MS-DOS批处理程序应用与技巧

宫文超 严立公 贺 翔 编著

清华大学出版社

MS-DOS批处理程序应用与技巧

MS-DOS 批处理程序应用与技巧

宫文超　严立公　贺　翔　编著

清华大学出版社

(京)新登字 158 号

内 容 提 要

随着广大微机应用人员对 DOS 应用水平的不断提高，介绍 DOS 的简单使用命令已不能满足读者的需要了。批处理程序设计，正是读者在熟悉了 DOS 之后应当进一步深入学习的基于 DOS 本身的一种简单易学的程序设计技术。

本书深入浅出地介绍了 DOS 批处理程序设计的基本概念，设计方法、语句控制，以及若干批处理程序的妙用——配置系统、建立菜单、文件归档、抗病毒等等。

本书适于广大 DOS 用户和电脑爱好者阅读。

清华大学软件部特别销售本书配套程序磁盘，联系电话 010-2594891。

图书在版编目(CIP)数据

MS-DOS 批处理程序应用与技巧/宫文超等编著. —北京：清华大学出版社，1995
ISBN 7-302-01954-1

Ⅰ. M… Ⅱ. 宫… Ⅲ. 微型计算机-磁盘操作系统-成批处理-程序设计 Ⅳ. TP316.3

中国版本图书馆 CIP 数据核字(95)第1[illegible]号

书 名：MS-DOS 批处理程序应用与技巧
作 者：宫文超 严立公 贺翔
出版者：清华大学出版社(北京清华大学校内，邮编 100084)
印刷者：国防工业出版社印刷厂
发行者：新华书店总店北京科技发行所
开 本：787×1092 1/16 印张：11.25 字数：263 千字
版 次：1995 年 10 月第 1 版 1995 年 10 月第 1 次印刷
印 数：0001—6000
书 号：ISBN 7-302-01954-1/TP・900
定 价：15.00 元

前　言

当前，微型计算机技术发展迅猛，应用领域不断扩大，已由单一的计算工具扩展到计算机辅助设计、工业控制、现代化管理，以及办公自动化等领域，甚至已进入家庭成为当代科学、技术、社会生活与家庭生活的主要部分。

微型计算机在任何领域的应用，都离不开硬件资源和信息资源的管理程序——磁盘操作系统 MS-DOS。目前使用较为广泛的微机操作系统是 MS-DOS 磁盘操作系统，也常可见到 IBM-DOS 或 PC-DOS、DR-DOS 等。随着微机应用的日益广泛和深入，MS-DOS 操作系统的版本不断升级，已由最初的 1.0 版本发展到目前的 6.0 版本，其功能也逐步完善。

多数介绍 MS-DOS 的书籍或手册，对于 MS-DOS 提供的一种特殊文件——.BAT 程序，即批处理程序都只作一般性的介绍。本书则以.BAT 文件为主要研讨方向。全书分 10 章详细介绍建立、使用批处理程序的方法。对批处理程序的功能进行了比较深入的分析和发掘。前 9 章分别阐述了 MS-DOS 的构成、组成批处理程序的语句，建立、使用批处理程序的方法等。第 10 章以实际例子介绍了批处理程序的应用技巧。

本书的编写，力求概念清楚、准确，通俗易懂。为了使读者掌握批处理程序的使用方法，书中还列举了一些实例，都在 AST 386/SX 微机上，在最为流行的 MS-DOS 3.30 版本上运行通过，如在其他机器上和 DOS 5.0 及其以上版本运行，屏幕显示结果可能会有差别，但功能上不至有大的变化。

本书的编写和出版得到了清华大学出版社编辑同志的大力支持和帮助，编者在此表示衷心的感谢。

为方便读者学习，特将书中所介绍的批处理程序专辑成盘。欲购者请与清华大学出版社软件部联系（电话 010-2594891）。

由于编者水平有限，书中错误及不当之处在所难免，恳请广大读者批评指正。

编　者

1995 年 6 月

目　录

第 1 章　磁盘操作系统 DOS 简介

1.1　DOS 概 要

1.1.1　DOS 的用途

所谓 DOS 是英文 Disk Operating System(磁盘操作系统)的缩写。DOS 是微型计算机系统的重要组成部分。它负责计算机各组成部分的互相配合协调,是微型计算机所有软硬件资源的组织者和管理者,DOS 提供了一个人机接口,对大多数用户而言,DOS 主要用于接受并执行用户输入的各种键盘命令,如同一个命令处理器。此外还有一种情形,即 DOS 运行后通过一个支撑应用程序来控制计算机,这种软件通常叫做命令解释器,它可以提供给用户可选择的屏幕菜单,用户根据菜单便可运行各种 DOS 命令,而不必记住命令本身。对于系统程序员而言,DOS 提供了磁盘、目录、文件和程序控制功能的低级接口。归纳起来 DOS 有以下几方面的基本用途:

(1) 管理计算机的全部软硬件资源。包括中央处理器、存储器、各种外部设备、磁盘驱动器、打印机、绘图仪、鼠标器等硬件资源,以及程序和数据等软件资源,使它们能被充分利用,有效地工作。

(2) 负责用户与计算机的接口,使用户不必详细了解计算机内部结构便可以顺利地完成各种操作任务。

(3) 在高版本的 DOS 中,它向用户提供了一些常用的应用软件。如在 DOS 3.3 中提供了 GWBasic 编译,在 DOS 4.0 以上版本中提供了 DOSSHELL 程序;在 DOS 5.0 和 6.0 中除 Qbasic 外还提供了许多类似于 PC 工具的实用程序。这些内容大多散见于以往的各种软件包中,使用 DOS 可以使用户不必另外购买,买上一套完整的 DOS 就可以得到常用的软件程序。

1.1.2　DOS 的组成

应用广泛的 MS-DOS 由四部分组成:

(1) 自举记录块(Boot Reccrd)。

系统启动时,由 ROM-BIOS 中的引导程序(INT 19H)将自举记录读入从 0∶7C00H 开始的内存区。再将控制权移给自举记录块,由它将 DOS 的主体装入内存。若系统通过软盘启动,则自举记录块放在软盘的 0 面 0 道 1 扇区;若系统由硬盘启动,则硬盘的第一扇区为主引导扇区,内有分区表和主自举程序。严格地说,自举记录块不能算是 DOS 的一部分。

(2) DOS-BIOS 模块,文件名为 IBMBIO.COM。

这一模块包括两个部分:一是系统初始化程序 SYSINT,完成系统初始化过程中的部分操作。如确定内存,定位 DOS-Kernel 模块;另外一部分是标准字符和块设备驱动,包含

一组设备驱动程序，由 DOS-Kernel 通过设备请求来调用。

(3) DOS-Kernel 模块，其文件名为 IBMDOS. COM。

该模块又叫做文件管理和系统调用模块，是 DOS 的核心。它由内核初始化程序和系统功能调用程序两个部分组成。前者完成 DOS 内部初始化工作，如设置 DOS 中断向量入口，检查常驻的设备驱动链等。后者主要是 DOS 系统调用程序 INT 21H，向用户提供一整套独立于硬件的系统功能。

(4) DOS-Shell 模块，文件名为 COMMAND. COM。

该模块是命令处理程序模块，是 DOS 的外壳程序，充当 DOS 和用户之间的接口，其任务是对用户输入的内部及外部命令加以解释并执行。该程序由如下三个部分组成：

- 常驻内存部分
- Shell 初始化程序
- 暂驻内存部分

在系统启动时，该程序装入内存并分为暂驻和常驻两个部分。启动成功后，Shell 初始化程序退出内存。

1.2 启动 DOS

1.2.1 DOS 的引导过程

DOS 启动也叫 DOS 系统引导，就是从系统引导盘上读取 DOS 并把它们装入内存的过程。系统可以从软盘上引导也可以从硬盘上引导。计算机开机以后，要进行加电自检，检查内存、驱动器、打印机等内外部设备，然后就进行 DOS 系统引导。计算机首先检查驱动器 A 中是否有磁盘，如果驱动器 A 中有软盘插入，并且小门关闭，计算机便试图从该软盘上引导装载 DOS。如果驱动器 A 中没有磁盘，或是小门没有关闭，计算机便试图从第一个硬盘上引导 DOS。引导 DOS 需要由 DOS 系统盘来完成。上面一节已谈到了 DOS 主要由四个部分组成。DOS 系统盘必须含有如下三个文件：IBMBIO. COM(或 IO. SYS)文件，IBMDOS. COM(或 MSDOS. SYS)和 COMMAND. COM，其中前两个文件是隐含文件，必须放在磁盘的根目录中。将一张软盘做成系统引导盘时简单地将这三个文件拷贝到磁盘上是不行的。在引导 DOS 时，如果计算机发现驱动器 A 中的软盘不是系统盘，便会提示这样的错误信息：

This disk is not bootable

……

Please insert a DOS diskette into the drive and strike any key…

———————

提示用户插入系统盘并按任意键。在引导 DOS 时有时会出现这样的错误信息：

Bad or missing command interpreter

丢失命令处理器或命令处理器错误

这是由于在引导盘的根目录中找不到 COMMAND. COM 文件。出现这样的信息

后，用户必须在系统盘的根目录中加入 COMMAND. COM 文件(以后会讲到 COMMAND. COM 也可以不在根目录中)。有的高档微机，如 AST386 等可以选择禁止从驱动器 A 上引导 DOS，一旦用户在 Setup 程序中选择了这一项，在引导 DOS 时就会出现这样的信息：

```
.....
Floppy Disk CNTRLR ERROR OR NO CNTRLR PRERSENT
.....
PRESS F1 KEY TO CONTINUE OR CTRL-ALT ESC FOR SETUP
```

这时，按一下 F1 键就可以从硬盘上引导 DOS。

1.2.2 冷启动

计算机在关机状态下开机引导系统叫冷启动。冷启动时计算机先进行加电自检，然后再引导 DOS。

1.2.3 热启动

计算机在主机通电的状态下也可以重新引导 DOS。同时按下 Ctrl、Alt、Del 这三个键就可以达到这个目的，这就是热启动。热启动时由于处在开机状态，所以计算机不执行加电自检。

1.2.4 如何制做系统盘

前面讲过系统盘必须含有 IBMBIO. COM ，IBMDOS. COM ，COMMAND. COM 这三个文件，前两个文件不能简单地拷贝，因为它们必须存放在磁盘 0 面 0 磁道 1 扇区上。制作系统盘是非常必要的，微机用户手头应该备有随时可用的系统引导盘，以备由于硬盘系统文件丢失、病毒破坏等原因导致计算机不能正常引导时，从软盘上引导系统。下面我们介绍三种制做系统盘的方法：

(1) 在格式化磁盘时选择“ /S ”参数。在格式化时只要键入 FORMAT A：/S，格式化完成后，磁盘便是一张系统引导盘。

(2) 用 DOS SYS 命令传输系统文件。前提是在格式化磁盘时必须选用“/B”参数。/B 参数负责在格式化磁盘时给系统文件留下特定的空间。

(3) 使用工具软件。许多应用软件，如 NORTON UTILITIES ，PCTOOLS，PCSHELL 等都可以用来制做系统盘。

IBMBIO. COM 和 IBMDOS. COM 是隐含文件，只能放在根目录中，而 COMMAND. COM 则不同，不但可以随意拷贝，而且也可以放在某一子目录中。在系统配置文件 CONFIG. SYS 中指定 COMMAND. COM 所在目录，并在 AUTOEXEC. BAT 中用 SHELL 命令指明 COMMAND. COM 所在的目录，CMMAND. COM 就可以正常工作。这在以后的章节中会详细介绍。

1.3 DOS 的文件

1.3.1 DOS 的文件类型与文件命名的规定

DOS 把存放在磁盘上的数据和程序叫做文件。每个 DOS 文件都有一个名称，文件名称由两部分组成：文件名与扩展名。文件名最多可包含 8 个字符，可用的字符是 ASCII 字符集中的字符。汉字可以被用来做为文件名，每个汉字占两个字符的位置。扩展名最多只可用三个字符，与文件名之间以圆点隔开，如：CCDOS.COM。下面列举的都是有效文件名：

CCED.EXE	YLG.1
GONGWEN.DAT	YAN-2
UCDOS.!!!	SYSTEM1.W

下面都是无效文件名：

1234-WSGDE	（字符过多）
CWS..COM	（句点为非法字符）
.EXE	（无文件名）
BLAN K.EXE	（空格非法）
OLD.DATA	（扩展名多于三个字符）

