C语言的汉字处理与图文数据库技术

刘路放 编著

西安交通大学出版社

C语言的汉字处理
与图文数据库技术

刘路放　　编著

西安交通大学出版社

内容简介

本书包括两部分内容：一是一个完整的汉字处理环境，是为了解决中文应用软件编程中经常遇到的与汉字操作系统的兼容性以及如何处理"半个汉字"等问题而设计的，包括一整套类型和宏的定义以及约100个库函数的使用方法和源程序；二是一个以C语言为宿主语言的图文数据系统的全部源代码和使用方法。该系统的数据库文件和B+树索引和Xbase兼容，同时又具备C语言的灵活而强大的编程功能。本书实为C语言程序员的最佳参考和初学者提高编程能力的良师益友。

本书配有高密软盘两张，内容包括书中所介绍的汉字处理系统HAN和图文数据库系统C_BASE的头文件、全部库函数的源程序、已经编译好的小、中、大、紧缩的巨型模式下的函数库以及16点阵简、繁体汉字库和24点阵宋、仿宋、黑和楷体汉字库。除此而外，盘中还包括以下应用软件(均为书中例题)的源程序、封面文件以及执行文件：

MKXZK(小字库制作)

READHAN(中西文文本阅读器)

TEXTSRCH(中西文全文检索工具)

WPS2TXT(WPS文件转换工具)

以及徒手画工具DWH。使用DWH制作应用软件的封面效果直观、速度快，制成的封面文件体积小，且可以使用盘中所附的LOADPIC.C中的函数调入内存并显示。

两张高密盘定价：60.00元

邮寄费：5.00元/套

汇款请寄：西安交通大学出版社

邮　　编：710049

户　　名：西安交通大学出版社

开户银行：西安市工商行互助路分理处

帐　　号：235-144260-90801

(陕)新登字007号

C语言的汉字处理与图文数据库技术

刘路放　编著

责任编辑　林全

*

西安交通大学出版社出版

(西安市咸宁西路28号　邮政编码710049)

陕西人口报印刷厂印装

陕西省新华书店经销

*

开本787×1092　1/16　印张　16　字数：390千字

1995年2月第版　1995年6月第2次印刷

印数：4001—12000

ISBN7-5605-0695-X/TP·85　定价：16.00元

前　言

C语言是一种结构化、模块化的通用程序设计语言，自其发明以来就一直以其结构简单、编码效率高、可移植性强等优点倍受系统软件开发人员的青睐。随着Turbo C、MS C等PC机版本的出现，C语言的使用者大大增加，C语言几乎代替了汇编语言而成为微机系统软件、应用软件开发的主流程序设计语言。

在我国，应用软件程序员最经常遇到的问题就是如何在应用软件中使用汉字信息。在应用软件中使用汉字要求所使用的程序设计语言的编译程序能够生成和中文操作系统兼容性好的目标程序。遗憾的是微机上常用的Turbo C、MS C等都不能达到这个要求。为了提高显示速度，微机C语言的各种版本所提供的输入输出库函数大都采用直接写屏技术显示字符，因此通过直接调用这些标准库函数编写的C语言程序编译后在中文操作系统下不能正常显示汉字。为了解决这一问题，程序开发者常常需要自行编写用于汉字输入输出的程序段，而这项工作对于缺乏相应的各种参考资料的程序员或初学者来说是相当困难的。同时在C语言中没有专门用于汉字的数据类型，只能使用原来为处理西文字符设计的字符类型或字符串类型来存放汉字。由于不得不使用两个字符型变量表示一个汉字，因此在处理中、西文符号混合存放的信息时常常会遇到"半个汉字"的问题，使得程序开发人员伤透脑筋。

本书的第一部分就是为了解决上述问题而编写的。在这一部分中，我们定义了一种新的数据类型han，用来处理汉字信息。同时编写了一整套用于han类型的头文件、宏、输入输出函数、串处理函数、窗口处理函数以及图形方式下的汉字处理函数，因而从根本上解决了使用C语言编写应用软件时的汉字处理问题。由于我们的工作是建立在BIOS中断的基础之上的，而绝大多数中文操作系统在中断一级上和MS－DOS兼容，所以使用本书提供的汉字处理功能编写的C语言应用程序编译后可以在任何中文操作系统下运行。

