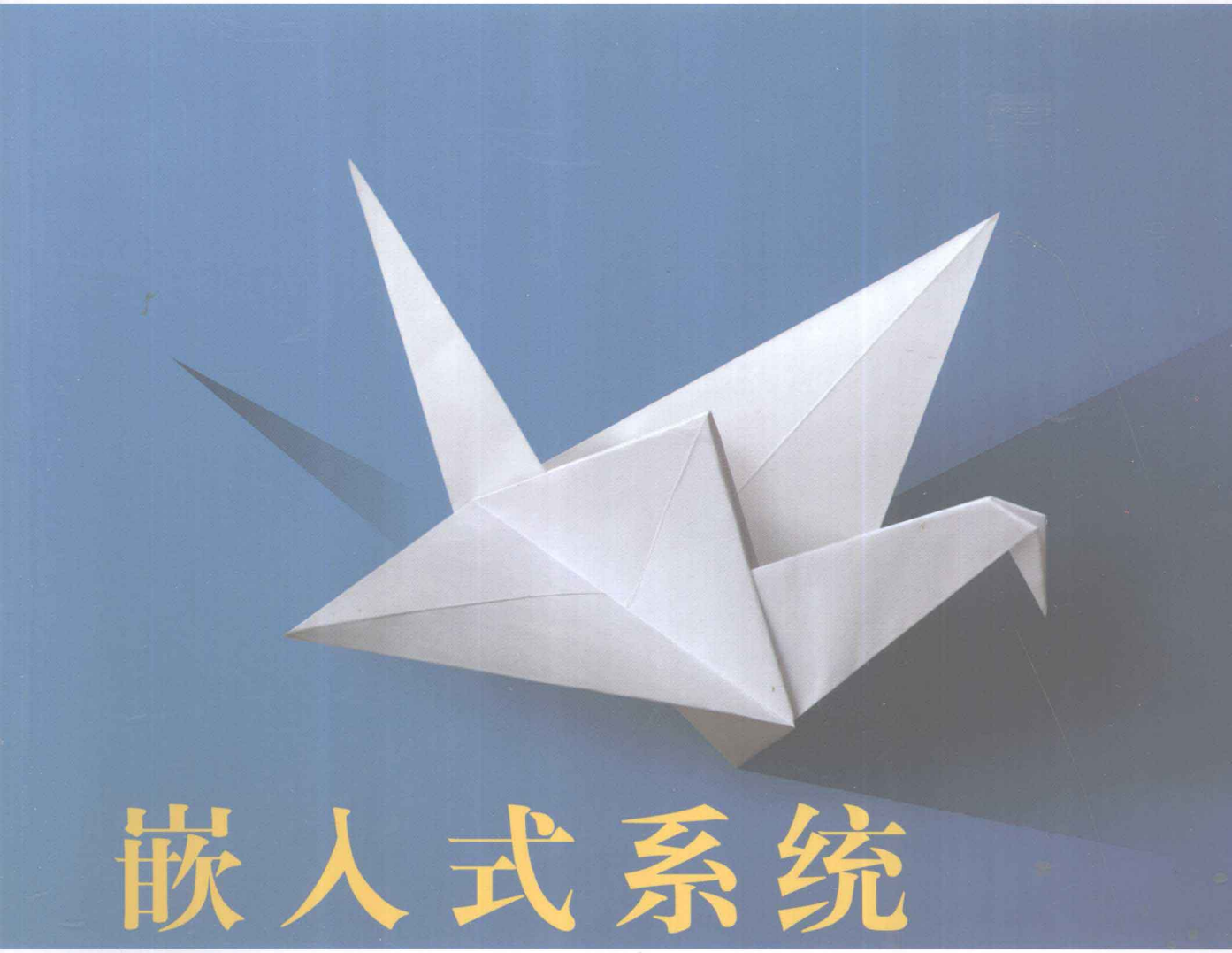
“十二五”高等院校规划教材



嵌入式系统
高级C语言编程

凌　明　编著

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

“十二五”

