C

# C语言

成都电讯工程学院

李智渊等 编译

四川科学技术出版社

责任编辑：罗孝昌　王仕德

版面设计：罗孝昌

# C 语 言

出版：四川科学技术出版社
印刷：成都电讯工程学院出版社印刷厂
发行：四川省新华书店

开本：787×1092毫米 1/32
印张：10.75　　字数：240千
插页：1　　印数：1—5,000册
版次：1985年9月第一版
印次：1985年9月第一次印刷

书号：15298·137
定价：2.25元

# 编 者 的 话

在对成都电讯工程学院的研究生开“C语言”一课时，原来是直接使用英语教材。后来因关心C语言的人越来越多，就有了把教材和有关资料编译出来为更多的读者服务的想法。当然，对更多的读者来说有一本中文版会更方便一些。

C语言被人称为是“近十年来对计算机程序设计实践的最重要的贡献之一”，近年来它颇为广大用户所赏识。它是一种功能很强而又比较简单的通用程序设计语言；它具有高级程序设计语言的许多特点，而又能作比较低级的汇编语言所能作的许多事。因此，它特别适宜用来编写各种软件。有名的UNIX系统就是用C语言编写的。C语言和UNIX的结合相当完美，相辅相成。如果你想使用UNIX，有一个C语言编译程序是必不可少的，你可能也不得不与C语言打交道。因此，学会C语言的程序设计技术也就很有意义了。随着UNIX的日益普及，许多过去使用COBOL、FORTRAN和其它语言的用户正在把他们的注意力转向C语言。

对用户来说，它有与汇编语言相类似的许多功能却又易于阅读，用起来非常灵活且结构严谨，所以有助于（在某种意义上甚至是迫使）用户编写出好的程序结构。C语言在执行速度快方面几乎可以同汇编语言媲美。它具有丰富的操作种类和各种各样的数据类型。由于它可移植性好，所占空间小，对目前日益增多的微型机用户来说更是一个佳音。实际上，目前许多微型计算机系统（如国内优选机型IBM PC等）都已配有C语言的编译程序。

本书的大部分内容取材于B. W. Kernighan和D. M. Rit-

chie 所著的“The C Programming Language”一书。由于该书作者对C语言和UNIX有重大的贡献和具有丰富的实践经验，使它成为一本了解C语言的权威著作。虽然该书的写法尚有值得改进之处，但其丰富的内容却给我们提供了极好的资料。在这一背景下，如果需要，今后我们还可以在如何改进方面作些文章。所以我们在对几种英文原版进行比较之后还是决定选它作为编译的基础。

首先，我们在假期里组织了几位研究生对该书大部内容进行了翻译，然后由我院计算机系李智渊老师对全书作了编译和审校、定稿工作。除纳入了该书的全部内容外，还编入了一个第九章和几个附录（B、C、D、E）。在它们的帮助下，纸面上的程序就容易变得“现实”起来。至于附录中关于如何使用UNIX操作系统命令、编辑程序等内容的介绍，也可供关心和使用它们的读者参考。总之，这些新加进去的内容，给本书增加了若干“实用性”；有了本书之后，基本上不需要参考别的材料，就可以在配有UNIX的系统上运行你的C语言程序了。这一点对那些想通过自学来掌握C语言的读者不无好处。对这些读者来说，通过阅读这些材料，再上机实践，就可以一步一步地深入学下去了。

本书已在成都电讯工程学院内部铅印出版一次。今又由李智渊同志对本书作了进一步的修订和增补。参加本书编译工作的有刘建军（第1、2章）、唐俊（第3章和附录A）、吉鸿宾（第4、5章）、胡健（第6、7章）和李智渊（第8、9章、附录B、C、D、E和全书的审校定稿）等几位同志。由于水平所限，成文仓促，错误缺点在所难免，敬请批评指正。

编　者

1984.11

# 目　录

# 导　论

C是一种通用的程序设计语言。由于这种语言是在UNIX操作系统上开发的，并且，UNIX和其软件也是用这种语言写的，所以它同UNIX联系紧密。但是，它并不受任何一种操作系统或机器的束缚。同时，尽管本语言因其在书写操作系统时好用而被叫做“系统程序设计语言”，用它编写大型的数值计算、行文处理及数据库程序也同样方便。

C是一种相对“低级”的高级语言。这样说并非轻视它，这仅是指它能处理绝大多数计算机所处理的“低级”对象，比方字符、数字或者地址而言的。可用实际机器上所实现的普通的算术和逻辑运算对这些对象进行组合和修改。实际上，这不但不是缺点，反而是C语言的一种优点。

C不提供直接处理复合对象（如作为一个整体看待的字符串、集合、表或数组等）的操作。比方它就没有象PL/I中那样的处理完整的数组的操作。本语言只有静态定义和由函数局部变量所提供的堆栈规则两种存储分配手段。没有定义任何其它存储分配手段。比如，就没有Algol 68那样的收集无用信息的手段。最后，C本身并不提供输入/输出手段：没有READ和WRITE语句。没有固定编排好的文件存取方法。所有这些高级操作是用显式调用函数来实现的。

