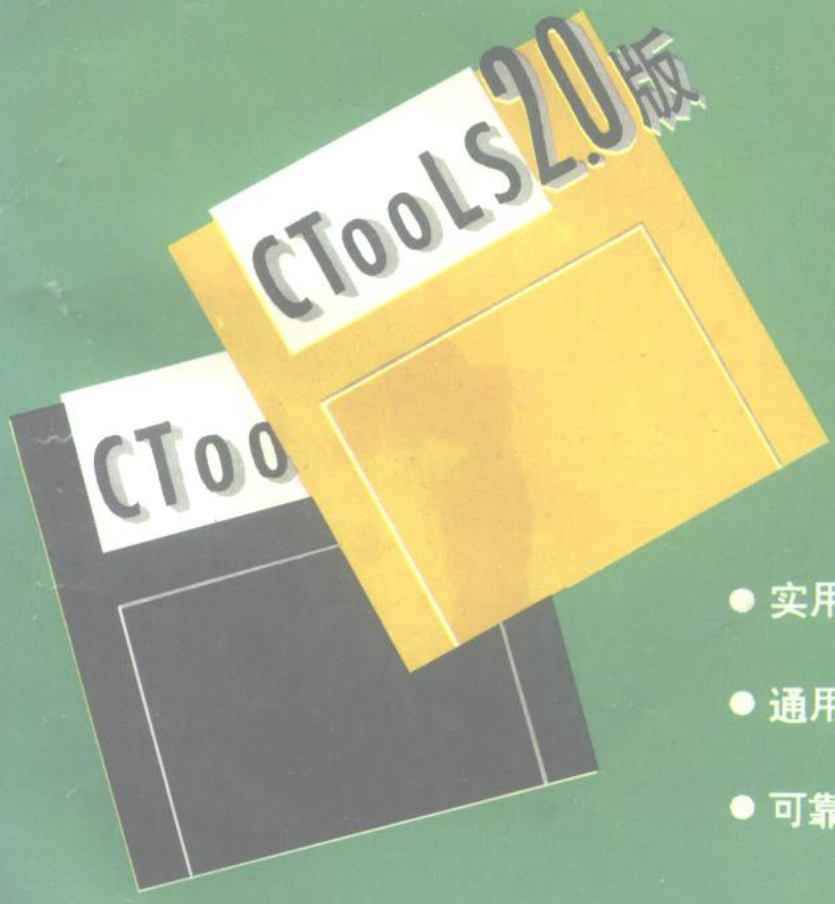
一本好的工具书可以使您的程序设计事半功倍

C语言实用程序与软件工具

郑庆华等 编著



- 实用性强：提供大量的实用函数和实例程序
- 通用性好：支持各种应用程序的开发
- 可靠性高：所有函数和程序均经上机调试通过，并附有软盘

西安交通大学出版社

C语言实用程序与软件工具

编著者(按姓氏)
王余蓝　王　斌　宋　洁
郑庆华　林　勇　胡　峻

西安交通大学出版社

内 容 提 要

这是一本面向C语言程序设计者开发DOS应用程序的实用工具书。书中介绍了C语言程序设计者编程提高的必备技能，以及程序设计中常见问题的解决方法和C函数具体实现。内容包括：C语言和汇编语言混合编程方法，输入/输出处理，窗口处理，图形、图象、汉字的显示与变换处理，菜单制作与生成，直接控制I/O设备（如显示器、打印机、鼠标、键盘、串行口、定时器等）编程，常见数据库文件（如.DBF文件）和图象文件（如.BMP、.PCX等）的格式和访问，以及扩展存储器的使用编程等7个部分。