本书的第二部分是关于图文关系数据库管理系统C_BASE的。众所周知，目前绝大多数微机数据库应用软件和管理信息系统(MIS)都是在FOXBASE或dBASE Ⅲ＋的环境下开发的。FOXBASE之类软件具有简单易学的优点，非常适合于初学者或非计算机专业人员使用。但由于其采用解释方式工作，因此处理速度慢，不易加密(FOXBASE的伪编译和源代码加密极易破解)，不适合于用作商品应用软件的编程环境。因此，我们设计了一个以C语言为宿主语言的关系数据库管理系统C_BASE，同时为了满足多媒体应用程序开发的需要，在C_BASE系统中增加了图文字段，这种字段不但可以存储变长的字符文本信息，而且可以存放变长的图形、图象和声音等信息，极大地拓宽了数据库系统的应用范围。

实际上，C_BASE是由头文件C_BASE.H和一组库函数构成的，程序开发人员只要将其和自己的C语言联结起来，就可以直接调用C_BASE中的数据库处理函数来编写数据库应用和管理信息系统的C语言程序了。这些C程序经过编译以后就可以产生后缀名为.EXE的可执行文件，可以直接在DOS环境下运行，速度快，也容易加密。

C_BASE中的数据库文件、B＋树索引文件和FOXBASE中的数据库文件、索引文件的结构保持一致，这样有利于使用C和FOXBASE混合编程。例如我们可以使用FOXBASE编写诸如数据输入、表格打印等处理模块，而使用C_BASE来编写那些核心的处理和运算模块。由于C语言拥有极其丰富的运算、图形函数，并且可以直接调用中断和汇编代码，因此可以

充分发挥 FOXBASE 编程简单和 C 语言功能强、速度快的优点。

本书的这两部分内容是密切相关、前后衔结的。在我国目前情况下，几乎所有的应用软件都要求具有汉化的人机界面，数据库应用软件和管理信息系统尤其如此。因此我们在设计 C_BASE 时就考虑了应用 han 类型数据及其处理函数的问题，使用 C_BASE 的程序设计人员可以直接在其应用程序中应用 han 类型处理中文信息。

虽然本书的内容是建立在 Turbo C 基础上的，但其基本思想同样适用于 MS C。书中的头文件和库函数只要经过少许修改即可用于 MS C。对于富有经验的程序员来说，也可以将其移植到 UNIX 的微机版本上去。

本书是为 C 语言程序开发人员编写的，为了方便读者，我们展示了全部头文件和库函数的源程序。我们相信无论是 C 语言的初学者，还是富有经验的程序员都可以从本书中找到自己感兴趣的内容。我们也非常乐意看到有读者来信指出书中的错误之处，或是提出自己的改进意见，或是谈谈应用 han 类型和 C_BASE 系统编写应用程序的体会，甚至愿意为 han 类型的处理和编写新的 C_BASE 库函数。

为了节省读者输入程序的时间，本书配有同名软盘(5 英寸高密度软盘两张)。该软盘中包括书中所有的头文件、源程序和已经编译好的目标文件，还有在图形方式下显示宋、仿、黑、楷各体汉字所必须的 24 点阵字库和 16 点阵字库。

在 C_BASE 系统的开发过程中，西安交通大学计算机系的花蓉同学作了许多有益的工作；在本书的写作过程中，作者还得到了西安交通大学计算机系冯博琴教授的许多指点和建议，特此表示感谢。

刘路放

西安交通大学计算机系

目　　录

第 1 章　中文操作系统与 Turbo C 语言

§1.1　DOS 和 BIOS

DOS 是微型计算机上应用最广泛的操作系统。除了能执行操作人员从键盘上输入的键盘命令以外，DOS 还为程序设计人员提供了两组系统服务软件：BIOS 和 DOS 的中断和功能调用。通过使用这些中断和功能调用使得应用程序能够访问微型计算机的硬件和外部设备，包括从键盘读取字符，在显示器显示信息，读写软、硬磁盘，主机向打印机传递信息等等。