另外，DOS 对某些设备具有保留名，这些保留名是被 DOS 使用的文件名，用户不能用它们建立文件。这些保留名是：

AUX	CON	LPT3
CLOCKS	LPT1	NUL
COM1	LPT2	PRN
COM2		

DOS 内部命令一般不能用来作为文件名，因为 DOS 内部命令在引导系统时已装入内存，是不能更改的。如果用 DOS 内部命令作为一个程序的文件名，这个程序便不能执行，但仍可以用 type 命令显示其内容。比如，用户编辑了这样一个文件：

```
DIR.BAT ↵
    echo off
    cd dos
    GWBASIC
```

这个批处理程序并不能运行，键入 DIR 将列出当前磁盘当前目录的全部文件。但是，DIR.BAT 这个文件仍存在于当前目录中。像这样的内部命令还有：MD，RD，CD 等，用户在编辑文件时尽量不要用它们作为文件名。关于 DOS 内部命令，以后的章节中会详细讲述。DOS 文件的扩展名一般用于指出文件的内容及属性。下面是通用的文件扩展名的含义：

COM　DOS 命令或可执行程序

EXE　DOS 可执行程序
BAT　DOS 批处理程序
SYS　可安装的设备驱动程序
TXT　文本文件
DAT　数据文件

1.3.2　DOS 文件标识符

DOS 命令是对文件标识符操作的。文件标识符同文件名的区别在于,它不但提供了文件名,还提供了查找文件存放地点所需要的信息。文件标识符格式是:

[drive] [Path] [filename] [.ext]

其中:

[drive]:　指示文件所在磁盘驱动器标识符(A,B,C 或 D 等) 缺省值为当前驱动器。

[path]:　指文件所在磁盘子目录的路径名称。该路径是 DOS 查找文件的必经之路。缺省值为当前工作目录。

[filename]:文件名。遵循有关 DOS 文件名的规定。DOS 通配符也是有效字符(下一节将介绍 DOS 通配符)。

[.ext]:　文件扩展名。遵循有关 DOS 文件扩展名的规定。DOS 通配符也是有效字符。

比如有这样一个文件:PC430.EXE,存放在 C 盘"YLG" 子目录中;那么它的文件标识符应是:C:\YLG\PC430.EXE。如果当前工作目录为 A:\CCED,要运行 PC430.EXE,须键入:

C:\YLG\PC430

当然,如果在 DOS 环境中用 path 命令设置了有关的路径,则只需键入 PC430 即可运行这个程序。

1.3.3　DOS 文件通配符

DOS 允许在查找文件时用字符"*"和"?"来代替其它字符。查找 DOS 文件时在文件名中加"?",它的含义是:任何字符都可以替换文件名中的这个位置。如:

C:\dir CCED?.TXT

这个命令执行的结果是,列出所有以 CCED 开头,后跟任何一个字符的.TXT 文件:

```
Volume in drive C has no label
Direetory of C:\
CCDE1.TXT     5034     19-10-92     10:00a
CCED2.TXT     3051     19-10-92     13:20p
CCED3.TXT     769      21-10-92     11:10a
```

```
CCED4. TXT      0       8-8-92     8 : 00a
CCED5. TXT    356       1-8-92     9 : 30a
```

又如：键入下面命令：

```
C : \dir cws???. EXE
```

结果是：

```
Volume in drive C has no label
Directory of C : \

CWSKEY. EXE
CWSCFG. EXE
```

CWS 后面的三个符号通知 DOS 对文件名后面的三个字符不必关心，只要前三个字符是 CWS，扩展名是 COM 即可。注意，文件名中的“?”只占据一个字符的位置，这与后面介绍的“*”字符有所不同。字符“*”不仅表示任何字符都可占据它的位置，而且只要在 DOS 对文件名和扩展名的规定范围内，则字符数不限。请看下面例子：

```
C : \> dir  *. BAT
```

这个命令的含义是，列出 C 盘根目录中所有以 BAT 为扩展名的文件：

```
YLG1. BAT
CWS. BAT
CCED. BAT
```

又如：

```
C : \> del  *. *
```

这个命令将删除 C 盘根目录中所有的文件。注意到通配符“?”与“*”使用上的区别，在使用文件通配符时就能够根据用户自己的需要正确合理地加以运用。

1.3.4 文件的类型及属性

DOS 文件的分类方式很多，没有一个固定的标准。本章 1.3.1 节已介绍了一些使用固定扩展名的文件，这也是一种文件的分类方式。对于同一类型文件固定使用同一扩展名是有好处的，尽管显得有些单调，但对于识别文件内容，管理好文件是非常方便的，更何况有些扩展名本身具有 DOS 命令的作用，如. BAT，DOS 将执行 BAT 文件中的所有 DOS 命令，换一个扩展名则起不到这样的作用。文本文件(. TXT)采用标准 ASCII 码，可以用 TYPE 命令显示其内容，也可以在某些编辑软件中显示、修改。而系统文件、程序文件等多是以二进制代码表示，无法将它们对应的内容显示出来。通过扩展名可以将这些文件很快地识别出来。这样，可以将 DOS 文件简单地加以分类：文本文件、程序文件、系统文件。下面所给出的是 DOS 常用文件的分类及其扩展名。

表 1-1 DOS 文件分类表

文本文件		程序文件		系统文件	
扩展名	内容	扩展名	内容	扩展名	内容
DAT	数据文件	C	C 语言文件	BAK	后备文件
DOC	文档文件	BAS	BASIC 文件	BAT	批处理程序
HLP	帮助文件	FOR	FORTRAN 文件	BIN	二进制文件
MSG	信息文件	PAS	PASCAL 文件	COM	系统执行文件
TXT	文本文件	ASM	汇编文件	EXE	可执行文件
LIS	列表文件	COB	COBOL 文件	OBJ	目标代码文件
		LIB	库文件	OVR	覆盖文件
		H	头部文件	SYS	系统文件
		MAP	映射文件		
		DBF	DBASE 数据库文件		
		PRG	Foxbase 程序文件		

在 DOS 文件操作的命令中有这样一个命令:ATTRIB,它控制改变文件的属性。文件的属性分为以下几类:

1. Read-Only(简称 R,只读)

文件只要带上这个属性,一般无法从 DOS 目录中随意删除、改变。因此,给文件加上这个属性可以在一定程度上保护该文件不被破坏。

2. Hidden(简称 H,隐含)

所谓隐含,就是在 DOS 提示符下不显示文件名,使用 dir 命令时查不到该文件,只有用 PCTOOLS 等工具软件才可以找到隐含文件。隐含文件在 DOS 提示符下拒绝 COPY,ERASE,DIR,DEL 命令。DOS 3.0以上版本,用 type 命令仍然可以显示文件内容。DOS 3.0以下版本对隐含文件不能进行任何操作。

3. System(简称 S,系统)

这类文件在 DOS 提示符下一般无法显示出来,并不接受 Copy ,del,ERASE 等命令。

4. Archive(简称 A,备份)

当某个文件被更改时,DOS 即把此文件标为"Archive",表明该文件需要备份。

文件的属性在 DOS 提示符下一般是无法知道的。在 DOS 提示符下对以上四类文件中的前三类进行操作会带来一些麻烦。下面举例说明。对于 Hidden 属性的文件,用户在使用 Copy,del,Erase,dir 等 DOS 命令对该文件进行操作时将得到以下的信息:

```
C:\> del YLG.BAT
File not found

C:\> copy YLG.BAT a:
```

File not found

而对于 Read-only 属性的文件，使用 del，erase 命令将得到：

C：\> del ylg. bat
access denied

对这两种属性的文件也不能进行编辑修改：

C：\> copy con YLG. BAT
echo off
echo OK! OK! OK!

File creation error
　　0 File(s) copied

上例中的 YLG. BAT 为只读或隐含文件，这两种属性的文件如果是. EXE，. COM，. BAT文件，在 DOS 提示符下仍然可以正常运行。同上面两种属性的文件有所不同的是，SYSTEM 属性的文件在 DOS 提示符下可以查找显示，但如果对它进行任何操作都将失败。

外部命令 ATTRIB 的完整语法格式是：

[drive：][path] ATTRIB [±r][±a][文件标识符][/s]

其中：

[drive：]　指 ATTRIB. COM 所在盘驱动器标识符。缺省值为当前驱动器。
[path]　指 ATTRIB. COM 所在子目录路径名。缺省值为当前子目录。
[±r]　表示设置(+r)或释放(-r)文件只读属性。
[±a]　表示设置(+A)或释放(-A)文件 Archive 属性。
[/s]　表示选择处理子目录中所有的文件。但对隐含文件和系统文件不能显示其属性，也不能改变其属性。

在 DOS 提示符下如想知道文件的属性，只有通过命令：

C：\> ATTRIB CONFIG. SYS(显示文件 CONFIG. SYS 的属性)

　　R　C：\CONFIG. SYS

又如：C：\> ATTRIB *. *（显示 C 盘根目录中所有文件的属性）

A R C：\ATTRIB. EXE
A R C：\COMMAND. COM
A R C：\AUTOEXEC. BAT
　　AUTOEXEC. BAK
……

若只想了解文件的某一属性：

```
C:\> ATTRIB *.* | Find"R"

R C:\AUTOEXEC.BAT
R C:\ATTRIB.EXE
.....
```

ATTRIB±a 只对 DOS3.2 以上版本适用。对隐含文件和系统文件的操作只有通过 PCTOOLS 等工具软件来实现，ATTRIB 命令对它们不起作用。一个文件一旦被加上隐含和系统文件属性，那么在使用 ATTRIB 显示其属性时将得到这样的提示信息：

```
Invalid path or file not found
```

1.4 DOS 命令简介

DOS 命令很多，随着 DOS 版本的不断升级，DOS 命令也在不断地增加。由于本书不是专门讨论 DOS 的，仅以 DOS3.3 为例作简单地介绍。DOS 命令可分为内部命令和外部命令两大类，这一节分别作以简单介绍。内部命令和外部命令的概念在前面已讲过了，这里不再重复。应当指出的是 DOS 的内部和外部命令的划分并不是一成不变的，随着 DOS 的升级，原来的外部命令可能会变成内部命令。由于内部命令随操作系统一同装入内存，大大提高了计算机运行速度，但另一方面因为占据了内存空间，一定程度上影响其他应用软件的装载运行。

1.4.1 DOS 内部命令

在这里不打算介绍 DOS 的全部内部命令，只就常用的一些命令作以简要介绍。有关 DOS 配置的命令以及批处理程序的命令将在以后的章节中另作详尽的讲述。

1. CD(CHDIR)：

格式： CD\[驱动器标识符] [路径名]

功能： 显示或改变指定当前目录。

举例： A> CD(显示当前工作目录)

```
A:\CPAV
```

又：

```
C:\CCED> CD\CPAV(从当前目录改变路径进入到 C:\CPAV 中)
C:\CPAV>
```

又：

```
A:\CPAV>  CD\(退出当前目录至根目录)
A:>
```

2. CLS

格式： CLS

功能:清除屏幕显示并将光标移至左上角。

3. COPY

功能:复制文件至新的目标盘驱动器、磁盘、子目录或文件。

格式:COPY [/A][/B][源文件标识符][/A][/B] [目标文件标识符[/A][/B][/N]

其中:

[/A] 表示 COPY 将文件按 ASCII 文件对待。拷贝文件内容直到 Ctrl—Z 标志。

[/B] 表示 COPY 长度以文件在目录表中的长度为准。

[/N] 表示 COPY 读出每个扇区,并检验每个扇区的数据记录。

COPY 命令支持文件通配符。

4. CTTY

功能:将标准输入输出设备指定为辅助设备。

格式:CTTY [设备名]

5. DEL

功能:删除磁盘文件

格式:DEL [文件标识符]

DEL 命令支持通配符,但不能删除子目录。

6. ERASE

功能:删除磁盘文件

格式:ERASE [文件标识符]

其功能同 DEL 命令完全相同。

7. DIR

功能:列出指定磁盘目录的文件名(包括子目录名)。

格式:DIR [文件标识符][/P][/W]

[文件标识符] 缺省时将列出指定目录中所有文件名。

[/P] 表示当名文件名显示满一屏时,暂停滚动,按任一键后继续。

[/W] 表示仅显示文件名,而忽略文件长度、建立日期、建立时间,文件名在屏幕上横向排列。

DIR 支持 DOS 通配符。

8. EXIT

功能:终止辅助命令处理器执行。

格式:EXIT

9. MD(MKDIR)

功能:建立子目录。

格式:MD [驱动器]\[子目录名]