本书是作者在总结和提炼多年软件开发经验之基础上，以软件生产工具化为指导思想编写的。书中介绍的所有实用程序和软件工具都是久经考验的，已成为我们软件开发的基础工具。

本书内容充实，条理清晰，理论和实践相结合，尤其注重能力、方法及实际应用经验的介绍。

本书适宜于具有一定C语言基础的程序设计者，特别是那些渴望自己的编程水平有很快提高的程序设计者。也可作为C语言高级程序设计的培训教材。

(陕)新登字007号

C语言实用程序与软件工具

郑庆华等编著

责任编辑 白居宪

*

西安交通大学出版社出版发行

（西安市咸宁西路28号　邮政编码：710049）

西安电子科技大学印刷厂印装

陕西省新华书店经销

开本：787×1092　1/16　印张：27.625　字数674千字

1996年2月第1版　1996年2月第1次印刷

印数：1——5000

ISBN7-5605-0784-0/TP·110　定价：25.00元

前 言

在总结、整理多年软件开发经验之基础上，以软件生产工具化为基本指导思想，我们编著了《C 语言实用程序与软件工具》一书。这是一本面向 C 语言程序设计者开发 DOS 应用程序的实用工具书。

多年程序设计的经验告诉我们：一个优秀的程序设计者除了具备一定的理论知识之外，更重要的还取决于他(她)掌握的编程经验、工具和技巧。虽然程序设计内容各异，要求不一，但其中碰到的很多问题和技术则是重复和类同的。例如窗口处理技术、菜单处理技术、扩展存储器使用技术，……等等。能否把这些共性的问题和技术提取出来，编写成可供各类程序调用的通用函数和过程，即通用工具，不仅直接影响了程序设计的效率，而且还影响了程序设计的规范性和准确性。一套良好的开发工具是取得事半功倍效率的最好途径。因此，一个优秀的程序设计者往往既善于编程，同时又善于提取和抽象各种通用工具。

本书内容简介：

这是一本理论和实践相结合的、以实践为主专题介绍 C 语言程序设计中遇到的各种问题及实现方法的技术工具书。内容涉及 I/O 程序设计，图形、图象、汉字处理，菜单制作和生成，直接控制 I/O 设备编程，常用数据库文件和图象文件的访问，以及扩展存储器的使用编程等方面。全书共分 7 章。第 0 章介绍了 C 语言程序设计者编程提高所必备的基本技能——C 语言内存管理模式及其和指针的关系，以及 C 语言和汇编语言混合编程的方法——OBJ 连接方法和嵌入汇编代码方法。第 1 章阐述了 I/O 程序设计中常用的若干工具和实用程序。主要介绍了文本和图形方式下输入窗口的设计，信息的输入编辑，图形方式消息窗口的设计和具体实现。第 2 章介绍了 VGA 图形方式下常用的图形、图象、汉字处理的主要问题及程序设计方法。从基本图形指令到各种复杂图形变换；从基本图象显示到图象的阴影、轮廓、旋转算法；从点阵和矢量汉字的基本结构到常用的汉字变换算法，都给出了详细的实现算法和对应的 C 实现函数。此外，还给出了通用的统计图形——直方图、饼图、折线图制作算法和实现函数，以及两种图象输出转换算法——灰度图象转为黑白二值图象和彩色图象转为黑白二值图象算法。第 3 章介绍了各种菜单设计方法，并给出了简易式、下拉式、弹出式及对话框、消息窗等常用菜单的具体实现函数。第 4 章阐述了直接控制 I/O 设备编程的主要问题，着重对显示器、打印机、鼠标器、串行口、键盘等 I/O 设备的编程作了详细说明，并给出了相应的程序设计实例。此外，本章还给出了几个非常实用的技术，如后台音乐演奏、淡入浅出技术、屏幕分割技术等等。第 5 章介绍了几种常见的文件格式及其访问方法，主要包括：.DBF，.BMP，PCX，.CUR，.ICO 等，并给出了相应的 r/w 访问算法和具体实现函数。第 6 章介绍了在 DOS 下突破 640Kbytes 常规 RAM 编程的 5 种使用扩展存储器的基本方法，即 ROM BIOS 的 int 15H，XMS 规范，EMS 规范，DOS Extender 及保护方式编程等，并就其中常用的 int 15H，XMS，EMS 三种方法作了详

