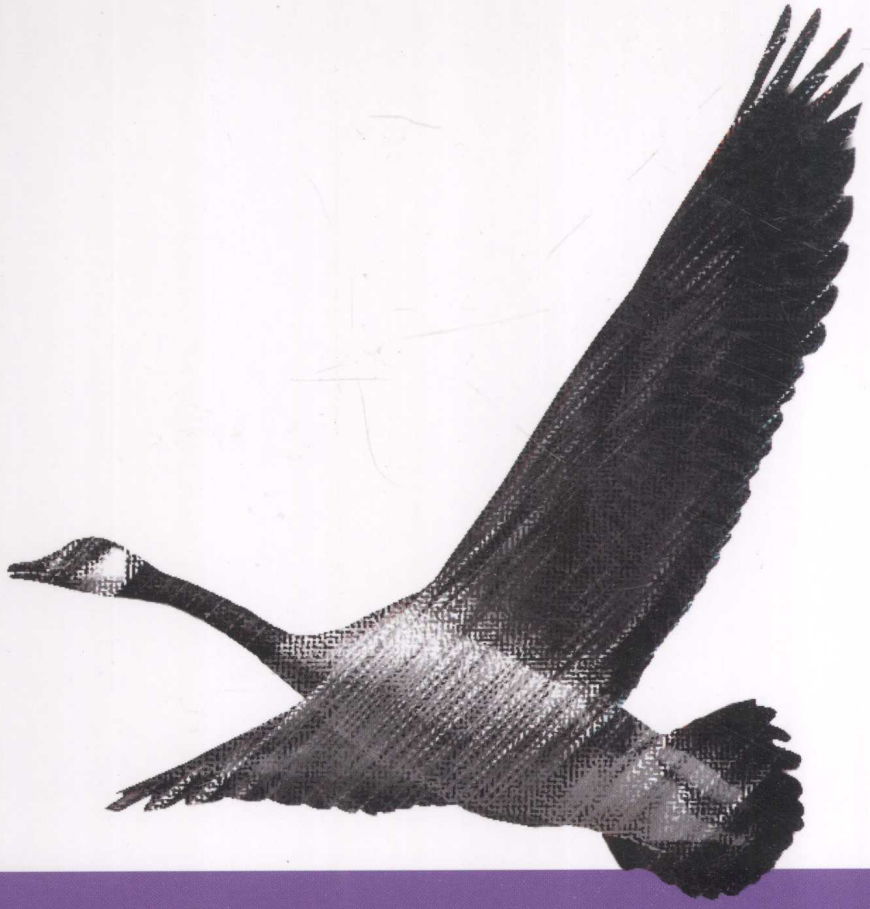
Broadview
www.broadview.com.cn

嵌入式技术丛书

亚嵌教育 AKA www.akaedu.org 作品系列



Linux C编程一站式学习

北京亚嵌教育研究中心 组编
宋劲杉 编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

嵌入式技术丛书

亚嵌教育 www.akaedu.org 作品系列

Linux C编程一站式学习

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书有两条线索，一条线索是以Linux平台为载体全面深入地介绍C语言的语法和程序的工作原理，另一条线索是介绍程序设计的基本思想和开发调试方法。本书分为两部分：第一部分讲解编程语言和程序设计的基本思想方法，让读者从概念上认识C语言；第二部分结合操作系统和体系结构的知识讲解程序的工作原理，让读者从本质上认识C语言。

本书适合做零基础的初学者学习C语言的第一本教材，帮助读者打下牢固的基础。有一定的编程经验但知识体系不够完整的读者也可以对照本书查缺补漏，从而更深入地理解程序的工作原理。本书最初是为北京亚嵌教育研究中心的嵌入式Linux系统工程师就业班课程量身定做的教材之一，也适合作为高等院校程序设计基础课程的教材。本书对于C语言的语法介绍得非常全面，对C99标准做了很多解读，因此也可以作为一本精简的C语言语法参考书。

图书在版编目（CIP）数据

Linux C编程一站式学习 / 北京亚嵌教育研究中心组编；宋劲杉编著. —北京：电子工业出版社，2009.12
（嵌入式技术丛书）
ISBN 978-7-121-09771-3

Ⅰ. L… Ⅱ. ①北… ②宋… Ⅲ. ①Linux操作系统—程序设计 ②C语言—程序设计
Ⅳ. TP316.89 TP312

中国版本图书馆CIP数据核字（2009）第195744号

责任编辑：李 冰
文字编辑：江 立
印 刷：北京东光印刷厂
装 订：三河市皇庄路通装订厂
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：787×1092 1/16 印张：30 字数：690千字
印 次：2009年12月第1次印刷
印 数：4000册 定价：60.00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。
质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。
服务热线：（010）88258888。