所建子目录的名称长度不可超过 62 个字符,而且要考虑所在目录串的长度,目录串的长度不能超过 255 个字符。

举例: C> MD A:CCED

将在 A 驱动器磁盘的当前目录下建立一个名为“CCED”的子目录。

10. REN(RENAME)

功能：重新命名文件。

格式：REN [原文件名标识符][新文件名]

REN 支持 DOS 通配符，它不能给子目录更名。

11. RD(RMDIR)

功能：删除子目录。

格式：RD [驱动器][子目录名]

RD 只能删除空目录，不能删除根目录和当前工作目录。注意，在删除子目录时一定要先删除目录中的所有文件，包括隐含文件和其它属性的非显示文件，否则将得到这样的信息：

Invaid path or directory not empty

12. TYPE

功能：在标准输出设备上显示文件内容。

格式：TYPE [文件标识符]

TYPE 命令不支持 DOS 通配符。

13. VER

功能：显示当前 DOS 版本。

格式：VER

14. VOL

功能：显示指定磁盘卷标。

格式：VOL [驱动器]

1.4.2 DOS 外部命令

DOS3.3 的外部命令一共有 33 个。其中有些命令在前面已讲过，比如 ATTRIB 等，还有一些在以后的章节会讲到。在这一小节中只讲述一部分常用命令以及和与本书内容有关的 DOS 外部命令。

1. APPEND

功能：设置 DOS 查找数据文件路径。

格式：APPEND[驱动器:][路径];[驱动器:][路径]…[/X][/E]

[/E] 将 APPEND 命令设置的路径存入 DOS 环境中。

[/X] 如果在使用 append 命令设置查找数据文件路径之前，使用 APPEND/X，那么 DOS 将首先查找当前目录；如果没有找到再按 APPEND 所设置的路径查找。

2. ASSIGN

功能：将一个驱动器标识符分配给另一个驱动器。

格式：ASSIGN [驱动器 1]=[驱动器 2]

举例：ASSIGN A=B

这个命令的结果是，将驱动器 A 同驱动 B 等同起来。这时，所有对 A 盘的访问请求均被指向 B 盘。注意，一旦启动了 ASSIGN 命令，禁止使用 BACKUP，RESTORE，LABEL，JOIN，SUBST，PRINT 等命令。

3. CHKDSK

功能：查寻指定驱动器上的磁盘信息，检查纠正磁盘错误。

格式：CHKDSK [驱动器：][路径名][/F][/V]

CHKDSK 是个非常有用的 DOS 命令，用户经常运行一下这个命令，以便及时发现磁盘上的错误并加以纠正。如果在这个命令后跟一个文件名，那么 CHKDSK 除了提供磁盘信息外，还会提供有关这个文件的信息。

[/F] 修改磁盘错误。如不加此参数，CHKDSK 只提供有关错误信息。

[/V] 检查磁盘时显示每个子目录中的每个文件的文件名。

注意，CHKDSK 对使用 SUBST 和 JOIN 命令生成的驱动器不起作用。

举例：

```
Sun 10-25-1992 17：40：18.49 C：\>chkdsk
VOLUME 1 created Oct 9,1992 3:40p
Errors found,F parameter not specified.
corrections will not be written to disk.
C：\DB3\RML.DBF
Allocation error,size adjusted.
33447936 bytes total disk space
5220352 bytes in 213 hidden files
  155648 bytes in 67 directories
27924480 bytes in 1684 user files
  147456 bytes available on disk

  655360 bytes total memory
  570800 bytes free
```

4. COMP

功能：比较两组文件的内容是否相同。

格式：COMP [驱动器：][文件名 1][驱动器：][文件名 2]

COMP 命令支持文件通配符。

举例：

```
Sun 10-25-1992 17：40：39.69 C：>
Sun 10-25-1992 18：02：50.04 C：\DOS>comp byg.bas map.bas
C：BYG.BAS and C：MAP.BAS
Files are different sizes
Compare more files(Y/N)? N
```

5. DISKCOMP

功能：比较源驱动器中磁盘同目标驱动器中磁盘的内容。

格式：DISKCOMP [驱动器 1：][驱动器 2：][/L][/8]

[/L] 指示 DISKCOMP 只比较磁盘的第一面。

[/8] 只比较磁盘每个磁道的前 8 个扇区。

举例：

```
Mon 11-23-1992 C：\>diskcomp a：b：/1/8
Insert FIRST diskette in/drive A：
Insert SECOND diskete in/drive B：
Press any key when ready ...
Comparing 40 tracks
8 sectors per track,1 side(s)
Compare error on side,0,track 0
Unrecoverable read error on drive B：
side 0,track 1
```

6. DISKCOPY

功能：将一张软盘上的内容完全复制到另一张软盘上（目标盘可以是未格式化的）。

格式：DISKCOPY [驱动器 1：][驱动器 2][/1]

[驱动器 I] 为源盘。

[驱动器 2] 为目标盘。

[/1] 允许用户只复制磁盘的一面。

注意，DISKCOPY 工作的对象是软盘，对硬盘不工作，而且源盘与目标盘必须匹配，即同是高密盘或双面双密度盘。用 DISKCOPY 命令复制磁盘时，目标盘被格式化，原来盘上的所有文件将被破坏。

7. FIND

功能：在指定文件中查找指定的字符串。

格式：FIND [/V][/C][/N]"字符串"[驱动器：][路径名]

[/V] 显示所有不含指定字符串的文件行。

[/C] 只显示每个文件中含有指定字符串的行数。

[/N] 显示指定字符串在文件中的行号。

注意，FIND 命令的命令行中，所指定的字符串一定要加引号。FIND 命令不支持 DOS 通配符。在使用该命令时如果同时加参数/C，/V，那么 FIND 将显示文件中不含指定字符串的行的行数；如果同时加参数/C，/N，那么 FIND 会忽略参数/N。

举例：C> DIR B:FIND/V"well"

这个命令将显示 B 盘上所有不含"well"的文件的文件名。

8. FORMAT

功能：格式化指定驱动器上的磁盘，以存放 DOS 文件。

格式：FORMAT [驱动器：][/1][/4][/8][/N][/S][/B]

[/1] 指示只格式化软盘的一面。

[/4] 标示在高密驱动器中格式化双面双密度软盘。

[/8] 指定每个磁道格式化八个扇区。

[/B] 指定在格式化时为系统文件留下空间。

[/S] 指定在格式化时加系统文件。

[/V] 指示 FORMAT 在格式化时提示用户加磁盘卷标，卷标的长度限定在 11 个字符之内。

9. JOIN

功能:将一个磁盘驱动器标识符赋予一个子目录。

格式:JOIN [驱动器]:[驱动器]:\[路径名]

注意,以下命令对 JOIN 命令所生成的驱动器无效;

CHKDSK

DISKCOPY

FDISK

FORMAT

LABEL

RECOVER

SYS

JOIN 只对第一级目录工作,如:

JOIN D: C:\CCED

而下面的操作无效:

JOIN D: C\CCED\YCG

10. LABEL

功能:在指定盘上建立、更换或删除卷标。

格式:LABEL [驱动器:][卷标]

以下这些字符不能用作卷标:

* ? / \ , . ; : + = < > [ ] ( ) & ^ 。

有时需要制做这样的卷标“Y. L. G”以区别“YLG”,这个问题用 NORTON UTILITIES 工具软件可以得到解决。

11. RECOVER

功能:恢复含有坏扇区的文件或磁盘。

格式:RECOVER [驱动器:][路径][文件名]

用 SUBST 或 JOIN 命令生成的驱动器不能使用 RECOVER 命令。

12. SUBST

功能:用一个驱动器标识符来代替一个子目录。

格式:RECOVER [驱动器][路径][文件名]

同样地,下面的命令不适用于 SUBST 命令定义的驱动器:

CHKDSK
DISKCOPY
FDISK
FORMAT
LABEL
RECOVER
SYS

举例：SUBST Z： C：\DOS

这个操作的结果是，驱动器 Z 将代替 C：\DOS 子目录，所有对 C：\DOS 的操作都可转向 Z，但是在 SUBST 命令中所指定的驱动器 Z 实际上不存在，上面的操作无效。若在配置文件 CONFIG.SYS 中写入：

LASTDRIVE=Z

那么上面的操作便可以工作。

13. SYS

功能：向指定盘中传输 DOS 系统文件。

格式：SYS [驱动器]

SYS 命令所传输的系统文件是：

IO.SYS(或 IBMBIO.COM)

MSDOS.SYS(或 IBMDOS.COM)

而 COMMAND.COM 则不能用 SYS 传输，需用 COPY 命令复制到指定盘上。

14. XCOPY

功能：复制文件或子目录到指定盘上。

格式：XCOPY [驱动器:][路径] [驱动器:][路径][/E][/S]

[/E]：复制所有子目录，包括空目录，必须同 /S 连用。

[/S]：复制子目录以及下一级子目录，不包括空目录。

XCOPY 命令比 DISKCOPY 命令优越之处在于，XCOPY 命令不破坏目标盘上原有的文件，也不要求源盘与目标盘必须相匹配。

第 2 章　批处理程序的基本概念

2.1　什么是批处理

在使用 DOS 命令的过程中，有时需要连续执行几条命令，每输入一条命令，必须等它完成后才能输入下一条命令。完全相同的一组 DOS 命令，有时需要反复执行，例如要把驱动器 A 中软盘上的所有文件复制到驱动器 B 中的软盘上，若用最原始的 DOS 命令的话，在开始复制文件之前，应先把驱动器 B 中的软盘格式化，执行的 DOS 命令是：

```
C>FORMAT B: /S/V
C>COPY A: *.* B:
```

如果这一组命令需要频繁地使用，那么用户就会觉得第一条命令输入相当麻烦。为解决这一问题，MS-DOS 提供了使用批处理程序的方法。批处理程序允许用户建立 DOS 命令程序，把一组 DOS 命令，按照从键盘输入的顺序放到批处理程序里，执行时 DOS 将从文件里读取命令，不再由用户从键盘上逐条输入。可以为上述两条命令建立一个名为 CPFR.BAT 的批处理程序：

批处理程序 CPFR.BAT

```
REM  This  is  a  batch file to  format  drive  B
REM  It  will  copy  all  information  from
REM  drive  A  to  drive  B
FORAMT B: /S/V
COPY A: *.* B:
```

执行这个批处理程序时，在 DOS 提示符下打入文件名：

```
C>CPFR  <Enter>
```

上例中的 REM 是批处理子命令，用于给用户提供信息。可以看出，批处理程序一旦建立，就可以多次调用执行。它改变了 DOS 命令的执行方式，可以成批自动执行 DOS 命令，这就给用户带来了方便。DOS 规定，批处理程序的扩展名必须是.BAT，除此规定外与一般文件的命名方法相同。

2.1.1　批处理程序——DOS 外部命令

DOS 命令分为内部命令和外部命令两种类型。内部命令是指那些使用频繁且功能较为简单的操作命令，作为命令处理程序 COMMAND.COM 的一部分常驻内存，它们包含在 DOS 内。每次启动时，内部命令即被调入内存。内部命令不占用分配给用户的磁盘空间。外部命令放在磁盘上，使用时由 DOS 从磁盘读入内存，执行完毕后退出并释放所占内存空间。不可能把所有 DOS 命令都设计成为内部命令，这样做会占用太多的内存

空间，影响到应用程序的运行。反之，如果把所有的 DOS 命令都设计成外部命令也是不可取的，因为执行外部命令时需要从盘区读入内存，会降低计算机的运行速度。

任何扩展名为.COM，.EXE 或.BAT 的文件都被 DOS 视为外部命令。这样便于用户开发自己的命令并加到系统中。用户建立的批处理程序，属于外部命令的范畴。批处理程序的文件名不可与 DOS 内部命令名相同。如果建立了名为 DIR.BAT 的批处理程序，并试图在系统提示符下用下面的命令来运行：

