软件改造技术

陆学斌 编著

华中师范大学出版社

软件改造技术

陆学斌　编著

华中师范大学出版社

鄂新登字11号

内 容 简 介

本书系统地阐述了MS－DOS操作系统下软件剖析和改造的技术，重点结合软件改造的主要领域—软件的汉化改造详细讲述了发现、分析以及解决问题的办法。其中对软件的剖析技术作了尤其详细的阐述。这些技术不仅对软件汉化适用，对软件的解密、修改等同样具有指导意义。全书注重实用性和技术性，对所讲方法均列举了大量实例加以说明，使读者容易理解和接受。

读者对象：广大计算机软件开发及应用人员。

软件改造技术
陆学斌 编著
*
华中师范大学出版社出版发行
（武昌桂子山）
新华书店湖北发行所经销
武汉测绘科技大学印刷厂印刷
*
开本 787×1092 1/16 印张 9.5 字数 236千字
1993年1月第1版 1993年1月第1次印刷
ISBN 7—5622—0947—2/T·02
印数：1—1000 定价：8.50元

前　言

软件改造是一项很实用的技术。我们在日常工作中，经常会碰到这样或那样的问题。例如：软件在某种机器、某种 CCDOS 上运行地很好，而换一个机器或操作系统就难以运行；发现机器已染病毒，您的软件已被破坏，但杀死该病毒的软件新版本尚未面世；买来的软件虽然功能还可以，但其有些操作方法却与您的习惯不符等等。事实上，大部分的问题通过改造软件是完全可以解决的。因此，如果我们掌握了软件改造的技术，将给我们的工作带来极大的便利。

软件改造也是一项很有价值的技术。通过剖析改造一个软件，不仅可以学到别人特别是国外的先进的程序设计方法，还可以获得大量的 IBM 和 MICROSOFT 秘而不宣的宝贵资料，从而提高您的开发水平。而通过汉化改造国外优秀的软件，使其洋为中用的话，就更有意义了。例如，众所周知、沿用至今的中文 WORDSTAR 和汉字 dBASEIII、FOXBASE，经过汉化之后，在不长的时间里便风靡全国，对促进我国计算机技术的普及、提高我国计算机应用水平做出了不可磨灭的贡献。

然而长期以来，这项技术却不为大多数人所掌握。究其原因，主要在于论述的专著太少，特别是关于软件剖析的方法论更是几乎无人提及。故而，初学者虽有修改软件的愿望，但面对浩如烟海的汇编语言指令，不是不知该从何处着手而望"洋"兴叹，便是因为没有正确的方法作指引，最终在茫茫的汇编语言指令大海之中迷失方向。鉴于这种现状，笔者不揣浅陋，系统地归纳和总结多年来从事软件汉化改造的经验写成此书，希望能给读者一点启迪和参考。

全书共分为五章，第一章讨论了汉化的必要性，讲述了西文软件不适应汉字处理的原因。第二章讲述了汉化的准备知识及工具，着重讲述了符号调试工具 SYMDEBUG 的用法。第三章讲述如何发现软件里存在的不适应汉字处理方面的问题。第四章讨论了软件剖析的技巧和方法，对软件剖析的各种技巧结合具体实例作了详细地阐述，并对 EXE 和 COM 文件的结构作了透彻的分析。第五章讨论了对 EXE 和 COM 文件具体改造的方法。此书虽然是结合软件的汉化来阐述方法的，但对软件的解密、修改同样适用。书中所述方法均结合大量流行软件举例加以阐述，使读者容易理解和接受。本书还对 MS－DOS、CCDOS、C 语言等的内部实现作了比较深入地分析。

本书初稿的内容已在两届研究生中讲述过，但最终能得以整理成书，归功于梁妙圆教授的热情鼓励和支持。没有她的帮助，本书是不可能完成的。赵永仪老师、刘武、陈云杰、任前尧、王玉等同学在书稿的打印过程中，也给了我很大的帮助，在此一并致谢！

作　者

1991 年 10 月于武昌桂子山

目　录

第一章　汉化问题的提出

引　言

近年来，随着 IBM－PC 系列机在我国的广泛流行，各种软件也随之涌现而来。由于我国软件产业起步较晚，加上一些客观条件的限制，目前能自主开发的软件尚不多见。目前使用的大部分软件都是从国外（主要是美国等英语国家）引进的。鉴于中国用户的普遍水平，直接使用这些西文软件不仅存在着语言障碍，而且也不符合中国人使用汉语的习惯。因而，国外不少优秀的、成熟的软件很难在我国推广使用。汉化正是在这种情况下应运而生了。它要做的工作便是针对我国的国情，把这些西文软件加以适当的改造，在保持其原有功能的前提下，使其具备处理汉字的功能。西文软件经过汉化后，便与一个由全中文开发的软件在使用时的感觉大致相同，而且其开发周期短，因而深受我国广大用户的欢迎。通过汉化西文软件，不仅可以加速各类应用软件的普及，提高我国计算机应用的水平，还可以从分析别人系统的过程中，学到国外先进的程序设计方法并获得大量的宝贵资料，进而提高我国计算机软件的开发水平。因此，汉化是一件十分有意义的工作。