前　言

本书最初是为北京亚嵌教育研究中心的嵌入式 Linux 系统工程师就业班课程量身定做的教材之一。该课程是为期四个月的全日制职业培训，要求学员毕业时具备非常 Solid 的 C 语言编程能力，能熟练地使用 Linux 系统，同时对计算机体系结构与指令集、操作系统原理和设备驱动程序都有比较深入的了解。然而学员入学时的水平是非常初级而且参差不齐的：学历有专科、本科也有研究生；专业有和计算机相关的，也有很不相关的（例如会计专业）；以前从事的职业有和技术相关的也有完全不相关的（例如 HR）；年龄从二十岁出头到三十五六岁的都有。这么多背景、基础、思维习惯和理解能力完全不同的人来听同一堂课，大家都迫切希望学会嵌入式开发技术，投身 IT 行业，这就是职业教育的特点，也是我编写本书时需要考虑的主要问题。

学习编程绝不是一件简单的事，尤其是对于零基础的初学者来说。大学的计算机专业有四年时间从零基础开始培养一个人，微积分、线性代数、概率论、离散数学、组合数学、自动机、编译原理、操作系统、计算机组成原理等一堆基础课，再加上 C/C++、Java、数据库、网络工程、软件工程、计算机图形学等一堆专业课，最后培养出一个能找到工作的学生。很遗憾这最后一条很多学校没有做好，据我们考查，来亚嵌培训的很多学生基础几乎为零，我不知道为什么。与之形成鲜明对比的是，只给我们四个月的时间，同样要求从零基础开始，最后培养出一个能找到工作的学生，而且还要保证他找到好工作，这就是职业教育的特点。

为什么我说“只给我们四个月的时间”？我们倒是想教四年呢，但学时的长短我们做不了主，是由市场规律决定的。四年的任务要求四个月做好，要怎么完成这样一个几乎不可能的任务呢？有些职业教育给出的答案是“实用主义”，打出了“有用就学，没有用就不学”的口号，大肆贬低说大学里教的基础课都是过时的、无用的，只有他们教的技术才是实用的。这种炒作很不好，我认为大学里教的每一门课都是非常有用的，基础知识在任何时候都不会过时，倒是那些时髦的“实用技术”有可能很快就会过时了。

四年的任务怎么才能用四个月做好？我们给出的答案是“优化”。现在大学里安排的课程体系最大的缺点就是根本不考虑优化。每个过来人都会有这样的感觉：大一大二学了好多数学课，却不知道都是干什么用的，不明白为什么要学。连它有什么用都不知道怎么能有兴趣学好呢？到大三大四学专业课时，用到以前的知识了，才发现以前学的数学是多么有用，然而早就忘得一干二净了，考完试都还

给老师了。回头重新学，才发现很多东西以前根本没学明白，现在真的学明白了，那么前两年的时间岂不是都浪费了？大学里的课程体系还有一个缺点就是不灵活，每门课必须占用一个学期，必须由一个老师教，不同课程的老师之间没有任何沟通和衔接，其实这些课程之间是相互依赖的，把它们强行拆开是不符合人的认知规律的。比如我刚上大学的时候，大一上半学期就被逼着学习 C 语言，其实 C 语言是一门很难的编程语言，不懂编译原理、操作系统和计算机体系结构根本不可能学明白，那半个学期自然就浪费掉了。当时几乎所有学校的计算机相关专业都是这样，大一刚来就学 C 语言，有的学校更疯狂，上来就学 C++，导致大多数学生都以为自己会 C 语言，但其实都是半吊子水平，到真正写代码的时候经常为一个 Bug 搞得焦头烂额，却没有机会再系统地学一遍 C 语言。因为在学校看来，C 语言早在大一就给你“上完了”，就像一顿饭已经吃完了，不管你吃饱没吃饱，不会再让你重吃一遍了。显而易见，如果要认真地对这些课程进行优化，的确是有很多水分可以挤的。