C>DIR　<Enter>

DOS 将启动内部命令 DIR，作列目录操作。如果用下面的命令来运行此文件：

C>DIR.BAT　<Enter>

DOS 就会给出出错信息。

每当 DOS 接到一个命令时，首先确定该命令的类型，如果是内部命令就立即执行。如果是外部命令，DOS 在当前目录中寻找，以键入的命令为文件名，扩展名为.COM，.EXE 或.BAT 的文件。如果文件在当前目录中，则执行此文件，否则 DOS 将按着.COM，.EXE，.BAT 的顺序在设置的路径(PATH)中找，如果发现，则执行；如果文件不存在，DOS 将给出错误报告：Bad command or file name。表 2-1 中给出了当路径为 PATH＝C：\；C：\FIRST；C：\SECOND；C：\THIRD 时，DOS 寻找文件的顺序。

表 2-1　DOS 命令执行的先后顺序

内部命令	(DOS 命令)
.COM	(当前目录中.COM 文件)
.EXE	(当前目录中.EXE 文件)
.BAT	(当前目录中批处理程序)
C：\FIRST\.COM	(.COM 文件)
C：\FIRST\.EXE	(.EXE 文件)
C：\FIRST\.BAT	(批处理程序)
C：\SECOND\.COM	(.COM 文件)
C：\SECOND\.EXE	(.EXE 文件)
C：\SECOND\.BAT	(批处理程序)
C：\THIRD\.COM	(.COM 文件)
C：\THIRD\.EXE	(.EXE 文件)
C：\THIRD\.BAT	(批处理程序)

2.1.2　批处理程序的特点

批处理程序可以一次建立，多次执行，DOS 将从批处理程序中获得要执行的命令。用户不必等一个命令执行完毕后再键入下一个命令。用准备执行的一批 DOS 命令为内

容，建立一个批处理程序，下次执行相同的操作时，调用该文件即可。批处理程序可以减少磁盘操作的击键次数，比如要列出驱动器 A 上的所有文件，可输入：

```
DIR A：/P <Enter>
```

需击键九次。以上述操作步骤构成一个批处理程序，在提示符 C>下键入：

```
C>COPY CON：A.BAT <Enter>
DIR A：/P <F6><Enter>
1  Files，copied
```

建立了批处理程序 A.BAT，把准备列目录的磁盘插入驱动器 A，键入：

```
C>A <Enter>
```

只需击键两次，可完成相同的磁盘操作过程。

某些 DOS 命令是由两个英语单词缩写组合成的，如 CHKDSK（CHecK DiSK），FDISK（FixeD dISK）等，这种类型的命令不是正常的英语词汇，较难记忆，如果建立一个名为 CHECK.BAT 的文件：

```
REM  CHECK.BAT
CHKDSK
```

使用 CHKDSK 时，键入 CHECK 即可，这样即便于记忆，又减少了击键次数。当然也可以把 CHKDSK.COM 改名为 CHECK.COM，利用 CHECK.COM 来完成同样的检查磁盘任务。这样做虽然便于记忆，然而别人却无法执行 CHKDSK.COM 命令了。最好的方法仍然是建立一个批处理程序，用户自己用起来方便，也不会影响别人进行正常的磁盘操作。

某些 DOS 命令含有多个参数，在使用之前，不免要查阅 DOS 手册。如果按上面讲的办法，建立含有这类命令的批处理程序，并给文件一个有意义的文件名，则免去了记忆那些冗长格式的麻烦。利用 BACKUP.COM 命令，定期把硬盘上的文件备份到软盘上是十分重要的。用下面的批处理程序完成这一任务，不仅减少击键次数，而且不必记忆 BACKUP.COM 命令的格式。需要制作备份时，键入 BACKUP1 即可批。

批处理程序 BACKUP1.BAT

REM BACKUP1.BAT	文件名注释
BACKUP C：\A：/S/M/A	执行 BACKUP 命令。由于该命令行没有指明 BACKUP.COM 所在的驱动器，也未指明查找的路径，批处理程序也没有改变目录而进入 BACKUP.COM 所在的目录，因此在路径中必须指明 BACKUP.COM 所在的目录。

使用批处理程序进行磁盘操作的另一个优点是可增加安全性，避免由于误操作而丢失文件。如 FORMAT、DEL 及 ERASE 等命令，如果使用不当，都会造成数据或文件的丢失。在系统提示符 C>下，准备格式化驱动器 A 里的软盘，若由于疏忽，键入的命令是 FORMAT 而不是 FORAMT A：，会造成全部硬盘文件的丢失；同样，若准备启动命令 ERASE 删除所有的.BAK 文件，却意外的键入了 ERASE *.BAT，必然会删除所有的

批处理程序，这就造成不可估量的损失。针对这类危险性很大的 DOS 命令，应建立相应的批处理程序来保护文件，防止上述情况发生。

批处理程序 FORMAT. BAT 可保护硬盘不被意外地格式化。在建立这个文件之前，应先把 FORMAT. COM 更名为 XYZ. COM。

批处理程序 FORMAT. BAT

ECHO OFF	关闭回显，文件中执行的命令将不在屏幕上显示。
REM FORMAT. BAT	文件名注释。
XYZ A：/V	格式化软盘命令，PATH 语句中必须包括 XYZ. COM 所在的子目录。

这样即使在系统提示符 C>下键入：

FORMAT

也不会把硬盘格式化。文件里的 REM 是批处理的注释子命令，将在后面详细介绍。

批处理程序 KILL. BAT 能够有效地防止由于使用 ERASE 命令的不慎而删除有用的文件。

批处理程序 KILL. BAT

ECHO OFF	关闭回显，将不显示执行的命令。
REM KILL. BAT	
DIR %1/W	宽行显示当前目录中的所有文件。
ECHO Hit Control Break to STOP the erase of	
	告诉用户如何停止批处理程序的执行。
ECHO these files	
PAUSE	暂停批处理程序的执行给用户一机会，如果输入了不准备删除的文件，可停止批处理程序的执行。
ERASE %1	按用户的意图删除文件。

如果键入：

C>KILL *. BAK

文件 KILL. BAT 中的%1 被 *. BAK 替换。DIR 命令列出的将是当前目录里的所有. BAK 文件。ERASE %1 将成为 ERASE *. BAK。KILL. BAT 将删除所有. BAK 文件。

2.2 建立批处理程序的方法

可以用 DOS 提供 COPY 命令，行编辑程序 EDLIN，或 DOS 5.0 版本中的 EDIT 及英文文字处理软件来建立批处理程序。

2.2.1 COPY 命令

建立批处理程序，最简单常用的方法就是使用 COPY 命令，例如建一个格式化新磁

盘，并检查磁盘的批处理程序，操作步骤如下：

```
C>COPY  CON  CHECKNEW.BAT  <ENTER>
```

此语句通知 DOS，把来自标准输入设备——键盘的信息复制到缺省驱动器的 CHECKNEW.BAT 文件中。应注意当按完 ENTER 后，并没有出现 DOS 提示符。现在再输入：

```
REM  THIS IS  A  FILE  TO  FORMAT  AND
REM  CHECK  NEW  DISKS
REM  THE  FILE  NAME  IS  CHECKNEW.BAT
PAUSE  INSERT  A NEW  DISK  IN  DRIVE B:
FORMAT  B: /V
CHKDSK  B: <F6><Enter>
```

注意每输完一行后应按 Enter 键，输完最后一行按 F6，然后再按回车键，F6 键的作用是把建立的批处理程序存入磁盘。全部输完后，信息“1 File(s) copied”将出现在屏幕上。

COPY 命令常用到如下几个逻辑设备：

CON：若用作输入设备，CON 即为键盘；用作输出设备时，CON 为屏幕。反之是没有意义的，因为键盘不可能作为输出设备。

PRN 或 LPT1：并行打印机。

NUL：DOS 黑洞或称不存在的设备。只有少数情况下能用到 NUL，例如在执行 I/O 双向输出或管道操作时。所有输出到 NUL 设备的文件全部消失。COPY 命令可在上述逻辑设备之间复制文件。

启动命令：

```
C>COPY  SHOWCON.DOC CON <Enter>
```

DOS 将文件 SHOWCON.DOC 的内容显示在屏幕上。

```
C>COPY SHOWCON.DOC CON <Enter>
REM  SHOWCON.DOC
This is an  example  of  an  ASCII  file like the files
created by some word processors.
        1  File(s)  copied

C>
```

有些逻辑设备之间的 COPY 操作无实际意义，例如从 NUL 设备或打印机向 CON 设备复制文件。

用 COPY CON 可建立任意长度的批处理程序，但是用 COPY CON 建立了一个批处理程序后，不能用 COPY 命令进行修改或编辑。因此 COPY CON 可用来建立较短的批处理程序，用该命令建立较长的文件或修改已存在的批处理程序是不方便的。

2.2.2 行编辑程序—EDLIN

行编辑程序 EDLIN 是 DOS 为用户提供的一个实用程序，可以用来建立、修改和显示文本文件。批处理程序是文本文件，所以可以用行编辑程序来建立。EDLIN 是行编辑程序，也就是说，它以行为单位进行文件编辑工作。这里行的意义是以 Enter 键为结尾的逻辑行，每行最多不得超过 253 个字符。

在当前子目录下启动行编辑程序，文件 EDLIN.COM 如不在当前子目录里，则在 PATH 语句中必须包括 EDLIN.COM 所在的目录。用 EDLIN 编辑一个文件，只需在 DOS 提示符下键入：

```
C>EDLIN [filename.extension]
```

如果命令中指定的文件不存在，则建立新文件并显示：

```
New  file
* —
```

如果指定的文件存在，则把指定的文件调入内存并在屏幕显示：

```
End  of  input  file
* -
```

这里的星号“*”是 EDLIN 的提示符，此时可以输入编辑命令。

当一个文件的编辑工作结束时，可以用 E(End)命令结束行编辑程序的运行，并把经过编辑的文件存入磁盘，返回 DOS，屏幕上出现 DOS 提示符。如果编辑的是一个存在的文件，那么 E 命令把编辑前后的文件都存盘，编辑前的文件在存盘时被重新命名为.BAK 文件，同时删除原来的.BAK 文件。使用 Q(Quit) 可退出行编辑程序，并且不把修改过的文件存盘，也不建立.BAK 文件。

和 E,Q 命令一样，行编辑程序的其它命令也是单个字母。E,Q 命令所影响的是整个文件，执行这两个命令时，无需附带任何参数，执行其它的命令时，往往需要同时输入命令参数，除 E,Q 两命令外，其余各命令用得最多的参数是行号。行编辑程序中，对于作为命令参数的行号应注意下面五种情况：

(1) 如果省略了行号，EDLIN 就从当前行开始工作。

例如删除命令[line][,line]D， 如果把两个行号省略，则删除当前行。

(2) 园点“·”总是使 EDLIN 从当前行开始工作。

(3) 如果指定的行号太大，EDLIN 将使用内存中或文件里的最后一行。

(4) 符号“#”使 EDLIN 使用内存中或文件里的最后一行。

(5) 输入+10 后，表示当前行之后的 10 行，同样—10 表示当前行之前的 10 行。

启动 EDLIN 后，并不能自动进入插入状态，必须用 I 命令进入插入状态，格式为[line]I，将从指定的行号开始插入，如果不指定行号，则在当前行插入。在插入状态下，EDLIN 不接受编辑命令。用 Ctrl-Break 或 Ctrl-C，可停止插入操作，返回 EDLIN 提示符 *。如果在按 Enter 键之前用了 Ctrl-Break 或 Ctrl-C 键，将取消当前正在编辑的行。

显示行命令 L(List)，显示指定范围内的各行，格式为[Line][,Line]L，其中第一个参数是显示的第一行的行号，第二个参数是显示的最后一行的行号，如果省略两个参数，将显示以当前行为中心的 23 行。如果使用命令 line,L 或 line L，将从规定的行号开始，共显示 23 行。

删除行命令 D(Delete)，删除指定范围内的各行，格式为[line][,line]D，其中第一个参数是在指定范围内要删除的第一行，第 2 个参数是准备删除的最后一行。省略两个参数只输入 D，将删除当前行。

编辑行命令，用来编辑一行文字。格式为[line]。执行编辑行命令，EELIN 将显示行号及其正文，并在该行下面显示同一个行号，在此行号后面可对该行进入编辑。

查找字符串命令 S(Search)，在规定的范围内查找指定的字符串。格式为[line][,line]S text，其中两个参数规定了查找字符串 text 的范围。如果第一个参数省略，则从当前行的下一行开始查找；如果第二个参数省略，则一直查找到文件的最后一行或者文件在内存里的最后一行。如果两个参数都省略，则查找范围从当前行的下一行开始到文件的最后一行或文件在内存里的最后一行为止。如果省略 text，EDLIN 将查找上次 S 命令指定的字符串。S(Search)命令查找文件中完全相同的字符，即查找文件中与 S 命令指定查找的字符串必须字母、顺序及大小写完全相同。