§1　汉化的任务

汉化具体要解决哪些问题呢？目前的 IBM－PC 系列机和国产的 0520 系列机，其软硬件环境都是西文的。西文软件可直接在上面运行。为能在这种西文计算机上运行中文软件，首先必须构造一个中文的操作系统环境，以具备汉字的输入、输出功能。然后，在中文操作系统的支持下，对西文软件进行汉化改造，使其具备汉字功能，即：汉字的输入输出、汉字信息的处理及传输、提示信息的汉字化等。这便是汉化要完成的两大任务。一般说来，构造中文操作系统环境的工作不需要我们去做。因为目前已有专门机构研制通用的汉字操作系统，另建一个不仅花费时间，而且用户不一定能够接受，因此，一般是利用现有的汉字操作系统。故后者才是我们要做的主要工作。

§2　CCDOS 实现原理

由于汉化是在 CCDOS 的基础上实现的，因此有必要对 CCDOS 的实现过程做简单的介绍。

2.1　汉字机内码的设计

汉字机内码（亦称汉字内码）是计算机系统内部处理和存储汉字而使用的代码。我们知

道，西文 ASCII 码是用一个字节来表示，一般只使用了一个字节的第一至第七位，而把高位作为奇偶校验或不用。1981 年，我国颁布了国家标准 GB2312－80 即《信息交换用汉字编码字符集》，规定了在计算机等信息处理领域使用的 6763 个汉字及 682 个图形字符。这些汉字和图形字符以区位方式存放。其中，第 1～9 区为图形符号，第 10～15 区是空白，留待将来扩充，第 16～55 区为一级字库，包含 3755 个汉字，第 56～87 区为二级字库，包含 3008 个一般常用汉字。GB2312－80 还规定，一个汉字用两个字节表示，每个字节也只用 7 位，高位未作定义。

为了保证系统的中西文兼容，系统的机内码必须在保持 ASCII 码使用的同时，允许汉字机内码的使用，而且两者之间没有冲突。如果直接用 GB2312－80 中的国标码作为汉字内码，则在系统中同时使用 ASCII 码和汉字时，将产生歧义性。例如，机内有两个字节的内容分别为 30H 和 21H，它们既可表示汉字“啊”的国标码，又可表示西文字符“0”和“1”的 ASCII 码。所以，为了区别 ASCII 码和汉字内码，有必要采用某种标志。目前经常使用的有两种汉字标志方式：

1. 引导符方式

2. 双字节 8 位、高位均置“1”的方式

所谓引导符方式，就是用一个特殊字符附在一个汉字或汉字字符串的开始和结束位置，以表示这个特殊字符之后跟的是汉字而不是 ASCII 码。这种方式的优点是原西文软件不做任何改造也能在一定程度上处理汉字，然而却存在着严重的问题：

1. 各个系统采用的引导符可能不同，因而数据资源不便共享；

2. 浪费存储空间，在一个汉字或汉字字符串上至少要加两个额外的字符，增加了系统存储的开销。

除此之外，引导符还存在着易与系统提供的特殊字符引起混淆及汉字编码的实际长度与其内部的长度不符等问题。因而目前已不多见这种方法。

目前大部分 CCDOS 及汉字终端上都采用的是双字节表示一个汉字、高位均置“1”的方法。这种标志方法既最大限度的节省了存储空间，又实现了与 ASCII 码的兼容与区分，被公认为是汉字内码的最佳表示方式。然而，由于其高位均置“1”，与 ASCII 码不同，因而提出了汉化改造的任务。

2.2 CCDOS 的实现方法

IBM－PC 系统的 BIOS（基本输入输出系统）存放于主板上的 ROM 中，所以又称为 ROM－BIOS。ROM－BIOS 由若干独立的外部设备驱动程序模块组成，各个模块的入口地址均被存放于绝对地址 0000:0000H～0000:03FFH 的中断向量表中。用户需使用这些外设时，直接发中断调用即可。例如，西文的输入是通过 9＃和 16H＃中断，显示是通过 10H＃中断、打印是通过 17H＃中断来分别实现的。不同的机器，硬件结构可能不同，但对外设调用都是使用同样的中断号，因而，外设的更改和增减都对用户透明。

根据上述情况，CCDOS 通过扩充 ROM－BIOS 中与汉字 I/O 相关的中断从而实现了汉字的输入输出。下面依次简述。

1. 汉字输入的实现。西文字符的输入过程是这样的：用户在键盘上按下一个键，计算机便产生一个 9＃中断，这个中断自动调用 9＃中断服务程序（在 ROM－BIOS 里），把按下的这个键加以正确识别，然后生成该键的 ASCII 码和扫描码，送到 0040H:001EH～0040H:

003DH 的键盘队列缓冲区中。应用程序要读取键盘时，再调用 16H＃中断的 0＃功能即可。16H＃中断的功能主要是从键盘队列缓冲区中取一个 ASCII 码和扫描码返给调用者。由于应用程序读取键盘时一般是调用 16H＃中断（只有极少数程序和多数游戏直接调用 9＃中断），故而在这一层替换，使其具备汉字功能的话，汉字的输入就实现了。目前国内流行的 CCDOS 大多数是替换 16H＃中断，台湾著名中文系统 ET 汉字系统则把这两个都替换了。

汉字 16H＃中断的实现原理很简单：调用原来的 16H＃中断服务程序或直接从键盘队列缓冲区里取一个 ASCII 码和一个扫描码，然后根据当前的输入状态进行处理，如为 ASCII 码输入方式，则直接返给调用者，否则根据当前的汉字输入方式（区位、五笔、拼音等）进行计算或查表，把这些 ASCII 码转换成汉字机内码，再分两次返给调用者。第一次返回高位，第二次返回低位。

2. 汉字显示的实现。西文显示是通过调用 10H＃中断的各个功能来实现的。其核心部分是：在文本方式下，往显示缓冲区内写入要显示字符的 ASCII 码和其属性值（即颜色）来实现显示。而汉字 10H＃中断为实现汉字的显示，则是把显示卡设为图形方式后，把要显示的汉字和 ASCII 码通过计算，得到其字模的地址后，再把相应的字模数据写入到显示缓冲区来实现的。因此，汉字和西文实现显示的过程是完全不同的，前者必须在图形方式下，而后者是在文本方式下实现的。因而，对于显示，汉化要做的工作比较多。

3. 汉字打印输出的实现。西文打印输出是通过 17H＃中断的 0＃功能来实现的，要打印一个西文字符，只需要把该字符的 ASCII 码送到打印机端口即可。汉字 17H.＃功能则是把打印机置为图形打印方式后，再把要打印的汉字或西文字符通过计算，得到打印字模地址，再把字模数据送到打印机来实现汉字及 ASCII 字符打印的，因此，汉字与西文打印的实现方式也有着截然的差别。

§3 汉化改造的必要性

西文软件在汉字操作系统环境下，为什么还要经过汉化改造才能处理汉字呢？这要从汉字的内码表示方法和汉字的输入输出谈起。

3.1 汉字内码表示方法存在的问题

前面已经谈过，采用双字节高位置“1”的方法进行内部编码有很多优点，而且从表面上看，这种编码方式与 ASCII 编码不会发生冲突。但是，仔细分析一下西文软件，就会发现这种方式存在着以下几个方面的问题：

1. 汉字由两个字节组成，而西文只要一个字节就可表示，因此，西文软件处理西文字符时，是以字符为单位来进行的，而这种处理方式对于汉字来说，无疑会产生错误。

2. 汉字内码高位置“1”的方式有时与西文字符产生混淆。某些西文软件也把普通西文字符高位置“1”作为特殊 ASCII 字符进行处理，例如，TURBO PASCAL 3.0 版显示主菜单时，有些字符需要高亮度显示以示其为关键字如 Compile 的“C”字，就把“C”的 ASCII 码 43H 高位置“1”而成为 C3H，变成了特殊字符，而其余的“ompile”的 ASCII 值不变，这样便可区别显示它们。而在汉字操作系统下，所有高位置“1”的 ASCII 码字符将被解释为汉字，因此，在这种情况下，西文字符将会被错误的处理。

3. 由于汉字内码值＞80H，不同于常规的 ASCII 码字符，有些西文软件将对其进行屏

蔽，而在另外一些软件里，汉字输入将被禁止。

例如，DEBUG 的"D"命令显示内存内容时，ASCII 字符在被显示之前，将经过下面的处理：

```
280A:0AFD AC        LODSB          ;取一个字符
280A:0AFE 3C7F      CMP    AL,7F   ;
280A:0B00 7304      JNB    0B06    ;大于 7FH 转 0B06
280A:0B02 3C20      CMP    AL,20   ;
280A:0B04 7302      JNB    0B08    ;大于 20H 转 0B08
280A:0B06 B02E      MOV    AL,2E   ;用句号'.'代替
280A:0B08 AA        STOSB          ;保存供显示
280A:0B09 E2F2      LOOP   0AFD    ;循环直至完
```

可见，每个 ASCII 字符如果＞7FH，则都会被替换成"."，这种处理对于普通 ASCII 字符不会出错，因为普通 ASCII 字符的 ASCII 值均＜80H，但恰恰所有汉字的 ASCII 值均＞7FH，故所有的汉字都将被屏蔽。在用 PCTOOLS 的 Edit/View 命令对文件进行编辑时，汉字也是无法输入的，因为该软件对输入的字符也有一道加工，如发现输入的字符其 ASCII 值＞7FH，就鸣笛警告，并拒绝接受该字符。