同样，C仅提供了直接单线的控制流结构：测试，循环，分组及子程序。而无多道程序设计，平行操作，同步机构或并行子程序。

尽管本语言缺少某些这样的特色而看起来似乎是个严重的缺陷，可是，让本语言保持恰如其分的规模却使人获益非浅。由于C相对较小，就能花较少时间来描述它，人们学起来也快。C语言的编译程序可以做得精练紧凑，也容易编写。用目前的技术，可望花上几个月时间就能为一台新机器配上编译程序，并且我们可以发现编译程序中80%的代码和现有的编译程序是相同的。这就提供了高度的语言可移植性。因为大多数现有的计算机直接支持C语言的数据类型和控制结构，故为实现程序封闭所需的运行库就相当小。例如，在PDP-11计算机上，它仅包含有做32位乘和除的例行程序和实现子程序出入序列的例行程序。当然，每一实现都提供了易理解的，兼容的函数程序库以执行I/O、字符串加工、存储分配操作。但由于这些只能显式调用，所以如果必须的话也可避免之，它们也可以用C本身来编写。

又因本语言反映了当前计算机的能力，所以C程序有足够的效率，因而就没有必要非要用汇编语言来代替它编写程序不可。最好的例子莫过于UNIX操作系统本身，它差不多全是用C写的。13000行的系统编码中，仅在最低层上有800行左右是用汇编语言写的。另外，基本上所有UNIX应用软件也是用C写的，大部分UNIX用户甚至不懂PDP-11汇编语言。

尽管C同许多计算机的能力相适应，它却独立于任何特定的机器结构，所以，稍加注意就能写出“可移植的”程序。即在一系列不同的机器上运行，而程序不加任何变动。现在，我们在UNIX上开发的程序，已移植到这里的Honeywell、IBM和Interdata系统中去了。实际上，在这四种机器上的C语言编译程序和运行时的支持要比设想中的ANSI的FORTRAN

标准文本兼容性好得多。UNIX 操作系统本身既在 PDP-11 又在 Interdata 8/32 上运行。除象多少都有些依赖机器的编译程序，汇编程序和纠错程序之外，用 C 写的软件在两种机器上完全一样。在操作系统内部，在除汇编语言支持和 I/O 设备处理程序之外的 7000 行代码中大约有 95% 完全一样。

对于那些熟悉其它语言的程序员来说，我们提及一些有关 C 的历史，技术和设计观点，以资对照比较，是有好处的。

C 的许多最重要的观点萌芽于语言 BCPL。该语言由 Matim Bichard 提出，比较老，但却还有生命力。BCPL 对 C 的影响是间接地通过 B 语言进行的。Ken Thompson 在 1970 年用 B 编写了第一个在 PDP-7 上运行的 UNIX 系统。

尽管 C 分享了 BCPL 的几处特色，但它并不是 BCPL 的一种方言。BCPL 和 B 是“无类型”语言：仅有的数据类型就是机器字，要访问其它类型的数据，就得用特殊的操作符或者函数调用。而 C 中，基本数据对象是字符，几种长度的整数及浮点数。另外，用指针，数组，结构，联合及函数构造了一个分层结构的推导数据类型。

C 提供了良好结构程序所需的基本流程控制结构：语句组合；判定(if)；终止测试在顶部的循环(while, for)，或在底部的循环(do)；在一系列可能情况中选择(switch)。(所有这些 BCPL 中也有，但语法有些不同；该语言加入到风行的结构程序设计中已有若干年了。)

C 提供了指针，并有做地址运算的能力。函数参量的传递是用抄写参量值的方式，这使得被调用的函数不可能改变调用程序的实际参数值。当想要达到“传地址”的效果时，可显式地传递指针，被调用函数此时就能改变指针所指对象了。数组名是用传递其起点单元的方法来实现的，所以，数组参

量传递是用的传地址的方法。

所有函数都可被递归调用，其局部变量是典型"自动的"或者每次引用时重新产生的。函数定义不能嵌套而变量说明可采用块结构方式。C程序的函数可以单独地编译。对一个函数来说，变量可以是内部量，也可以是一个单独源文件内已知的外部量，或者是完全全局变量。内部变量可以是自动的或者静态的。自动变量可放入寄存器内以便提高效率，但寄存器说明仅是对于编译程序的提示，并非是指特定的机器寄存器。

按 Pascal 或 Algol 68 的概念，C 并不是一个强类型的语言。尽管它不象在 PL/I 中那样任意地自动转换数据类型，但数据转换在C中还是比较随意的。现有的编译程序没有提供运行时的数组下标、参量类型等的检查功能。

在希望有强类型检查的地方，要用独立的编译程序版本。该程序叫做 lint，显然是由于它能找出程序中的错误。lint 不产生代码，而提供尽可能多的编译时和装入时的严格检验。它发现类型不符，不一致的参量用法，没有用过的和显然未经初始化的变量，以及潜在的移植性困难等等。除了极少数的例外，凡是顺利通过了 lint 的程序，就可能象 Algol 68 程序那样完全地免于类型错误。我们将在后面需要时，再提及 lint 的其它功能。