细阐述，并给出了编程实例。最后的附录以简表形式给出了随书出售软盘的详细目录及其使用方法。

书中涉及的所有软件工具和实用程序都经作者们上机验证、调试通过。

本书的第0章、第6章、第1章的第1节、第2章的第1,11,12节以及第4章的第1,7节由郑庆华、王余蓝执笔；第1章、第2章及第4章的余下部分由林勇、宋洁执笔；第3章和第5章由王斌、胡峻执笔。全书由郑庆华主编并统一定稿。

本书读者对象：

本书适合于具有一定C语言基础的程序设计者，尤其是那些渴望编制出色程序，但又缺乏经验和系统训练的程序设计者。也可作为C语言高级程序设计培训教材。

衷心希望本书能成为您软件开发的得力助手。

本书配套软盘说明：

随书附软盘一片，包含书上所介绍的全部实用程序和软件工具的源程序和目标程序，此外还包含了本书因篇幅限止而无法包含的例子程序。务请您注意一起购买！

作者简介：

郑庆华：1990年毕业于西安交通大学计算机系计算机软件专业，并于同年被推荐攻读本校计算机组织与系统结构硕士学位，1993年获该专业工学硕士学位，后留校任教于计算机系，并于次年(94年)被推荐攻读西安交通大学系统工程研究所“智能控制”专业的博士。

自90年以来，作者一直从事于计算机软件的开发，主要成果有：国家级火炬计划项目“CAiT图文系统”、“CAiT鼠标造字系统”的研制和实现；国家“七五”重点科技攻关项目“0530多微处理机操作系统”的设计和实现；此外，还成功地开发了“通用报表生成及其图形分析系统”和“通用图文文档信息系统”两个商品化软件；目前主要从事CSCW(计算机支持的协同工作)方向的研究和开发。

近年来，作者已发表论文10篇，出版《Visual Basic for Windows 3.0程序设计指南》书籍一本。主要获奖项目：国家教委科技进步二等奖一项，校优秀科技成果二等奖一项，全国大学生课外学术、科技作品“挑战杯”竞赛二等奖优胜奖各一项。

王余蓝：外语系教师，毕业于西安交通大学计算机系。林勇：在校学生。宋洁、王斌、胡峻西安交通大学计算机系在校研究生。

最后，我们还要提请读者注意：虽然书中的程序和算法是经我们上机验证通过的，但未必是最好的、最高效的。此外，我们总结的编程工具也还不够齐全。因此，我们真诚希望读者能返馈给我们更好的实现算法和更多、更全的软件工具。

衷心感谢为本书出版做出各种奉献的学友，亲朋以及担任本书责任编辑白居宪老师！

作者

1995年9月

目 录

第0章 C语言编程提高基础

不论您是一位初级刚入门的C语言程序设计者，还是一个已编过若干C程序具有一定实际经验的程序设计者，我们都希望您认真阅读本章。这不仅对于您理解后面各章会有帮助，而且对于您正确地理解C的工作模式、指针使用以及扩展C语言编程、实现C和汇编的混合编程等都大有益处。

本章阐述了C指针的概念及属性。也许您已能熟练地使用指针数据类型，但您未必真正理解三种指针——near，far 和 huge 的含义、相互区别，以及指针和内存模式之间的相互关系。或许您已尝试过C和ASM混合编程，本章将告诉您更多的有关C和ASM混合编程的方法，除了C程序可以直接调用ASM代码外，ASM代码也可调用C的代码和函数，而且两者还可相互引用变量。

本章首先介绍MS-DOS下内存管理及寻址机构的基本原理，在此基础上说明C语言三种指针——near，far 及 huge 的基本概念与使用方法，以及 Borland C++ & Turbo C++六种存储模式的基本含义、存储模式和指针间的相互关系；接着介绍C语言和汇编语言混合编程的两种方法——.OBJ 文件连接方式和嵌入汇编方式，并通过具体的实例说明混合编程中要注意的接口问题——参数传递、变量引用、参数返回和伪变量引用等。