嵌入式系统高级C语言编程

凌　明　编著

北京航空航天大学出版社

内容简介

本书主要介绍针对嵌入式系统基于C语言的软件项目开发流程、较为复杂的C语言编程知识与技巧、编程风格及调试习惯，并通过对一个具体的软件模块(ASIX Window GUI)的分析，介绍分析代码的方法以及设计软件系统需要考虑的各要素。本书以实际项目中的代码为例来进行介绍，详细分析在嵌入式系统开发中程序员应该注意的方法、技巧和存在的陷阱。

本书适合用作学习嵌入式系统的高年级本科生或硕士研究生的教学用书，也可作为从事嵌入式系统编程的软、硬件工程师的技术参考用书。

图书在版编目(CIP)数据

嵌入式系统高级C语言编程/凌明编著. --北京：
北京航空航天大学出版社，2011.1
ISBN 978-7-5124-0308-6

Ⅰ.①嵌… Ⅱ.①凌… Ⅲ.①微型计算机—
C语言—程序设计 Ⅳ.①TP360.21②TP312

中国版本图书馆CIP数据核字(2011)第004037号

嵌入式系统高级C语言编程
凌 明 编著
责任编辑 冯 颖
*
北京航空航天大学出版社出版发行
北京市海淀区学院路37号(邮编100191) http://www.buaapress.com.cn
发行部电话：(010)82317024 传真：(010)82328026
读者信箱：emsbook@gmail.com 邮购电话：(010)82316936
涿州市新华印刷有限公司印装 各地书店经销
*
开本：787×960 1/16 印张：21 字数：470千字
2011年1月第1版 2011年1月第1次印刷 印数：4 000册
ISBN 978-7-5124-0308-6 定价：39.00元

前 言

最早接触编程语言是1990年我在东南大学电子工程系(现在已经改名为东南大学电子科学与工程学院)读本科一年级时学习的FORTRAN编程语言。当时我对计算机的基本组成原理以及计算机是如何工作的基本上是一无所知,我和同学们稀里糊涂地在计算中心的UNIX主机上编写自己的FORTRAN代码,最后稀里糊涂地通过考试、拿到学分,但确实没有留下什么深刻的印象(老师听到要生气了,呵呵)。大二的时候系里面组建了自己的PC机房,我疯狂地迷上了在PC上的编程。当时的主要编程工具是微软的Quick BASIC和Borland公司的Turbo C,当然这些都是作为自娱自乐和自我陶醉的小把戏,没有真正做出什么有用的东西。

1998年我考入东南大学国家专用集成电路系统工程技术研究中心(以下简称ASIC工程中心)攻读硕士学位。那时中心有一个基于Motorola公司68000内核(现在该公司的半导体部门已经独立成为Freescale公司)68EZ328龙珠处理器的PDA研发项目(2000年的时候这个项目的研发成果实现了产业化,深圳的一家公司后来推出了"蓝火"随身E无线金融掌上电脑),我和师兄郑凯东的研发任务是为这个PDA项目开发TCP/IP协议栈。当时国内做这方面研发的人还比较少,可以参考的资料更是有限,郑凯东从互联网上下载了开源的OS-Karn写的一个KA9Q。KA9Q是非常精致的基于DOS的网络操作系统,它在DOS之上模拟了一个协作式多任务内核,并基于这个内核实现了非常完整的TCP/IP协议栈。我们的任务就是将这个由上百个C文件构成的系统移植到68000内核上。我们几乎没有任何文档,KA9Q中的注释也非常精练,唯一的办法就是苦读这些源文件,然后一点一点将这些文件移植到68EZ328自带的PPSM操作系统上。在协议完整地移植后,我们还为这个PDA编写了自己的POP3协议、SMTP协议以及与之相关的电子邮件收发应用程序。现在看起来,花了大半年的时间阅读高手编写的代码对于后来自己的成长是非常有帮助的,这个经历让我真正理解了编写大型软件的思维方式,同时也对C语言的博大精深有了深刻的体会。

在完成了PDA项目的TCP/IP协议移植后,我的主要精力转移到了为PDA应用程序编写统一软键盘模块。在当时的软件负责人胡晨老师的指导下,我将这个键盘模块设计成为采用消息过滤和回调机制的软件架构,这为后来编写ASIC工程中心自主的嵌入式图形用户界面ASIX Window打下了基础。后来我们又将ASIX Window移植到EPSON公司的C33209

