近十年来对计算机程序设计
实践的最重要的贡献之一

C 语言

·修订本·

李智渊 编著

成都电讯工程学院出版社

C 语言

·修订本·

李智洲 编著

成都电讯工程学院出版社

·1988·

C 语 言

·修订本·

李智湖 编著

成都电讯工程学院出版社出版
四川省石油局印刷厂印刷
四川省新华书店经销

开本 787×1092 1/16 印张 16.25 字数 384千字
版次 1988年6月第二版 印次 1988年6月第一次印刷
印数 1—10000册
中国标准书 ISBN 7—81016—035—4/TP.5
(15452.38) 定价：4.00元

编者的话

自《C语言》一书1985年出版以来，深受广大读者的欢迎，后来竟出现供不应求的局面。根据广大读者的建议和要求，以及我们自己的教学实践，现重新编写了这一新的版本。本书在结构和内容上都有较大的变动，文字上也作了许多修改。其主要的特点为：

（1）保持了原书由浅入深，深度和广度兼顾的特点，从而宜于作为教材使用；

（2）增强了可读性。特别是对不太好懂，或比较重要，或易于混淆的地方，作了深入浅出的解释，便于读者自学；

（3）本书基本上自成系统，读者学习时，几乎不需要参考别的资料，就可以在UNIX系统上运行自己的C语言程序；

（4）增加了若干原来没有的新内容，使本书的内容更为丰富。

C语言被人们称为是“近十年来对计算机程序设计实践的最重要的贡献之一”，近些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；它兼有高级程序设计语言特点的同时，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。现在，许多软件都是用C语言编写的。例如，有名的UNIX系统就是用C语言编写的，当前微型机上的许多软件，如dBASEⅢ等也是用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和其他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）用户编写出良好的程序结构。C语言在执行速度快方面几乎可以同汇编语言媲美。它具有丰富的的操作种类和各种各样的数据类型，功能强。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，目前许多微型机系统（如国内优选机型IBM PC等）都已配有多种C语言的编译程序。

值得指出的是，在1985年版本正式出版之前，曾经在成都电讯工程学院作为内部资料印过一次。当时的研究生刘建军、唐俊、吉鸿宾、胡建等同学曾分别参加了部分章节的工作，这里向他们致以衷心的感谢。

水平所限，错误缺点在所难免，敬请批评指正。来信请寄成都电讯工程学院微型计算机研究所编者收。

编　者

目　录

第一章　C语言及其简史

C语言是一种相当新的程序设计语言。它是1972年左右才在美国的贝尔实验室(Bell Laboratories)开发出来的。因为它的功能强，所占空间小，表达简捷，且具通用程序设计语言的特性，故在程序开发人员中，C语言被广泛认为是一种十分有效、实用的软件开发工具语言。

什么是C语言呢？C语言的许多用户把它称作是“汇编语言的速记形式”。因汇编语言程序编写起来较慢，而C语言有高级语言的特点。编起来快得多，另外C语言还具有汇编语言所具有的代码紧凑，和执行速度非常快的特点，的确，在大家所关心的程序设计语言范围内，C语言确有其奇特之处：既未被归到象Pascal、Fortran、COBOL、以及BASIC等那样的高级语言一类，也未被分到汇编语言那样的低级语言一类。C语言常被称之为“低级语言的高级形式”，或者被叫作“高级语言的低级形式”。这就是说，C语言是介于高级语言和低级语言之间的一种语言。不难想象，C语言兼有高级语言和低级语言的许多优点，这也是它的极为独特之处。当我们说C语言是一种相对“低级”的高级语言的时候，并没有轻视它的意思，这仅仅是指它能处理大多数计算机所能处理的“低级”对象，比方字位、字符、数字或者地址等而言的。因为它能比较深入地处理一些细微的东西，为许多高级语言所不及，所以我们可以说，这不但不是一种缺点，反而是C语言的一种优点。