一般来说，应用程序可以通过四种方式控制微机的硬件和外部设备：直接访问硬件和外部设备，使用 BIOS 提供的功能，使用 DOS 提供的功能和使用高级语言提供的功能。这四种方式在程序设计的复杂性和程序的可移植性方面各有利弊。

使用微机的最基本的方式是利用输入和输出指令直接访问硬件和外部设备。例如，应用程序可以通过执行 OUT 指令从 I/O 端口向外发送信息和指令，也可以通过执行 IN 指令从 I/O 端口接收外部信息，从而使外部设备完成指定的操作。然而，这种程序的编写是十分繁杂的。例如当访问软盘驱动器时，接通驱动电机，移动磁头到指定位置，对一个扇区进行读写等操作都需要用复杂的设备驱动指令直接干预。如果不是为了得到更高的执行效率和获得某些 BIOS 和 DOS 不支持的功能，应用程序不应该直接和硬件以及外部设备打交道。一般来说，直接访问硬件的程序的可移植性相当差。在一个厂商生产的机器上调试通过的程序，在其它厂商生产的兼容机上可能根本无法运行，甚至在同一厂家生产的不同型号的微机上也无法运行。

BIOS(Basis Input Output System)是 DOS 中的一组低级支撑软件，它为编程人员提供了一个简单的接口，避免了应用程序直接和硬件、外部设备打交道。

BIOS 驻留在系统板上的只读存储器(ROM)中，计算机加电后就可以随时调用 BIOS 的中断服务程序。因为 BIOS 提供最基本的低层服务，所以使用 BIOS 而不直接访问硬件将使程序更加简炼。由于无论是否使用 BIOS，它总是驻留在机器中，只用几条指令就可以调用到用户所需要的 BIOS 程序，所以许多商业上取得成功的软件都采用了 BIOS 的功能调用。由于设计微机中文操作系统时的主要目标就是改造原西文 BIOS，使之能处理汉字信息，同时还要能兼容处理原有的西文软件，所以在 BIOS 中断一级上开发的中文应用软件具有相当好的可移植性。

DOS 在更高一级的层次上提供了与 BIOS 相当的功能。例如，既可以通过调用 BIOS，也可以通过调用 DOS 实现磁盘读写。调用 BIOS 时必须准确地说明读写位置(磁头、磁道和扇区号)，才能正确读写信息。但是，在 DOS 系统中已经建立了文件、目录、子目录等概念，因此调用 DOS 的应用程序就不必指出读写信息在磁盘上的物理位置，而只需打开文件，在文件内找到指定位置，就可以进行读写操作了。

一般来说使用 DOS 提供的功能比使用 BIOS 提供的功能更容易。因为使用 DOS 提供的

功能时程序设计人员不需要对硬件有更多的了解。通过调用 DOS 还可以充分利用操作系统提供的所有功能，如分级文件系统、装载和执行程序、分配内存等。但是并非 BIOS 的所有功能都能通过调用 DOS 来完成，由于在编写 DOS 时对可能调用它的情况做了某些假设，因此有些 BIOS 所提供的功能无法用 DOS 实现，例如显示器的 I/O 操作，DOS 仅仅提供了简单的字符和字符串输出能力，如果需要使用图形方式，置字符的属性和颜色，或者需要提高输出显示的速度，就必须使用 BIOS 中断，或者直接访问硬件。

在可能的情况下，尽量使用 DOS 而不是 BIOS，这样程序将具有良好的可移植性。由于 DOS 对复杂操作提供了一个简单接口，程序既容易编写，又方便调试。

如果用 C 或 Pascal 等高级语言编写应用程序，可以充分利用嵌在高级语言中的许多操作系统的功能。例如，可以直接使用高级语言提供的语句从键盘接收信息，向显示器写数据和读写磁盘文件。如果高级语言提供的功能完全能够满足编程的需要，就没有必要直接使用 BIOS 和 DOS 功能调用。这样将使应用程序具有良好的可移植性（只要把源程序在有编译程序的机器上重新编译一次即可），而且接口比较简单。但是高级语言提供的 I/O 功能比 DOS 要少，所以有些操作仅仅使用高级语言提供的语句将无法完成。

