Brady

THE PETER NORTON 编程宝典

程序员疑难问题详解

PETER NORTON 编程宝典

PROGRAMMER'S PROBLEM SOLVER

〔美〕 Robert Jourdain 著

郝金川 刘涌 梅宝富 译

本书是美国最畅销计算机图书之一，是程序员案头必备的DOS编程宝典。书中包括用MS C、Turbo Pascal和Turbo C 编写的数百个精湛的编程示例和全部源代码。

電子工業出版社

程序员疑难问题详解

Peter Norton 编程宝库

PROGRAMMER'S PROBLEM SOLVER

[美] Robert Jourdain 著
郝金川 刘涌 梅宝富 译
陈一飞 景利 审校

電子工業出版社

（京）新登字055号

内 容 提 要

本书针对软件人员在程序设计中所遇到的种种疑难问题，全面深入地介绍了微机程序设计涉及的基本知识和各种技术，包括如何使用微机的各类资源，如何对各种硬件设备编程控制等。全书共十七章，分别就各类设备的编程技术做了详尽的说明，同时提供了用BASIC、Pascal、C和汇编等四种常用语言编写的编程示例。

本书结构合理，例子丰富，实用性很强。从系统设计的函数使用、系统调用和低层访问三个不同层次，为软件人员提供了完成程序设计任务的大量信息。

PROGRAMMER'S PROBLEM SOLVER

Robert Jourdain

* * *

程序员疑难问题详解

Peter Norton编程宝库

[美]罗勃特·约丹 著

郝金川 刘涌 梅宝富 译

陈一飞 景利 审校

特约编辑 张成全

*

电子工业出版社出版

电子工业出版社发行 各地新华书店经销

顺义县天竺颖华印刷厂 印刷

*

开本：787×1092 毫米1/16 印张：22.5 字数：548千字

1994年10月第 1 版 1994年10月第 1 次印刷

印数：0001－6000册 定价：42.50元

ISBN 7－5053－2788－7／TP·893

前　言

如今大多数人已经发现，虽然信息时代给我们提供了大量的信息，但组织这些信息也是一件十分繁琐的工作。计算机程序设计人员对此有更深切的体会，因为他们每天接触到的是成千上万有关硬件标准、操作系统版本以及编译器库等形形色色的内容。程序员的书架上塞满了各类资料，它们通常内容有序，但往往不能满足我们的需要。

本书致力于把IBM PC程序设计基本的信息组织起来，使你能迅速找到你所需要的东西，并能读到用你熟悉的语言所写成的示例。有时，在浩如烟海的资料中查找必要的东西，是相当困难的。但若本书在手，则只要查看一下目录，问题即可迎刃而解。

本书围绕硬件进行安排，包括有关打印、计时及磁盘操作等章节。在许多示例中，你会看到在下面三个层次如何编程的说明：高层如何使用编译器库函数，中层如何调用操作系统，而底层则如何直接访问串行端口和视频控制卡之类的外设芯片。若缺少高级编译子程序，你会看到用高级语言在中层或低层编程的例子。通过研究不同的方法，你会学习到高级语言是如何调用操作系统的，而操作系统又是如何来管理硬件的。

几乎对每个程序设计任务，你都会看到用四种最常用的程序设计语言写成的例子：即BASIC，Pascal，C和8086汇编语言。当然，编译器库因厂商而异，所以对于哪种程序示例应用哪种编译器需要进行选择。对BASIC和Pascal而言，选择并不困难，选用Microsoft的BASIC和Borland的Turbo Pascal是无可争议的。选择C编译器却不是那么简单。在写本书之时，Borland与Microsoft C编译器之间的王位之争仍在继续。因此，若有差异，则给出两种例子。汇编语言示例是通用的，可与任何汇编程序一起使用，或作为直插式汇编代码用在编译器中。

你可能怀疑这些信息是否还象过去一样有用。在DOS主宰PC操作系统10余年之后，已现老矣，现在轮到Microsoft Windows了，还值得为DOS做这么大的努力么？回答是完全肯定的。面向DOS的程序设计不会很快消失。成千上百万台缺少运行Windows所需速度和内存的PC，还将一如既往在多年之内用DOS运行下去。Microsoft知道这个市场非常之大，不容忽视，并计划继续改进DOS。同时，Windows应用程序已经把用户对软件易学易用的期望值提高了。现在，许多程序员必须同时开发软件的DOS和Windows版本。这使得DOS编程更加困难，并要求对PC硬件有更深入的了解才能使DOS应用程序达到Microsoft的标准。本书在这方面会对你有所帮助。

