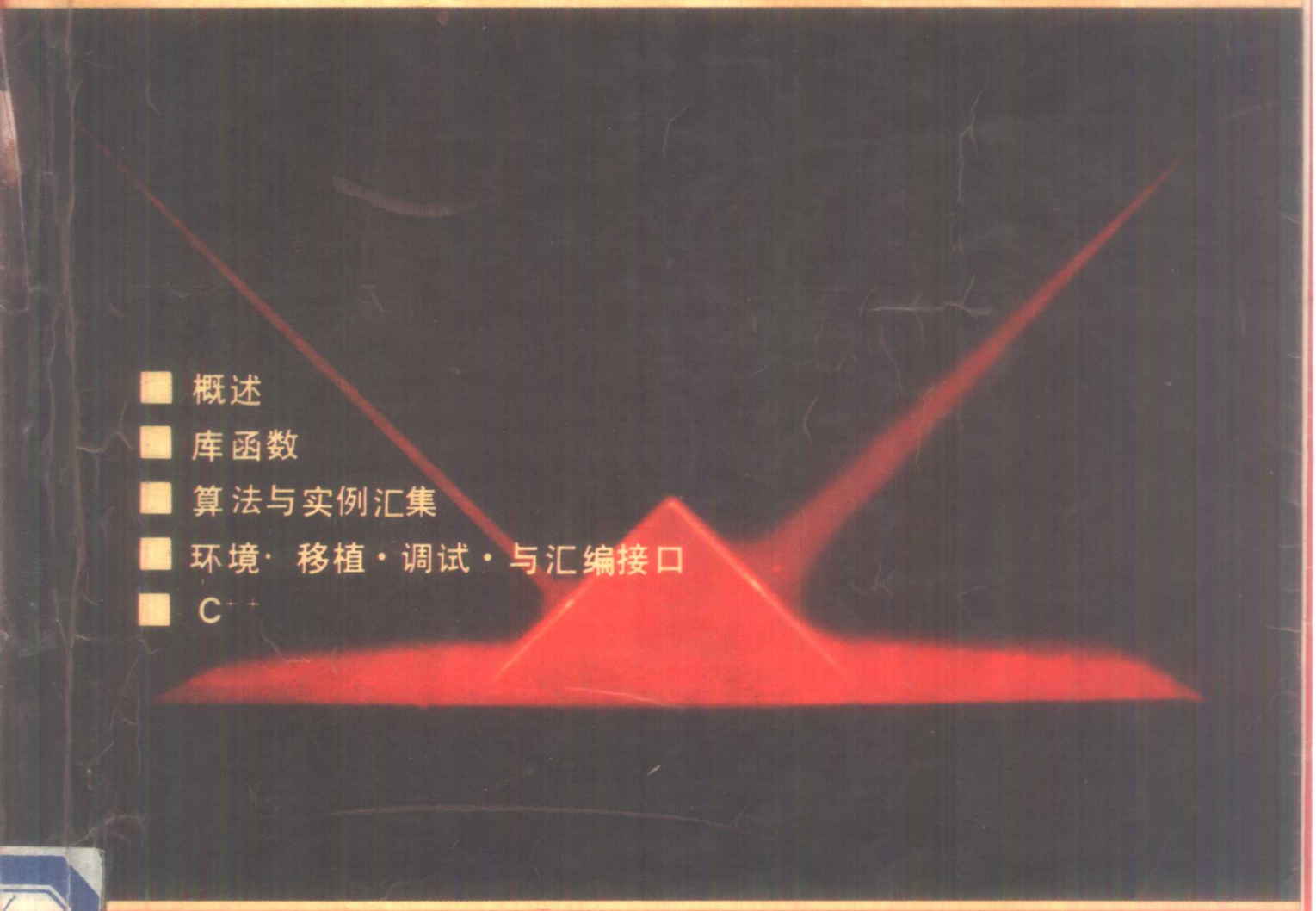
ANSIC·UNIXC· MicrosoftC·C++

C 语 言 大 全

〔美〕Herb schildt 著　　吴双译



中国科学院希望高级电脑技术公司
一九九〇年七月

ANSIC·UNIXC·MSC·C++

C 语 言 大 全

〔美〕 Herb Schildt 著

吴 双 编译

中国科学院希望高级电脑技术公司

一九八九年七月

前　言

本书是关于C语言命令、库函数、代码、应用程序以及最新进展的百科全书。

原作者是C语言方面的专家，中国科学院希望电脑技术公司已出版他的多部著作。这次我们编译的《C语言大全》是C语言使用者案头必备的参考书。无论对初学者还是经验丰富的程序员，本书都能提供有益的信息。

