents

第一篇 新手篇

第 1 章 C 语言概述

1.1 C 语言的前世今生.....	2
1.2 C 语言的优势与劣势.....	3
1.3 C 语言的当前标准.....	4
1.4 C 语言的编程机制.....	4

第 2 章 C 语言的骨骼——基础数据类型

2.1 常量与变量.....	6
2.1.1 常量.....	6
2.1.2 变量.....	8
2.2 关键字和保留标识符.....	9
2.3 整数类型.....	11
2.4 浮点类型.....	11
2.5 字符和字符串类型.....	13
2.6 类型之间的转换.....	14
2.7 程序注释.....	18

第 3 章 C 语言的肉身——运算符

3.1 算术运算符.....	20
3.2 赋值运算符.....	21

3.3	逻辑运算符.....	21
3.4	移位运算符.....	22
3.5	关系运算符.....	23
3.6	增量运算符.....	24
3.7	位运算符.....	25
3.8	条件运算符.....	26
3.9	逗号运算符.....	27
3.10	运算符的优先级.....	28

第4章 C 语言的血液——控制流

4.1	顺序流.....	30
4.2	条件分支流.....	31
4.3	循环控制流.....	35
4.4	输入输出流.....	38
4.4.1	scanf/printf 函数.....	38
4.4.2	getchar/putchar 函数.....	40
4.4.3	gets/puts 函数.....	41
4.5	语句嵌套.....	42

第5章 C 语言的灵魂——函数

5.1	函数定义.....	44
5.2	函数声明.....	45
5.3	函数参数.....	46
5.4	函数调用.....	48
5.5	函数递归.....	52
5.6	可变参数列表.....	54

第6章 YY 学步——构建第一个程序

6.1	main 函数.....	56
6.2	程序风格.....	58
6.3	第一个 C 程序.....	60
6.4	编译执行.....	61

第二篇 进阶篇

第7章 成长的烦恼——数组和指针

7.1 一维数组.....	64
7.2 多维数组.....	65
7.3 变长数组.....	66
7.4 指针与地址.....	68
7.5 指针数组.....	71
7.6 指向函数的指针.....	73
7.7 指向指针的指针.....	74
7.8 指针和数组的区别.....	75

第8章 成长的积累——结构体、联合体及其他数据形式

8.1 结构体基础知识.....	77
8.2 结构的存储与对齐.....	79
8.3 结构数组.....	84
8.4 指向结构的指针.....	85
8.5 结构体自引用.....	87
8.6 联合体基础知识.....	88
8.7 枚举类型.....	90
8.8 位字段.....	93
8.9 typedef 简介.....	95

第9章 成长的惊喜——预处理器

9.1 宏定义.....	98
9.2 文件包含.....	102
9.3 条件编译.....	103

第10章 成人礼——第一次构建多文件工程

10.1 多源文件编译.....	107
10.2 动态库和静态库.....	113

10.2.1 静态库	114
10.2.2 动态库	115

第三篇 实战篇

第 11 章 骨骼的发育——经典数据结构

11.1 栈	120
11.2 链表	123
11.3 队列	126
11.4 树	129
11.5 堆	134
11.6 散列表	137
11.7 图	139
11.7.1 邻接矩阵	141
11.7.2 邻接表	142
11.7.3 十字链表	142
11.7.4 邻接多重表	143
11.8 一个具体的例子——协议识别引擎	144

第 12 章 社会经验的积累——经典设计模式

12.1 程序设计理念	151
12.2 设计模式原则	152
12.3 单件模式	154
12.4 工厂模式	157
12.5 抽象工厂模式	158
12.6 创建者模式	161
12.7 原型模式	162
12.8 适配器模式	163
12.9 装饰器模式	164
12.10 代理模式	166
12.11 外观模式	167
12.12 桥接模式	169
12.13 组合模式	170

12.14	享元模式.....	172
12.15	策略模式.....	174
12.16	模板方法模式.....	175
12.17	观察者模式.....	177
12.18	迭代器模式.....	179
12.19	责任链模式.....	180
12.20	命令模式.....	182
12.21	备忘录模式.....	183
12.22	状态模式.....	185
12.23	访问者模式.....	186
12.24	中介者模式.....	188
12.25	解释器模式.....	190

第 13 章 成长的挫折——再论程序调试

13.1	断言.....	194
13.2	万能的打印.....	197
13.3	GDB 调试浅谈.....	199
13.3.1	基础命令.....	199
13.3.2	进阶多线程命令.....	205
13.3.3	调试 core 文件.....	206
13.4	符号表与反汇编.....	210
13.5	core 文件的配置.....	212