另一个趋势是面向对象的程序设计，特别是C语言。结构完好的类库最终可能取代目前编译器所用的过程库。但类库才刚刚开始发展，远未标准化，故本书在此不拟加以讨论。本书中所描述的过程和函数很容易用于面向对象的程序，因此，本书信息也适用于C++编程。

当然，还有许多东西在本书中找不到。本书不打算包罗每个程序库中的所有函数以及所涉及函数的各个细节，那是编译器文档的任务。而本书的目的是要在绝大多数情况下使你能迅速获得答案，并同时教会你许多有关PC硬件的知识。

Robert Jourdain

出版者的话

为了促进我国计算机技术的发展，我社与美国的MACMILLAN出版公司达成协议，近期即将翻译出版一批美国新出版的计算机软件图书。其中在1994年年底及1995年年初即将出版发行的有：

1. Tom Swan's Code Secret
2. Programmer's Problem Solver（Second Edition）
3. C++By Example
4. Visual Basic for Applications By Example
5. The 'Data Recovery Bible
6. Tricks of The Windows 3.1 Masters
7. Teach Yourself visual C++1.5 in 21 Days（Revised Editions）
8. DOS Secrets Unleashed
9. Tricks of the Graphic Gurus
10. Visual C++ Object Oriented Programming
11. Using Novell DOS 7
12. Visual C++ Developer's Guide

作为领导世界计算机图书潮流的MACMILLAN公司所出版的以上图书内容新颖、通俗易懂、图文并茂、讲解生动。这些书基本上都是美国一些计算机编程大师们的经验结晶。我社翻译出版这些图书，希望对我国从事计算机应用、研究、开发的有关人员业务提高有所帮助。热忱欢迎广大读者朋友踊跃选购。

我们还应提出的是：美国CAPAO（Chinese American Publishing Company）的Michael Howard先生为了开展中美文化交流，在电子工业出版社与Macmillan公司签订出书合同方面起到了牵线搭桥作用，在此深表感谢。

目　录

第1章　程序组织

- 简要说明:程序的管理
- 截取命令行参数
- 读入或修改 DOS 环境变量
- 从一个程序中运行另一个程序
- 给 DOS 返回一个退出代码
- 使程序驻留内存
- 把程序从 .EXE 类型转换为 .COM 类型

1.1 简要说明:程序的管理

当程序装入后,它处在一个复杂的环境中,有许多因素决定着它如何运行。程序需要找出程序使用者是否规定了命令行参数。它还需要查找由 DOS 设置的向机器中运行的所有软件说明配置信息的环境变量。有时,程序需要运行其它程序。并当程序终止时,它们可能需要继续驻留内存或者给 DOS 返回一个错误代码。

DOS 程序使用两种格式之一:EXE 或 COM。与 COM 程序不同的是,EXE 程序可以大于 64K,但它们在装入内存时要求 DOS 做些处理工作。COM 程序却已是最终形式。COM 程序特别适用于较短的实用程序。在任一情况下,组成程序的代码在内存中都要由一个程序段前缀(PSP)加以引导。这是一个 100H 字节大小的区,它存放 DOS 运行程序所要求的特殊信息。PSP 还为文件I/O操作提供一个小缓冲区。

下面是 PSP 字段的一个对应表,供参考:

表 1－1　PSP 字段的对应表

偏移	字段大小	用　途
0h	dw	用来调用程序终止中断的机器代码。虽然该技术现在已经过时,程序仍可照此设计,以便终止 RET 指令把控制传送到该点。
2	dw	最高位内存的段地址。该值用于计算可用常规内存,如第 2 章"确认常规内存的可用性"所示。
4	dw	保留区。
6	dd	对 DOS 功能发送程序的长调用。
A	dd	结束地址(IP,CS)。指定程序结束时控制要传送的地方。
E	dd	Ctrl-Break 退出地址 (IP,CS)。Ctrl-Break 功能在第 7 章"编写一个 Ctrl-Break 例程"中加以讨论。
12	dd	严重错误退出地址(IP,CX)。严重错误处理程序在第 10 章"检测和恢复磁盘错误"中加以讨论。
16	22 字节	保留区。
2C	dw	程序环境的段#。参见下面"读入或修改 DOS 环境变量"章节,有关如何使用该值进入程序当前或主环境的说明。
2E	46 字节	保留区。