为了便于读者更快地找到所需要的内容，本书分为五个部分，分别讨论了C语言的每一个重要分枝。

第一部分是对C语言的一般论述，讨论了关键字数据类型，操作符，函数、指针，文件I/O，动态存贮分配及其它基本知识。

第二部分按照类别详述了C语言的库函数，读者可以在这里找到很多标准用法。本书所提供的函数不仅包括最新的ANSI标准用法，也覆盖了传统的UNIX C用法。其范围之广是少见的。

第三部分是关于用C语言实现的算法和用C语言编写的实例的汇集。这里包括查找和排序算法；栈，队列，链表和二叉树的实现及算法；人工智能算法；图形学算法及表达式分析算法。

第四部分讨论了C语言的使用环境。包括如何提高C语言效率，C语言的移植和如何调试C语言程序。同时还提供了C语言与汇编语言的接口，其中有许多内容是只能在实践中才能获得的经验。

第五部分是为对C语言最新发展感兴趣的人提供的。这里介绍了C++语言，它是最新出现并得到迅速发展的C语言分枝。

ANSIC·UNIXC·MSC·C++

C 语 言 大 全

〔美〕　Herb　Schildt　著

吴　双　译

责任编辑：　鞠玉兰　秦人华

· 北京市新闻出版局

准印证号：30026-900026

· 订购单位：北京8721信箱资料部

· 邮　码：100080

· 电　话：2562329

· 乘车：320、332、302路车至海淀黄庄下车

· 办公地点：希望公司大楼101房间

目　录

第一部分　C　语　言

第二部分　C语言标准函数库

第三部分　算法及其应用

第四部分 用C语言开发软件

第五部分 新的进展

第一部分 C 语 言

本书的第一部分对C程序设计语言给出了一个全面的论述。第一章是对C语言的总论，有更高要求的程序员可以直接跳到第二章。第二章分析了C语言的数据类型，变量，操作符和表达式。第三章给出了程序的控制性语句。在第四章中研究了函数以及C语言的块结构。第五章讨论了数组和指针。第六章讨论了结构，联合和位域操作。第七章分析了用户自定义的类型和枚举类型。第八章研究了C语言的输入输出功能。第九章总结了编译命令和代码生成过程。

第一章 C语言概述

本章的目的是对C程序设计语言给出一个轮廓性的描述，同时讨论它的发展过程，应用范围以及基本结构。

1.1 C语言的起源

C语言的最初设计和实现是由Dennis Ritchie 在DEC PDP-11上的 UNIX 操作系统环境下完成的。C 语言的前身是 BCPL语言，后者在欧洲目前还有应用。BCPL 语言是由Martin Richards 设计的，它深刻地影响了 Ken Thompson 设计的B语言的风格。B语言在1970年发展成C语言。

很多年以来，C语言的标准版一直由 UNIX 操作系统第五版所支持。它的经典描述是 Brian Kernighan 和 Dennis Ritchie 在1978年出版的《C程序设计语言》这本书提供的。随着微型机的大众化，各种C语言的实用版本纷纷出现。在不同版本上，C语言的源程序可以奇迹般地保持高度兼容。另一方面，由于没有一个标准，它们彼此之间也有一些差异。为了改变这种情况，美国国家标准局成立了一个委员会，他们在1983年夏初建立了一个C语言的标准。在本书的写作过程中，这个标准已经很完整了，它已在1987年被采用。

1.2 C是一种中级程序设计语言

一般都把C语言叫做中级计算机语言。所谓中级语言并不是指它的能力弱或者难于使用，也不是指它不如BASIC 或 PASCAL 之类语言高级。C语言并不象汇编语言那样经常给使用者造成麻烦，它只不过是把高级语言的一些特征同汇编语言的能力结合起来形成了所谓的中级语言。下面我们列出有关语言的分类以及C在这些语言中的位置。

高级语言	中级语言	低级语言
Ada	C	Assembler
Modula-2	FORTH	
Pascal	Macro-assembler	
COBOL		
FORTRAN		
BASIC		