4. 西文软件对标识符的定义排除了汉字作为其组成元素的合法性。所谓标识符，是指文件名、数据库名、字段名、变量名等。西文软件对标识符的定义如下：

〈标识符〉::=〈英文字母〉|〈下划线〉[[〈英文字母〉|〈下划线〉|〈数字〉]]

〈英文字母〉::=〈小写字母〉|〈大写字母〉

〈大写字母〉::=A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

〈小写字母〉::=a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

〈数字〉::=0|1|2|3|4|5|6|7|8|9

〈下划线〉::=—

由此可见，汉字不能作为合法的标识符，例如 dBASE Ⅲ 的字段名便不能使用汉字。因此，对标识符的规定需要加以改造，使之成为下面的定义：

〈标识符〉::=〈英文字母〉|〈下划线〉|〈汉字〉[[〈英文字母〉|〈数字〉|〈汉字〉]]

其中，英文字母、数字、下划线的定义如前面所述，汉字的定义如下：

〈汉字〉::=〈1～9 区 682 个图形符号〉|〈16～87 区全部 6763 个汉字〉

3.2 汉字的显示及打印输出存在的问题

前面讨论了由于汉字内码的表示方法引起的问题，下面让我们看看汉字的显示和打印输出存在的问题。

如前所述，CCDOS 实现字符显示用的是图形方式，这与西文显示所用的文本方式截然不同。但是，如果西文软件显示字符是调用 10H＃中断的话，则在汉字环境下仍然是可以显示西文或汉字的。例如，DOS 的内部和外部命令、GWBASIC、BASICA 等常见软件，便是调用 10H＃中断实现显示的，故无论在西文态还是在中文态，这些软件一样能显示字符。但是现在越来越多的软件为了提高显示速度，采用了直接对硬件进行操作而避开 10H＃中断的方法。这种方法的实质在于往显示缓冲区内写入需要显示的字符的 ASCII 码和属性值，因而，这种方法在汉字环境的图形方式下是根本不能显示字符的。例如，西文 WORDSTAR，在西文方式下显示的是正确的西文信息，而在 CCDOS 的 10 行显示方式下，屏幕上出现的是一些类

似虚线的东西。这是因为在中文方式下，把这些 ASCII 码解释成了图形点阵信息的缘故。除此之外，CCDOS 的显示方式对西文软件还存在着下列问题：

1. 西文方式下，一屏一般能显示 25 行西文字符(EGA、VGA 特有的方式下，一屏甚至能够显示 43 行乃至更多的西文字符)，而 CCDOS 一般达不到这些要求。如 CGA 卡只能显示 11(实际 10)行汉字，MDA 卡 25(实际 24)行汉字，EGA 卡一般只能显示 24 行(UCDOS 采用压缩字模及提示行采用弹出式窗口的技术达到了 25 行)，VGA 卡作到了显示 25 行。因此，原软件 25 行的运行环境移植到这些显示卡上时，必然要做改造，否则就会出现我们经常看到的屏幕翻滚现象。

2. 边界问题。由于汉字是由两个字节组成的，这两个字节是一个不可分开的整体，而西文软件处理西文字符是以一个字节为单位进行处理的，故汉字往往在边界处被分成两半，形成所谓“边界问题”。

3. 窗口边框问题。现在流行的西文软件都采用了窗口工作方式如下拉式、弹出式等，这些窗口一般都有边框。而边框字符的 ASCII 码值均>80H，与汉字具有相同的特性。例如横线“═”的 ASCII 码值为 CDH，因而在显示时，两个横线字符连在一起就被汉字显示模块解释为 CDCDH 即为“屯”字而加以显示，相信读者对此现象不会陌生。

西文软件的打印输出碰到的问题要少一些，这是因为绝大多数西文软件在打印输出时都是调用 17H＃中断来实现的，故而在 17H＃中断被 CCDOS 的汉字 17H＃中断取代后，汉字便可在打印机上输出了。少数软件如 WORDSTAR 等绕过了 17H＃中断而直接对打印机适配器端口进行操作，这时其打印功能必须进行汉化改造。

上面讲述的各种问题统属于西文软件在逻辑上不适应汉字处理方面的问题，西文软件还存在另一大问题即提示信息也必须汉化。因为英文软件一般都有大量英语提示信息，特别是数据库、字处理等软件提示信息更多，作为一个高质量的汉化软件，这些提示信息应改变成准确的中文信息。

第二章　汉化的准备知识及工具

"工欲善其事，必先利其器"。在汉化西文软件之前，必须具备必要的知识并掌握一定的工具。本章主要讲述汉化的基本知识及常用工具的用法。

§1　汉化的准备知识

汉化是一项综和性的技术，它的实质是对被改造的西文软件的理解。而西文软件的功能则千差万别：操作系统、数据库、管理信息系统、CAD、各类工具软件、各种语言的编辑及编译工具等等，几乎涉及到 IBM－PC 软硬件的各个领域。因此，这就要求我们具备比较全面的知识，这些知识包括：