偏移	字段大小	用　途
5C	16 字节	参数区 1(格式化作未打开的 FCB)。该字段及下一个字段用于初始化现已不用的文件控制块,在第 12 章的"简要说明"中加以讨论。
6C	20 字节	参数区 2(格式化作未打开的 FCB)。
80	128 字节	缺省的磁盘传输区。启动时,该字段接收命令行数据。它也可以用作磁盘 I/O 的一个工作区。

1.2 截取命令行参数

当装入时,许多程序允许用户在 DOS 命令行指定参数,通常用来指明程序首先所要运行的文件名及路径。该信息被转储到程序段前缀的一个 128 字节的字段中(在本章"简要说明"中讨论过),从偏移 80H 开始。该字段也可用于磁盘传输操作,因此,若未及时访问,命令行信息可能被改写。信息写入 PSP 的形式与它被键入时完全一样。它作为一个 Pascal 串存储,其第一个字节(在偏移 80H)说明串长度。

BASIC

在 Microsoft BASIC 中,COMMAND$ 函数返回整个命令行,把它转换成大写字母,并去掉任何引导空格。你所要做的就是编写一个例程,把命令行按其各个部分分开。

```
TheLines$ = COMMAND$
```

Pascal

Turbo Pascal 的 ParamCount 函数说明有多少参数传送到了命令行,ParamStr 函数则返回每个参数。ParamCount 返回一个整型结果。ParamStr 取一个整数值,计算参数数量,从 1 向上数,并把每个参数作为一个串返回。若想列出所有参数,可如下写:

```
for I:= 1 to ParamCount do Writeln(ParamStr(I));
```

C

在 C 语言中,如下说明 main 函数:

```
main(int argc, char *argv[])
```

编译器将初始化 argc,以存放命令行参数数量值,而且,它将用指向命令行每个元素(按其被键入时的顺序排列)的一个指针填充 argv 的各元素。数组的最后一个元素是空指针。若想列出命令行的参数,可如下写:

```
for(i=0; i<argc; i++) printf("%s\n",argv[i]);
```

汇编语言

汇编语言程序必须直接从 PSP 中取命令行数据。EXE 程序可以使用中断 21h 的功能 62h,将 PSP 段地址送回存放在 BX 中。在 COM 程序中,CS 总是指向 PSP。下面的例子使命令行移到变量 COMMAND_LINE,并把它转换成类似 C 语言的以空值结束的串。

```
;---取命令行:
COMMAND_LINE db 128 dup (?)
    mov   ah,seg COMMAND_LINE          ;把 ES:DI 指向 COMMAND_LINE
```

```
        mov   es,ax                          ;
        mov   di,offset COMMAND_LINE
        push  ds                             ;保存 DS
        mov   ah,62h                         ;功能号
        int   21h                            ;找出 PSP
        mov   ax,bx                          ;移位 PSP 到 AX
        mov   ds,ax                          ;现在 DS 指向 PSP
        mov   si,80h                         ;PSP 字段的偏移
        sub   cx,cx                          ;清 CX 作计数器
        mov   cl,[si]                        ;得到串长度
        cmp   cl, 0                          ;无参数测试
        je    NO_PARAMS                      ;若无则转移
        inc   si                             ;转发 ptr 到第一字节
        cld                                  ;设置方向标志
rep   movsb                                  ;传送命令行
NO_PARAMS:                                   ;
        mov   byte ptr es:[di],0             ;空值结束串
        pop   ds                             ;恢复 DS
```

1.3 读入或修改 DOS 环境变量

DOS 维护一个“环境”，其中记录任何程序都可能使用的信息。某些 DOS 命令，如 PATH 和 COMSPEC，写或修改环境中的变量。当程序启动时，DOS 生成环境的一个备份供程序使用。例如，它可能在环境的 PATH 变量中查找目录路径，供搜索文件时使用。