其实，在微机 C 语言（包括 Turbo C 和 MS C）中提供了一批直接调用 BIOS 和 DOS 功能调用的函数和直接嵌入汇编语言程序段的能力，为我们在应用程序的设计中根据实际情况选用最恰当的方法访问硬件和外部设备提供了灵活的手段。

§1.2 中文操作系统

众所周知，汉化微机操作系统（DOS）要解决好三个问题：汉字输入、汉字输出和汉字在计算机内的表示方法（编码）。目前绝大多数中文操作系统都是利用微机原有的输入输出设备如显示器、键盘和点阵式打印机来输入输出汉字，所以汉化 DOS 的主要工作也是修改原西文 DOS 的输入输出部分（即 BIOS），使之支持汉字的输入输出。

为了利用原来为西文输入而设计的键盘来输入汉字，就必须对汉字进行编码。这种编码称为输入码，其设计原则是易学、好记和重码率低。目前已经有数百种编码方式，其中最常用的有区位码、拼音码、五笔字型码、双拼码和苍颉码等。除了区位码以外，其他各种输入码的码长都不固定；而且都不能完全避免重码，即不同的汉字可能具有相同的输入码。

图形显示器的工作原理是将要输出的符号的点阵图形在显示器上显示出来。通常西文 DOS 把西文符号（ASCII 码）分解为 8×8，8×14 或 8×16 点阵。由于汉字的结构比西文符号复杂，所以通常在汉字操作系统中用 16×16 点阵表示汉字。点阵中的一个点需要一位二进制码表示，因此一个汉字的点阵就需要 32 个字节（每个字节为 8 个二进制位）才能存放。通常中文操作系统将所用到的所有汉字和符号（大约 7 000 个左右）的显示点阵信息存放在一个文件中或者固化在汉字卡中的只读存储器中，称为显示汉字库。为了有比较好的输出效果，用于打印机的汉字点阵通常还要大些，例如 24×24 点阵，48×48 点阵等。比较完善的中文操作系统还备有不同字体的打印字库，可以得到更加完美的输出效果。

汉字输入码的码长不固定，而且可能有重码，输出点阵占用的存储空间太大，它们都不适合于在计算机作为汉字的存储和处理代码。因此，在中文操作系统中，还有一种机内码用来表示汉字。为了使不同的中文操作系统能够交流含有汉字信息的软件和数据，就必须制定

一套统一的汉字编码方案。

《汉字信息交换通用代码字符集 GB2312－80》就是我国制定的一套汉字编码方案。在 GB2312－80 中，一共规定了 6 000 多个汉字和 300 多个符号的代码。每个汉字或符号的代码由两部分组成，第一部分称为区码，第二部分称为位码，每一部分占 7 个二进制位。通常把这种汉字代码称为国标码，又称区位码。

为了和微机原有的西文处理功能兼容，通常中文操作系统使用两个字节存放一个国标码。由于表示西文字符的 ASCII 码只使用了一个字节的低 7 位，最高位为 0，所以用两个最高位为 1 的字节存放国标码。这样就可以在计算机中同时处理西文 ASCII 码和中文国标码了。这种最高位为 1 的国标码也称为异型国标码。

中文操作系统的主要任务就是接收使用者从键盘上键入的输入码，并将其转换为汉字机内码(异型国标码)和根据汉字机内码在显示屏上显示出(或者在打印机上打印出)对应的汉字点阵来。

本书所介绍的 han 系统是为在 Turbo C 语言中处理汉字而设计的，为了处理方便，使用了一种由国标码变形而来的 16 位的汉字编码，其构成和使用方法见第二章。

§1.3 Turbo C

Turbo C 是美国 Borland 公司在微机上实现的一个高效、优化的 C 编译程序。除了标准 C 语言的功能以外，TurboXIVC 还增加了许多和微机操作系统 DOS 有关的库函数和功能，其中中断调用函数、准变量、存储模式和图形工作方式等和本书内容的关系比较密切。下面我们简要介绍这些内容。

