[美] Joe Campbell 原著
游疆来 杨飞强 黄 昱 译
周少柏 查良钿 审校

串行通信编程指南

北京科海培训中心
一九九〇年十月

串行通信编程指南

[美] Joe Campbell 原著

杨飞强 游疆来 黄 昱 译

周少柏 查良钿 审校

北京科海培训中心

发　　行：北京科海培训中心资料组
地　　址：北京海淀路82号科海培训中心
邮　　码：100080
联系电话：2562449　2562954
乘　　车：320　332　302路车至海淀黄庄下车
　　海淀文化馆北
印　　刷：河北省蔚县印刷厂

北京市新闻出版局准印证号：3133—90133

译 者 的 话

从复杂的巨型机到简单的个人微机，在运行和使用上都离不开通信问题。计算机与计算机之间、计算机与外部设备之间都以通信的方式传递、交换数据或信息。

计算机通信有两种基本方式：并行通信和串行通信。并行通信的特点是传输速率高；串行通信虽然速度较慢，但它节省线路。在计算机内部及计算机与外部块数据设备之间多采用并行方式。串行通信往往应用于数据传输速度要求不甚高和通信双方相距较远的情形。一般计算机控制台与主机之间、计算机与绘图机之间，通过电话网络相连的两台计算机之间的通信多采用串行方式。

数据的串行通信问题是一个专门的学科。它涉及的内容非常广泛，包括计算机接口、调制解调器、数据传输协议等等。近年来出版的有关书籍大多只是蜻蜓点水似地给以介绍，对于一般读者很难获得系统的知识。针对这种情况，我们选择翻译了这本由Joe Campbell所著的《C Programmer's Guide to Serial Communication》专著，真诚地希望能为国内的广大读者提供一定的帮助。

本书从介绍串行通信的历史及演化过程入手，由浅入深地引出各个方面的专门问题。书中不光讲述了硬件原理和外围知识，还为读者提供了实用的编程方法和系统的C语言程序。

全书分三个部分。第一部分介绍基础知识：包括ASCII码表的由来和在通信上的应用，异步串行通信的发展过程，错误及错误检测，信息传输协议，调制解调器及其控制，通用异步接收发送器的软硬件原理(其中包括RS-232接口，UART中断，8250，Zilog Z80SIO)，专门的产品介绍：Hayes Smartmodem。第二部分讲述如何用C语言对异步串行通信编程：包括基本串行I/O库的设计，可移植性，波特率和数据格式管理，RS-232控制，格式化输入输出，对Hayes Smartmodem的编程，XModem文件传输协议，CRC计算，中断编程等等。第三部分为附录。

其中第1～4章及附录由游疆来翻译，第5～7，9～12章由杨飞强翻译，第8，13～16，18，19章由黄昱翻译，第17章由余恒翻译。

本书在编辑、审校的过程中，得到了中科院软件所周少柏、查良钿同志的热情帮助，在此表示衷心感谢。

译者

一九九〇年十二月

目　录

第一章 ASCII 字 符 集

我们在进入通信的主题之前，先来讨论通信的媒介。尽管所有的高科技都是以计算机为基础的，但分析计算机通信则要从基本的词法讨论入手。这样我们能够把分析人与人之间通信的方法应用到分析计算机与计算机之间的通信上。

在过去一两千年间，人类的通信大多是借助书写语言的。

从根本上说，一种书写语言最终可以被看作为一种编码体系。拼写相当于按照一定的规则编码，而阅读刚好是快速的解码。语言主要是由编码单元(字符)和编码集(字母表)所标志的。例如，一个有效的英语单词可以仅由26个不同字母中的几个所组成。由于在英语中，没有对单词长度的限制，从理论上说，只要我们愿意容忍用无限长的时间去读出一个单词，那便有可能用这26个字母构造出无限多个单词。按照英语的规则，某些字母的次序是遵循一定规律的（如"u"总跟在"q"的后面），以避免发音上的不流畅。总之，人类语言是极其复杂的。