另一方面，由于它有高级语言的特点，因此用它来写各种应用程序也同样方便。

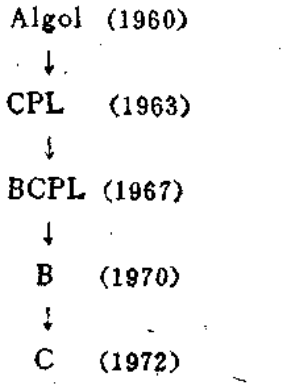
C语言是由贝尔实验室的Dennis Ritchie设计并编写出来的，D.Ritchie后来又和Ken Thompson一起做UNIX操作系统的工作。在开发UNIX的时候，就设想用它来为软件工作者提供一套工具和良好的工作环境。而C语言则是它们之中最基本的工具。几乎由UNIX提供的所有软件工具，包括操作系统和C语言编译程序本身，现在都是用C语言编写的。

来自伯克莱加洲大学(University of California Berkeley)的Ken Thompson被认为是世界上最出色的软件工作者之一，在工作中，他为B语言写了一个解释程序，这个语言与BCPL语言类似。后者是一个移植性颇好的语言，它只需要对它在其上工作的机器作相当简单的假设即可。结果，程序可以相当容易地，从一个机器，移植到另一个机器之上去。Ken Thompson很欣赏BCPL语言的一点是，它编写起程序来相当容易，于是他对BCPL语言的一部分进行优化，而提出了B语言。选B作为这个语言名字的理由，被认为它是BCPL语言的一个很小和很简单的部分，因此就把作为母语言BCPL的第一个字母B用来命名。

当Dennis Ritchie参加进来的时候，他对Thompson所写的解释程序非常感兴趣，结果他写了一个叫做C语言的编译程序。但是，选C作为名字，并不是因为在英文

字母表中，C是在B之后，而是因为它是BCPL 的第二个字母。

C语言的“家谱”如下图示：



Algol是由一国际委员会设计的，虽然它比Fortran的出现就晚那么几年，但它却比Fortran 更为精致和完善，因此，对程序设计语言的设计有较大的影响。Algol的设计者对语法的规则性，模块结构及大家倾向于认为的若干“现代”的特征，给予了很大的注意。不幸，在计算机用得最广的美国，Algol 却从来没有得到真正的理解和推广。其原因大概是Algol太抽象、太笼统了。CPL的目标，就是想把Algol带到现实中，和实际的计算机更好地联系起来。与Algol 相象，CPL也很大，在功能上有所加强，但难于学习和实现。CPL是Combined Programming Language（复合程序设计语言）的简称，它是由剑桥和伦敦大学于1963年推出的。1967年，剑桥的Martin Richards研制出了BCPL语言，BCPL的目标，是要把CPL语言加以浓缩，以得到C PL的较好的基本特性而又较易学习和实现。BCPL是Basic Combined Programming Language（基本复合程序设计语言）的简称。而1970年由Ken Thompson所写的B语言又是BCPL 的进一步简化。而Ritchie在C语上的成就，主要在于他通过巧妙地使用各种数据类型，而恢复了已经失去的通用性。

许多程序设计语言差不多都反映了作者本人从事的专业领域。Dennis Ritchie 的领域是系统软件（语言、操作系统、程序产生器等），因此C语言最长于用来实现这一类工作性的软件。为什么呢？有两个理由，但都与C语言本身的特性有关，首先，如前所述，它是一种相对低级的语言，因此容易用它来实现程序中的许多细节，以达到能最大地发挥计算机的效率，其次，由于它又是一种相对高级的语言，因此，程序员又不需要去关心与计算机本身有关的许多细节，从而可使编程的效率大为提高。

用简单的语言成份，来编写大型的复杂程序的能力，是C语言的一个很大的长处。

在七十年代中期，UNIX已在贝尔实验室普及，并推广到了许多大学。在没有做任何宣传鼓动的情况下，C语言就逐渐代替了在UNIX上运行的，人们更为熟悉的其他语言。变成了贝尔实验室的“官方”语言。其名声日振，用户日多。许多程序员都越来越喜欢C语言，而与一些老的语言如Fortran或PL/I，甚至相当新的语言Pascal和APL等疏远。这一情况，甚至使C语言的开发者Ritchie都感到十分惊异，真是“无心插柳，柳成荫”。1980年以前已有若干C编译程序在市场上推出，而且在非UNIX系统的机器上，也有一些C语言编译程序能够运行。甚至在单片机这样小的机器上，也出现了一些