1. IBM－PC 的硬件知识。对 IBM－PC 的硬件组成有一个大致的了解，而对系统 BIOS 则要有很清楚的了解。特别是其中常用的中断调用如 10H＃（显示）、16H＃、9H＃（键盘输入）、17H＃（打印）、13H＃（磁盘 I/O）等中断更要重点掌握。关于 IBM－PC 的硬件知识读者可参阅张载鸿的《IBM－PC/XT 软硬件系统分析与应用》一书，这本书讲的较为完整。除此之外，还可参阅 IBM－PC 的硬件手册。对于 ROM－BIOS，IBM－PC 硬件手册后附有全部的 8088 宏汇编语言程序清单，并附有详细的中文注释，读者如能全部通读一遍，相信会大有裨益的。

2. 8088 宏汇编语言的知识。汉化基本上全是与汇编语言打交道，因此，IBM－PC 的汇编语言是汉化的最基本知识。同时，读者最好能够掌握一到两门高级语言如 C、PASCAL、BASIC 等，其中，尤以 C 语言最为重要。因为现在大多数的软件都是用 C 语言开发的，如 dBASE Ⅲ、FOXBASE 等我们熟悉的数据库管理系统软件，就是用 C 语言和汇编语言混合编程的。我们掌握了 C 语言后，一方面可以熟悉它的编程风格和程序的结构，帮助我们在以后的汉化中分析、理解由这种语言开发的软件；另一方面，在汉化过程中，经常要自己开发一些汉化辅助工具，借助于这些高级语言，可以大大提高开发的效率。

目前市面上流行的有两种 C 语言版本，一种是 Borland 公司的 Turbo C，另一种是 Microsoft 公司的 MSC 及 Quick C。这两种版本各有千秋，其中，Turbo C 的函数功能更多一些，而且其运行速度也比 Quick C 快。其界面比较友好，很受初学者喜爱。但好象也有一种说法，认为 Turbo C 的稳定性不如 Quick C。笔者更[illegible] MSC，因为宏汇编语言最早是由 Microsoft 公司开发的，因而与 MSC（或 Quick C）混合编[illegible]较好。而且，笔者看到很多西文软件都是用 MSC 开发的。当然，最好能同时掌握这两种版本。

3. DOS 的内部结构及其功能调用。几乎所有的应用软件都是在 PC－DOS（或 MS－DOS）的支持下运行的，DOS 的核心部分向用户提供了一整套独立于硬件的系统功能，这些功能有：

- 文件和记录的管理
- 内存管理

·字符设备的输入/输出

·“假脱机”

·提取和设置实时时钟

这些功能都是在 DOS 的核心部分(对于 PC—DOS,该文件为 IBMDOS. COM,对于 MS—DOS,为 MSDOS. SYS)以中断的方式提供给应用程序的。这个中断号为 21H,是 DOS 最宝贵的软件资源之一。这个中断提供了 90 多个功能,应用程序需要使用什么功能,只需把功能号放在 AH 里,再在一些寄存器里设置一些必要的值,然后,发中断 INT 21H 即可。西文软件里,调用数量最多的中断往往便是 21H#,因此,熟悉和掌握 DOS 的 21H# 中断调用也是汉化的基本知识之一。有关 DOS 及其中断调用讲的比较好的有贺志强等翻译的《DOS 磁盘操作系统高级程序员指南》和《IBM—PC/XT/AT 高级程序员编程指南》等书。其中,前一本书附录有 DOS 常用中断的功能描述、调用方法及其实例,内容翔实,很适合作手册用,后一本书对 IBM—PC 的软硬件资料进行了仔细的[illegible],对常用的功能调用及编程技巧进行了详细的阐述,并附有 BASIC 语言和汇编语言[illegible]是一本比较好的资料。

4. 被汉化软件的知识。除了上面要掌握的基础知识外,对将要汉化的软件本身也要了解,对该软件的所有功能都应做到心中有数,不然,汉化工作就可能存在隐患。还有一点也很重要,那就是该软件开发时所用的编程语言。如果能够找到这方面的背景资料,对以后的汉化工作是大有好处的(关于这一点,以后的章节还要阐述)。