一、中断调用函数

在 Turbo C 中可以使用几种不同的方法调用 DOS 或 BIOS 中断。第一种方法是使用 int86 函数：

```
#include <dos.h>

int int86(int intr_num,union REGS *inregs,union REGS *outregs);
int int86x (int intr_num,union REGS *inregs,union REGS *outregs,
        struct SREGS *segregs);
```

这两个函数都执行一个由参数 intr_num 所指定的 8086 软件中断。执行软件中断前，这两个函数都把 inregs 中的寄存器值复制到寄存器中。另外，在执行软件中断前，int86x 还把 segregs 的值复制到段寄存器中去，因此可以用于调用需要使用段寄存器的中断。联合类型 REGS 和结构类型 SREGS 的定义在头文件 dos.h 中：

```
struct WORDREGS
{
```

```
  unsigned int ax, bx, cx, dx, si, di, cflag, flags;
};

struct BYTEREGS
{
  unsigned char al, ah, bl, bh, cl, ch, dl, dh;
};

union REGS
{
  struct WORDREGS x;
  struct BYTEREGS h;
};

struct SREGS
{
  unsigned int es;
  unsigned int cs;
  unsigned int ss;
  unsigned int ds;
};
```

函数 int86 和 int86x 使用方便，是调用中断时比较常用的方法。但是比起下文要说明的使用伪变量和 geninterrupt 函数的方法来说，速度稍微慢了一点。如果需要调用 DOS 中断 0x21，也可以使用以下两个函数：

```
int intdos(union REGS * inregs,union REGS * outregs);
int intdosx (union REGS * inregs,union REGS * outregs, struct SREGS * seg regs);
```

调用时参数 inregs 中的域 h.al 的值指定了所要调用的功能。在中断 0x21 返回以后，这两个函数都把当前寄存器的值复制到 outregs 中去，并且把系统进位标志状态复制到 outregs 中的 x.flag 域中。另外，函数 intdosx 还恢复数据段寄存器 DS 的值。

例 1-1 清屏。

```
#include <dos.h>

main()
{
  union REGS r;
```

```
    r.h.ah=0x06;          /* 窗口上滚        */
    r.h.al=0;             /* 清屏            */
    r.h.cl=0;             /* 窗口左上列位置  */
    r.h.ch=0;             /* 窗口左上行位置  */
    r.h.dl=79;            /* 窗口右下列位置  */
    r.h.dh=24;            /* 窗口右下行位置  */
    int86(0x10,&r,&r);    /* 调用显示器中断  */
}
```

该程序执行清屏功能。

除了以上两组通用中断调用函数以外，Turbo C 中还有许多其他中断调用函数，这里就不一一列举了。

二、伪变量

Turbo C 允许使用伪变量直接访问 8086 寄存器。表 1-1 中列出了所有 Turbo C 中可用的伪变量及其类型和相应的寄存器。

表 1-1 Torbo C 中的伪变量

伪变量	类　型	寄存器	基本用途
_AX	unsigned int	AX	通用寄存器/累加器
_AL	unsigned cShar	AL	AX 低字节
_AH	unsigned char	AH	AX 高字节
_BX	unsigned int	BX	通用寄存器/变址器
_BL	unsigned char	BL	BX 低字节
_BH	unsigned char	BH	BX 高字节
_CX	unsigned int	CX	通用寄存器/计数和循环
_CL	unsigned char	CL	CX 低字节
_CH	unsigned char	CH	CX 高字节
_DX	unsigned int	DX	通用寄存器/存放数据
_DL	unsigned char	DL	DX 低字节
_DH	unsigned char	DH	DX 高字节
_CS	unsigned int	CS	代码段地址
_DS	unsigned int	DS	数据段地址
_SS	unsigned int	SS	栈段地址
_ES	unsigned int	ES	附加段地址
_SP	unsigned int	SP	栈指针(SS 位移)
_BP	unsigned int	BP	基指针(SS 位移)
_DI	unsigned int	DI	用于寄存器变量
_SI	unsigned int	SI	用于寄存器变量