C语言作为一种中级程序语言，它允许对字节，字位和地址直接操作，而地址是计算机的基本工作单位。C语言也很便于移植，所谓可移植性是指一个软件可以从一种计算机类型上转移到另一种计算机类型上仍能保持其功能。例如，如果一个在 Apple 上写的程序可以在 IBM PC 机上运行，我们就说这个程序是可移植的。

所有的高级程序设计语言都支持数据类型这个概念。一个数据类型定义了一组值，可以通过一组操作来存取具有这种类型的变量。普通的数据类型包括整型，字符型和实型。尽管C有五种基本类型，但它并不象 pascal 或Ada语言那样对类型要求的那么严格。C 语言几乎允许所有类型互相转换。例如，在很多表达式中，C语言都允许整型和字符型自由混用。C语言也不进行象数组越界或者变量类型不匹配这一类的运行错误检查。不过在有些版本中，对除零错误和堆栈溢出错误还是提供报告的。上面的那些检查都是程序员的职责。

C 语言的特殊性表现在它允许对位域、字节，字和指针的直接操作。这就使它很适合频繁使用这些操作的系统级程序设计。C语言的另一个特征是仅有32个关键字（Kernighan 和 Ritchie定义了27个，ANSI 标准委员会增加了5个）。这些关键字构成了C语言的骨架。与此对应的是 IBM PC 机上的 BASIC 语言有159个关键字。

1.3 C是一种结构化程序语言

所谓C语言的块状结构并不是一种严格的说法，我们一般把C语言称为结构化程序语言，这是因为它们结构很象 ALGOL，PASCAL 和Modula-2。严格的块结构定义是指允许在过程或函数内部说明另外的过程或函数。在这种情况下，全局变量和局部变量的概念被作用域规则所代替，该规则控制变量或过程的可见性。因此，既然C语言不允许在函数内部建立函数，它也就不能叫做块状结构的语言。

结构化语言的一个显著性质就是代码和数据的独立性。这种语言可以把为完成特定任务所用的信息和指令同程序的其它部分分隔开，所用的方法就是使用局部变量和子程序。利用局部变量，程序员可以写出对程序其它部分没有付作用的子程序。这样，他就可以很容易地写出需要共享的代码。换句话说，如果你要开发一个独立的函数，你只需要知道这个函数要做什么，而不需要知道它怎样做。需要注意的是，过份地使用全局变量（在整个程序内都起作用的变量）会导致不应有的副作用，从而使程序出错（使用过BASIC的程序员都会遇到过这个麻烦）。

一种结构化程序语言允许你在程序设计中有很大的选择余地，它直接支持各种循环结构如：while，do-while 和 for 语句。在结构化语言中，goto 语句的使用是被禁止或者不提倡的，它不象在 BASIC 或 FORTRAN语言中那样作为一种常见的程序控制结构。另外，结构化语言允许你采用缩进语句书写方法，同时也没有严格的语句位置概念（FORTRAN则严格要求语句的位置）。

下面我们列出一些结构化语言和非结构化语言：

非结构化的	结构化的
FORTRAN	Pascal
BASIC	Ada
COBOL	c
	Modula-2

结构化语言是现代的发展趋势，而非结构化语言正在过时。实际上，老牌计算机语言的标志就是它是非结构化的。目前广泛认为，结构化语言要比非结构化语言更便于编写程序，同时，结构化语言程序的清晰性使得维护它们也更加容易。

C 语言的主要结构成份就是函数(也就是 C 中的独立子程序)。在 C 语言中，函数被描写成一个块状结构，所有的程序动作却在其内部进行。该语言允许把程序的每个任务的定义和编码分开实现，因而使你的程序易于模块化。一旦函数被建立，你可以把它放在不同环境下工作，而不必担心会对程序的其他部份产生副作用。能够建立独立性强的函数是大型项目中非常严格的要求，在那里，一个程序员写的代码不会影响另一个人的代码是非常必要的。在二十六章我们将讨论如何在大型项目的开发中消除副作用。

C 语言中另一个结构化和独立化的方法是使用代码块。一个代码块是指一组程序语句在逻辑上可以当作一个单位来使用。C 语言中的代码块是靠在一组语句的前后加上{ }来构成，例如：

```
if ( x<10 )
    printf ( "too low, try again" );
    reset\counter ( -1 );
}
```

if 语句后面的两条语句作为一个整体在x<10的情况下被执行。这两条语句构成了一个代码块。它们是一个逻辑单位，必须同时执行。需要注意的是，每一个语句既可以是单个语句，也可以是语句块。使用代码块既可以使算法更加清晰，优美和高效，也可以帮助程序员看清一个过程的本来面目。