总之，通过本章的学习定会使您对C语言编程有更深的认识和提高。

0.1 Turbo C 和 Borland C++系列的存储模式

Borland C++ 和 Turbo C++提供了6种内存模式：微模式(Tiny)、小模式(Small)、中模式(Medium)、紧缩模式(Compact)、大模式(Large)和巨模式(Huge)。每种模式对应了不同的程序代码结构和代码大小，并以不同的方式使用内存、管理用户程序的代码和数据，并影响程序执行的速度。存储模式对用户程序的执行及系统资源的访问能力影响很大。因此，用户程序的编译必须选择一个适当的存储模式。为了更加深入、明确地理解存储模式的概念，我们首先介绍MS-DOS内存管理的基本方法——段式内存管理。

0.1.1 8086的段式内存管理和地址计算

Intel 8086 微处理器有20根地址线，因此其最大寻址能力可达 $2^{20}=1$Mbyte，但是，8086中的寄存器均为16位，最大值仅为 $2^{16}=64$Kbytes，如何用16位的寄存器表示1Mbtes的地址空间呢？8086以至后来的80286，80386均采用了段：段内偏移量的结构来解决，称作“段式内存结构”。

段是指内存中一块连续的区域，段最小为16bytes，最大为64Kbytes。虽然8086的寻址能力可达到1Mbytes，但在某个特定时刻，可直接访问的内存地址只能为64Kbytes。

为了访问段，8086中设置了四个16位段寄存器：CS，DS，ES，SS。每个段寄存器对应于一

个段，分别为代码段(Code Segment)、数据段(Data Segment)、附加数据段和堆栈段(Stack Segment)。每个段寄存器指示了该段的起始地址。四个段寄存器的值可以相同，也可以不同，如果相同，则表示代码、数据、堆栈共用一段，其总值不超过 64Kbytes，如.COM 文件。

为了表示 20 位的地址，段式结构采用了下面的方法：将段值(由段寄存器指示)左移 4 位后加上段内偏移值，由此产生一个 20 位的数据值，表示实际地址。例如：

假设数据段地址为 5F84H，即 DS＝5F84H，基址寄存器 BX＝3047H，那么 DS:[BX]指示的数据其实际地址应为：

DS 值左移 4 位　　0101,1111,1000,0100,0000=5F840H

偏移值 ⊕　　0011,0000,0100,0111=3047H

实际地址　　0110,0010,1000,1000,0111=62887H

一个段的首地址总是一个 20 位的值，即 5 位 16 进制数，其最末一位为 0。这是因为当 16 位的段寄存器值左移 4 位后，其最低 4 位总设成全零，因此，段值总是起始于内存的 16bytes 倍数地址上。如上例所示，如果 DS 保存的值为 5F84H，那么数据段的实际起始地址为 5F840H。

段式结构对一个地址采用段:段内偏移量的标准表示法，这种方法表示的地址称为逻辑地址。如前例中的地址可被写为 5F84:3047。值得注意的是，段值与偏移量之间是可以相互覆盖的，即一个给定的内存地址(又称物理地址)，其表示形式不是唯一确定的。例如：物理地址或内存地址 00345，采用段式结构的逻辑地址表示法，可以表示为：

0030:0045

0004:0305

0034:0005

0020:0145

0022:0125

⋮

由此可见，一个内存单元对应一个唯一的物理地址，但却可以表示为多个逻辑地址。正由于段式存储管理的这一特点，使得下一节中介绍的 far 类型指针和 huge 类型指针有了显著的区别。

所谓指针，即是指向内存中某个存储单元的地址。这一地址可以是段内偏移量，可以是段值，也可以是一个完整的逻辑地址:(段值，段内偏移量)，不同的表示对应了指针的不同性质，也决定了其访问数据和代码的能力。

0.1.2 指针

指针是 C 语言程序设计中最常用的概念之一。指针具有不同类型，按照指针所指的对象，可分为数据指针和函数指针(代码指针)。在 C 和 C++中，指针与内存模式是直接相关的，内存模式的类型决定了代码和数据指针的缺省类型，当然也可以显式地说明一个指针具有特定的类型，而不管正在使用的内存模式。在 TC 和 BC++系列中，指针有下列四种类型：近指针(near)16 位；远指针(far)32 位；巨型指针(huge)32 位；段指针(32 位)。