替换字符串命令 R(Replace)，与查找字符串命令类似，在规定的范围内，用新字符串取代旧字符串，格式为[line] [,line]R old new。第一个参数为替换命令开始的行号，如省略，R(Replace)命令使用当前行。第二个参数为结束替换命令执行的行号，如果省略，R 命令使用内存里的最后一行。“old”是旧字符串，“new” 为新字符串，“^Z”是新旧字符串之间的分隔符。

被编辑的文件太长，内存中容纳不下时，可以用写命令 W(write) 把文件前部数行写回磁盘。W 命令的格式为[line]W，其中的参数是写回磁盘的行数，如果省略，W 命令把占四分之一内存的行数写入磁盘。写盘后，文件在内存中的剩余部分将重编行号，剩余部分的第一行新编号为第 1 行。

A(Append) 命令可以把磁盘上的文件未编辑部分装入内存，格式为[line]A，其中参数是调入内存的行数，如果不指定装入内存的行数，就到装满可用内存的 75%为止。A 命令把文件的最后一行装入内存后，EDLIN 显示 End of file input 信息。

从 DOS2.0 开始，EDLIN 增加了四个命令，它们是复制行命令 C(Copy)，移动行命令 M(Move)，传输行命令 T(Transfer)及页命令 P(Page)。

复制行命令 C　把规定范围内的各行复制到由第三个参数指定行的前面，复制的次数由第四个参数指定。C 命令的格式为[line],[line],[line][,count]C；例如命令 1,3,7,2,C 把 1 至 3 行在第 7 行前面复制两次。

移动行命令 M　把规定范围内各行移到第三个参数指定行数前面，命令格式为[line],[line],line M；例如命令 1,3, 7 M 把 1 至 3 行移到第 7 行的前面。

传输命令 T　把指定文件的内容合并到当前文件里。命令格式为 [line][filename,]该命令把 filename 指定的文件，插入当前正在编辑的文件里，插入位置由参数 line 指定。

页命令 P 和显示行命令几乎相同，与 L 命令的唯一区别是它改变当前行。

由于行编辑程序 EDLIN 是 DOS 的组成部分，因此凡是以 DOS 作为操作系统的用户，都有 EDLIN。所以 EDLIN 使用极为广泛，DOS 用户不一定人人都有建立批处理程序的有力工具——英文文字处理软件，例如 WS（Word Star），WP（Word Perfect）等。EDLIN 的另外两个优点是它只有 8K 字节长，建立的文件是 ASCII 码文件。

EDLIN 是命令驱动程序，命令多，难以记忆，不具有选择命令的菜单，而且是个行编辑程序，光标只能在当前行移动，按了回车键后不可能返回原行重新编辑，这些都是行编辑程序的缺点。

2.2.3 全屏幕编辑程序—EDIT

低于 5.0 的所有 DOS 版本提供的编辑文本文件的程序只有 EDLIN，EDLIN 只能编辑简单短小的文本文件，DOS 5.0 及以上版本中除了 EDLIN 外，又增加了功能很强的全屏幕文本文件编辑程序——EDIT。EDIT 是建立、编辑、修改批处理程序的强有力工具。

全屏幕编辑程序 EDIT 是介于窗口程序与 QuickBasic 编辑程序之间的产物，是一个利用下拉式菜单驱动的全屏幕编辑程序，操作与 Quick Basic 的 Editor 极为相似。EDIT 编辑修改文件时光标可移到屏幕的任何位置，而不是像 EDLIN 那样，只能在当前行移动光标。

EDIT 编辑程序有 File，Edit，Search 及 Options 四个菜单 。每个菜单设有若干个命令，按下 ALT 键，4 个菜单中都产生一个加亮字母，按下准备启动菜单对应的加亮字母，即可调出菜单。File 菜单有 6 条命令：

New　建立新文件，用户可键入写好的批处理程序进行编辑。

Open　把驻留盘区的文件调入内存，进行修改、编辑或打印。

Save　把修改后的文件以原来的文件名存盘，驻留盘区的同名文件将被改写。

Save as　主要用来把新建立的文件存盘，或把调入内存的文件修改后以新的文件名存盘。也用来把修改后的文件以相同的文件名存入不同的磁盘。

Print　打印整个文件或文字段，打印文字段前，必须用 shift-光标键把准备打印的文字段加亮。

Exit　退出 Edit 编辑程序，退出前提示用户是否把文件存盘。

Edit　菜单的四条命令如下：

Cut　把文件中加亮的文字段移至编辑缓冲器，文字段从屏幕上消失。

Copy　把文件中加亮的文字段复制至编辑缓冲器，文字段仍然存在屏幕上。

Paste　把编辑缓冲器中的内容复制到屏幕上的光标位置，编辑缓冲器中的文字保持不变。

Clear　删除屏幕上加亮的文字段。

Search　菜单有三条命令：

Find　确定给定文字段在文件中的位置。

Repeat Last Find　确定上次查找的文字段在文中的下一个位置，

Change　替换文字段命令。

Options　菜单有两条命令：

Display　　设置屏幕前景、背景颜色。

Help path　　指明帮助文件 Edit.hlp 所在目录。

利用全屏幕编辑程序 EDIT 替代行编辑程序 EDLIN，无疑会提高工作效率。

另外，大多数英文文字处理软件，如 Word Star，Word Perfect 等，都可以建立、编辑批处理程序。

第 3 章　简单的批处理语句

DOS 为批处理程序配置了一些专用命令，以加强批处理程序的功能，这些专用命令全部是 DOS 内部命令。从本章开始，将陆续介绍。

3.1　REM(注释)子命令

REM 是 REMARK(注释)一词的缩写，注释子命令的作用是在批处理程序内加入注解，以使自己和别人都明白批处理程序的功能，例如批处理程序的名字，某些程序行的作用及功能等。REM 子命令是非执行命令，即在批处理程序的运行过程中，遇到该命令时，只是 REM 后面的文字按原样全部显示出来，对批处理程序内其它命令或语句的执行无任何影响。REM 子命令在文件中仅起注释和备忘的作用。REM 子命令的格式为：

REM (字符串)

整个命令行的长度不得超过 127 个字符。先看一个批处理程序：

批处理程序 YESDOC.BAT

```
:START
ERASE  %1.BAK
COPY %1.DOC B:
COPY  C:\WORD\NORMAL\%1.STY  B:
ERASE  C:\WORD\NORMAL\%1.BAK
COPY  \WORD\%1.CMP  B:
SHIFT
IF (%1)==( ) GOTO END
GOTO  START
:END
```

在上面的批处理程序里，有些命令还没有介绍，但是这并不重要。关键是由于这个文件内没有 REM 语句，对于初学者，很难理解这个文件的功能。下面同样是一个批处理程序，所有的工作命令与上面的文件完全相同，唯一的区别是增加了 REM 语句，对该文件进行了详细的注释和说明。虽然文件内包括了一些本书后面再介绍的命令，但是即使是初学者也能理解每个命令行的作用及功能。

批处理程序 YESDOC.BAT

```
REM YESDOC.BAT
REM Batch file to copy new files to floppy disk
REM Name top of loop
```

```
:START
REM Erase the backup file
ERASE %1.BAK
REM Copy the document file to floppy disk
COPY %1.DOC B:
REM Copy the associated style sheet to floppy disk
COPY C:\WORD\NORMAL\%1.STY B:
REM Erase backup copy of style sheet
ERASE C:\WORD\NORMAL\%1.BAK
REM Copy document dictionary to floppy disk
COPY \WORD\%1.CMP B:
REM Get next name
SHIFT
REM Test for next name existing and go to
REM bottom if does not exist
IF (%1)==( ) GOTO END
REM Go to top since there is another file
GOTO START
:END
REM Finished copy
```

如果子命令回显处于关闭(OFF)状态,REM 子命令行将不显示在屏幕上。为了便于阅读,可以使用后面没有任何信息的 REM 子命令,这样在批处理程序内产生一个空白行。

3.2 ECHO(显示命令)子命令

在批处理程序执行时,用批处理子命令 ECHO 将允许或禁止 DOS 命令在屏幕上的显示。ECHO 不影响由于 DOS 命令的执行产生显示。ECHO 子命令以下面四种形式出现在批处理程序里:

```
ECHO  ON
ECHO  OFF
ECHO  (字符串)
ECHO
```

当 ECHO 处于打开(ON)状态时,DOS 将把所有执行的批处理程序内的命令显示在屏幕上。在开机或者重新启动系统时,ECHO 处于 ON 状态。大多数 DOS 版本,在启动一个批处理程序时,如果该文件的起始行是 ECHO OFF,在屏幕上仍然会看到 ECHO OFF 的字样。从 DOS3.3 开始,在批处理程序命令前面加上字符@,这样,即使回显处于 ON 状态,这个命令也不显示。因此很多用户不准备显示执行的 DOS 命令时,喜欢把@ECHO OFF 作为批处理程序的第一行,以避免 ECHO OFF 在屏幕上的显示。

ECHO OFF 只是不显示所执行的DOS 命令，对于DOS 命令的执行而产生的提示信息，仍然显示。比如 ECHO 处于 OFF 状态，并在批处理程序内执行 COPY 命令时，1 File(s)Copicd 的提示信息照样显示。

ECHO 本身，即不带任何参数的 ECHO 子命令，则显示 ECHO 的当前状态：ON 或 OFF，例如：

```
C>ECHO <Enter>

ECHO  is  on
```

不管 ECHO 是处于 ON 还是 OFF 状态，ECHO message 总是把信息显示在屏幕上。这样，即使 ECHO 已关闭时，仍然显示指定的信息。当 ECHO 是 ON 状态时，命令行 ECHO message 把信息实际显示两次，DOS 显示批处理命令时，则显示一次；当 DOS 执行完命令时，又显示一次。如果 ECHO 是 OFF 状态，那么在 DOS 执行 ECHO 命令时，显示一次 ECHO message 中的信息。批处理程序 ECHO1.BAT 说明了这种情况。

批处理程序 ECHO1.BAT

```
REM ECHO1.BAT                          文件名注释
ECHO                                   显示 ECHO 当前状态
ECHO You should see a message saying ECHO is on
                                       显示信息
ECHO OFF                               关闭回显
ECHO                                   显示 ECHO 当前状态
ECHO You should not have seen the ECHO command
                                       显示信息
ECHO However, you should have seen the ECHO is off message
                                       显示信息
ECHO Unlike the first message, you should see these
                                       显示信息
ECHO messages only once
                                       显示信息
ECHO ON                                打开回显
ECHO ECHO is now back ON, and you should see this twice
                                       显示信息
```

执行 ECHO1 后屏幕显示为：

```
D>echo1

D>REM ECHO1.BAT

D>ECHO
ECHO is on
```

D>ECHO You should see a message saying ECHO is on
You should see a message saying ECHO is on

D>ECHO OFF
ECHO is off
You should not have seen the ECHO command
However, you should have seen the ECHO is off message
Unlike the first message, you should see these
messages only once

D>ECHO ECHO is now back ON, and you should see this twice
ECHO is now back ON, and you should see this twice
D>

利用 ECHO 子命令与 ASCII 字符，可得到特殊的屏幕显示效果。文件 FANCYECHO. BAT，使信息位于长方形的中心。

批处理程序 FANCYECHO. BAT

```
ECHO OFF
REM FANCYECHO. BAT
ECHO
ECHO
ECHO
ECHO
ECHO If your word processor will not handle these characters, then...
ECHO ----------
ECHO |                    |
ECHO |  YOUR MESSAGE HERE |
ECHO |                    |
ECHO ----------
```

```
┌──────────────────┐
│ YOUR MESSAGE HERE │
└──────────────────┘
```

下面是用于显示图形的扩展 ASCII 字符，输入时先按下 Alt 键，再从右侧小键盘键入数字，敲完一个字符，松开 Alt 键，该字符出现在屏幕上。

文件 FANCYECHO. BAT 的执行结果为：

C：>fancyecho

C：>ECHO OFF

```
+-----------------------+
|                       |
|   YOUR MESSAGE HERE   |
|                       |
+-----------------------+
```