本书有什么特点

- 本书不是孤立地讲 C 语言，而是和编译原理、操作系统、计算机体系结构结合起来讲。或者说，本书的内容只是以 C 语言为载体，真正讲的是计算机和程序的原理。
- 强调基本概念和基本原理，在编排顺序上重视概念之间的依赖关系，每次引入一个新的概念，只依赖于前面章节已经讲过的概念，而绝不会依赖于后面章节要讲的概念。有些地方为了叙述得完整，也会引用后面要讲的内容，比如说“有关××我们到第×章再仔细讲解”，凡是这种引用都不是必要的依赖，可以当它不存在，只管继续往下学习就行了。
- 尽量做到每个知识点直到要用的时候才引入。过早引入一个知识点，讲完了又不用它，读者很快就会遗忘，这是不符合认知规律的。

本书面向什么样的读者

这是一本从零基础开始学习编程的书，不要求读者有任何编程经验，但读者至少需要具备以下素质：

- 熟悉 Linux 系统的基本操作。如果不具备这一点，请先参考其他教材学习相关知识，熟练之后再学习本书，《鸟哥的 Linux 私房菜》据说是 Linux 系统管理和应用方面比较好的一本书。但学习本书并不需要会很多系统管理技术，只要会用基本命令、会自己安装系统和软件包就足够了。
- 具有高中毕业的数学水平。本书会用到高中的数学知识。事实上，如果不具有高中毕业的数学水平，也不必考虑做程序员了。但并不是说只要具有高中毕业的数学水平就足够做程序员了，只能说看这本书应该没有问题，数学是程序员最重要的修养，计算机科学其实就是数学的一个分支，如果你的数学功底很差，日后还需要恶补一下。

- 具有高中毕业的英文水平。理由同上。
- 对计算机的原理和本质深感兴趣，不是为就业而学习，不是为拿高薪而学习，而是真的感兴趣，想把一切来龙去脉搞得清清楚楚而学习。
- 勤于思考。本书尽最大努力理清概念之间的依赖关系，力求一站式学习，读者不需要为了找一个概念的定义去翻阅其他书籍，也不需要为了搞清楚一个概念在本书中乱翻一通，只需要从前到后按顺序学习即可。但一站式学习并不等于傻瓜式学习，有些章节有一定的难度，需要读者积极思考才能领会。本书可以替你节省时间，但不能替你思考，不要指望像看小说一样走马观花看一遍就能学会。

为什么要学这本书而不是 K&R

《The C Programming Language》（后文简称[K&R]）是公认的世界上最经典的 C 语言教程之一，这点毫无疑问。在 C 标准出台之前，K&R 第一版就是事实上的 C 标准。C89 标准出台之后，K&R 跟着推出了第二版，可惜此后就没有更新过了，所以不能反映 C89 之后 C 语言的发展以及最新的 C99 标准。本书在这方面做了很多补充。本书与其说是讲 C 语言，不如说是以 C 语言为载体讲计算机和操作系统的原理，而 K&R 只是为了讲 C 语言而讲 C 语言，侧重点不同，内容编排也很不相同。K&R 写得非常好，代码和语言都非常简洁，但很可惜，只有会 C 语言的人才懂得欣赏它，K&R 是非常不适合入门学习的，尤其不适合零基础的学生学习。

本书“是什么”和“不是什么”

本书包括两大部分：

- C 语言入门。介绍基本的 C 语法，帮助没有任何编程经验的读者理解什么是程序以及怎么写程序，培养程序员的思维习惯，找到编程的感觉。前半部分改编自《How To Think Like A Computer Scientist: Learning with C++》（后文简称[ThinkCpp]）。
- C 语言本质。结合计算机和操作系统的原理讲解 C 程序是怎么编译、链接、运行的，同时全面介绍 C 的语法。位运算的章节改编自林小竹老师的讲义；链表和二叉树的章节改编自朱仲涛老师的讲义；汇编语言的章节改编自《Programming from the Ground Up: An Introduction to Programming using Linux Assembly Language》（后文简称[GroundUp]），在该书的最后一章中提到，学习编程有两种 Approach，一种是“Bottom Up”，一种是“Top Down”，它们各有优缺点，而我们需要将两者结合起来。所以我编写本书的思路是：第一部分 Top Down；第二部分 Bottom Up；第三部分可以算填补了中间的空隙，三部分全都围绕 C 语言展开。

这本书定位在入门级，虽然内容很多，但不是一本百科全书，除了 C 语言的基础

知识要讲透之外其他内容都不深入，书中列出了很多参考资料，是读者进一步学习的起点。[K&R]的第 1 章是一个 Whirlwind Tour，把全书的内容简单概括了一遍，然后再逐个深入讲解。本书也可以看作是计算机专业课程体系的一个 Whirlwind Tour，学习完本书之后读者有了一个全局观，再去学习那些参考资料就应该很容易上手了。