最后，C就象其它语言一样，也有自己的不足之处。一些运算符有错误的优先权；句法的一些部分可能不是很好；有几种互相有点不同的语言文本。然而，就很多种不同的程序设计应用来说，已证明C是一特别有效，表达能力很强的语言。

第一章对C的中心部分作了个人指导性介绍，目的是让

读者尽快开始写程序，因为我们确信学习一种新语言唯一的方法是用它编程序。假定读者有程序设计的一般知识，就不再对什么叫计算机、编译、表达式 n = n + 1 是何意思之类的问题再作解释了。尽管我们将在可能的地方出示若干有用的程序设计技巧，但本书并不打算成为一本数据结构和算法方面的参考书；在必须选择的情况下，我们都把注意力集中于语言本身之上。

第二章至第六章更详细地讨论了C的各个不同方面，并且用比第一章更形式化的方法，重点仍是完整的有用的程序实例而不是孤立的片断。第二章讲解了基本数据类型、算符及表达式。第三章处理流程控制：if-else、while、for 等。第四章则是函数和程序结构——外部变量，作用域规则等等。第五章讨论指针及地址算术运算。第六章包括结构和联合的细节。

第七章描述了标准C I/O程序库，这提供了一个到操作系统的公共的接口。在所有支持C的机器上都有这种I/O程序库，所以用它来作输入、输出及其它系统功能的程序基本不作变更就可以从一个系统移到另一个系统上去。

第八章描述C和UNIX操作系统之间的接口，集中谈输入/输出，文件系统及可移植性。尽管本章有一些特定于UNIX的内容，但不打算使用UNIX的程序员也仍能从中找到有用资料，包括象一种文本的标准程序库如何实现，及取得关于可移植编码的建议。

第九章对用C语言编写程序时常见的错误和如何排除这些错误作了比较全面的介绍。其内容对读者编写自己的程序大有益处。

附录A包含C参考手册，这是C的语法和语义的"正式"

说明（除了自己的编译程序外）以及先前各章的含糊及省略之处的最后公断。

由于C是在一系列系统中不断演变的语言，某一特定系统的最新发展可能同本书的一些内容不一致，我们也努力去澄清过这样的问题，提请对可能产生的问题和困难的注意。然而，有疑问时，我们一般选择描述 PDP-11 UNIX 的情形，因为那是大多数C程序员的工作环境。附录A也描述在各主要的C系统上付诸实施时的差异。

此外，为了方便读者在计算机系统上实地运行自己的C语言程序，我们还在本书的末尾加上了附录B、C、D、E。

# 第一章　C语言的概貌

这里对C作快捷的介绍，目的是用真实程序来展示C的基本内容，而不陷入细节、形式化规则及种种例外中去。为此，先不追求完整或精练（仅保证例子正确）。为尽快编出有用的程序，就得集中全力到基本点上：变量和常量，算术运算，流程控制，函数及基本的输入和输出。经过考虑，在本章中略去了C的一些特点，而这些在书写大一些的程序时特别重要，包括指针、结构、C丰富的算符集合、几个流程控制语句及许多细节。当然，这种方法有其缺陷。最令人注目的就是语言特点的完整说明不能在一个地方找到。同时，由于本章的简略，也可能使人产生误解。还有就是所举的例子都没有充分发挥C的全部威力，从而也就显得不象它们本来应该有的那样简明精练。我们将努力减小这些影响，但先告诫读者一声。

另一个缺陷就是后来的各章会不得不重新提及本章的一些内容。我们希望读者能从内容的重复中得益，而不是感到烦燥。

无论如何，有经验的程序设计人员会从本章中找到他们所需要的东西。而初学者则应通过亲自编写类似的小程序来补充巩固。这两类人都可用这一章来作为框架子，在此之上不断增添那些从第二章开始的更加细化了的内容。

## 1.1 如何着手

学习一种新的程序设计语言的唯一方法是用它编写程序。要编的第一个程序如下（几乎所有语言都是这样）：

打印出下列的词

```
hello, world
```

这是基本的障碍，要越过它就得有能力在某处写出一个程序正文，成功的编译，装入运行，得到其输出结果。精通了这些技巧，其余的就相对简单了。

在C中，要打印“hello, world”的程序是

```
main( )
{
    printf("hello, world\n");
}
```

如何运行这个程序要视所用的系统而定。作为特例，在UNIX操作系统中，必须在以“.c”作为其名字结尾的文件（如hello.c）上创建源程序，然后用以下命令编译它：

```
cc hello.c
```

如果没有任何出错，比方没有漏掉字符和拼写出错，编译就静静地进行，并产生一个叫做a.out的可执行文件。再用以下命令运行之

```
a.out
```

这就产生输出：

```
hello, world
```

在其它系统上，规则将会有所不同；可请有关专家加以指导。

**练习1-1** 在你的系统上运行以上程序。有意去掉程序