If your word processor will not handle these characters, then...

```
-----------------------------------------------------
!                                                   !
:              YOUR MESSAGE HERE                    !
!                                                   !
-----------------------------------------------------
C:>
```

ECHO 命令有一个鲜为人知的功能：如果在 DOS 提示符下输入 ECHO OFF，DOS 提示符消失。此时如不重新使用 ECHO ON 命令，那么对于所有的批处理程序，ECHO 的缺省值都是 OFF；而不是如前所述那样，当一个批处理程序运行结束后，无论文件内是否使用了 ECHO OFF 命令，ECHO 都恢复为 ON 状态。

在 DOS3.0 之前的 DOS 版本中，可使用园点“·”取代 ECHO，所以

```
·Don't  use  the  dot  now
```

与

```
ECHO Don't use the dot now
```

具有相同的作用，都把信息 Don't use the dot now 显示在屏幕上。并不是所有的 DOS 版本都支持园点去代替 ECHO。在批处理程序中显示信息时应避免使用园点。

一些较早的 DOS 版本不按用户的意图在 ECHO 子命令的信息部分显示空格。无论用户在 ECHO 与 message 之间设置几个空格，在这些 DOS 版本下只有一个有效。用不同的 DOS 版本运行下面的文件，运行结果的屏幕显示清楚地说明了这一点。

批处理程序 SPACE1.BAT

```
ECHO OFF                关闭回显
REM SPACE1.BAT          文件名注释
VER                     显示 DOS 版本
ECHO ONE SPACE          ECHO 与信息之间显示一个空格
ECHO  TWO SPACES        ECHO 与信息之间显示二个空格
ECHO  THREE SPACES      ECHO 与信息之间显示三个空格
ECHO   FOUR SPACES      ECHO 与信息之间显示四个空格
```

两种 DOS 版本的运行结果为：

```
C>SPACE1

C>ECHO OFF

COMPAQ Personal Computer DOS Version 2.11
ONE SPACE
TWO SPACES
THREE SPACES
FOUR SPACES

C>

C>SPACE1

C>ECHO OFF

COMPAQ Personal Computer DOS Version 3.20

ONE SPACE
TWO SPACES
  THREE SPACES
   FOUR SPACES

C>
```

在空格前面加上一个前导字符“!”就可以解决这一问题。用任何 DOS 版本运行文件 SPACE2.BAT,结果都是一样的。

批处理程序 SPACE2.BAT

```
ECHO OFF
REM SPACE2.BAT
VER
ECHO ! ONE SPACE
ECHO !    TWO SPACES
ECHO !     THREE SPACES
ECHO !      FOUR SPACES
```

用 DOS2.11 运行 SPACE2.BAT 的结果为:

```
C>SPACE2

C>ECHO OFF

COMPAQ Personal Computer DOS Version 2.11
```

```
! ONE SPACE
!   TWO SPACES
!    THREE SPACES
!     FOUR SPACES

C>
```

Ctrl 键与一些字母键的组合，可产生其它的效果。比如 ^ G，可鸣铃，来报警或通知已完成批处理程序的执行。文件 BELL.BAT 鸣铃三次。

批处理程序 BELL.BAT

```
ECHO  OFF
REM  BELL.BAT
ECHO  ^G^G^G
```

有些用户为了增加屏幕信息的可读性，想在两个 ECHO message 行之间生产一个空行。如果所用的是 3.0 以下的 DOS 版本，在批处理程序里的 ECHO 后面跟两个空格，即可产生一个空行：

批处理程序 LINE1.BAT

```
ECHO OFF
REM LINE1.BAT
ECHO The next line is intended to be blank 显示信息
ECHO                                       在两行信息间
                                           加一空行
ECHO The above line should be blank        显示信息
```

用 3.0 或高于 3.0 的 DOS 版本运行 LINE1.BAT 的结果如下：

```
C>LINE1

C>ECHO OFF
The next line is intended to be blank
ECHO is off
The above line should be blank

C>
```

可以看出，文件中第四行的 ECHO 没有产生空行，却显示出当前 ECHO 的状态，说明 3.0 以上的 DOS 版本不支持这种作法。

ECHO 的信息部分如果含有形式参数(%0—%9)，或者环境变量(例如%PATH%)，那么用实际参数替换时，可能使整个 ECHO 语句的长度超过 127 个字符。环境变量 PATH 所设置的路径长可达 122 个字符("PATH="需 5 个字符)。实际参数的字符太多也会使 ECHO 子命令超长。

对于 ECHO 子命令行超长的问题，对于同样的 DOS 版本，不同的计算机的反应有时不同。有的计算机发生“死机”现象；另外一些计算机只认为前 127 个字符有效，超长部分不予考虑。因此在批处理程序内用 ECHO 子命令显示可替换参数或环境变量的值时应特别注意。在必须这样做时，让一个 ECHO 语句的信息部分只包括一个形式参数或环境变量，这样用实际参数替换时，ECHO 命令行一般不会超过 127 个字符。

不能在 ECHO 语句内使用<，>，>>，以及 |。它们是标准输入输出重定向符和管道操作符，如果 ECHO 语句内使用了它们，DOS 将不会把它们当作字符来显示，而是进行信息输入输出的重新定向，或者进行管道操作，从而产生预料不到的后果。

ECHO 语句中如果含有%，DOS 会把它当作变量。如果需要在屏幕上显示%，应在 ECHO 语句中使用%%而不是%。

ECHO 语句中的信息部分如果以 ON＋空格或 OFF ＋空格开头，DOS 就会认为它们是 ECHO ON 或 ECHO OFF 命令，所以应避免这样做。

3.3 批处理程序的形式参数与实际参数

使用计算机高级语言编写源程序时，经常使用变量。在程序中设置一些变量，可根据需要，在每次运行时赋给变量不同的值，使程序简化、灵活，给使用者带来方便。

批处理程序提供了类似变量的形式参数，实际参数是形式参数的值。批处理程序共有 10 个形式参数，在建立批处理程序时，可以根据需要安排一些形式参数，执行文件时，由命令行上的实际参数取代文件内的形式参数。文件 SHOWPAR1. BAT 设置了形式参数%1。

批处理程序 SHOWPAR1. BAT

```
REM   SHOWPAR1. BAT
DIR   %1   执行时应输入实际参数，取代形式参数%1。
ECHO %1   显示输入的实际参数，如果没有输入实际参数，将显示“ECHO is on”。
REM   %1   说明实际参数已取代了形式参数。
```

在当前目录 C：\BAT 下键入：

```
C：\BAT>SHOWPAR1   *. BAK (Enter)
```

执行结果为：

```
C>REM   SHOWPAR1.BAT

C>DIR *. BAK
Volume  in  drive  c  is  Daqing
Directory  of  C：\BAT
CHAPTER2  BAK     10996  8-15-91  910P
CHAPTER3  BAK     3712   8-17-91  938P
CHAPTER4  BAK     38016  8-19-91  226P
INTR0     BAK     7168   8-15-91  859P
```

```
CHAPTER5  BAK     19968  8-19-91  903a

        5  File(s)  2158582  bytes  free

C:BAT\>ECHO *.BAK

*.BAK
C:BAT\>REM *.BAK
```

上例中，执行该文件时，键入文件名的同时键入了实际参数*.BAK，执行文件时，用实际参数*.BAK 取代了形式参数%1。对于*.DOC 文件，要执行与 *.BAK 文件同样的操作，执行文件 SHOWPAR1.BAT 时用*.DOC 作为实际参数即可。执行文件 SHOWPAR1.BAT 时，如果只键入文件名，不输入实际参数：

```
C>SHOWPAR1
```

那么执行 DIR 命令时，将列出当前目录中的所有文件，ECHO 将显示“ECHO is on”。可以看出，含有形式参数的批处理程序，执行时如果不输入实际参数，DOS 将认为文件中的形式参数不存在。

在执行批处理程序的命令行上输入实际参数时，参数之间应用空格或逗号隔开：

```
Batch  Para1  Para2  Para3 Para4
```

上面的命令行包括五个实际参数，用来替换形式参数%0—%4。DOS 规定%0 对应的实际参数总是批处理程序名。

DOS 命令也可以作为实际参数，例如文件 SHOWREPL.BAT 中，执行时用 DIR 命令取代第一个形式参数%1：

批处理程序 SHOWREPL.BAT

```
REM  SHOWREPL.BAT
%1  %2  %3  %4
```

执行时键入

```
C>SHOWREPL  DIR  *.BAK
```

文件中虽然有 4 个形式参数，执行时只输入了两个。%3，%4 没有实际参数取代，DOS 将认为这两个形式参数不存在。

某些 DOS 版本不允许在 AUTOEXEC.BAT 文件内使用形式参数，如果使用了可能会发生死机现象。每次机器启动时，命令解释程序 COMMAND.COM 自动运行 AUTOEXEC.BAT 文件，因此在文件 AUTOEXEC.BAT 内使用形式参数就没有必要。

3.4 SHIFT(左移实际参数)子命令

在批处理程序中可以设置 10 个(%0—%9)形式参数，这就限制了实际参数不能超过

10 个。SHIFT 子命令的使用，允许在执行批处理程序的命令行上输入 10 个以上的实际参数。批处理程序中每执行一次 SHIFT 命令，命令行上的实际参数都左移一个位置，即用对应于%1 的实际参数取代%0，对应于%2 的实际参数取代%1，以此类推。例如：

执行 SHIFT 命令之前实际参数与形式参数的对应关系：

实际参数：	A	B	C	D	E	F	G	H	I	J	K	L
	/	/	/	/	/	/	/	/	/	/		
形式参数：	%0	%1	%2	%3	%4	%5	%6	%7	%8	%9		

执行一次 SHIFT 命令之后，实际参数与形式参数的对应关系：

实际参数：	A	B	C	D	E	F	G	H	I	J	K	L
	/	/	/	/	/	/	/	/	/	/	/	
形式参数：消失	%0	%1	%2	%3	%4	%5	%6	%7	%8	%9		

执行两次 SHIFT 命令后，实际参数与形式参数的对应关系：

实际参数：	B	C	D	E	F	G	H	I	J	K	L
	/	/	/	/	/	/	/	/	/	/	/
形式参数：消失	%0	%1	%2	%3	%4	%5	%6	%7	%8	%9	

下面的批处理程序清楚地说明了 SHIFT 命令的功能和使用方法：

批处理程序 REPLACE1. BAT

```
REM REPLACE1. BAT
ECHO OFF
ECHO 1=%1 2=%2 3=%3 4=%4 5=%5 6=%6 7=%7 8=%8 9=%9
SHIFT
ECHO 1=%1 2=%2 3=%3 4=%4 5=%5 6=%6 7=%7 8=%8 9=%9
SHIFT
ECHO 1=%1 2=%2 3=%3 4=%4 5=%5 6=%6 7=%7 8=%8 9=%9
SHIFT
ECHO 1=%1 2=%2 3=%3 4=%4 5=%5 6=%6 7=%7 8=%8 9=%9
SHIFT
ECHO 1=%1 2=%2 3=%3 4=%4 5=%5 6=%6 7=%7 8=%8 9=%9
SHIFT
```

执行 REPLACE1. BAT，键入：

```
C>REPLACE1 FIRST SECOND THIRD 4TH FIFTH SIXTH SEVENTH EIGHTH NINTH
TENTH
11 12 13 14
```

执行后，屏幕显示的结果为：

```
C>REM REPLACE1. BAT

C>ECHO OFF
```

```
1=FIRST 2=SECOND 3=THIRD 4=4TH 5=FIFTH 6=SIXTH 7=SEVENTH 8=EIGHTH
9=NINTH
1=SECOND 2=THIRD 3=4TH 4=FIFTH 5=SIXTH 6=SEVENTH 7=EIGHTH 8=NINTH
9=TENTH
1=THIRD 2=4TH 3=FIFTH 4=SIXTH 5=SEVENTH 6=EIGHTH 7=NINTH 8=TENTH
9=11
1=4TH 2=FIFTH 3=SIXTH 4=SEVENTH 5=EIGHTH 6=NINTH 7=TENTH 8=11 9=12
1=FIFTH 2=SIXTH 3=SEVENTH 4=EIGHTH 5=NINTH 6=TENTH 7=11 8=12 9=13

C>
```

上面执行 REPLACE1. BAT 的命令行里，输入了 14 个实际参数，并且占据了两行，这是因为命令行的长度不得超过 127 个字符，同时也注意到实际参数(14)没有使用。