该备份即称作当前环境，而原来的则叫作主环境。当程序结束时，当前环境便不再存在。因此，通常没有改变当前环境的必要。然而，各种实用程序和软件安装程序有时需要在主环境中做些长时间有效的修改。

程序当前环境的地址存放在程序的 PSP 的偏移 2Ch 处。该地址是个段值，环境在其中从偏移 0 开始。主环境的位置在基本 shell 即 COMMAND.COM 的 PSP 中的同一偏移处。中断向量 2Eh 指向该 PSP。下面的汇编语言示例说明了如何存取它。

DOS 环境是你可以配置软件的一种方法。你不必生成配置程序，可以让用户利用 DOS SET 命令在环境中设置配置信息。SET 将接受任何名称的变量，因此，你可以确定一个只有你的程序使用的变量；你可以对变量所含的信息设计你所喜欢的任何格式。该方法有助于避免配置文件与程序分离。然而，许多用户对于使用不熟悉的 DOS 命令和编辑他们自己的 AUTOEXEC.BAT 文件有畏难情绪。而且，DOS 环境很容易占满，(由于环境使用的是 SHELL 命令)一般用户又难以使之增大。

BASIC

在 Microsoft BASIC 中，ENVIRON $ 函数从环境中返回变量。其唯一的参数是一个串，用于命名一个环境变量。该串必须是大写的，否则，ENVIRON $ 返回一个空串。若想查出 PATH 变量的当前设置，请写：

PathString$ = ENVIRON$("PATH")

作为选择，也可以赋予该参数一个数值。若该值是1，ENVIRON$则返回环境中的第一个变量；若值是2，则返回第二个变量；如此类推。若没有与所给数字相对应的变量，则返回空串。可以使用这种办法搜索整个环境。

若想修改程序的备份环境，可使用BASIC的ENVIRON语句。若如下写：

ENVIRON "PATH=C:\DOS"

则环境中的PATH变量将被改变成指定的串。若想在现有的PATH变量上增加内容，可用ENVIRON$将其读入，补上追加的目录路径，然后使用ENVIRON重新插入整个串。若想从环境中去掉PATH变量，则只写一个分号即可：

ENVIRON "PATH=;"

Pascal

在Turbo Pascal中，GetEnv函数从环境中返回一个指定的变量。若想查出DOS PATH变量的当前设置，请写：

```
uses DOS;
Var
    DOSPath:string[128];
    DOSPath:=GetEnv('PATH');
```

变量名可用大写或小写字母，其中不可含有"="号。若未找到变量则返回一个空串。

可以使用EnvCount函数查出环境中包含有多少个串。该函数返回的整数，可在一个循环中用EnvStr读整个环境的内容；EnvStr取一个整型参数，若参数值是1，则返回第一个串，若是2，则返回第二个参数，以此类推。若想显示整个环境，可如下写：

```
for I:= 1 to EnvCount do Writeln(EnvStr(I));
```

Turbo Pascal未提供能在其备份环境中进行修改的过程。如果必须这样做，可在程序段前缀中查找环境的地址。Turbo Pascal的System单元包含有全局整型变量PrefixSeg，它说明了PSP的段地址。使用它可找到程序的环境的段址，存放在偏移2Ch。

C

Borland和Microsoft编译器都使用getenv函数从DOS环境中读取变量。该函数只取一个参数——即一个指针，该指针指向一个串。该串为欲取的变旺命名，可以是大写或小写。返回一个指向该变量的指针。若变量未找到则返回一个空串。可如下了解PATH的当前设置：

```
#include <stdlib.h>

char *return_string;

return_string = getenv("PATH");
```

两种编译器都使用putenv函数修改环境。其唯一的参数是个给出环境变量新值的串。若修改成功，该函数返回0。给PATH变量加上C:\DOS：

```
putenv("PATH=C:\\DOS");
```