处理器上，并且和自己编写的抢占式多任务内核 ASIX OS Kernel 融合到了一起，构成了一个比较完整的嵌入式软件开发平台。在 2001 年与北京大学微处理器研究中心合作的众志 805 微处理器项目中，我们为这个处理器开发了集成开发环境(IDE)，并且也成功地将我们的 ASIX OS 移植到了该处理器。2003 年 ASIC 工程中心成为国内第一家 ARM 大学计划的授权用户，成功研发了基于 ARM7TDMI 的 SEP3203 嵌入式微处理器，我们又将 ASIX OS 移植到了 ARM 平台。2005 年 SEP3203 处理器进入量产阶段。作为芯片的设计单位，ASIC 中心为 SEP3203 处理器开发了大量的底层软件平台和相关的应用方案，这些方案包括无线数据传输终端(DTU)、EPOS(电子支付终端)解决方案、继电保护终端、电机保护器等。就在本书快要成稿时，SEP3203 处理器的升级版本 SEP4020 嵌入式微处理器也进入了量产阶段，为移动导航终端(PND)、手持多媒体播放器(PMP)等应用设计的基于 ARM1136 处理器内核的 SEP0718 处理器也正式立项，并推出了第一个版本的测试芯片。我们为这两款处理器移植了嵌入式 Linux 和 Windows CE 操作系统，并开发了基于这些操作系统的底层软件包和相关驱动。

在上面所介绍的研发过程中，大量的研究生参与了其中的工作。由于考入 ASIC 工程中心的研究生多是电子类专业背景，因此他们的编程经验大多和我刚读硕士时一样的贫乏。C 语言本质上(至少从它诞生之初起)不是一门面向初学者的编程语言，其灵活多变、语法检查不严格、对底层存储器的直接操作(主要是通过各种形式的指针)等特征使得 C 编程成为一次布满陷阱的冒险，我的这些师弟师妹们也和我当初一样成为这些陷阱的受害者。如何帮助他们尽量避免或者少走当初我自己所遇到的弯路，使他们尽快地成为合格的程序员？事实上，他们作为初学者在比较大型的嵌入式软件系统开发过程中所遇到的这些问题具有一定的普遍性，至少我在刚刚接触的时候所遇到的挫折与麻烦与他们现在所遇到的问题是惊人的相似。这个想法最终在 2005 年底的时候变成了积极的行动，东南大学集成电路学院院长时龙兴教授(时老师同时也是国家专用集成电路系统工程技术研究中心的主任和我攻读硕士、博士期间的指导老师)建议我在学院开设关于嵌入式系统高级 C 语言编程的硕士选修课程，以帮助硕士研究生弥补他们所学习的 C 语言知识与实际工程研发过程中所需要的能力之间的差距。在本书成稿的时候，“嵌入式系统高级 C 语言编程”这门选修课已成功开设了 5 届，本书也是在该课程讲义的基础上做了进一步的完善和补充。

关于本书

首先，本书不是 C 语言的入门教材。在本书写作之初，我假设读者已经具备了初步的 C 语法知识，并且至少有过使用 C 编写程序的经验。但从教材的完整性和系统性上考虑，本书在第 2 章和第 3 章给出了关于 C 语法提纲式的复习与总结。现在市面上有关 C 语言的基础教材比比皆是，如果你对 C 还不是非常熟悉可以参阅第 1 章中给出的 C 语言基础教材。读者也可以利用附录 C 所提供的一份测试样卷测试一下自己对 C 语言的掌握程度，以便有针对性地阅读本书的相关章节。

其次，本书也不是专门介绍C编程技巧的。本书将主要介绍针对嵌入式系统基于C语言的软件项目的开发流程，以及较为复杂的C语言编程知识与技巧、编程风格和调试习惯，并通过对一个具体软件模块(ASIX Window GUI)的分析，介绍分析代码的方法以及设计软件系统需要考虑的各要素。本书将以实际项目中的代码作为实例来进行介绍，详细分析在嵌入式系统开发中程序员应该注意的方法、技巧和陷阱。我试图将本书编写成为一本适合作为高校相关专业高年级本科生和硕士研究生教学使用的教材，而不仅仅是一本参考书(虽然，显而易见地，我希望本书同样也能适合工程师的需要)。因此，在本书的写作过程中我尽可能地保证相关知识的系统性，同时我也注意适合教学的范例代码和课后习题的编写。