寄存器变量可以和函数 geninterrupt 配合使用调用 DOS 中断和 BIOS 中断。geninterrupt 函数的调用方法为：

```
void geninterrupt(int intr_num);
```

其中参数 intr_num 指出所要调用的中断号。而调用中断时各个寄存器的值可以通过使用伪变量来设置。

例 1-2 清屏。

```
#include <dos.h>

main()
{
  _CL=0;              /* 窗口左上列位置   */
  _CH=0;              /* 窗口左上行位置   */
  _DL=79;             /* 窗口右下列位置   */
  _DH=24;             /* 窗口右下行位置   */
  _AL=0;              /* 清屏             */
  _AH=0x06;           /* 窗口上滚         */
  geninterrupt(0x10); /* 调用显示器中断   */
}
```

该程序的执行结果和实例 1-1 相同。

一般情况下，采用伪变量和 genintrrrupt 函数配合方式调用中断要比使用中断调用函数 int86 等的速度快，所以在本书介绍的各个函数中调用中断时均使用伪变量和 geninterrupt 函数。

三、直接插入汇编代码

在 Turbo C 中，还可以在程序中直接插入汇编代码。由于在本书中没有用到这方面的内容，所以在此不作详细介绍。对此有兴趣的读者可以参考有关书籍的内容。

四、指针和存储模式

Intelx 86 系列微处理器采用分段内存结构，其总地址空间可达 1M 字节。但在设计上只能同时访问 64K 字节，即一个段。这是因为 1M 内存的地址空间需要 20 位的地址数，而 Intel 86 系列是 16 位微处理器，各寄存器均只有 16 位，所以在设计上采用了用 16 位的段寄存器和 16 位的偏移寄存器组合起来的方法表示内存地址，如图 1-1 所示。

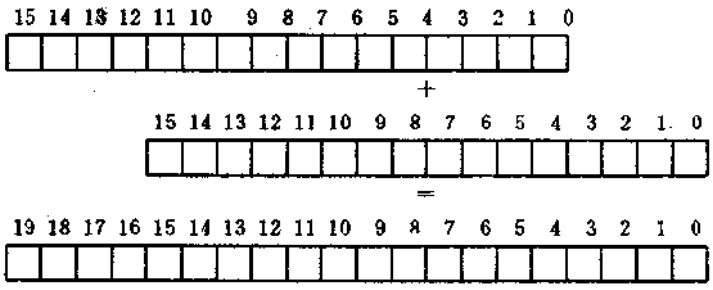

图 1-1 86 系列微处理器的地址构成方法

从图 1-1 可以看出，实际内存地址是由将段寄存器的内容左移 4 位后再加上位移后构成的。因此，我们通常将地址写为段：偏移的形式。

在 Turbo C 中是通过存储模式和指针来实现 20 位地址调用的。指针共有三类，即近指针(16 位)、远指针(32 位)和特大指针(32 位)。

16 位的近指针的地址计算依赖于某一段寄存器。例如近数据指针的地址是将其 16 位的值加上数据段寄存器(DS)左移 4 位后的值，而函数指针的地址是相对于代码段寄存器(CS)的位移。使用近指针只能访问 64K 以内的地址空间，但其使用方便，适合小数据量和小程序使用。

32 位的远指针由 16 位的段地址和 16 位的偏移组成，因此可以同时访问多个代码段或数据段。但是使用远指针时要注意几个问题。首先，由于同一内存地址可以表示为不同的地址形式，例如远指针(0000：0120)、(0010：0020)和(0012：0000)实际上指的使同一内存地址。但是如果我们使用"==="比较上述远指针时，会发现结果是不等的。同样，在使用比较运算符">"，"<"，">="，"<="和"<>"也有类似的问题。实际上，在比较远指针时只有位移参加了运算。另外还要注意的是，如果加一数到远指针，那么只会改变位移。要是加的数大到使位移超过 FFFF(最大可能值)，指针又回到段的开始位置；例如 1234：FFFF 加 1 的结果为 1234：0000，而不是 2234：0000。这在大多数情况下是不希望的。因此，如果需要用到指针的比较时，最安全的方法是使用近指针或特大指针。