1. near 指针

near 指针仅表示段内偏移量，是 16 位的。near 指针在进行地址计算时还需要一个段寄存器。在 C 编译时约定：一个函数的 near 指针(代码指针)应和代码段 CS 寄存器结合，一起形成完整的逻辑地址，而一个数据 near 指针仅是相对于数据段 DS 偏移，它和 DS 寄存器一起形成完整的逻辑地址。由于 near 指针未考虑段地址，因而操作简单安全。

2. 远(far)指针

32 位的远指针不仅包含了段内偏移量，也包含了段地址(均为 16 位)，即为一个逻辑地址。段地址左移 4 位后与偏移量相加，即得到物理地址，通过使用远指针可以支持多个代码段；允许程序代码大于 64Kbytes，同时也可访问超过 64Kbytes 的数据。

在使用 far 指针时，要注意避免发生一些指针运算错误。我们假设存在下面三个 far 类型的指针：ptr1 为 0000:0340；ptr2 为 0030:0040；ptr3 为 0034:0000，那么，由 far 指针计算得到的物理地址均为 00340H(20 位)。此时，下面的表达式都存在一些容易忽视的问题：

```
if (ptr1==ptr2)…
if (ptr1==ptr3)…
if (ptr2==ptr3)…
```

这些表达式的值将为假。这是因为等于(==)和不等于(!=)操作符把 32 位指针值当作 unsigned long 类型值处理，而不作为实际的内存地址处理，即把段地址作为高 16 位，偏移地址作为低 16 位，而没有将指针转化为实际地址。也因为这一特性，far 类型指针可以与 NULL 指针(空指针 0000:0000)相比较。

但是，对于 ptr1，ptr2，ptr3 三个指针，下面三个表达式的值将为真：

```
if (ptr1>ptr2)…
if (ptr2>ptr3)…
if (ptr3>ptr1)…
```

这是因为使用>，>=，<，<=运算符对远指针作比较时，仅对偏移量(作为 unsigned)进行比较。

如果将一个值加到远指针上，那么只有偏移量被改变。当加的值使偏移量大于 0FFFFH 时，偏移量将改变为 0(溢出)，而段值不变，减法也是一样。

我们建议，不要使用 far 指针作比较。若需要作比较时，最好使用 near 或 huge 类型的指针。

3. huge 指针

far 指针在作比较或移动(加、减)时，存在着隐患，而 huge 指针则避免了这些问题。

huge 指针同样是 32 位的，同 far 指针一样，huge 指针含有一个段地址和一个偏移量。但不同于 far 指针的是，huge 指针的值是经过规范化的，从而避免了远指针中存在的问题。

规范化的指针有 32 位，采用最大的段地址加上偏移量来表示一个实际物理地址。由于一个段可以开始于任何能被 16 整除的边界上，huge 指针通过修改段地址和偏移量来形成规范化指针。

规范化的规则是：对于 20 位的物理地址，取其左边 16 位为段地址，取其右边 4 位为偏移地址。例如

0040:0056　　规范化:0045:0006

2F84:0532　　规范化:2FD7:0002

7418:D03F　　规范化:811B:000F

把 huge 指针规范化,其重要意义在于:

(1) 给定的内存地址对应唯一的 huge 指针,只可能有一种段:偏移的表示形式。这使得==和!=操作符能够返回正确值;

(2) 其余的比较运算符使用 32 位值,规范化后使得比较结果也是正确的;

(3) 规范化后的巨型指针偏移量每增加到 16 就自动回绕,与此同时,段值也作了调整,假如:将 684E:FFFF 加 1 后,偏移量变为 0000,而段值也改变为 684F,对于减法也是这样。正由于巨型指针的这一特点,使得 huge 指针可以处理大于 64Kbytes 的数据结构。

虽然规范化的 huge 指针增加了运算的正确性,但付出了时间的代价,因为巨型指针运算是通过调用特定的过程来完成的,因此,较之 far 和 near 指针,huge 指针的计算显得比较慢。