然而，我们如何来描述一种计算机语言*呢？，记住，计算机语言必须能够完全被计算机所理解。与人类语言比较起来，计算机词汇是相当简单的；它仅只处理两个"字母"，并且所有的"单词"都具有同一的长度——8个"字母"。从这点上说来，可以简单地得出结论：计算机语言的字典，仅由2^8（或256个）"单词"组成，它既没有成语和习语，也没有拼写规则。

众所周知，计算机"字母表"的两个"字母"是0和1。一个计算机"单词"被当作为一个"字节"，而一个计算机"字符"被当成一个位。

• 上下文语义

用这么小的词汇表，计算机不可能表达清楚任何语义(试想：如果你仅只知道256个单词，你的谈话将是多么的单调和贫乏)。那么，问题是如何用这256个单词来表达数目众多的客观对象。解决这一问题的办法是用上下文语义。

在人类语言中，单词通常都有确定的含义，即一个单词仅只表示一件事物。一个单词具有单个意义的程度有时称为"真值"（truth value）。然而，由于词源学中的奇异性，许多单词的意义都不能从它们的拼写上准确无误地推断出来——譬如："bow"、"row"和"sanction"，这些单词必须按照它们所在句子的意义来解释。换句话说，它们的含义仅在上、下文中是清晰的。

上下文语义是拓展计算机的这256个字母字汇表的表达能力的唯一办法。这意味着我们可以赋予不同的意义给256个字母，其确切的含义取决于我们想对它们作何种解释。每一种上下文语义的规定称为一个集合。

---

* 这里所讲的计算机语言同计算机编程语言（如BASIC PASCAL，C等）完全不同，只是作为与人类语言的类比。

(A) 机器上下文关系：指令系统

字节0100-0001被用作为8080微处理器的一个指令，它的作用是使微处理器将C寄存器的内容送到B寄存器。每一种微处理器解释这256个“单词”的方式都不一样，例如8086微处理器就将字节0100-0001解释为使CX寄存器加1。有时希望在不同处理器的指令系统之间存在兼容性，即一个处理器的指令系统包含另一个处理器的指令系统，例如Z80的指令系统除了增加某些扩充指令（如字节10H，20H，30H等）外，基本等同于8080的指令系统。当这种关系存在时，扩充指令系统被称作为原集合的“超集”。

(B) 人类上下文关系：字符集

上述两个例子说明了计算机词汇表是可以按照非人类上下文语义来约定的。显然，也可用这样的256个“单字”定义一种上下文关系来表示普通的英语字母、标点符号、数字等等。这种上下文关系被称为“字符集”。

字符集是根据需要而创造出来的。实际上，当今有数百种字符集在使用中。只要在一个学科范围内存在共同点，便要衍生出标准。在计算机生产厂家之间也存在着相似的情形。任何一个厂家都希望自己的设备同其它设备有广泛的兼容性。不言而喻，如果每个生产厂家都发明一套其自家的字符集，那么在不同的设备之间的数据交换将会格外困难。出于这种原因，工业标准已经由不同的标准化组织在同计算机工业合作中建立起来。

迄今，在讲英语的国家范围内，最主要的字符集是“美国信息交换标准代码”（英文为American Standard Code For Information Interchange，简写为ASCII），ASCII代码是美国国家标准协会(ANSI)制定的，此处它指的是ANSI标准X 3.4—1977(1983年修订)信息交换码，或者就称为ASCII。实际上，还有另外两个等同的标准，它们是ISO(国际标准化组织）的用于信息处理的7位码字符集和CCITT（国际电话、电报咨询委员会）的字母表No.5。

其实，世界上几乎每一种计算机都采用了上述几种人类可读字符集中的一种，但IBM（国际商用机器公司）却是一个显著的例外，他们创造了自己的字符集EBCDIC（扩展的二进制编码的十进制代码，英文为：the Extended Binary Coded Decimal Interchange Code）。不幸的是，它不与上述的所有字符集兼容，但是，为了追随微计算机工业的既定趋势，IBM还是在它的微机产品上选择使用了ASCII代码。