特大指针和远指针一样，也是 32 位，含有段地址和偏移。但是特大指针是经过规格化处理的，可以解决远指针存在的问题。规格化的方法是先将远指针的 32 位地址转换为 20 位地址，然后取左边 16 位作为段地址，余下 4 位即位移。因此对于 1M 内存中的任意地址，只有唯一一个特大指针与之对应。这样，就可以保证特大指针地址比较结果正确。在使用特大指针运算时，同样是有 32 位参加运算，可以保证在处理大于 64K 的数据结构时不出错误。但是由于处理特大指针需要调用专用子程序，所以特大指针的处理要比处理近指针或远指针慢。

为了提高编程的灵活性，Turbo C 提供了 6 种存储模式可供编程人员选择：

1. 极小模式：各段寄存器(CS，DS，SS 和 ES)均取相同的地址值，所以代码和数据的总空间为 64K。极小模式使用近指针。使用极小模式的程序经编译后生成的可执行文件可以转换成.COM 格式。

2. 小模式：数据段和代码段不同且不重叠，所以可以有 64K 代码和 64K 静态数据。栈段(SS)、附加段(ES)和数据段(DS)取相同值。小模式使用近指针，适用于一般应用问题。

3. 中模式：代码使用远指针而数据使用近指针。所以静态数据最多可以使用 64K，而代码长度可以超过 64K。适合于大程序、小数据的情况。

4. 紧缩模式：和中模式正好相反，数据使用远指针而代码使用近指针。因此适用于小程序、大数据的情况。

5. 大模式：代码和数据均使用远指针，所以可以处理很大的应用程序。

6. 巨大模式：代码和数据均使用远指针。Turbo C 限制静态数据最多为 64K，而巨大模式可以打破这一限制。

在紧缩模式、大模式和巨大模式中，可以使用特大指针。特大指针的使用实例可以参看本书第 4 章 § 4.6：中西文全文检索工具。

本书的大部分例题都使用小模式编译。只有少数应用程序是采用远数据指针模式，如紧缩模式和大模式编译的。

五、显示器管理

在 Turbo C 中提供了两类显示器管理工作模式，以弥补标准 C 中只有电传方式的终端设备的不足。一类是字符屏幕管理方式，一类是图形工作方式。由于字符屏幕管理方式和许多中文操作系统不兼容，所以我们在第四章中提供了一批相应的汉字处理函数。Turbo C 的图形工作方式下的字符串输出函数也不能处理汉字，因此在第五章中专门解决图形方式下汉字的输入输出问题。

第 2 章　han 类型

由于各种程序设计语言(包括 C 语言在内)都没有专门的用于处理中文信息的数据类型,所以通常在编写程序时都是利用字符串类型存放汉字或汉字串。在中文操作系统中每个汉字的机内码占用两个字节,而每个西文字符(ASCII 码)只占用一个字节,因此程序设计人员经常会遇到如何切分字符串中的汉字的问题,特别是处理中、西文混合的字符串变量时,"半个汉字"是一个令人感到十分头痛的问题。

问题的关键是字符串类型的基本组成单位是字符(一个字节)。我们能否使用某种基本组成单位就是两个字节的数据类型呢?比如说,C 语言中的 int 类型就是占用两个字节的数据类型。我们可以使用一个 int 型的变量存放单个汉字的机内码,而使用 int 类型的数组存放汉字串。但是这样做会出现下列问题:第一,C 语言拥有的非常丰富的字符串处理库函数就无法利用了;第二,C 语言用于文件 I/O 和控制台 I/O 的库函数也无法用来处理这种汉字或汉字串。这个问题比第一个问题更加严重,因为大多数中文处理应用软件离不开汉字的输入和输出操作;第三,中、西文混合的信息很难处理。

为了解决以上三个问题,我们编写了一整套专门应用于汉字和汉字串处理的 C 语言系统软件,包括头文件 han.h 和近百个库函数。只要把这些软件加入你的 Turbo C 中去,你就可以在编写自己的 C 语言应用程序时非常轻松地处理汉字信息了。