4. 段指针

TC 系列中有四个特殊的 near 型指针_cs,_ds,_ss,_es,它们分别与代码段,数据段,堆栈段和附加段相连,称为"段指针"。我们将在下一节中详细介绍段指针。

0.1.3 地址修饰符

TC 和 BC 系列中引入了 8 个 ANSI C 标准没有定义的关键字:near, far, huge,_cs,_ds,_ss,_es 和_seg。它们都作为指针的修饰符(某些情况下也可以用于函数)。

可以使用关键字 near,far 或 huge 修饰函数和指针。near 函数以 near 方式调用激活,并以 near 方式返回(近返回);far 函数以 far 方式调用激活,并以 far 方式返回(远返回);huge 函数与 far 函数一样,只是 huge 函数将 DS 值置为一个新值,而 far 不这样。

有 4 个特殊的 near 数据指针:_cs,_ds,_ss,_es。它们都是 16 位的,并分别与对应的段寄存器相连。例如下面的一个指针:

```
char _es *str;
```

则 str 将包含相对于附加数据段内的 16 位偏移量,即 str 被限定为指向附加数据段内的一个地址。

若函数或指针是 near 型的,则它们将自动地与 CS 或 DS 相连。表 0-1 给出了不同存储模式下函数或指针的缺省形式,指针的长度根据指针访问空间的大小而定:小于 64Kbytes 对应于 near 指针;大于 64Kbytes 而小于 1Mbytes 对应于 far 指针。

表 0-1　存储模式与指针

存储模式	函数指针	数据指针
Tiny	near,_cs	near,_ds
Small	near,_cs	near,_ds
Medium	far	near,_ds
Compact	near,_cs	far
Large	far	far
Huge	far	far

_seg 称为段指针类型说明符，它表示 16 位段指针。例如，指针 segment _ptr 以下列方式定义：

```
int _seg * segment _ptr;
```

那么，segment _ptr 表示一个 16 位的整型段指针。段指针仅包含段值，而不包含偏移量。当对 segment _ptr 进行间接寻址时，对应了一个偏移量为 0 的逻辑地址。

关于段指针，还需说明以下几点

(1) 段指针不能使用＋、－操作符，即不允许增加也不能减小段指针。当把一个整数加到段指针或从段指针上减去一个整数时，段指针被隐式地转化为一个远指针，且算术运算同 far 指针一样；

(2) 段指针与 near 指针相加，结果为 far 指针，它由段指针中的段值和近指针中的偏移量组成；同时，两个指针必须指向相同的数据类型或其中一个为 void 类型。此外，不允许偏移量作乘法操作；

(3) 段指针用于间接寻址时，将被隐式地转换为 far 指针；

(4) 段指针可以用于分配空间、初始化、I/O 函数或比较运算中。段指针作比较运算时，指针的值视为 unsigned int；

0.1.4 六种存储模式

TC 系列提供了六种存储模式：Tiny ，Small，Medium ，Compact，Large 和 Huge。

1. Tiny 微模式

在 Tiny 模式下，四个段寄存器(CS，DS，ES，SS)被置成相同的地址，代码、数据和堆栈共享 64Kbytes 的空间，并总是使用 near 指针。当用 TLINK 的/t 参数选择项时，Tiny 模式的应用程序可以生成.COM 格式的可执行文件。

2. Small 小模式

代码段和数据段不同，而且不能重叠，因而程序可有 64Kbytes 的代码和 64Kbytes 的数据段和堆栈段，而且 DS＝ES＝SS。所有指针使用 near 型。