最后，虽然嵌入式系统的C语言编程在很大程度上与PC编程甚至UNIX或者Linux服务器编程没有太本质的区别，它们都需要遵循基本的软件编程思想和编程规范，但是嵌入式系统还是有其自身特点的，比如嵌入式软件的开发环境一般而言都比PC编程的开发环境要复杂得多，初学者在刚刚接触交叉编译的开发环境、仿真器、目标系统的时候往往会不知所措并充满挫折感，其实这些都是因为没有真正理解嵌入式软件开发环境的基本原理而造成的。因此，第4章专门介绍了嵌入式软件开发的基本流程和工具链，以及这些流程工具所起到的作用。另外，虽然嵌入式系统开发的概念远不仅局限于基于ARM处理器的嵌入式软件开发，但是由于ARM在消费类电子领域取得的巨大成功以及32位处理器在传统嵌入式系统的广泛采用，本书在涉及具体CPU或者具体系统设计的时候往往以东南大学国家专用集成电路系统工程技术研究中心研发的SEP3203和SEP4020嵌入式微处理器(以ARM7TDMI为内核)为例进行介绍。

基于本书的课程安排

正如我在前面所介绍的，本书的主要内容来自于东南大学集成电路学院“嵌入式系统高级C语言编程”这门硕士选修课程。作为SOC与嵌入式系统专业方向课程体系的一部分，“嵌入式系统高级C语言编程”这门课程旨在帮助学生掌握针对嵌入式系统的基于C语言的软件项目开发流程，掌握较为复杂的C语言编程知识和技巧，培养良好的编程风格和调试习惯，并通过对一个具体的软件模块(ASIX Window GUI)的分析，使学生掌握分析代码的方法以及设计软件系统需要考虑的各要素。本课程一共36学时，其中课堂授课部分为33学时，学生实验和课外作业与讨论折合为3学时(实际我希望学生在课外所花的时间远多于这个数量)。下面是我们在东南大学集成电路学院的课程安排(未与本书各章节内容严格对应，仅供参考)：

第1讲	概论	3学时
第2讲	C语言基本语法复习	6学时
第3讲	编译、汇编、链接与调试	3学时
第4讲	存储器与指针	6学时
第5讲	中断与设备驱动	6学时
第6讲	编码风格	3学时

第7讲　代码的调试	3学时
第8讲　ASIX Win GUI设计详解一	3学时（可选）
第9讲　ASIX Win GUI设计详解二	3学时（可选）
课程项目：ASIX Windows的控件开发	未定（可选）

关于实验与第8、9讲的设计案例分析以及最后的课程项目，教师可以根据所从事科研项目的具体情况和学生的接受能力，选择适合本校的教学内容。**本课程的多媒体课件和课程中使用的部分教学代码，读者可以到http://www.armfans.net论坛下载。**为了大家讨论的方便，我们在这个网站还将专门开设关于嵌入式C的讨论区。

致　谢

这本书能够诞生，首先要感谢东南大学国家专用集成电路系统工程技术研究中心主任时龙兴教授和副主任胡晨教授。如果没有时老师的鼓励和鞭策，我很难在东南大学集成电路学院开设“嵌入式系统高级C语言编程”这门课程，当然也就不会有这本书。胡晨老师是我进入嵌入式系统C编程的启蒙老师，在我攻读硕士学位以及以后的工作过程中，胡老师对我的指导使我受益匪浅，是胡老师使我真正认识到软件作为一个系统的概念，以及软件中的架构分层、封装与接口设计的重要性。

另外，我要感谢曾经在东南大学国家专用集成电路系统工程技术研究中心与我共事的研究生同学，在与他们的项目合作中使我逐步积累了本书中所写的心得与体会。他们是郑凯东、浦汉来、张宇、戚隆宁、金晶等。