C语言编译程序，根据1986年国外期刊指导，在8051系列的单片机上，已推出了一种叫做MICRO/C—51的C编译程序。目前，在各种机型，在各种操作系统上运行的C语言编译程序已不胜枚举。不过在本书中，以讨论在UNIX上运行的例子为主。

因为C语言具有高级语言的形式，所以学起来还是比较容易的。同其他高级语言的情形一样，计算机也不能直接处理C语言。计算机仅仅能执行机器指令。为了使用C语言编写的程序能够在机器上运行，需要把用C语言写的程序翻译成等效的机器语言。执行翻译工作的这一个软件，称之为编译程序。

当我们需要编一个程序来完成某件工作时，如果我们要使用C语言，首先就要用C语言来编写一个程序。这个程序被称之为源程序。编译程序的任务，就是接受这一源程序，并把它翻译成计算机能够理解和执行的机器指令。编译程序的输出称之为可执行代码或执行程序。换句话说，它是程序的另一种形式，可为计算机所直接执行。不同的机器可要求不同形式的C编译程序，这是因为它们的机器语言不同的缘故。对不同的机器来说，源程序总是（或基本上是）相同的，但可执行代码却随机器不同而异。

从源程序变到可执行程序，要经过若干个中间步骤。首先，程序员启动计算机，用编辑程序编写一个程序。然后把它存入一个文件中，这个文件称之为源程序文件。然后，通过在键盘上打入某个命令来启动编译过程，通常，在UNIX系统中，打入CC。这一命令触发了整个翻译过程，在每个阶段，执行翻译工作的软件接受用户编写的程序，或其某个中间形式，然后再把它变换成较为低层的形式。

编好的源程序首先交给预处理程序加以处理，它把源程序中的某些缩写形式加以扩展，其输出是经过扩展后的源程序。扩展后的源程序送给编译程序处理，其输出是变换成汇编语言形式的源程序。然后交汇编程序处理，汇编程序将它变换成浮动目标代码。目标代码是一种中间形式，它既不可以由程序员读，也不可以由计算机运行。为什么要这样做呢？因为所有的C语言程序，都必须和C语言程序库中的那些支持程序连接起来。连接程序执行这一任务，它把所有需要的程序连接起来，并把它翻译成可执行程序文件。之后，把这一文件送给装入程序就可以运行了。在UNIX中相当简单，通过打入文件即可（即以后要谈的a.out）。

从写一个程序到运行它，看起来是一个相当长的过程。幸好在实际中并不需要去考虑这个长过程中一个个的步骤。编译过程是隐藏起来了，至少在UNIX系统中是如此。我们仅仅需要打入CC命令，等上几秒钟，就可以得到一个可执行程序。这个可执行程序是否就能象我们原来设想的那样运行，则还不一定。如果不行，则我们就要找出程序中的问题，用编辑程序修改源程序，然后再重复上述的编译过程。

第二章　C语言的快速入门

为了让读者尽快对C语言的概貌有所了解，在本章我们将首先让读者看一看C语言的程序是什么样子，对一些符号和表现的方式有所了解，并且看一看一个C语言程序。如何在UNIX操作系统下（这是一种典型情况）编译和执行。

本章不打算让读者一开始就纠缠到许多细节里去，其原因，是我们希望读者能尽快编出自己的程序，那怕很小，也能取得一些实感。为此，我们就应该首先把注意力集中到一些基本的东西上，比如：常数和变量，算术运算，流程控制，函数及基本的输入输出等。至于其他的许多内容，如指针、结构，C语言中十分丰富的算符集合、某些流程控制语句及许多细节，这里暂时予以搁置不管。因为在本章我们介绍的只是C语言全部功能和特点的一个子集，因此，难于使读者一开始就体会到它的全部威力，用C语言编程的那种十分简练的特色也不能圆满地表现出来。如果出现了一些一时还不能完全得到解答的问题，请读者耐心，以后各章会给出比较满意的回答。