若想修改现存变量，首先要用getenv读取它，然后修改，再用putenv把它重新插入环境中。若

想删除变量，则用 putenv，但不要指定任何内容。例如，想删除 PATH 变量，可如下写：

```
putenv("PATH=");
```

汇编程序

若想用汇编语言访问程序的备份环境，需要找到环境并遍历它。首先，找出程序的 PSP 的地址。在 COM 程序中，CS 总是指向 PSP。在 EXE 程序中，要调用 INT 21h 的功能 62h。它把 PSP 段返回到 BX 中。可在 PSP 中的偏移 2Ch 找到环境的段地址。

环境的大小，以 16 字节的节表示，放在位于环境段紧前的那一节中偏移 0003h 处的两个字节中。环境中的所有变量都由包含大写字母并以零为结束符的串组成。变量的格式与其在 SET 命令中的完全相同：首先是变量，之后是"="符号，再后是赋给该变量的值。下例把 DS:SI 指向环境的第一个字节，并把环境长度放入 CX：

```
;--- 指向环境：
    mov   ah,62h                ;功能号
    int   21h                   ;取 PSP 段
    mov   es,bx                 ;把它放入 ES
    mov   ax,es:[2Ch]           ;取环境段
    dec   ax                    ;上挪一节地址
    mov   ds,ax                 ;把它放入 DS
    mov   cx,[3]                ;取环境的大小值
    inc   ax                    ;移回指针
    mov   dx,ax                 ;
    sub   si,si                 ;DS:SI pts 到 env,准备好 MOVSB
```

如上所述，用同样方法可找到程序的主环境，但在 COMMAND.COM 的 PSP 中。该 PSP 的地址存放在中断向量 2Eh 中。下面的例子定位主环境：

```
    mov   ah,35h                ;取中断向量的功能号
    mov   al,2Eh                ;向量号
    int   21h                   ;EX:BX (EX:0000) 指向 PSP
    mov   ax,es:[002Ch]         ;取环境地址
    mov   es,ax                 ;把 ES 指向环境
    sub   di,di                 ;使用 DI 作为到 env 的偏移量
```

1.4 从一个程序中运行另一个程序

DOS 提供了 EXEC 功能(中断 21h 的 4B)，可从一个程序中运行另一个程序。第一个程序称作父进程；被装入并运行的程序叫作子进程。子进程可以是 COMMAND.COM 的第二个备份，在其中可以运行 DOS 命令。说明前可以加一个初始命令，用符号/C 起始，如下面的 Pascal 例子所示。

BASIC

BASIC 的 SHELL 语句可以装入并运行另一个程序。其格式是 SHELL command-string。命令串可以只是程序的名字，也可以是命令加上在命令行中通常跟在程序名之后的参数。如果没有指

定命令串，则装入 COMMAND.COM 的一个拷贝，并显示 DOS 提示符。任何 DOS 命令即可键入，结束时，键入 EXIT 把控制返回到 BASIC 程序。

使用 SHELL 有一些限制。例如，如果被装入的程序改变了屏幕方式，则在返回时不会作自动修正。该程序装入之前所有文件都必须关闭，而且，在结束后不得有程序继续驻留内存。其它问题参见 BASIC 手册。

在下例中，SHELL 装入程序 C:\UTIL\TRASHER.COM：

```
Shell "C:\UTIL\TRASHER.COM"
```

Pascal

Turbo Pascal 的 Exec 过程装入并运行程序。它取两个串参数。第一个串给出程序的名字和路径，第二串指定命令行参数。下面的示例装入 COMMAND.COM 的第二个备份，并要求 DIR 命令：

```
Exec('C:\COMMAND.COM', '/C DIR *.*');
```

被编译的程序需要内存来运行 EXEC 式的进程。因此，必须限制堆的大小，即必须以下面的格式用一个伪指令给程序开头：

```
{$M StackSize, MinimumHeap, MaximumHeap}
```

若无堆分配，堆则可以删除，如下例中只提供了一个 4K 的栈：

```
{$M $1000, 0, 0}
```

C

Borland 和 Microsoft 编译器都用 execl 函数装入并运行子进程。该函数取不定数目的参数，都是指向串的指针。第一个参数是将被装入的程序名和路径。后续参数是一个接一个的命令行自变量。空字符标志参数表的结束。下例运行一个叫作 DSKDEATH.COM 的程序，并把参数“512”和“UNDO”传递给它：

```
#include <process.h>
execl("C:\\UTIL\\DSKDEATH.COM","512","UNDO",NULL);
```