为什么要在 Linux 平台上学 C 语言？用 Windows 学 C 语言不好吗？

用 Windows 还真的是学不好 C 语言。C 语言是一种面向底层的编程语言，要写好 C 程序，必须对操作系统的工作原理非常清楚，因为操作系统也是用 C 语言编写的，我们用 C 语言编写应用程序可以直接使用操作系统提供的接口。既然你选择了本书，你一定了解：Linux 是一种开源的操作系统，你有任何疑问都可以从源代码和文档中找到答案，即使你看不懂源代码，也找不到文档，也很容易找个高手教你，各种邮件列表、新闻组和论坛上从来都不缺乐于助人的高手；而 Windows 是一种封闭的操作系统，除了微软的员工别人都看不到它的源代码，只能通过文档去猜测它的工作原理。更糟糕的是，微软向来喜欢藏着掖着，好用的功能留着自己用，而不会写到文档里公开。本书的第一部分在 Linux 或 Windows 平台上学习都可以，但第二部分和第三部分介绍了很多 Linux 操作系统的原理以帮助读者更深入地理解 C 语言，所以后两部分只能在 Linux 平台上学习。

Windows 平台上的开发工具往往和各种集成开发环境（Integrated Development Environment，IDE）绑在一起，例如 Visual Studio、Eclipse 等。使用 IDE 确实很便捷，但 IDE 对于初学者绝对不是好东西。微软喜欢宣扬傻瓜式编程的理念，告诉你用鼠标拖几个控件，然后单击一个按钮就可以编译出程序来，但是真正有用的程序有哪个是这么拖出来的？很多从 Windows 平台入门学编程的人，编了好几年程序，还是只知道编完程序单击一个按钮就完事了，把几个源文件拖到一个项目里就可以编译到一起了，如果有更复杂的需求他们就傻眼了，因为他们脑子里只有按钮、菜单的概念，根本没有编译器、链接器、Makefile 的概念，甚至连命令行都没用过，然而这些都是初学编程就应该建立起来的基本概念。另一方面，编译器、链接器和 C 语言的语法有着密切的关系，不了解编译器、链接器的工作原理，也不可能真正掌握 C 语言的语法。所以，IDE 并没有帮助你学习，而是阻碍了你的学习，本来要学好 C 编程只要把语法和编译命令学会就行了，现在有了 IDE，除了学会语法和编译命令，你还得弄清楚编译命令和 IDE 是怎么集成的，这才算学明白了，本来就很复杂的学习任务被 IDE 搞得更加复杂了。Linux 用户的使用习惯从来都是以敲命令为主，以鼠标操作为辅，从学编程的第一天起就要敲命令编译程序，等到你把这些基本概念都搞清楚了，你觉得哪个 IDE 好用你再去用，不过到那时候你可能会更喜欢 `vi` 或 `emacs` 而不是 IDE 了。

体例说明

像 The quick brown fox jumps over the lazy dog 这样的字体在本书中是代码字体。这种字体的名称是 Dejavu Sans Mono，为什么我要提倡用这种字体呢？第一，它是等宽字体，因此适合做代码字体。第二，它的 1 和 l、0 和 O 区分得非常清楚（我在教学中发现初学者很容易把这些字符抄错），因此它比 Courier New 更适合做代码字体。第三，它是我的 Linux 图形终端的默认字体，采用这种字体排版可以使得看书和看屏幕的感觉很一致，希望读者在看这本书时也会有这种 Dejavu（似曾相识）的感觉。

像下面这样有边线的是代码：

```
#! /bin/sh
VAR=1
VAR=$(($VAR+1))
echo $VAR
```

没有边线的是终端显示，包括输入的命令和程序运行结果，例如：

```
$ VAR=1
$ VAR=$(($VAR+1))
$ echo $VAR
2
```

本书中统一用$表示 Shell 提示符。

加粗的字句表示强调。

在定义一个名词时会给出它的英文名称，例如集成开发环境（Integrated Development Environment，IDE），通过书后的索引可以找到这些定义在书中首次出现的位置。