显然，这样做的结果会使以后各章再次提及本章的一些内容。但再次提及时并不是简单的重复，而是相应内容的加深和扩展。

通过本章的学习，掌握其他程序设计语言的读者会对C语言的特点和风格很快有所了解。而初学者通过本章的学习，再编写一些练习程序会发现，C语言不但很容易学，而且用C语言编程序在某种程度上是一种享受。总之，对各种不同的读者来说，本章的学习可以看成是全面深入学习C语言的一个框架。

§2.1　起　　步

当前，人们可以使用的程序设计语言多种多样，各有其针对性和特色，但学习和掌握这些语言的方法还是基本一致的。首先要学习这些语言本身，了解它们的语法规则。而且更重要的，是要用它来编写程序，只有通过大量地编写程序，才能做到对语言的真正理解和掌握。

要我们编写的第一个程序当然要尽可能简单，对各种语言，通常都差不多，首先是在荧光屏或打印机上，显示或打印出几个表示问候的字，例如：

```
打印出下列的词
hello, world
```

要输出这两个英文单词，我们首先应编写出一小段程序，然后对之进行编译。由于程序十分简单，所以编译必然成功。编译成功后，即可将之装入运行，进而得到所要的输出结果。我们所编出的第一个C语言程序如下：

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

有了程序，首先要进行编译。如何编译和运行一个程序，视所用的系统而定。由于C语言和UNIX操作系统的联系紧密，本书将以在UNIX系统下运行C语言程序作为主要考虑的对象。在UNIX操作系统的工作环境下，所编写的C语言源程序必须在文件后加上后缀“.c”，如果我们把这个程序要输出的关键词hello用在文件名中，则这个文件名可以是hello.c。在UNIX系统下，我们用“cc”命令来对这个文件进行编译：

```
cc hello.c
```

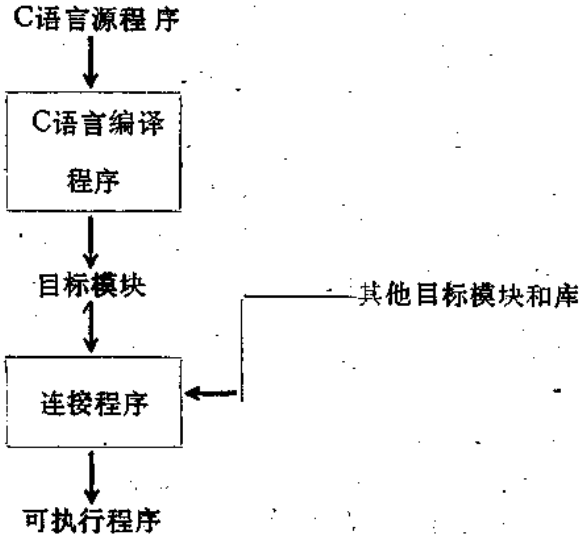
如果程序没有什么错误，编译就会顺利地进行，并产生一个叫做 a.out 的可执行文件，运行这个可执行文件，就会产生所要的输出结果。运行这个可执行文件很简单，只消在键盘上打入文件名a.out即可，打入

```
a.out
```

该文件即运行，并得到输出结果：

```
hello, world
```

如前一章所述，在UNIX系统中，编译过程是隐藏起来了，所以我们只要简单地使用“cc”命令即可。但在非UNIX的其他操作系统环境下，编译、连接、运行等规则又有所不同，甚至有很大的差别，一般化的过程如下图所示