3. Medium 中模式

远指针被使用在代码上，但不用于数据，因而代码段可以达到 1Mbytes，而数据和堆栈段则受限于 64Kbytes。这适合于代码大而数据量小的程序。

4. Compact 紧缩模式

远指针被用于数据段，而代码段使用 near 指针，这刚好和 Medium 相反。这适合于处理数据量大而代码小的程序。

5. Large 大模式

代码和数据都使用 far 指针，因此都可以使用 1Mbytes。这适合于大型软件的开发。

6. Huge 巨模式

代码和数据也使用 far 指针，但取消了静态数据不超过 64Kbytes 的限制，允许静态数据超过 64Kbytes 的空间。

图 0-1 至图 0-6 说明了 PC 机内存在上述六种存储模式下的分配方式。表 0-2 给出了六种存储模式的比较。

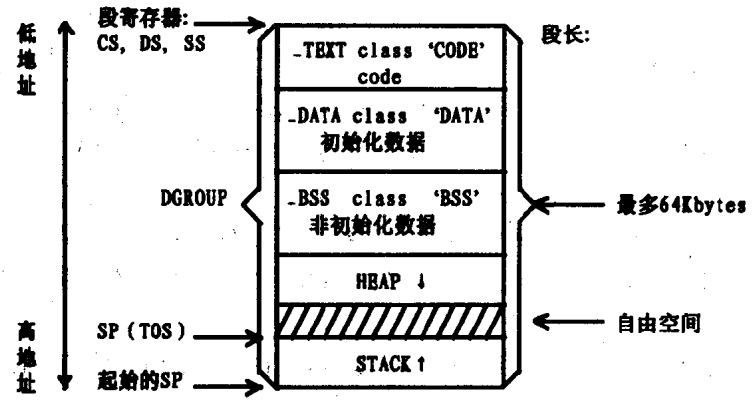


图 0-1　Tiny 微型模式内存分段

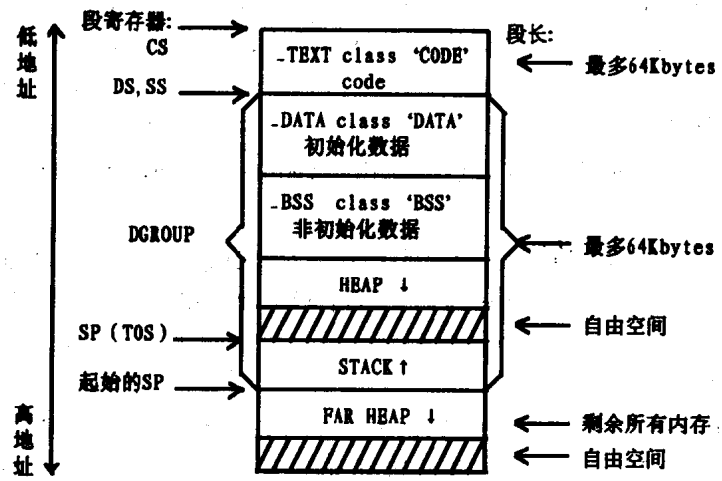


图 0-2　Small 小型模式内存分段

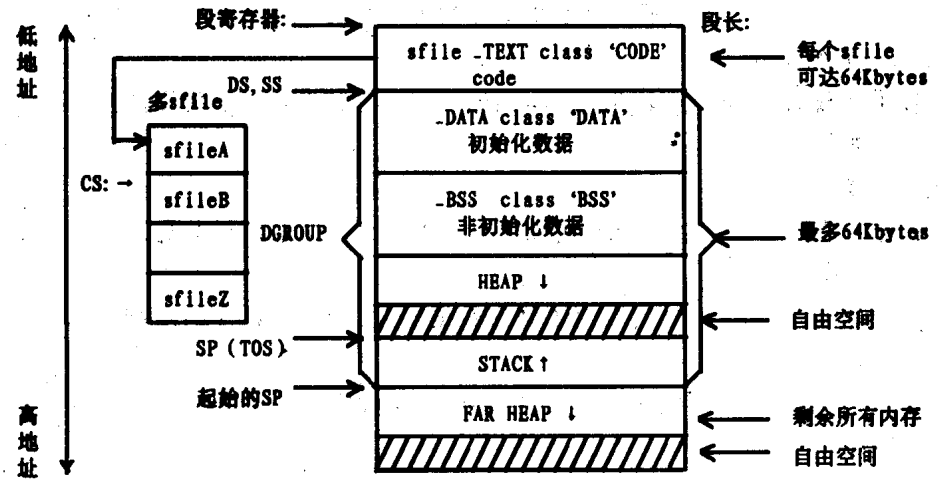


图 0-3　Medium 中型模式内存分段

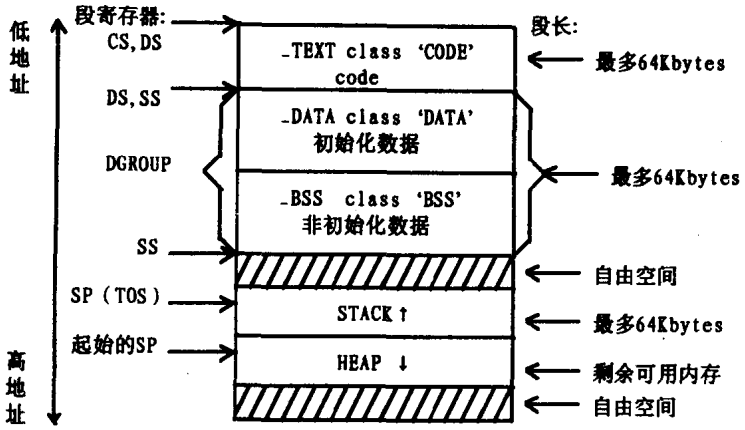


图 0-4　Compact 紧缩型模式内存分段

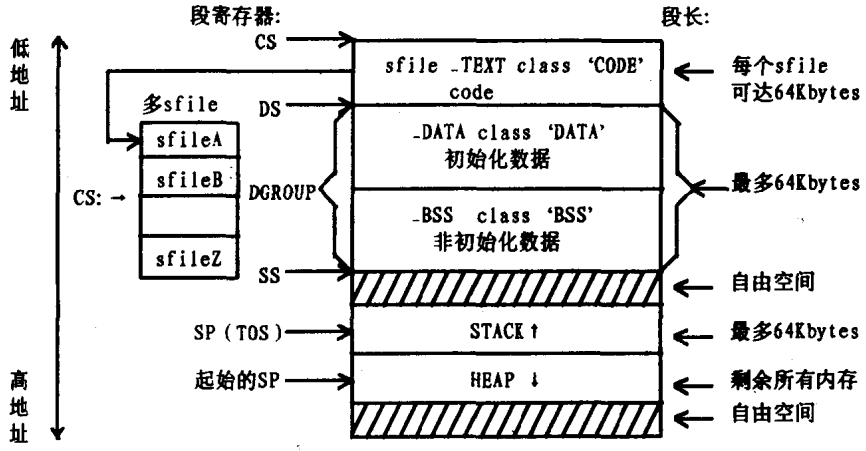


图 0-5　Large 大型模式内存分段

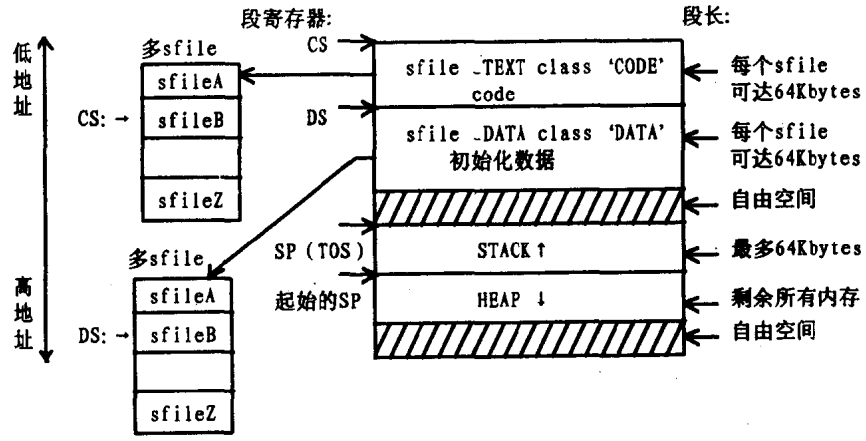


图 0-6　Huge 巨型模式内存分段