致谢

本书的写作得到了北京亚嵌教育研究中心的全力支持，尤其感谢李明老师和何家胜老师。没有公司的支持，我不可能有时间有条件写这本书，也不可能有机会将这本书公开在网上。

然后要感谢亚嵌教育的历届学员和各位老师，在教学和讨论的过程中我经常会得到有益的启发，这些都促使本书更加完善。在本书的写作过程中，很多读者为本书提出了很有价值的建议，很多建议是热心网友通过在线评论提出的，有些网友我只知道 ID 或 E-mail。在此向他们表示感谢。

感谢帮助过我的老师们：李明、何家胜、邸海霞、郎铁山、朱仲涛、廖文江、韩超、秦蔚、吴岳、张颐、邢文鹏、何晓龙、林小竹、卫剑钒、郭同彬、王波、王磊。

感谢热心网友：ddd、wuyulei、commapopo、田伟、田雨、daidai、邓楠、杜朴风、Zoom.Quiet、陈莉君老师、杨景、章钰、chen、Jiawei Zhang、waterloo、张现超、曾宇、董俊波、RobinXiang、刘艳明、been2100、cleverd、juicerococo、徐斌、cyy、Linux_Xfce、冯海云、侯延祥、churchmice、codycody23、syfeagle、王公仆、刘敏、Laciq、yuchen、陆杨、陈杨希、love_wc3、姚磊、芝麻、wadenx、沈震、sunbingfly、mick、baaluck、曹帅军、zhoudy、朱夜光、刺猬、leezhenfeng、王兆宏、徐凯、码匠、况海斌、尹志伟、王星。

还要感谢电子工业出版社博文视点资讯有限公司的周筠老师和李冰老师的大力支持，感谢江立编辑严谨细致的工作。

在写作过程中我遇到过很多困难：工作繁忙、对未来迷茫、生活压力大、缺乏安全感、个人琐事等。然而有这么多热心的同学、老师、朋友、网友在等着阅读我的书在线更新的内容，给我提建议，希望我把书改得更完善，这是我坚持写下去的最大动力。谢谢你们！

由于作者水平十分有限，没写过C编译器和C标准库，所以疏漏之处在所难免，如有错误欢迎广大读者朋友批评指正。写书是一件严肃的事，书中的错误所有人都看得见，白纸黑字赖不掉的。我教过的很多学生都在大学里学过C语言，甚至考过二级，但程序写得一塌糊涂，连最基本的概念都搞错了，以前学过的C语言教材中的错误在他们脑子里根深蒂固，即使我纠正多次，他们仍然只记得以前学过的错误概念。这种有基础的学生还不如没有任何基础的学生教起来容易。我非常害怕我教给别人的知识也是错的，所以我仔细研究了C99之后才敢动笔写书。这本书涵盖的话题比较广泛，我竭尽全力也不足以保证书中的内容全部正确，还要依靠社区的力量一起来完善这本书，这样才能真正对读者负责，所以我选择将这本书开源。本书的在线版本位于http://learn.akae.cn/。

希望本书能成为你求学道路上的第一个伙伴。

宋劲杉

2009年7月22日

目　录

上篇　C 语言入门

下篇 C 语言本质

上篇 C 语言入门

第 1 章

程序的基本概念

1.1 程序和编程语言

程序（Program）告诉计算机应该如何完成一个计算任务，这里的计算可以是数学运算（如解方程），也可以是符号运算（如查找和替换文档中的某个单词）。从根本上说，计算机是由数字电路组成的运算机器，只能对数字进行运算，程序之所以能进行符号运算，是因为符号在计算机内部也是用数字表示的。此外，程序还可以处理声音和图像。声音和图像在计算机内部必然也是用数字表示的，这些数字经过专门的硬件设备转换成人可以听到的声音和看到的图像。

程序由一系列指令（Instruction）组成，指令是指示计算机进行某种运算的命令，通常包括以下几类：

输入（Input）

从键盘、文件或者其他设备获取数据。

输出（Output）

把数据显示到屏幕，或者存入一个文件，或者发送到其他设备。

基本运算

执行最基本的数学运算（加减乘除）和数据存取。

测试和分支

测试某个条件，然后根据不同的测试结果执行不同的后续指令。

循环

重复执行一系列操作。

对于程序来说，有上面这几类指令就足够了。你曾用过的任何一个程序，不管它有多么复杂，都是由这几类指令组成的。程序是那么复杂，而编写程序可以用的指令却只有这么简单的几种，这中间巨大的落差就要由程序员去填补了，所以编写程序理应是一件相当复杂的工作。**编写程序可以说就是这样一个过程：把复杂的任务分解成子任务，把子任务再分解成更简单的任务，层层分解，直到最后简单得可以用以上指令来完成。**