编译程序把源程序变成“立即形式”的目标模块。和源程序一样，目标模块也不能直接运行，还要把它同C语言库中提供的支持程序和其他有关的目标模块连接起来才行。连接后就产生了可直接运行的可执行程序。学过BASIC、Pascal或其他某些语言的读者可能不熟悉连接程序，因为在这些语言中，通常不需要通过连接来产生可执行程序。不管是隐式的（如UNIX系统下）或是显式的进行连接。连接本身给C语言带来了

强大的威力，这是因为C语言强烈地依赖于C语言库提供的支持程序，这些支持程序提供了许多额外的资源，使我们可以很迅速地建立起自己的可执行程序。

由于操作系统支撑环境和各C语言版本的微小区别，在实地编制和运行自己的程序时，除了参考教科书外，特别要注意参阅自己工作环境下的有关手册。

现在对程序本身作些解释。一个C程序无论其大小，都是由一个或多个“函数”组成，而由“函数”决定要做的实际计算操作。C函数同Fortran程序中的函数和子程序及PL/I，Pascal等的过程是类似的。在上一例子中，main就是这样的函数。一般地说，无论给函数取什么名字都可以，但是main是一个特殊的名字——程序总是从main的开头执行。这就是说，任一程序必须在某处有个main。main一般将调用其他函数来完成它的工作。这些函数有的是源于同一程序本身，有的来源于先前写进程序库中的那些函数。

现在，我们用下图来对main（ ）函数加以进一步的说明，图中的圆点用来代表函数体中的诸语句。

```
main (  )          ←—— 函数名
{                  ←—— 函数开始
   ⋮               }←—— 函数体
{                  ←—— 函数结束
```

各函数之间，可用参数来交换数据。函数名后的一对括号内括着一个参数表。现在的这个main函数没有使用参数，用一对空的括号（ ）跟在main的后面。实际上，函数main（ ）也是可以带参数的，后面将会谈到。花括弧{ }用以包围构成函数的语句；有些象PL/I中的DO-END或者Pascal，Algol等中的begin-end。函数通过名字来调用，这个名字后面跟有一带括号的参数表。这里没有Fortran或PL/I中的CALL语句。即使没有参数，括号也不能少写。

下面这一行是一个函数调用

```
printf ("hello, world/n") ;
```

它调用一个名叫printf的函数，带有参数“hello, word/n”。printf是一个在终端上（除非指明了其他目的地）打印输出的库函数。在这里，它打印出作为其参量的字符串。

由双引号“…”所括的任何数目的字符序列叫做字符串或串常数。目前，仅用字符串作为printf或其他函数的参数。

串中的序列\n是C中用作换行字符的记号，当遇到它时，终端上的光标前进到下一行的左首端。如果漏掉了\n（这是一个值得做一做的试验），会发现输出结束时并不由换行。把换行符送入printf参量中去的唯一办法是加上\n；如果不信，可试试下面的程序

```
printf ("hello, world

");
```

C编译程序就会打印出报告引号丢失的诊断信息。

Printf决不会自动换行，所以可用多次调用以得到一个长的输出行，如是，第一个

程序也可写成：

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

其输出完全一样。

注意：\n仅表示一个单独的字符。象\n这样的转义序列提供了一种通用和扩展的手段，以表示不易获得或不可见的特殊字符。

这里，我们顺便讨论一下所谓转义序列，它是用来表示象“回车”，“退格”等特殊字符所最常用的一种方法。在C语言中，一个反斜线“\”后带一个字符表示转义序列。转义序列前的这个反斜线用来告之编译程序：后面跟的字符具有特殊的意义。C语言中定义的转义序列有：

\n	换行符 (newline)
\t	制表符 (tab)
\b	退格符 (backspace)
\r	回车符 (carriage return)
\f	换页符 (form feed)
\\	反斜线 (backslash)
\'	单引号 (single guote)
\o	空字符 (null)

按国际标准化组织（ISO）的定义，空字符是用来填充媒体或填充时间的一种控制字符，当空字符插入一个字符序列或从中去掉时并不影响该序列的含义。有时，空字符的含义与空格符（space）的意义相同。