北京航空航天大学出版社的胡晓柏老师的鼓励和支持使得本书最终得以出版，在此表示由衷的感谢。

东南大学集成电路学院的研究生张阳、徐继新以及南京邮电大学的冯海东同学参与了文字的校稿和相关材料收集与整理工作；东南大学国家ASIC工程中心的张黎明同学和史先强同学分别编写了本书6.4节和6.5节的内容。在本书的写作过程中，这些同学还给出了很多中肯的意见，在此表示特别的感谢！

书中引用了部分来自互联网的文章，博客的内容，在此对这些文章和博客的作者表示感谢。

最后感谢我的妻子与儿子，是他们给了我本书写作过程中的鼓励与支持！

限于作者的水平，书中错误和不妥之处敬请各位读者批评指正，并提出宝贵意见。读者也可以通过E-mail与我联系交流：trio@seu.edu.cn。

作　者

2010年9月

目 录

第1章 概 述

1.1 C语言的历史和特点

在C语言诞生以前，系统软件主要是用汇编语言编写的。由于汇编语言程序依赖于计算机硬件，其可读性和可移植性都很差，但一般的高级语言又难以实现对计算机硬件的直接操作（这正是汇编语言的优势），于是人们盼望有一种兼有汇编语言和高级语言特性的新语言出现。

具有讽刺意味的是，C的诞生是从失败开始的。1969年由通用电气、麻省理工、贝尔实验室联合研制的Multics操作系统几乎彻底失败，该操作系统实在是太庞大、太复杂了，以至于超出了开发团队的控制程度。从Multics项目撤出后，贝尔实验室的工程师Ken Thompson和Dennis Ritchie开始利用业余时间将Thompson写的一个小游戏“太空旅行”移植到PDP-7小型机上，这个小游戏模拟了太阳系的行星系统，游戏者可以驾驶飞船降落在某个行星上；与此同时，Thompson还为PDP-7小型机设计了一个比Multics更简单也更轻量级的操作系统，1970年Brian Kernighan模仿Multics的名字将这个新操作系统戏称为“UNIX”（Multi换成了Uni，以示这个新操作系统较之原来要简单、单纯得多）。与早期的操作系统一样，最早的UNIX采用的PDP-7是用汇编语言编写的，但是汇编语言在处理复杂数据结构时难以编码，同时也难以调试和理解。Thompson希望能够采用高级语言来编写，在尝试FORTAN失败后，他将一种研究性的高级语言BCPL（Basic Combined Programming Language，是由伦敦大学和剑桥大学合作研发的早期高级语言）简化为一种他称之为“B”的高级语言，以使得B语言的解释器能够运行在PDP-7 8K的存储器中。然而由于硬件资源的限制而采用的解释执行使B的效率不高，因此B语言并不适合作为UNIX系统的编程语言，以至于Thompson在1970年将UNIX移植到PDP-11小型机的时候依然采用了汇编语言。Dennis Ritchie利用PDP-11更强大的硬件功能创立了“New B”语言。这种新的语言支持多种数据类型，同时因为采用编译的运行方式而提高了性能，很快人们将“New B”称为“C”语言。

经过几年的演变和完善，到了20世纪70年代中期C语言已经和今天我们使用的C语言相差无几了，虽然后续的完善一直持续不断（比如增加了新的关键字unsigned和long等）。

1978年，Steve Johnson编写了PCC编译器(Portable C Compiler，可移植的C编译器)。由于这个编译器的源码可以在贝尔实验室之外公开，故该编译器被广泛地移植到不同的处理器上，成为当时C编译器的共同基石。同年，C语言的经典著作《C编程语言》("The C Programming Language")出版，为了表示对该书两位作者Brain Kernighan和Dennis Ritchie的敬意，书中的C版本被称为"K&R C"(出版社当时估计能卖掉1 000本就不错了，然而截至1994年这本书一共卖了150万册)。

到20世纪80年代早期，C语言已经被业界广泛采用了，但是随之而来的是多种不同的实现和版本。比如为了适应80X86的特殊地址架构，微软公司的C语言版本增加了一些新的关键字(如far、near等)。随着越来越多的非pcc基础的C语言版本出现，C语言逐渐形成了类似于BASIC语言一样的松散语言家族。1983年，美国国家标准化协会(ANSI)根据C语言问世以来各种版本对C语言的发展和扩充制定了ANSI C标准，1989年12月再次做了修订，并最终确认了该标准。国际标准化组织(ISO)随后接受了该标准作为国际标准。ANSI C标准有4个主要部分，分别是第4部分"简介"、第5部分"环境"、第6部分"C语言"、第7部分"C运行库"。该标准还有几个有用的附件，比如附件F"一般警告信息"、附件G"可移植性问题"等。