批处理程序 TOA1. BAT 的作用是从 C 盘不同的子目录中，把扩展名为. CMP，. DOC及. STY 的文件复制到软盘上，同时删除扩展名为. BAK 的文件。

批处理程序 TOA1. BAT

REM TOA1. BAT	文件名注释。文件中没有使用 ECHO OFF 子命令，所有的命 令将显示在屏幕上。
ERASE %1. BAK	删除. BAK 文件。如果文件不存在，将出现“File not found”提示信息。
COPY %1. DOC A：	把. DOC 文件复制到 A 盘。. DOC 文件的文件名是执行 TOA1. BAT 的命令行的第一个实际参数。 如果文件不存在，可能出现“File notnot found ”的提示信息。
COPY C：\WORD\%1. CMP A：	把. CMP 文件复制到 A 盘。. CMP 文件的全名是执行 TOA1. BAT 命令行上的第一个实际参数。如果文件不存在，可能出现“Filenot found”的提示信息。
COPY C：\WORD\NORMAL\%1. STY A：	把. STY 文件复制到 A 盘。. STY 文件的文件名是执行 TOA1. BAT 命令行上的第一个实际参数。 如果文件不存在，可能出现“File not found” 的提示信息。
ERASE %2. BAK COPY %2. DOC A： COPY C：\WORD\%2. CMP A： COPY C：\WORD\NORMAL\%2. STY A： ERASE %3. BAK COPY %3. DOC A： COPY C：\WORD\%3. CMP A： COPY C：\WORD\NORMAL\%3. STY A：	对余下的形式参数(%2—%9)， 执行与形式参数%1 完全相同 的操作。

```
ERASE %4.BAK
COPY %4.DOC A:
COPY C:\WORD\%4.CMP A:
COPY C:\WORD\NORMAL\%4.STY A:
ERASE %5.BAK
COPY %5.DOC A:
COPY C:\WORD\%5.CMP A:
COPY C:\WORD\NORMAL\%5.STY A:
ERASE %6.BAK
COPY %6.DOC A:
COPY C:\WORD\%6.CMP A:
COPY C:\WORD\NORMAL\%6.STY A:
ERASE %7.BAK
COPY %7.DOC A:
COPY C:\WORD\%7.CMP A:
COPY C:\WORD\NORMAL\%7.STY A:
ERASE %8.BAK
COPY %8.DOC A:
COPY C:\WORD\%8.CMP A:
COPY C:\WORD\NORMAL\%8.STY A:
ERASE %9.BAK
COPY %9.DOC A:
COPY C:\WORD\%9.CMP A:
COPY C:\WORD\NORMAL\%9.STY A:
```

执行该文件时，应输入：

```
C>TOA1  file1  file2  file3
```

批处理程序 TOA1.BAT 存在几个缺点：首先，即使输入的实际参数小于 9 个，该文件也总是企图复制 9 次；其次是它最多只能复制 9 个文件；再就是如果要准备复制的每一个文件所属的子目录都改变了，那么必须需修改文件路径达 9 次之多；另外，该文件显得非常冗长。批处理程序 TOA2.BAT 的作用与 TOA1.BAT 完全相同，但是由于使用了 SHIFT 命令及其他一些批处理程序子命令(将在后面介绍)，因而克服了 TOA1.BAT 中存在的弱点。TOA2. BAT 的基本结构是循环，文件中只用了一个形式参数，处理的文件却可以多于 9 个。

批处理程序 TOA2.BAT

```
REM TOA1.BAT
:TOP                                    循环体起始的标号行。
    ERASE %1.BAK
    COPY %1.DOC A:
    COPY C:\WORD\%1.CMP A:
    COPY C:\WORD\NORMAL\%1.STY A:
```

```
SHIFT                          把实际参数左移一个位置。
IF (%1)==() GOTO END           如果实际参数已用完，
                               转移至标号 END 后，退出批处理程序。
GOTO TOP                       转移到循环体的起始标号。
END                            标号行 END。
```

使用 SHIFT 命令，尽管在批处理程序内只设置一个形式参数，却能把执行批处理程序命令行上的所有的实际参数逐个传递给它，这就给修改批处理程序时带来了方便，批处理程序本身也可编写得短小灵活，而且功能更强。

3.5 批处理程序内部环境变量的使用

利用 DOS 提供的机制，可设置一系列的环境变量，这样批处程序及其它应用程序可以调用这些变量，在批处理程序之间以及批处理程序和其他文件之间传递信息。通常的做法是把一些经常用到的变量设置为环境变量。

3.5.1 DOS 环境

DOS 环境是 DOS 用来存储特殊信息的一块内存空间。DOS 环境所占内存空间大小的缺省值为 160 个字节，即能储存 160 个字节的信息量。在 DOS 环境区中通常有以下几项内容：

(1) COMMAND.COM 所在的目录；

(2) 设置的路径(PATH 语句)；

(3) PROMPT(设置系统提示符)语句。

输入不带任何参数的 SET 命令，可在屏幕上显示 DOS 环境的当前设置，例如：

```
C>set
COMSPEC=C:\COMMAND.COM
TEMP=C:\DOS;
PATH=C:\SUPER;C:\;C:\DOS;C:\LOTUS;C:\WPS;C:\BAT;C:\PC;C:\WP51;
D:\
C>
```

DOS 环境设置是由 COMMAND. COM 来完成的，用户可以用 COMMAND.COM 来修改环境设置，也可以在 DOS 提示符下，用 SET 命令或者执行有关的批文件来修改环境设置。除此以外，其它的程序不能完成这一任务。在 DOS 提示符下运行一个程序，或者用一个正在运行的程序来调用另一个程序时，该程序从 COMMAND. COM 获得一个环境设置的副本。这对内存驻留程序是十分不利的，因为每当内存驻留程序调入时，它会得到一个环境设置的副本，在此之后，如果用 SET 命令对环境设置进行了修改，内存驻留程序所得到的环境设置副本并不随着修改。这样那些依靠环境变量才能正常运行的内存驻留程序必然会产生问题。

当一个程序运行结束后，它的环境设置副本也随着消失，例如在运行 Lotus 1-2-3 时，用 system 命令暂时返回 DOS 来改变 DOS 提示符，Lotus 利用 COMMAND.COM 的副本暂时访问 DOS，所修改的提示符储存在这个副本里，利用 EXIT 命令返回 Louts 1-2-3 时，COMMAND.COM 的副本消失。

3.5.2 DOS 环境的扩充

如前所述，DOS 环境区大小的缺省值为 160 个字节，即能够储存 160 个字节的信息。这 160 个字符的空间首先必须用来储存 COMSPEC 语句、语句 PATH 和 PROMPT 语句，此外还有其它一些必须使用的变量。有时 160 个字节的空间不够用，必须扩充环境空间。

使用 DOS2.x 版本的用户，必须修改 COMMAND.COM 才能扩充环境空间。可以用调试程序 Debug 修改 COMMAND.COM。Debug.COM 是 DOS 提供的一个实用程序，当用它来修改 COMMAND.COM 时，首先用命令 FORMAT/S 格式化一张软盘，由于使用了/S 参数，软盘可以用来启动系统，COMMAND.COM 也已在软盘上，接着把 Debug.COM复制到这张盘上，修改 COMMAND.COM 的工作就在这张软盘上进行。确信修改正确无误后，把修改过的 COMMAND.COM 复制到硬盘上。DOS2.X 规定，扩充后可达到的最大环境空间为 992 字节。扩充环境空间的具体步骤如下：

键入命令：

A：<Enter> 转到 A 盘上。	将在驱动器 A 内的软盘上修改 COMMAND.COM
DEBUG COMMAND.COM	启动 DEBUG 命令，修改 COMMAND.COM。
-	这是 DEBUG 命令的提示符，当它出现时即可键入命令。
-S 100L 1000 bb 0a 00	这是搜寻 COMMAND.COM 中储存环境空间大小值的命令，0a 00是环境空间的缺省值 160(十六进制)字节(0a00H＝160)。
XXXX:YYYY	DEBUG 找到字符串后，将显示由分号(:)分开的两个数字。由于是十六进制，显示的数字里可能包括字母，要用的是右边的数。
-e YYYY bb 3e 00	这是修改 COMMAND.COM 的 DEBUG 命令，它将给环境分配更多的空间，本命令把环境空间扩充为 992 字节(3e00H)。
-W	把改变后的环境区大小值写盘，信息“Writing ZZZZ bytes”将出现在屏幕上。
-q	退出 DEBUG 命令。

使用 DOS2.0 或 2.1 的 IBM 或其他 IBM 兼容机的用户，做了这种修改后，都能扩充环境空间。可用下面的批处理程序检验扩充后环境空间的大小：

批处理程序 TESTENVI.BAT

```
ECHO OFF
REM TESTENVI.BAT
ECHO Use this variable to test your environment.
ECHO The last line in this batch file will display
```

```
ECHO        the environment.
ECHO Count the characters you see before the variable 1.
ECHO Be sure to count the variable names and the equal sign.
ECHO Then add 50 for each numeric variable except the last.
ECHO Since the last variable may have been cut short,
ECHO        you must count those characters. The resulting
ECHO        number is the current size of the environment.
ECHO NOTE: If you see all 19 variables, then your
ECHO        environment is too large to measure using
ECHO        this batch file.
SET 1=12345678901234567890123456789012345678901234567890123456789012345678
SET 2=12345678901234567890123456789012345678901234567890123456789012345678
SET 3=12345678901234567890123456789012345678901234567890123456789012345678
SET 4=12345678901234567890123456789012345678901234567890123456789012345678
SET 5=12345678901234567890123456789012345678901234567890123456789012345678
SET 6=12345678901234567890123456789012345678901234567890123456789012345678
SET 7=12345678901234567890123456789012345678901234567890123456789012345678
SET 8=12345678901234567890123456789012345678901234567890123456789012345678
SET 9=12345678901234567890123456789012345678901234567890123456789012345678
SET 10=12345678901234567890123456789012345678901234567890123456789012345678
SET 11=12345678901234567890123456789012345678901234567890123456789012345678
SET 12=12345678901234567890123456789012345678901234567890123456789012345678
SET 13=12345678901234567890123456789012345678901234567890123456789012345678
SET 14=12345678901234567890123456789012345678901234567890123456789012345678
SET 15=12345678901234567890123456789012345678901234567890123456789012345678
SET 16=12345678901234567890123456789012345678901234567890123456789012345678
SET 17=12345678901234567890123456789012345678901234567890123456789012345678
SET 18=12345678901234567890123456789012345678901234567890123456789012345678
SET 19=12345678901234567890123456789012345678901234567890123456789012345678
SET
```

ESTENVI.BAT 共设置 19 个环境变量(1—19)，文件最后的 SET 命令显示环境的当前设置。每个变量包括 50 个字符(从变量名开始)，即占用 50 个字节。运行该文件，根据变量在屏幕上的显示情况，可以确定环境空间的字节数。例如屏幕显示结果为：

```
1=12345678901234567890123456789012345678901234567890123456789012345678
2=12345678901234567890123456789012345678901234567890123456789012345678
3=12345678901234567890123456789012345678901234567890123456789012345678
4=12345678
```

那么环境空间的大小为 160 字节。

完成了 COMMAND.COM的修改后，利用这张软盘重新启动系统，然后运行批处理程序 TESTENVI.BAT 来检验环境的大小。如果工作正常，即可把 MMAND.COM文

件复制到硬盘上。这样就完成了环境空间的扩充。

使用 DOS 3.0 或更高版本的用户，把下列各行中的任意一行作为 CONFIG.SYS 文件的第一行，就可以扩充环境空间：

```
Version 3.0 SHELL=C:\COMMAND.COM/E:xx/P
Version 3.1 SHELL=C:\COMMAND.COM/E:xx/P
Version 3.2 SHELL=C:\COMMAND.COM/E:yyyy/P
Version 3.3 SHELL=C:\COMMAND.COM/E:yyyy/P
Version 4.0 SHELL=C:\COMMAND.COM/E:yyyy/P
Version 5.0 SHELL=C:\COMMAND.COM/E:yyyy/P
```

如果使用软盘，则应该把上面各行中的 C:\改为 A:\。第 1,2 行中的 XX 是 10 至 62 间的任意值，代表环境空间包括长度为 16 字节的段的个数。如果 XX 等于 20，环境空间就是 320 字节；XX 等于 62 时，就达到了环境空间的最大值(992 字节)。