§2 汉化的工具综述

我们知道,无论是从国外引进的还是国内开发的软件,用户买来后,手上的程序大多是直接在 DOS 下能运行的 EXE 或 COM 文件。这些 EXE 和 COM 文件是软件开发者用编译器对源码进行编译后生成的二进制机器语言程序。这种机器语言程序较之源码读起来真是宛如天书一般。例如,BASIC 语言里的一条简单的"PRINT"语句,经编译后,生成的机器语言程序指令竟达 100 多条。试想:我们读完这 100 多条指令后方才得知这是一条 PRINT 语句的功能的话,比起一目了然看一条语句 PRINT 的工作量来,相差竟 100 多倍!因此,长期以来,便有不少人从事逆向工程的研究,试图把这种机器语言程序转换成容易理解的高级语言程序。然而令人遗憾的是,由于编译过程的不可逆性,这些机器语言程序经反汇编或反编译后,往往只能生成与其等价的宏汇编语言程序或高级语言程序(一般为 C 语言),而且,这个转换程序往往要借助于人工干预才能顺利实现。这些程序基本上只能做到与源码功能等价,其结构可能完全不同,而且由于源码里的注释信息经过编译后便统统“过滤”掉了,绝无恢复之可能,因而对理解和汉化几乎没有多大的帮助。

目前市面上可以见到的多为反汇编工具,因为从机器语言到汇编语言这一级的转换技术较为简单。这类工具比较常用的有 ASMGEN,PC—KIT,SOURCE 等,而目前,以 SOURCE 软件最为流行。SOURCE 是专门用于把机器语言转换成宏汇编语言的工具,其功能很强。虽然如此,SOURCE 也不是万能的。笔者曾经作过一个试验,让 SOURCE 把一个 39K 字节长的机器语言程序反汇编成宏汇编语言,结果,最后生成的宏汇编语言程序长达 580K 之多。对于这么大的源程序进行分析,倒不如直接看机器语言程序来得方便。可想而知,对于大型的软件,其反汇编出来的源程序文本该有多大了!因此,SOURCE 在这种情况下几乎起不了多大的作用。不过,对于小型的软件,SOURCE 的确还是一个强有力的辅助分析与汉化工具。

实际上，目前使用的最多、最广泛的分析工具还是 Microsoft 公司随 DOS 盘带的 DEBUG、SYMDEBUG、CODEVIEW 等，这些调试软件虽然到处可见，但是功能却很强，只要运用得当，可以有效的对软件进行汉化、加密、解密工作。从 DOS2.00 起就配有调试软件 DEBUG，该程序较小，约 12K，提供了 A、C、D、E、F、G、H、I、L、M、N、O、P(早先的版本未提供此命令)、Q、R、S、T、U、W 共 19 条命令。该软件只能调试机器语言程序，不支持符号调试，且命令功能比较弱，特别是断点功能较差，这对调试汇编语言程序来说，无疑是很不方便的。从 DOS3.XX 开始配备符号调试程序 SYMDEBUG(即 SYMBOL DEBUG)，它扩充了原来 DEBUG 很多命令的功能，另外还增加了许多新的命令如表 2.1 所示。

表 2.1 SYMDEBUG 中新增加的命令

功能分类	命令	含义	功能分类	命令	含义
存储器显示	D	存储器显示(初始缺省方式为字节方式，如果不是初始进入方式则跟前一命令方式相同)	断点	BP	设置断点
				BC	清除断点
				BD	临时禁止
				BE	恢复断点
				BL	断点列表
	DA	ASCII 字符方式	符号	X	列出符号名
	DB	字节方式		XO	置符号图式
	DW	字方式		Z	给符号赋值
	DD	双字方式	高级语言调试	S	指定跟踪的单位以及反汇编时的显示形式
	DS	短实数方式			
	DL	长实数方式		V	显示源程序行
	DT	临时实数方式		.	显示当前源程序行
				K	显示堆栈上的参数
存储器设置	E	存储器设置(初始缺省方式为字节，如果不是初进入方式则跟前一命令方式相同)	重定向	<,(	输入指定
				>,)	输出指定
				=,~	输入输出同时指定
	EA	ASCII 字符方式	其它	P	跟踪执行但对 CALL 的目标不进行单步跟踪
	EB	字节方式			
	EW	字方式		?	HELP/计算器
	ED	双字方式		*	注释
	ES	短实数方式		!	执行 DOS 命令
	EL	长实数方式			
	ET	临时实数方式			

SYMDEBUG 的主要特征表现在以下 6 点：

1. 能够进行符号调试。

2. 能够进行高级语言源代码级的调试。

3. 具有丰富的断点功能。

4. 存储器内容的显示/赋值可以 7 种方式进行，即字节、ASCII 字符、字、双字、短实数、长实数及临时实数。

5. 支持输入/输出的重定向。

6. 在 SYMDEBUG 中可执行 DOS 命令。

继 SYMDEBUG 后，Microsoft 公司于 1987 年又推出了功能更强的新一代调试工具 CodeView 即 CV。这是一种基于窗口(也可以 SYMDEBUG 方式运行)工作方式的高级语言源程序的调试工具，可支持该公司的各种高级语言编译器：MSC(4.0 版以上)，Fortran(4.0 版以上)，PASCAL(4.0 版以上)，BASIC(4.0 版以上)及 MASM(5.0 版以上)。其符号调试功能很强(这是 CV 的最大特色)，加上窗口工作方式、鼠标器、详尽的 HELP 功能以及新增强的多条命令，使得该软件不仅功能完备，而且界面也非常友好，因而一经推出，即受到程序员的欢迎并受到广泛的好评。