需要说明的是，虽然ANSI C标准规范了C语言的实现，但是在实际情况中，各家C语言提供商都会根据各平台的不同情况对ANSI C进行一定的扩展，比如我们上面提到的微软的C语言实现中增加了关键字far、near；又比如在嵌入式领域ARM的C编译器增加了关键字long long以支持64位整数，增加了关键字_irq以支持C语言编写的中断处理程序(注意：在有些编译器中有类似的关键字#intrrupt)。如图1-1所示，我们可以将现实中的C语言实现看作是ANSI C的一个超集，这些厂商对ANSI C的扩展部分有可能彼此不兼容，从而使得C程序的移植需要对这些非标准的部分特别小心。在这个问题上比较有代表性的例子是Linux的gcc编译器。由于该编译器对ANSI C进行了非常多的扩展，Linux的内核源码基本上只能在gcc上进行编译，希望通过其他C编译器编译Linux内核几乎是不可能的。另外一个需要注意的问题是，虽然ANSI C对C语言的规范进行了非常详细的约定，但是由于C语言的实现平台纵跨了从8位单片机CPU到32位甚至64位CPU的硬件环境，因此在数据类型的约定上标准C必须有足够的灵活性。比如ANSI C只规定了char数据类型是一个8位的数据，但是并没有规定int、short、long类型应该是多少位。这就造成了不同C编译器对于这些数据类型的不同约定，比如Borland公司的Turbo C规定int类型是16位整数；但是ARM的编译器规定int类型是32位整数，Freescale的68000编译器关于int、short、long类型的数据宽度是可以配置的。因此，嵌入式软件程序员在编写C代码时或者从其他处理器平台移植C代码时必须非常谨慎地处理这些与编译器相关的内容。

C语言的特点主要有以下几点：

① 语言简洁、紧凑，使用方便、灵活。C只有32个关键字，9种控制语句。较之其他高级语言，C语言的关键字非常少，一方面是语言本身的设计使然，另一个重要的原因是因为C语言将所有与外围硬件设备相关的输入/输出操作统统放在C运行库中实现，比如从键盘输入、向屏幕输出、文件的操作等都没有作为C语言关键字出现，而是以库函数的方法加以实现。这样做的好处一方面使得语言的实现变得比较简洁(编译器的实现也会比较简单)，另一方面由于与硬件设备相关的功能以函数的方法实现，使得C语言本身尽可能与硬件平台无关，这也是C语言能够在如此众多的硬件平台上实现的重要原因。

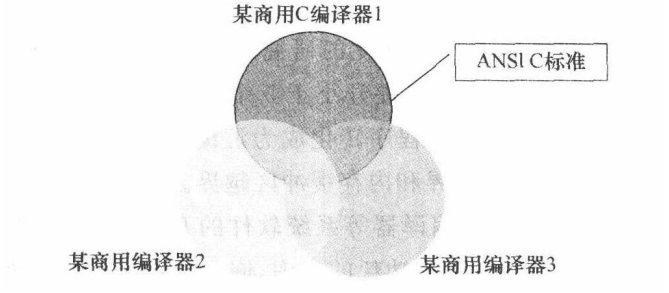


图1-1 ANSI C与商用C编译器的关系

② 运算符很丰富，C语言一共有34种运算符，但关键字只有32个。C语言中包含了一些特有的运算符。比如：自增自减运算符++和--；针对指针运算的取内容运算符*和取地址运算符&；针对位运算的移位运算符<<和>>；按位与&、按位或|、按位异或^、按位取反~；等。这些运算符大大方便了程序员在进行底层代码编写的过程中对存储器、控制寄存器等硬件资源进行操作。