本章和以后几章就分别介绍头文件 han.h 和这些库函数的内容。

§2.1　han 类型的定义

我们定义 han 类型为:

```
typedef int han;
```

即把 han 类型定义为 int 类型。由于在 C 语言中 char 类型和 int 类型等价,所以我们的 han 类型也可以兼容处理字符类型,反之原来用于 char 类型处理的许多函数也可以用于 han 类型,这一点是十分重要的。但是要注意的是,在 C 语言中 int 类型和 char 数组类型(也就是字符串类型)是不同的,前者占用两个字节而在后者中每个字符只占用一个字节。

为了在表示汉字时使 han 类型的最高位保持为 0(即在把 han 类型看成是 int 类型时其值为一正整数,这在许多情况下是有利的),我们对组成一个汉字内码的两个字节作如下处理:

第二个字节的低 7 位存放汉字的位码,第一个字节的低 6 位和第二个字节的最高位(第 7 位)存放区码,第一个字节的次高位(第 14 位)置 1,而第一字节的最高位(第 15 位)总是置为 0,如图 2-1 所示。

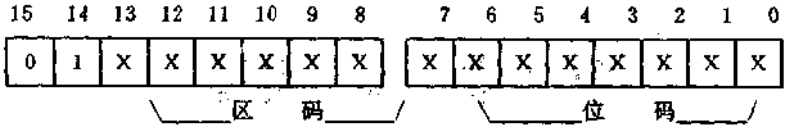

图 2-1 汉字在 han 类型中的表示

同时，为了在应用 han 类型时能够兼容处理 ASCII 码、扩展 ASCII 码和键盘扫描码，我们规定：如果一个 han 类型的变量的值小于 128，那么就表示存储的是一个 ASCII 码，如果其值在 128－255 之间，就表示存储的是一个扩展 ASCII 码，如果其值在 256－511 之间，就表示存储的是一个键盘扫描码，如图 2-2 所示。

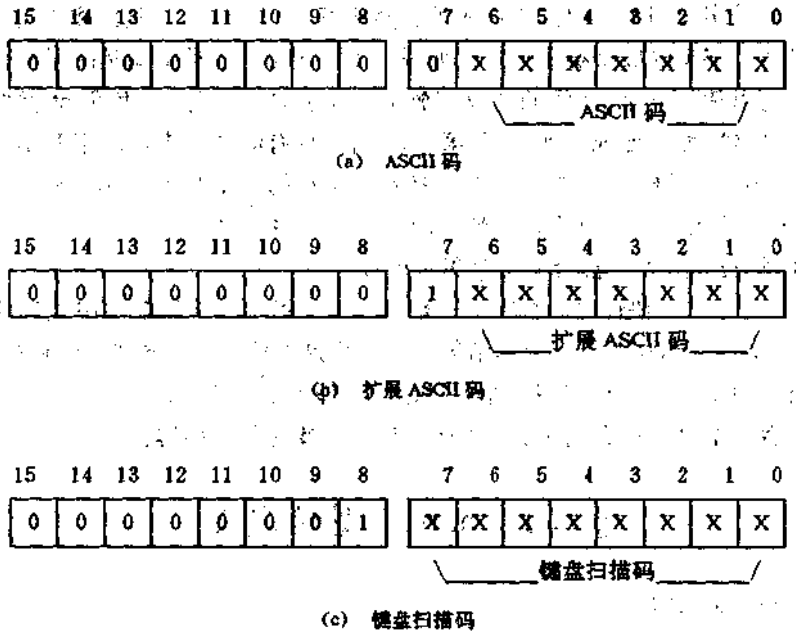

图 2-2 (扩展)ASCII 码和键盘扫描码在 han 类型中的表示

有了 han 类型以后，就可以定义 han 类型的变量了。

例 2-1 han 类型变量。

```
han h=0;
han key,code;
```

我们也可以定义 han 类型的数组和指向 han 类型的指针。如例 2-2。

例 2-2 han 类型的数组和指针。

```
han hstr [20],name [MAX_NAME];
han *hp,*hq;
```