从汉化的角度考虑，笔者认为 SYMDEBUG 最为理想，因为在对机器语言的调试功能上，CV 和 SYMDEBUG 的功能相差不大。CV 的主要功能是在窗口方式下直观地调试高级语言源程序，而这一点对于无源码情况的汉化来说，基本上没有用处。虽然 CV 也能在窗口方式下调试机器语言程序，但与 SYMDEBUG 比较起来，显得很不灵活。由于其功能繁多，占用的内存空间也大，加上在中文系统下该软件的窗口功能不能正确使用，而且不支持汉字处理(它也需要"汉化"!)，因而，不太适合作为汉化的工具。而 SYMDEBUG 既有 CV 的优点，又克服了常用的 DEBUG 的不足，因而是汉化、加密、解密等的比较理想的工具。

§3 SYMDEBUG 的使用

本节讲述 SYMDEBUG 在汉化中的使用。SYMDEBUG 有很多功能，灵活的使用这些功能，往往可以起到事半功倍的效果。

3.1 SYMDEBUG 的启动

可以两种方式启动 SYMDEBUG，一种带参数，另一种不带参数。所谓带参数是指其后跟上要调试的文件名(一般为 COM 或 EXE 文件)。例如，我们用 SYMDEBUG 调试它本身，则启动后的屏幕显示信息如下：

```
C>symdebug symdebug.exe (CR)
Microsoft (R) Symbolic Debug Utility Version 4.00
Copyright (C) Microsoft Corp 1984, 1985. All rights reserved.

Processor is [80386]
-
```

在版本号和版权信息后，SYMDEBUG 还显示出了当前机器的处理器型号为 80386，之后便是 SYMDEBUG 的提示符"—"，此时 SYMDEBUG 启动完成，系统等待用户键入命令。

3.2 SYMDEBUG 在汉化中常用的命令

SYMDEBUG 的命令很多，我们只讲述在汉化中常常使用的一些命令。

3.2.1 SYMDEBUG 中的数制

SYMDEBUG 可用二进制、八进制、十进制和十六进制表示整数。每一种数制的基如下：

进制	基	合法数字
二进制	Y	0,1
八进制	O 或 Q	0,1,2,3,4,5,6,7
十进制	T	0,1,2,3,4,5,6,7,8,9
十六进制	H	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

输入数字时，后面都要带上基。如十进制的 22 在 SYMDEBUG 中的各种进制表示方法为：10110Y(二进制)，26Q(八进制)，22T(十进制)，16H(十六进制)。SYMDEBUG 的默认值为十六进制。

3.2.2 地址及范围 (Address and Range)

SYMDEBUG 的地址均由两个字组成，即 SEGMENT:OFFSET。这与 80X86 系列特有的寻址方式是一致的。因此，在命令中指定地址对象时，应该以上述格式来加以指定，否则 SYMDEBUG 会使用默认值。在 SEGMENT 缺省的情况下，SYMDEBUG 的 A、G、L、P、、T、U、W 命令使用 CS 作为 SEGMENT，其它的命令一般用 DS 或 SS 作 SEGMENT。

SYMDEBUG 表示地址范围有两种方法即绝对法和相对法。绝对法的表示方法为：

startaddress,endaddress (起始地址，结束地址)

相对法的表示方法为：

startaddress L count (起始地址 L 长度)

3.2.3 常用命令

1. 输入汇编程序

[语法] A [address]

[功能] 从 CS:IP(缺省)或 address 开始输入 8086 的小汇编程序。键入 A 命令后，即可逐行输入小汇编程序，SYMDEBUG 自动把它转换成机器语言代码存放于 CS:IP 开始的区域，退出 A 命令态按 Ctrl+Break 键或直接按 Enter 键即可。小汇编程序除了不能使用符号地址外，基本的语法与宏汇编相同。此命令经常用作修改西文软件。

2. 显示所有寄存器的值或修改某个寄存器的内容

[语法] R [register]

[功能] 分析软件的过程中，经常要察看 CPU 各寄存器的值及标志位。使用 R 命令(不带参数)即可完成这个功能。R 命令显示完各寄存器的值后，还显示下一条指令。

[例]用 symdebug symdebug.exe 命令启动后，键入 R 命令，得到下面的结果：

```
—R (CR)
AX=0000  BX=0000  CX=8E9D  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=669E  ES=669E  SS=6F98  CS=66AE  IP=0109  NV UP EI PL NZ NA PO NC
66AE:0109  A2A66D          MOV  [6DA6],AL          DS:6DA6=00
```

R 命令显示了 AX、BX、CX、DX、SP、BP、SI、DI、DS、ES、SS、CS、IP 的值，之后便是标志寄存器当前的状态。该状态是用符号的形式来显示其内容的，符号的含义如下：