§1.1 ASCII码表介绍

在我们讨论ASCII字符集的细节之前，首先来对它的某些开发思想作一概述。

ASCII的出发点主要是想作为一种人类可读的代码。然而必须指出，人和计算机的可读代码的区别是人为的，因为在物理的意义上，人不能读计算机代码。计算机代码只是存在于半导体隔离层之间的电势。简而言之，所有的计算机代码都是设计成在某种低级水平上同其它非人的电子设备的相互作用。需要记住的要点是：ASCII字符集要能为普通的设备所用（这种设备最终须为人的视觉或听觉所识别）。由于要求这种设备能把计算机代码翻译成人能够阅读的形式，故这种代码中也须包含一些用于控制显示设备本身的字符。所以一部分ASCII代码被明确指定为控制设备之用。

从人类可读性考虑，ASCII代码的制定者未免有些偏颇。由于世界上绝大多数计算机都是用于数据处理的，这样，数据处理的需要自然地放到了较高的位置上。就象我们将要看到的，在ASCII码表中，某些字符以及字符的顺序明显具有数据处理的特征。同样，任何一种推荐使用的代码，都必须服务于数据通信领域。从本质上说来，数据通信对代码的依赖性是非常重要的。除此之外，我们还会意识到，如果把ASCII码表当作为计算机的标准词汇表的话，它还得包纳计算机程序员编程时所需用的字符。因此ASCII码表也包含了普遍编程语言FORTRAN，BASIC，COBOL需求的字符。

甚至ASCII代码的制定者还曾想过把各个民族的不同字母表都包纳进去，以使得不同的民族使用确定的代码、诸如特有的字符、标点和分隔标记。

§1.1.1 "ASCII"的歧义性

ASCII是一种7位代码，这样从128到255之间的128个位置是没有定义的。虽然也有采用第八位的字符集，但原则上不是ASCII码。尽管ASCII表的大小和范围在X3.4文件中作了毫不含糊的定义，然而对"ASCII"一词的含义仍然产生了某种混淆。例如Commodore公司称它自己的字符集为"PETASCII"(由PET计算机所命名)，其实它只具有与ASCII很少的相似性。IBM把包括IBM PC上使用的"特殊键"(如功能键)编码体系归并为"扩展ASCII"，同样这些键如F1，Alt和PgDn同ASCII是没有任何相干的。许多字处理程序采用了未定义的高位128个字符（从80H到FFH）作为行结束、空边和定位标记，例如，在Wordstar程序中，字节A0H（SP第七位为true）表示为行调整而增加的空格。Wordstar也把每个单词（包括回车，换行）的最后字符的第七位置1，以便于行调整。

§1.1.2 ASCII码表

ASCII码表如图1-1所示，为便于查询，其中多增加了一栏内容。通过研究这个表的结构和组成可以了解许多计算机通信的本质性问题。

在ASCII文件（ANSI X3.4）中，这个码表是用八进制表示的。即是说它的宽度为8、长度为16。然而，对于微机程序员来说，采用16进制更为适合。这也就是我们给出的格式，其宽度为16、长度为8。在每个字符图案的上方，是代码的十进制和二进制值。任一字符的位置都可由它的行列号来表示："A"是4/1，"S"是2/4等等。ASCII文件是用十进制给出的行列坐标。但在这本书中，我们将全部使用十六进制。不同于ANSI的码表形式，采用十六进制意味着字符在码表中的坐标也是它在码表中的序号。例如：在ANSI的码表中把字母"k"当作61/1而在我们的码表中则为6BH。

ASCII代码的最明显特征就是它仅定义了256个可能的码位中的128个。特别是，ASCII字符集是一种7位码，这意味着在采用8位字节的系统中（包括所有微机），ASCII字符集是用低7位表示的，其取值范围为0～127。有时，当我们看到一个字节在80H～FFH范围之内时，不免产生疑问：它的ASCII码对应值是多少呢？显然应该是它的最高位减1。若是16进制表示形式，相当于高位数减8。例如若希望找出E4H的ASCII码值，可以简单地用E减8、得6，这便得到E4H的对应ASCII码值为64H。