第 14 章 适应社会——可移植性

14.1	为什么需要可移植.....	214
14.2	如何设计可移植的数据结构.....	215
14.3	如何设计可移植的程序.....	217

第四篇 高手篇

第 15 章 找出自身的不足——性能调试

15.1	程序 Cycle 的意义.....	220
------	-------------------	-----

15.2	性能测试工具的使用	221
15.3	变量的优化	224
15.4	高性能函数	228
15.5	嵌入式汇编	234
15.6	编译优化	237
第 16 章 做事需未雨绸缪——Cache 技术		
16.1	为什么要使用 Cache	241
16.2	Cache 有多少级	242
16.3	Cache Line 的介绍	244
16.4	与 Cache 结合的 CPU 指令	246
16.5	Cache 的淘汰策略	250
16.6	让程序爱上 Cache	252
第 17 章 找到亲近的人与事——NUMA 技术		
17.1	NUMA 简介	254
17.2	NUMA 存储管理	257
17.3	NUMA 相关工具	258
17.4	NUMA 读写实测	260
17.5	让程序爱上 NUMA	261
第 18 章 社会更新换代——大页技术		
18.1	大页简介	262
18.2	Linux 如何配置大页	263
18.3	简述 Hugetlbfs 实现	266
18.4	程序如何使用大页	269
第 19 章 自我修炼——多线程技术		
19.1	进程与线程的区别	271
19.2	多线程编程	275
19.2.1	线程的创建和结束	275
19.2.2	线程同步	277

19.2.3	线程互斥	281
19.2.4	定义线程独有变量	283
19.3	CPU 亲和性	284
19.3.1	RTC 模式	287
19.3.2	Pipeline 模式	288
19.4	多线程调试	289

附录 A 术语表

附录 B 操作符优先级表

附录 C Linux 信号表

1.1 C 语言的前世今生

了解和学习掌握一门编程语言，如果不知道其历史和现状，而只知道一味地苦学和研究，就很容易陷入“一叶障目不见泰山”的境地。因此在本书开头，作者会先介绍 C 语言的诞生、发展和现状。

C 语言于 1973 年诞生在 AT&T 公司设立的贝尔实验室，它最初是为了解决 UNIX 操作系统使用汇编语言实现，从而难以移植的问题而产生的。UNIX 操作系统在 1971 年开发完成最初的版本，该系统当时是使用汇编语言实现的。后来由于里奇和汤普森认为汇编语言实现的操作系统难以移植，他们希望通过一种高级语言重新优化 UNIX 系统。于是在 1973 年，一种新的高级编程语言即 C 语言就此诞生，由 C 语言实现的最新版本的 UNIX 也从而问世。由于 C 语言远胜于汇编语言的可移植性，其对后续的操作系统产生了巨大的影响，其中最著名的就是芬兰人 Linus.Torvalds 开发的 Linux 操作系统，这个开源操作系统已经成为现在流行的操作系统内核之一。由于在这方面的卓越贡献，汤普森和里奇在 1983 年获得了有“计算机界诺贝尔奖”之称的图灵奖。

1977 年，D.M.Ritchie 发表了不依赖于具体机器系统的 C 语言编译文本《可移植的 C 语言编译程序》。

1982 年，很多有识之士和美国国家标准协会为了使 C 语言健康地发展下去，决定成立 C 标准委员会，制订 C 语言的标准。

1989 年，ANSI 发布了第一个完整的 C 语言标准（ANSI X3.159—1989），简称 C89，目前习惯称之为 ANSI C。

1990 年，C89 被国际标准组织 ISO (International Organization for Standardization) 一字不改地采纳，命名为 ISO/IEC 9899，所以 ISO/IEC 9899: 1990 通常被简称为

C90。

1999年,ISO在做了一些必要的修正和完善后,发布了新的C语言标准,命名为ISO/IEC 9899:1999,简称为C99。

2011年12月8日,ISO又正式发布了新的标准,称为ISO/IEC 9899:2011,简称为C11。

以上便是C语言的发展简史,长久以来,作为最底层的基础开发语言,C语言的地位从来没有被撼动过,并且在嵌入式领域仍然是主流语言。随着技术的发展,新的编程语言层出不穷,更新换代,但是,C语言作为基础,在目前来看是基本不可能被替换的。比如Linux内核经过无数代的发展,目前仍以C语言为主导,存在即合理。

1.2 C语言的优势与劣势

C语言的优势如下:

(1) 可移植性高。C语言就是为了可移植性而产生的,它可以在不同的软硬件平台上运行。

(2) 效率高。C语言可以像汇编语言一样直接访问物理地址,对计算机硬件的3种工作单元(位、字节、地址)进行操作。因此其兼备了高级语言和低级语言的功能,可用来编写系统软件。

(3) 程序执行效率高。C语言是编译型语言,生成目标代码质量高,程序执行效率一般只比汇编语言低1~2层。

(4) 语法丰富。C语言拥有丰富的数据结构(第2、7、8章介绍),能够实现各种复杂的数据结构运算,同时包含广泛的运算符(第3章介绍),可以使得表达式多样化,拥有多种程序控制语句(第4章介绍),可以实现其他高级语言难以实现的运算。

(5) 结构式语言。最显著的特点是代码和数据分割化,使得程序的各个部分最大限度地彼此独立,C语言是以函数(第5章介绍)的形式提供给用户的,可以方便地调用这些函数,并且能够通过控制流控制其程序流向,从而使程序完全结构化。

(6) 语法限制少。虽然C语言是强制性语言,但是其程序设计自由度较大,语法较灵活,允许编写者有最大程度的发挥。

(7) 强大的绘图能力。C语言具有强大的绘图能力和数据处理能力,适于编写二维、三维图形和动画。

(8) 库函数众多。C语言经过了多年的发展,提供了大量的库函数,其中包括系统生成的函数和用户定义的函数。举例来说,C编译器自带的头文件,其中就包括可用于开发程序的许多基本功能列表。

同时 C 语言为许多其他目前已知的语言构建模块。因此，学习 C 语言也是了解其他高级语言的基石。

C 语言最大的劣势在于需要自己做内存管理，没有其他高级语言的内存管理回收机制，同时 C 语言在处理复杂的数据结构时，会滋生出大量 bug（如缓冲区溢出、数据越界、动态内存申请释放、空指针等，这些在大部分语言中看起来都不是重点需要关注的点），因此调试 C 程序是成为合格的 C 程序员必备的技能，甚至成为衡量 C 程序员技能经验水平的重要标志，本书在第 13 章会讨论程序的调试手段。

总而言之，我们不应该指望一门编程语言能解决所有的问题。至于 C 语言本身，笔者认为它将在很长的一段时间，带着它的优势和劣势，继续扮演着在计算机世界中不可取代的角色。

1.3 C 语言的当前标准

当下，C 语言的标准分为 GNU C、ANSI C、ISO C。

- GNU C：软件自由基金会制定的标准，它是美国的一个民间非营利组织，致力于推进自由软件，其中 Linux 与 GNU 就是由这个组织在维护。
- ANSI C：由美国国家标准学会制定的标准，它是美国用于制定国家各类标准的组织。
- ISO C：由国际标准化组织制定的标准，它的作用同美国国家标准协会相似，只是这个组织的目标更远大一些，致力于制定国际标准。

大约在 20 世纪 90 年代，美国国家标准学会与国际标准化组织相互接纳吸收对方的标准，所以 ISO C 与 ANSI C 的标准其实是一样的。GNU C 主要应用于 Linux 开发，比标准 C 支持更多的特性，使用起来更加灵活，所以标准 C=ISO C=ANSI C<=GNU C。

1.4 C 语言的编程机制

学习 C 语言，我们首先需要了解下形成可执行文件的大致流程（如图 1.1 所示）：

编辑→预处理→编译→汇编→连接→加载

1. 编辑：通过编译器，进行 C 语言代码的编写，这里推荐一款平台——Source Insight，它可以整理出原文件内部各种关系，并通过丰富的界面进行展示。

2. 预处理：这个阶段用来处理代码中的条件编译指令（#if、#ifdef 等）、展开头文件（#include）、替换宏定义（#define）、删除所有注释、添加行号和文件名标示（当编译错误时提示的错误位置就是由此而来）。

3. 编译：对经过预处理的文件进行词法分析、语法分析、语义分析及优化后，产生相应的汇编代码文件。

4. 汇编：将上述的汇编代码文件翻译成机器指令（每一条汇编语句基本都可翻译成一条机器指令），并生成目标程序的.o 文件，该文件为二进制文件，字节编码是机器指令。

5. 连接：通过连接器将上述.o 文件以及程序指定包含的库文件连接在一起，生成一个完整的可执行程序。

6. 加载：将上述可执行程序加载到内存中，并获取 CPU 的执行权限，开始运行。



图 1.1 C 语言编译执行机制