标志位	设置	清除
上溢 (Overflow)	OV	NV
方向 (Direction)	DN (减少)	UP (增加)
中断 (Interrupt)	EI (允许)	DI (屏蔽)
符号 (Sign)	NG (负)	PL (正)
零 (Zero)	ZR	NZ
辅助进位(Auxiliary Carry)	AG	NA
奇偶 (Parity)	PE (偶)	PO (奇)
进位 (Carry)	CY	NC

除此之外，我们在执行程序的过程中，往往需要动态修改某个寄存器的值，可使用命令：

R 寄存器名　(CR)

寄存器名必须是 13 个寄存器 AX、BX、CX、DX、SP、BP、SI、DI、DS、ES、SS、CS、IP 之一。

[例]把 AX 寄存器的内容由 0000H 改为 FFFFH，则键入下列命令：

```
—RAX (CR)
AX  0000
:FFFF  (CR)
```

3. 反汇编命令

[语法] U [range]

[功能]如果指定了 range，则在相应的地址范围内反汇编。否则，SYMDEBUG 默认为在 CS 段内进行反汇编。如果是第一次使用 U 命令，则为从 CS:IP 处开始反汇编出 8 条指令，否则从上一次 U 命令反汇编显示的最后字节的下一个字节开始。

[例]用 symdebug　symdebug.exe 命令启动后，发 U 命令，则屏幕上显示下列结果：

```
—U (CR)
66AE:0109   2A66D      MOV   [6DA6],AL
66AE:010C   FA         CLI
66AE:010D   8CC8       MOV   AX,CS
66AE:010F   8ED0       MOV   SS,AX
66AE:0111   BCA08F     MOV   SP,8FA0
66AE:0114   FB         STI
66AE:0115   8EC0       MOV   ES,AX
66AE:0117   33F6       XOR   SI,SI
```

需要注意的是，由于可以随意指定 U 命令反汇编的范围，因此，有时 U 命令反汇编的区域不是代码区，因而反汇编出来的程序无意义，但 SYMDEBUG 并无任何提示信息。在这种情况下，需要我们具有较丰富的经验和判断能力。

[例]从 SYMDEBUG.EXE 的 013C 处开始反汇编，列表如下：

```
-U13C  (CR)
66AE:013C   4D            DEC    BP
66AE:013D   6963726F73    IMUL   SP,[BP+DI+72],736F
66AE:0142   6F            OUTSW
66AE:0143   66            DB     66
66AE:0144   7420          JZ     0166
66AE:0146   285229        SUB    [BP+SI+29],DL
66AE:0149   205379        AND    [BP+DI+79],DL
66AE:014C   6D            INSW
```

读者一看就应该知道，上述指令毫无意义。事实上，这是 SYMDEBUG 的数据区域。

4. 跟踪命令

[语法] T[=startaddress][count]

P[=startaddress][count]

[功能]T 和 P 都是用于单步跟踪的命令，两个只在几个地方有所差别。如未指定=startaddress，则执行当前 CS:IP 所指的一条指令，然后 CS:IP 指向下一条指令，显示所有寄存器的值(等同于 R 命令)，重又回到命令态，这就是"单步"的含义。如果指定了=startaddress，相当于把当前的 CS:IP 无条件的指向 startaddress，然后执行。count 可指定单步跟踪的次数。

[例]先用 R 命令显示当前的 CS:IP 所指向的一条指令，再用 T 命令执行该指令，结果 CS:IP 指向下一条指令，且显示所有寄存器的值。

```
-R  (CR)
AX=0000  BX=0000  CX=8E9D  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=669E  ES=669E  SS=6F98  CS=66AE  IP=0109  NV UP EI PL NZ NA PO NC
66AE:0109  A2A66D          MOV  [6DA6],AL          DS:6DA6=00
-T  (CR)
AX=0000  BX=0000  CX=8E9D  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=669E  ES=669E  SS=6F98  CS=66AE  IP=010C  NV UP EI PL NZ NA PO NC
66AE:010C  FA          CLI
```

P 命令与 T 命令的不同之处在于以下三点：

(1)对中断调用的处理。除了 DOS 的 21H # 中断 T 命令跟踪时不会陷入外，其余中断 T 都会跟踪进去，而 P 则把指令一次性地执行完毕。

(2)对 LOOP 指令的处理。如用 T 命令，则一直跟踪整个循环体，直至 CX 减为 0 为止。而用 P 命令可一次性地把循环执行完毕。

(3)对调用子程序的处理。对于 CALL XXXX 指令，如用 T 命令跟踪，如同中断一样会陷入子程序里去，而用 P 命令则一次性地把该程序执行完毕。

在实践中，灵活使用 P 命令和 T 命令可以大大提高效率。

5. 断点操作命令

SYMDEBUG 的特色之一就是其断点处理功能较强，它可以在程序里设置十个"胶状"断点，而且可以随时关闭、激活和清除，下面依次分述。