ASCII有它自己的组织原则。为了便于分析，我们可以把它分成几个小的字符组(或字符子集）。最普遍的分法是分成两组：图案字符和控制字符。控制字符是从0～1FH及7FH

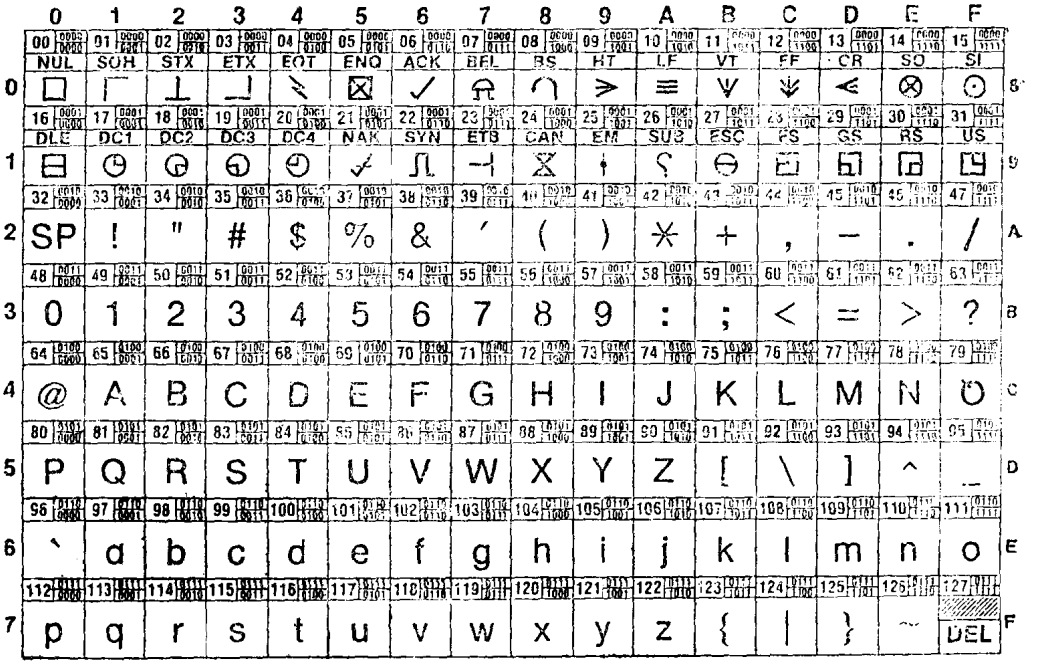

图 1-1 ASCII码表

(DEL)，而剩下的都是图案字符。

§1.2 图案字符

图案字符又可分为几个子集，其中的两个——数字和拉丁字母——是广为熟悉的，我们将首先讨论它们；剩下的称作为特殊字符，我们需要作更仔细的讨论。

§1.2.1 数字字符

数字字符如图1-2所示。数字字符既不同于数码，也不同于数值。ASCII数码1（位图0000-0001）指的是码表的01H位置，而数字"1"则位于码表的31H位置。数字和数码之间的区别对许多人来说是一个奥妙的问题，它是产生混淆的根本所在。数值是用在算术运算中的，诸如加、减法运算，它们是按照二进制形式存储的，即数值1被存作为0000-0001，2存作为0000-0010，3存作为0000-0011等等。另一方面，数字是人为规定的图案字符，换言之，数值（码）9存储时的原始二进制值是0000-1001；，但用于显示图案"9"的ASCII字符（数字）在ASCII码表中，则位于39H（0011-1001）位置。

只要所涉及到的数量是在0～9的范围之内，以ASCII数字形式存贮不存在空间上的浪费——两种情形都只占用单个字节。当数量变大时，以ASCII数字形式存贮将需要占用大得多的空间。因为每个十进制位都要用一个字节来存放。