③ 数据结构丰富，C语言的数据类型支持整型、实型、字型符、数组、指针、结构(struct)、共用体(union)、枚举类型(enum)。与其他高级语言不太一样的是，C语言没有字符串类型。这也是我个人认为C语言在处理字符串问题时比较不方便的原因。事实上，在很多需要对字符串进行处理的应用中(比如脚本的解释程序，像早期Web应用中的CGI脚本)往往更多地采用非常适合字符串处理的Perl语言进行编写，而不经常采用C语言。

④ 具有结构化的控制语句，在C语言中支持if…else、while、do…while、switch…case、for这些结构化的控制语句，我们后面会专门讨论这些控制语句。虽然C语言和绝大多数高级语言一样保留了goto关键字，而且C的语言结构也没有PASCAL那样严格和规范，但是总的来说C语言依然是非常好的结构化编程语言。

⑤ C语言的语法限制不太严格，程序设计自由度大。这是一个双刃剑。C语法非常宽容。比如C语言里面不检查数组越界，它是C语言里面很重要的一个特点，虽然这看起来是一个不好的特点，但是在一些优秀的程序员手中，这个特点也可以变成一个非常灵活的、并且

富有技巧性的方法。所以虽然说C很危险、很灵活，但是在高手手里面这些都是可以利用的，也就是留给程序员的空间非常大。C语言就像是一种非常厉害的兵器，比如流星锤，他要求玩这个兵器的人要很厉害，但如果一个新手玩，就很可能被流星锤打中自己。

⑥ C语言允许直接访问物理地址，能进行位(bit)操作，可以直接对硬件操作。这是C语言非常重要的一个特点。C对物理地址的访问主要是通过指针，而指针又是C里面最灵活的部分，也是初学者最难掌握的内容。但是如果没有指针的话，实际上也就意味着C不能够访问硬件或者说访问硬件会变得很困难，那么对内存的操作也就变得很困难。

⑦ 生成目标代码质量高，程序执行效率高。相对于其他高级语言，C的编译器效率可能是最高的。这一方面是因为对C编译器优化的研究已经达到了非常成熟的程度，这一点从ARM公司的C编译器在性能上每年仅仅提升小于5个百分点可以得到印证；另一方面是因为C语言本身的语言特性，使得在将C程序转化成为汇编代码时，需要额外增加的检查代码要少得多。比如C语言不检查数组越界和内存缓冲区越界。也正因为C目标代码的高效性，使得C语言非常适合诸如操作系统、编译器等系统软件的开发，同时也使C语言成为嵌入式软件开发的首选高级语言(基于对成本和功耗的考虑，嵌入式系统的硬件性能往往受到严格的限制)。

⑧ 可移植性好。理论上讲，任何一个高级语言都应该具有很好的可移植性，但是实际的情况却不尽如人意，这是因为各个厂商推出的编译器往往会扩展一些自己的特性。C语言的可移植性是比较好的，从巨型机到单片机都可以使用C语言。这主要有两个原因：第一，C语言在20世纪80年代就制定了相关的标准(ANSI C标准)，因此虽然各家编译器厂商推出的C语言各不相同，但是都保证与ANSI C兼容；第二，也是往往被大家忽略的原因，就是C语言本身的标准中并没有设计输入/输出的操作，C的关键字中没有与计算机系统相关的输入/输出功能，所有的这些功能都是由C运行库中的库函数完成的。从这个意义上来说，C语言本身是和硬件无关的。当然，C的可移植性是相对的，实际的工程项目中移植依然是一个不容小视的问题。

C语言在1997～2009年之间都是嵌入式软件开发使用最多的语言。近五年来，C与C++语言更瓜分了大部分原属于汇编语言的版图。其中较高阶的C++发展速度虽不如预期，但仍在嵌入式软件设计领域维持27%左右的占有率。整体看来，C++语言使用率在20世纪90年代晚期加速上升，在2001年达到高峰，然后稍微下滑，之后维持稳定。无论如何，嵌入式软件设计师不会在短时间内放弃使用C语言，原因有很多个：首先，C语言编译器支持大多数的8位、16位与32位CPU；其次，C语言在处理器与驱动程序层级，兼具高低级语言的特色。请看图1-2所示的关于嵌入式软件工程师所使用编程语言的调查数据(来自TechInsights 2009年嵌入式系统市场研究)。