本函数有几种变体；其中有些把命令行自变量作为数组，有些能把环境传递给子进程。你也可控制是否在装入时为子进程进行 DOS PATH 搜索。详情参见有关资料。

汇编程序

功能 4Bh 比其它大多数功能都要复杂一些，它要求四个准备步骤：

(1)为程序准备好可用内存空间。

(2)建立一个参数块。

(3)建立一个驱动器、路径及程序名串。

(4)在变量中保存 SS 和 SP 寄存器。

内存中必须辟出空间，因为程序在装入时 DOS 把整个内存都分配给它。若不辟出自由空间，则无法装入第二个程序。第 3 章“分配/释放常规内存”说明如何使用 SETBLOCK 功能做此工作。内存空间辟出之后，只要在 BX 中放入所需的 16 字节大小的节的数目，把 4Ah 放入 AH，并执行中断 21h 缩小内存分配表，这样，程序只能使用所请求节数的内存用量。

EX:BX 必须指向的 parameter block，是一个 14 字节的内存块，其中可以存放下列四条信息：

```
dw      环境串的段地址
```

dd　　命令行的段/偏移
dd　　第一个文件控制块的段/偏移
dd　　第二个文件控制块的段/偏移

环境串必须从节边界开始。这是因为指向该串的参数块中的输入项存放的只是两字节的段值。如果新程序与装入它的那个程序运行环境相同，则这一切都可以避免，只要把 ASCII 码 0 放入参数块的头两个字节即可。

参数块的下面四个字节指向用于正被装入程序的命令行。命令行开头的字节含有串的字符数量，串结束字节是 ASCII 码 13(它不计入串长度内)。

参数块的最后八个字节指向文件控制块(FCB)。FCB 中存放有关命令行中指定的文件的信息。如果没有准备以此方式打开的文件(实际上几乎从来没有)，那么用 ASCII 码 0 填入这八个字节。

最后，必须建立一个驱动器、路径和文件名串。该串对被装入的程序命名。当执行 EXEC 时，DS:DX 指向该串。该串是个标准的"ASCII Z 串"，它只是个驱动器说明符、一个树形目录路径和文件名及扩展名，以 ASCII 码 0 字节结束。

上述信息设置完毕后，还有最后一项工作要做。程序被调用时，所有寄存器都被更新。栈段和栈指针必须保存起来，以便当控制返回调用程序时能够加以恢复。留出一些变量来做这项工作。因为 DS 也被破坏，这些变量必须相对于 CS 进行存取。SS 和 SP 保存之后，把 0 放入 AL，选择"装入并运行"选项。然后，把 4Ah 放入 AH，并调用中断 21h。此时，实际上有两个程序在运行，父程序继续"挂起"。DOS 提供一种方法，让子程序把返回代码传送给父程序，这样便可以报告错误和状态。下面的"给 DOS 返回一个退出代码"一节说明了如何做到这点。至少，如果出现过错误，则在返回时设置进位标志；在这种情况下，若功能号无效，AX 返回 1；未找到文件，返回 2；磁盘问题，返回 5；内存不够，返回 8；环境串无效，返回 10；格式无效则返回 11。

下例是可能出现的最简单的情况，但通常 EXEC 过程也不会要求更多。置整个参数块为零，不要建立环境串。这意味着没有命令行传送给被装入的程序，且环境将与调用程序相同。你只要修改内存分配、设置文件名和(空)参数块，并保存 SS 和 SP 即可。

```
;---在数据段:
FILENAME    db  'C:TRIAL.EXE',0;load TRIAL.EXE from drive C
;---重新分配内存:
    mov     bx,sp                   ;SP 指向栈顶
    mov     cl,4                    ;用16除
    shr     bx,cl                   ;
    mov     ax,S3                   ;加入栈段
    add     bx,ax                   ;程序结束处的地址
    mov     ax,es                   ;取程序开始时的节#
    sub     bx,ax                   ;计算以节表示的程序的大小
    inc     bx                      ;加1保证安全
    mov     ah,4Ah                  ;功能号
    int     21h                     ;进行重新分配
;---指向参数块:
    push    CS                      ;ES 保存段
```