作为一个例子，考虑数值26，581（67D5H）的存贮。它可用两个字节的二进制值表示：

0110-0111 1101-0101

然而，它的ASCII表示则需要5个字节：

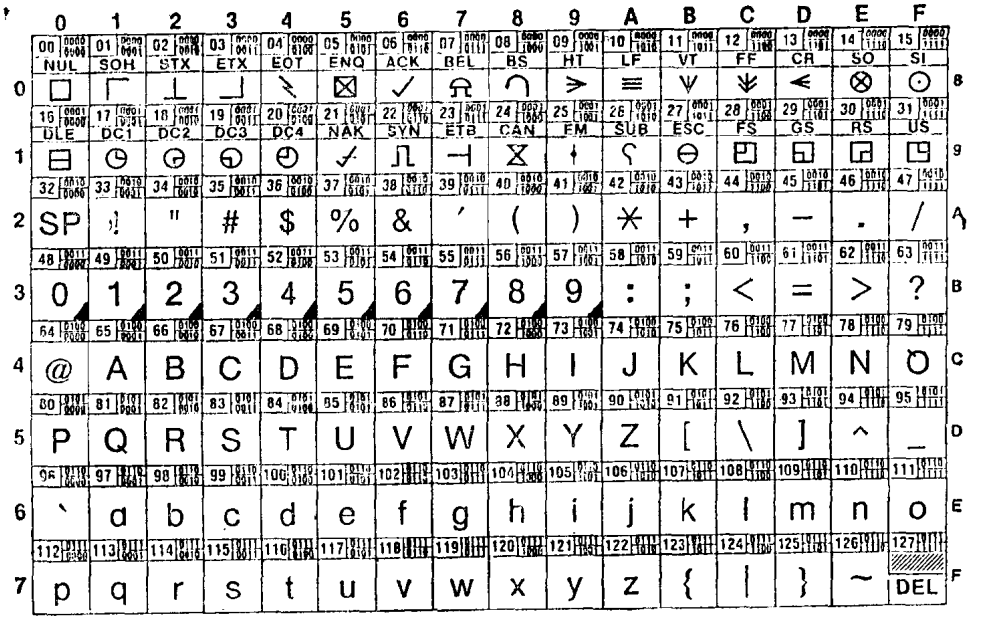

图 1-2 数字字符

0011-0010 (32H) →“2”

0011-0110 (36H) →“6”

0011-0101 (35H) →“5”

0011-1000 (38H) →“8”

0011-0001 (31H) →“1”

通过采用压缩ASCII码的方法，这个问题可以得到部分地缓解。从上面可以看出，每个字节中只用了低四位。如果把两个ASCII数字编码成一个字节，便可把存贮压缩一半。上述数值26，581可表示为：

0000-0010→“2”

0110-0101→“65”

1000-0001→“81”

用ASCII数字来表示数值，不论在编程还是在存贮上都会带来额外的开销。即使使用BCD格式也是如此。但能够对压缩ASCII数值数据作算术运算（甚至浮点算术运算)的系统仍然很普遍。例如8086微处理器提供了一组指令来调整计算压缩BCD格式运算量的算术指令的结果。尽管实现起来很方便，但对ASCII的计算与同样的二进制数值运算比较起来显得格外的慢。

依这些严重的不利因素——速度、存贮量、额外的编码——看来，你也许要问：这种表示形式为何还存在呢？答案是简单的——因为对ASCII格式数据的存取比较方便。换句话说，因为ASCII编码是普遍认识的，实际上，每个计算机系统都装备了处理它的功能。

• 阿拉伯零

阿拉伯零与拉丁字母“O”是极其相似的。奇怪的是，这种缺陷仍然在ASCII中保留了下来。尽管许多显示设备已经把零显示为加斜杠零（“Ø”），但这种形式又容易同斯堪的纳维亚字母表中的一个字母混淆。为了避免混同，我们的码表用了一个特殊的大写