1.4 C是面向程序员的语言

有人会对这句话提出疑问，难道不是所有计算机语言都是面向程序员的吗？答案是："不"有些语言如COBOL 和 BASIC 就是面向非程序员的经典例子。COBOL 语言的设计目标不是为了使程序员更自由，而是为了改善代码的可读性。它不关心代码的可靠性，也不关心代码执行的速度，只是想让非程序员能读懂这个程序。BASIC语言最初也是为非程序员编写简单的计算机程序而设计的。与此相反，C 语言完全是被真正的实际程序员所建立，影响和使用的语言。结果是 C 语言提供了程序员想要的一切：很少的限制，没有什么修饰，块结构，独立函数能力和很少的关键字。使用 C 语言可以使程序员达到使用汇编语言的执行效率，同时还保留 ALGOL 和 Modula-2 的结构化特性。毫无疑问，在非常优秀的程序员中，C 语言是一种很通用的程序设计语言。

C 语言可以在程序员之间流行的一个重要因素是它可以代替汇编语言。汇编语言用符号来表示计算机能够直接执行的二进制代码。每一条汇编语言的语句都对应着计算机实际执行的一个动作。尽管汇编语言可以给程序员完成其任务的最大灵活性和执行效率，但却很难用汇编语言来开发和调试程序。而且由于汇编语言是非结构化的，最后写出的程序象一团乱麻，充满了转移语句，调用语句和变址语句。这种缺乏结构性的汇编语言程序很难于阅读，改进

和维护，更重要的是，汇编语言无法在不同的机器（CPU）之间移植。

最初，C 语言是用来进行系统程序设计的。系统程序是在大量计算机程序中构成操作系统及其支持程序的部分。下列程序一般被叫做系统程序：

- 操作系统
- 解释系统
- 编辑程序
- 汇 编 器
- 编译程序
- 数据库管理系统。

随着 C 语言的流行，很多程序员利用它的可移植性和高效性开始编制各类程序。因为很多机器上都有 C 语言编译程序，所以可以把在一种机器上编写的程序很少或不加以改变就能运行在另外的机器上。这种可移植性节省了时间和金钱。C 编译程序还可以产生来非常紧凑和高效的目标代码，这要比 BASIC编译程序好得多。

使C 语言能用于各种场合的一个重要原因就是程序员喜欢用它。C 语言有汇编语言的速度和 FORTH 语言的能力，而又没有 Pascal 和 Modula-2 的限制。每个C 语言程序员都可以建立和维护特有的函数库，以适应他自己的风格并能应用于各种不同的程序之中。因力C 语言允许并鼓励独立编译，所以这使程序员可以方便地管理大型项目，并减少重复性工作。

1.5 编译器和解释器

编译和解释是指程序的执行方式。从理论上说，任何程序语言都可以既被编译执行，又被解释执行，但是，有一些语言通常采用其中的一种形式。例如，BASIC 语言解释执行的，而 C 语言则是编译的（最近有一些 C 语言解释器出现，用来帮助调试程序）。一个程序的执行方式并不取决于它用哪种语言书写。解释和编译的概念只是说明怎样对你的源程序进行处理。

一个解释器读入你的程序的源代码，一次一行，并执行这一行给出的特殊指令。一个编译器则一次读入整个程序，并把它转换成目标代码，这种目标代码可以直接在机器上执行。目标代码也被称为二进制代码或机器代码。一旦程序编译后，源代码中的行就不会在程序执行过程中体现出来了。

当使用解释器时，它必须在每次你要运行程序时存在。例如，你使用 BASIC 语言时，就必须先执行 BASIC 解释程序，然后调入你的程序，并用 RUN命令启动它。BASIC 解释器每一次查看你程序中的一行，先排除错误再执行它。每次程序执行时，都要重复这个缓慢的过程。与此相反，编译器把你的程序转换成可以直接在计算机上运行的目标代码。由于编译器只对你的程序进行一次转换，以后你就可以直接执行你的程序，换句话说，仅仅需要输入程序的名字就可以了。总之，编译器只做一次工作，而解释器则在程序的每次运行时都要工作。