要注意的是，虽然转义序列由反斜线\和另一字符组成，要敲两次键才能输入，但它却是一个“单一字符”，与其他单个字符的性质相同。

现在，为了给一个字符型变量linefeed赋以换行符的值，可以用

```
Char linefeed
  ⋮
linefeed='\n'
```

来得到。转义序列也可以和其他字符自由混合使用，例如下面这个语句

```
printf ("A\nBC\nDEF\n") ;
```

将得到格式如下的输出

```
A
BC
DEF
```

练习2-1　在你的系统上运行以上程序。有意去掉程序的某些部分，看看能得到一些什么样的错误信息。

练习2-2　做做实验，看printf参数串在带有\x时会发生什么。其中x是以上未列出的某一字符。

§2.2　变量、语句和算术运算

下一个程序打印下面的华氏温度和其对应的摄氏温度对照表，转换公式为C = (5/9) × (F-32)。下面是对照表（华氏0-300度）

0	-17.8
20	-6.7
40	4.4
60	15.6
...	...
260	126.7
280	137.8
300	148.9

为了完成此任务，我们已经编出了下面的程序

```
/* print Fahrenheit-Celsius table
   for f = 0, 20, ···, 300 */
main ( )
{
    int lower, upper, step;
    float fahr, celsius;
    lower = 0;    /* lower limit of temperature table */
    upper = 300;  /* upper limit */
    step = 20;    /* step size */
    fahr = lower;
    while (fahr <= upper) {
        celsius = (5.0/9.0) * (fahr-32.0);
        prinf ("%4.0f%6.1f\n", fahr, celsius);
        fahr = fahr + step;
    }
}
```

程序中的头两行，为我们所加的注释。这里，用注释简单明了地说明了在华氏0—300度的范围内，打印出一个华氏—摄氏温度对照表，每隔华氏20度打印一次。

符号/*和*/是用来加注释用的，我们可以在这两个符号之间加入任意的说明文字作为注释。在程序中加入适当的注释是一个良好的习惯，在大型程序中更有必要。当编

译程序对程序进行扫描时，这一对注释符号之间的任何内容都弃之不顾。注释可出现在当空白字符和换行符可以出现的任何地方。

在C中，所有变量在使用之前必须加以说明，通常把说明放在函数的开始部分，在任何可执行语句之前。如果忘了说明，编译程序就会给出一个出错的诊断信息。说明由类型及该类型变量的一个变量表组成，比如在

```
int lower, upper, step;
float fahr, celsius;
```

中，类型int表示所得到的变量是整数；float代表浮点数，即可带有小数部分的数。int和float的精度视所用的特定机器而定。比如，在PDP-11上，整数是有16个二进位的带符号数，范围在-32768至+32767之间。浮点数有32个二进位，它相当于大约七位有效数字，量值在10^{-38}至10^{+38}之间。

除int和float之外，C还提供了其他几种基本数据类型：

char	字符——一个字节
short	短整数
long	长整数
double	双精度浮点数

以上类型的实际大小也视各种机器而定，在第三章将进一步讨论。此外，还有这些基本类型的数组、构造及联合，指向它们的指针以及返回这些基本类型值的函数。所有这些，也将在以后适当时候讨论。

在函数体中包含一系列的语句，这些语句描述了该函数要做的工作。每个语句都要完成一定的工作。每个语句都必须以分号结束。

温度转换程序的实际计算由以下赋值语句开始：

```
lower = 0;
upper = 300;
step = 20;
fahr = lower;
```

以上各语句把变量置为它们的初始值。

因为该表的每一行都是用同样的方法计算的，所以采用循环，每行重复一次；使用while语句的目的就在于此。

```
while (fahr <= upper) {
    ...
}
```

括号内的条件被测试，若为真（fahr小于等于upper），循环体（用花括弧圈着的所有语句）就执行一次。然后再测试条件，若仍为真，就再执行循环体。当测试为假后（fahr超过upper），循环才结束，并继续执行本循环之后的语句。因为本程序没有后继语句，故程序终止。

while后面的花括弧里的循环体，可有一个或多个包在花括号内的语句。只有单个