编译过程本身是需要时间的，但是这替你节省了每次运行起程序时所花费的时间。编译过的程序运行起来要比被解释的程序快得多。只有一种情况下，就是你的程序非常短（小于50行）而且没有循环语句时才不是这样。

在本书和你的编译手册中可以经常看到的两个词是编译时间和执行时间。编译时间是指整个编译过程所需要的时间。执行时间是指程序实际运行时所需要的时间。不过，有时这两个词也被用来指明错误类型，如编译期间错误和解释期间错误。

1.6 C语言的形式

下面我们列出 C 语言的32个关键字，它们同语法一起构成了 C 语言的骨架。

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

C 语言的关键字

所有的关键字都是小写字母。在 C 语言中大写字母和小写字母是不同的，小写的else是关键字，而大写的 ELSE 则不是。关键字不能在 C 语言程序中用于其他目的，也就是说，不能用来作为变量或函数的名字。

所有的 C 语言程序都包含一个或多个函数。唯一不可缺少的函数叫做main( )。它在程序开始执行后第一个被调用。在优秀的 C 语言程序中，main( ) 一般包含程序的轮廓。这种轮廓由一系列函数调用组成。尽管 main 这个词并不是 C 语言的一部分，但我们经常要把它看成是这样。不要用 main 作为变量的名字，因为这会给编译器带来极大的混乱。

C 语言程序的一般形式如下，在这里 f1( ) 到 fn( ) 代表用户定义的函数。

```
global declarations
main( )
{
   local variables
   statement sequence
}
f1( )
{
   local variables
   statement sequence
}
f2( )
{
   local variables
   statement sequence
}
.
.
.
fN( )
{
   local variables
   statement sequence
}
```

C 程序的一般形式

1.6.1 函数库和连接

从本质上说，完全用程序员自己写的语句就可以构成一个有用的和实际的C语言程序。不过这种情况很少见，因为 C 语言（作为一种语言定义）并没有提供任何执行输入输出 操作的手册。实际上，很多程序中都调用了 C 语言标准程序库中的函数来完成这件事。

所有的 C 语言编译器都附带一个标准函数库，用来提供常见的操作。有些 C 语言的 版本中，函数库是一个很大的文件；在另一些版本中，则提供多个小文件，并用一个装配程序保证其有效性和实用性。为简便起见，本书只提供库函数这个词来指明这两种情况。

在你的 C 语言编译器中会有很多你所需要的通用函数。当你要用你的某个程序本身 来提供的函数时，C 语言编译器会记住这个名字，然后连接程序会把你写的程序与标准函数库中的目标代码进行连接。这个过程叫做程序的连接。有些 C 编译器有自己的连 接 程 序，另一些则用操作系统提供的标准连接程序。

函数库中的函数采用的是可重新定位的格式。这就是说，机器代码的内存地址并不是绝对的，仅仅保留相对位移信息。当你的程序同标准库中的函数进行连接时，这些内存位移被变为实际使用的地址。有一些技术手册和教科书专门来解释这个过程。不过你不必 在 C 语言程序中关心实际重定位过程。

所谓标准库函数只是一种说法，在 C 语言中并没有精确规定哪些函数要放在标准 库中以及如何具体实现它们。这些完全由 C 语言编译器的设计者们自行决定。不过大部分商 用的 C 语言编译器都使用 UNIX标准库函数。 C 语言最初是在 UNIX 操作系统下开 发的，所以虽然没有直接的联系，一般都认为这个版本是标准的。后来，当 ANSI 答案 出 现时，情况得到了改变。ANSI 版本规定，任何编译器的库函数都要包括一组特 定 的 函 数集合，只要它承认是遵循 ANSI 标准的。但实际上，很多 C 语言的版本都在 ANSI标准之外又附加了许多有用的内容。本书着重讨论 ANSI 标准定义了的那些函数。

尽管许多C编译器声称自己是UNIX标准版本，但它们的函数名字与标准函数名字 是 有所不同的。例如把一个字符串拷贝到另一个字符串中的函数名字一般叫做 strcpy( )，但是也有叫做 strcopy( ) 或 stringcpy( ) 的。你应该查对一下你的编译手册中具体的名字 是什么。不过这种情况在 ANSI 标准中是不存在的。除非特殊声明，本书中所有的例子都 遵循 ANSI 标准使用函数的名字和调用约定。

你在编写程序中所需要的大部分函数都会在标准库中找到。你所做的仅仅是把这些砖块搭在一起。如果你编写了一个经常要使用的函数，你最好把它放在函数库中。有些编译器允许你把你自己的函数加入标准库；另一些则要求你建立一个附加数库。不论哪种方法，你都可以不断使用你所编写的这个函数。

1.6.2 分别编译

很多短小的 C 语言程序完全可以装在一个源文件中。但是，随着程序长度的增加， 编译的时间会越来越长。C 语言允许把程序分成几块，分别放在不同的文件中，并且分别编译这些文件。一旦所有的文件都编译好，就可以把它们以及其他的标准库函数连接起来，构成完整的目标代码。分别编译的优点是当你只改变一个文件中的代码时，你不必重新编译整个程序。除了简单的程序以外，时间的节省是很可观的。在第四部分中我们将详细讨论这个问题。

1.6.3 编译一个C语言程序

编译一个 C 语言程序包括下面四个步骤：

1．建立你的程序

2．对你的程序进行编译

3．把程序同必要的库函数连接在一起

4．执行程序

很多 C 语言编译器需要一个独立的编辑程序来建立你的源程序。如果你用 CP／M，你就要用它们标准编辑器 ED。如果使用 MS-DOS 或 PC-DOS，你就要用 EDLIN。除此之外，有很多优秀的屏幕编辑程序也适合用来建立程序。编译器只接收标准的正文文件。一般的编译器不识别某些字处理软件的结果，因为它们含有控制字符和不可打印字符。如果这方面不把握，可以去询问一下有经验的人。

1.6.4 用批命令编译你的程序

CP／M，MS－DOS 和 PC－DOS 允许你定义一个批命令来自动完成一系列任务。你可以用批命令建立整个编译过程。下面的批命令中包括编译，连接和运行三个部分。这仅仅是个例子，你可以根据你的 C 语言手册做出精确的编译步骤。

CP／M	MS—DOS 和 PC—DOS
cc 1.c	cc %1.c
link 1, lib.c	link %1, lib.c
1	%1

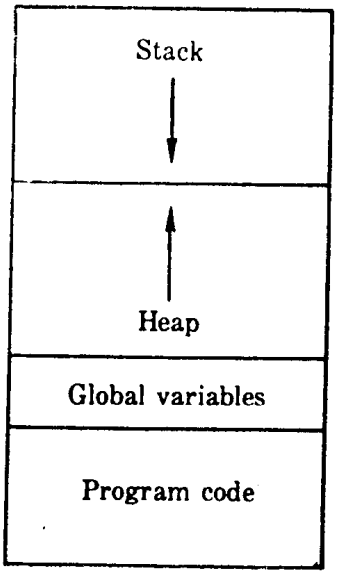
其中符号%1 和1代表你的程序名字。

如果你用的操作系统不是 CP／M，MS—DOS 或 PC—DOS，你就要查有关手册决定批文件的格式。

1.6.5 C 的内存图

一个编译好的C 程序建立并使用四个独立的逻辑区域，这些内存中的区域用于不同目的。第一个内存区域用来存放程序的代码部分。下一个区域用来存贮全局变量。余下的两个部分分别称为栈和堆。栈在程序运行时起着很大作用，它保留函数调用的返回地址，函数的变量以及局部变量。它也用来保存当前CPU的状态。所谓堆是一块空闲存贮区，当你的程序要使用C 语言的动态存贮分配功能时，就要用到它们。

尽管由于CPU类型和C 语言版本的不同，造成这四个区域实际分布的位置不同，但下面我们给出一般性的C语言内存分布图。



C 语言内存分布图

1.7 概念复习

本书经常用到下面这些概念，所以你必须完全理解它们的实际含义。

源程序：

用户可以阅读的程序正文。源程序最终要作为 C 语言编译器的输入。

目标代码：

由源程序转换成的机器代码，计算机可以识别并执行它。它是连接程序的输入。

连接程序：

一个把分立的函数合并成一个程序的程序。它把你写的程序同标准库函数合并起来。连接程序的输出就是可执行程序。

函数库：

一个包含标准函数的文件。这些函数可以被你的程序使用，它们包括所有的输入输出操作及其它一些有用的过程。

编译时间：

指编译你的程序所需要的时间。在这期间出现的错误称为语法错误。

运行时间：

你的程序执行所用的时间。