从 DOS3.2 开始，可允许建立更大的环境空间，YYYY 的值在 160 至 32768 之间，这样环境空间的最大值可达 32K。上面各行中的参数/P 的功能之一是使 COMMAND.COM 自动运行 AUTOEXEC.BAT 文件，这是因为 SHELL 命令是在文件 CONFIG.SYS 里面，而 COMFIG.SYS 又在 AUTOEXEC.BAT 之前执行。如果不包括参数/P，将越过 AUTOEXEC.BAT 文件。

在上面的各个 SHELL 语句中，使用了参数/E 和/P，它们并不是 SHELL 的参数，而是 COMMAND.COM 的参数。全部参数如下：

/C 告诉 COMMAND.COM 去执行列在/C 后面的命令。这样可把批处理程序名列在/C 的后面，以执行批处理程序，并可把批处理程序中的最后一个命令安排为 EXIT，以清除 COMMAND.COM 的副本。

/D 使 COMMAND.COM 不运行 AUTOEXEC.BAT 程序。

/E 用来改变 DOS 环境空间的大小。首次应用在 DOS3.0 版本里；从 DOS3.1 版本开始，正式列为 COMMAND.COM 的参数。在 DOS3.0 版本中的工作方式也与后来的版本不同。

/F 首次出现在 DOS3.1 里，在出现提示信息 Abort, retry or fail 或 Abort,ignore, retry or fail 时，该参数使 DOS 自动反应，并把反应信息显示在屏幕上。

/P 使调入的辅助命令处理程序常驻内存，此时要回到命令处理状态是不可能的，只有重新启动才行。另外，该参数迫使 COMMAND.COM 运行 AUTOEXEC.BAT。

值得注意的是，在使用上述 SHELL 语句时，必须把 COMMAND 的扩展名.COM 也写上。上述语句及其各项参数设置完毕后，必须重新启动系统，对环境空间的改变才能起作用。在任何时候都应保存一张用于启动的系统软盘，这样在扩充环境空间时，如果由于误操作使硬盘不能启动，就可用这张系统软盘来启动系统。某些错误会导致死机现象，此时可用软盘重新启动，然后转到硬盘上来作相应处理。实际上在进行扩充环境操作前，应把文件 CONFIG.SYS 复制到软盘上。

3.5.3 批处理程序对环境变量的调用

批处理程序调用环境变量之前，被调用变量必须存在于环境中，即用 SET 命令事先在环境内设置，然后才能调用。SET 命令的格式为：

SET variable=value

SET 与 variable 之间应有一空格。在等号两侧安排空格也是可以的，但空格将成为 variable 或者 value 的一部分，例如：

SET TEMP = C：\JUNK

上面 SET 命令规定的变量名是 TEMP 和一个空格，其值为空格+C：\JUNK，这样做是不可取的。应避免把空格作为变量名或变量值的一部分，这样可减少很多麻烦。由于变量值是字符串，任何信息都可以储存在环境里。

COMSPEC，PATH 和 PROMPT 是三个专用的环境变量，在环境中设置它们时，前面不必使用 SET。

环境变量 COMSPEC 是 COMmand 与 SPECification 两个英文词的缩写。当 COMMAND.COM 的暂驻部分被应用程序冲掉时，DOS 通过 COMSPEC 找到 COMMAND.COM 所在的目录，这样可以重新把 COMMAND.COM 调入内存。COMSPEC 指示寻找 COMMAND.COM 的路径，例如：

SET COMSPEC=C：\COMMAND.COM

改变 COMSPEC 的值是危险的，如果规定了不正确的路径，DOS 将无法找到 COMMAND.COM，提示信息 Connot load COMMAND，system halted 将指示系统出错，此时只有重新启动才行。

环境变量可由批处理程序或其它命令调用。在批处理程序中调用环境变量的格式是在变量名的两侧各加上一个%：

%variable%

下面是批处理程序调用环境变量的例子：

批处理程序 SET1.BAT

```
REM  SET1.BAT
DIR  %TODAY%
ECHO %TODAY%
```

如果事先使用了 SET TODAY=*.BAK，那么执行这个批处理程序的结果为：

```
C>SET1

C>REM SET1.BAT

C>DIR *.BAK
```

```
Volume in drive C is MS\|DOS_5
Volume Serial Number is 1AE3-7711
Directory of C : \BAT\BAT

01         BAK        470 11-25-90    8 : 59p
02         BAK        481 11-25-90    8 : 59p
1          BAK         26 11-25-90    8 : 59p
1-LOG      BAK        354 08-15-89    10 : 08a
2          BAK         21 11-25-90    9 : 00p
         5 file(s)     1352 bytes
                      15753216 bytes free

C>ECHO *.BAK
*.BAK

C>
```

在批处理程序中调用环境变量，可以给一些应用程序加上口令，以防止程序被盗用，批处理程序 PASSWORD.BAT 就是这样一个例子。如果要运行 PASSWORD.BAT 中名为 Kr 的程序，必须事先在 DOS 提示符下，用 SET 命令给变量 PASSWORD 赋值，该值即为运行 Kr 程序的口令。

批处理程序 PASSWORD.BAT

REM PASWORD.BAT	文件名注释
IF (%)==( ) GOTO NOPAS	如果不输入口令字，就转移到标号 NOPAS。
IF (%PASSWORD%)==( ) GOTO NOTSET	如果事先没有设置环境变量，则转移到标号 NOTSET。
IF NOT %1==%PASSWORD% GOTO NO	如果输入了错误的口令，则转移至标号 NO。
Kr	输入的口令正确，则运行。
GOTO END	Kr 程序，然后退出批处理程序。
:NOPAS	标号行。
ECHO you must enter a password	告诉用户输入口令。
GOTO END	
:NOTSET	如果事先没有设置环境变
ECHO Password not set	量就转到这里，显示信息后，退出。
GOTO END	
:NO	输入的口令不正确就转到
ECHO Incrorrect password	这里。产生“死机”。
ECHO Locking system	
PAUSE>NUL	模拟“死机”现象并把信息输入到 NUL。按

```
                                        任一键将重新启动批处理程序。
GOTO   END                              退出批处理程序。
:END                                    标号行。
```

下面是四种不同情况的执行结果：

(1) 在执行批处理程序的命令行上没有输入口令。

```
C:\BAT\BAT>ECHO OFF
You must enter a password.
See manual for syntax.
C:\BAT\BAT>
```

(2) 如果没有设置环境变量，但在执行批处理程序的命令行上输入了口令。

```
C>PASSWORD PETRO
C>ECHO OFF
PASSWORD NOT SET
```

(3) 设置了环境变量，但是在执行批处理程序的命令行上输入的口令不对。

```
C>SET  PASSWORD=Petro

C>PASSWORD  PETRO
C>ECHO OFF
Incorrect  password
Locking  system
```

注意，用 SET 命令设置的口令为 Petro，而输入的口令为 PETRO，字母完全相同，只是大小写不同。

(4) 设置了口令，在执行批处理程序的命令行上也输入了正确的口令，但由于 Kr.COM程序并不存在，因而出现了 DOS 提示信息：

```
C>PASSWORD  Petro
Bad  command  or  file  name
```

批处理程序亦可以调用环境变量的缺省值。批处理程序 NEWPROMPT.BAT 的功能是：如果 DOS 没有设置提示符，则进行设置；否则保持 DOS 提示符不变。

批处理程序 NEWPROMPT.BAT

```
ECHO  OFF                               关闭 ECHO。
REM  NEWPROMPT                          文件名注释。
IF NOT (%PROMPT%)==( ) GOTO END         如果已设置提示符，则保持不变，并退出批处理
                                        程序。
SET PROMPT=$p$g                         设置 DOS 提示符。
:END                                    标号行 END。
```

通过调用一个或几个环境变量的值，可在批处理程序之间传递信息。批处理程序 ACCOUNT1.BAT 完成每月银行帐户的结算工作，如果该文件运行正常，程序 CLOSE.EXE

把 ERRORLEVEL 设置为 0，否则设置 ERRORLEVEL 为 1。ACCOUNT1. BAT 利用 ERRORLEVEL 与 ACCOUNT2. BAT 之间传递信息。在运行了 ACCOUNT1. BAT 后，很可能执行其他的 DOS 命令或程序改变了 ERRORLEVEL 的值。为解决这一问题，ACCOUNT1. BAT 设置一个环境变量值，ACCOUNT2. BAT 根据它判断 ACCOUNT1. BAT的运行情况（这里只是说明批处理程序之间如何通过环境变量相互联系，其中的 IF 语句，ERRORLEVEL 等将在后面详细介绍）。

批处理程序 ACCOUNT1. BAT

```
ECHO  OFF                  关闭 ECHO。
REM  ACCOUNT1.BAT          注释文件名。
CLOSE                      运行 CLOSE.EXE。
IF  ERRORLEVEL 1           如果 CLOSE.EXE 运行过程中把 ERRORLEVEL。
    SET  ACCOUNT=NO        设置为 1，则 SET ACCOUNT=NO，CLOSE.EXE。
IF ERRORLEVEL 1            运行结束，转移至标号行 END。
   GOTO END
IF  ERRORLEVEL 0           如果 CLOSE.EXE 运行过程
    SET  ACCOUNT=YES       中把 ERRORLEVEL 设置为 0，则 SET
                           ACCOUNT=YES。
: END                      标号行 END。
```

上述文件中，很自然地产生一个问题：为什么程序 CLOSE. EXE 本身不设置环境变量 ACCOUNT，而必须在 CLOSE. EXE 运行结束后，根据 CLOSE. EXE 运行过程中设置的 ERRORLEVEL 值，由 ACCOUNT1. BAT 来设置？我们知道，一般程序开始运行时，都能得到一个完整的环境变量设置的副本，程序运行过程中对环境变量做的任何改变，都只是对环境变量设置副本的改变，而环境变量设置的原件保持不变。程序运行结束时，其副本也随之消失，因而用程序 CLOSE. EXE 来设置变量 ACCOUNT 是不可能的。只有在 DOS 提示符下用 SET 命令或用批处理程序才能对环境变量设置进行修改。ERRORLEVEL 不是环境变量，当程序 CLOSE. EXE 运行结束时，它设置的 ERRORLEVEL 值仍然存在。根据批处理程序 ACCOUNT1. BAT 设置的环境变量 ACCOUNT 的值决定 ACCOUNT2. BAT 是否执行，如果批处理程序 ACCOUNT1. BAT 运行正常（ACCOUNT 的值为 YES），则运行 ACCOUNT2. BAT，否则不运行（ACCOUNT 的值为 NO）。

批处理程序 ACCOUNT2. BAT

```
ECHO  OFF                          关闭 ECHO。
REM  ACCOUNT2.BAT                  注释文件名。
1F (%ACCOUNT%)==() GOTO   ERROR    如果没有设置环境变量
                                   ACCOUNT 的值，转向标号行 ERROR。
1F %ACCOUNT%==NO GOTO NO           如果环境变量 ACCOUNT 的值为 NO，转向
                                   标号 NO。
CLOSE2                             如果批处理程序能运行到这里，执行名为
                                   CLOSE2. EXE 的程序，此时 ACCOUNT 的
```

值一定为 YES，环境变量 ACCOUNT 的值是由文件 ACCOUNT1.BAT 设置的，只有 YES/NO 两种可能。

```
GOTO   END                          CLOSE2 运行结束后，退出批处理程序。
:ERROR                              标号行 ERROR。
ECHO   RUN   ACCOUNT1   FIRST       告诉用户首先运行批处理程序ACCOUNT1.BAT。

PAUSE                               暂停批处理程序的运行，给用户提供时间，阅读信息，按任意一键，即继续运行。

GOTO   END                          退出批处理程序。
:NO                                 如果 CLOSE.EXE 运行过程中出错，则转移到此。

ECHO   WARNING:                     给用户显示信息。
       CLOSE  DOES  NOT
       RUN  SUCCESSFULLY
ECHO  FIX  THIS  ERROR
       BEFORE  RUNNING
       ACCOUNT2
PAUSE                               暂停批处理程序的执行，给用户时间阅读信息。

:END                                标号行 END。
```

第 4 章　批处理程序中的循环

为了执行需要重复多次的操作，可在批处理程序内设置循环。批处理程序中的循环与高级语言中的循环相同，即由批处理命令语句组成循环体，循环体内设置条件语句，用来控制循环的流向。

下面详细介绍批处理程序中组成循环的各个语句。

4.1　GOTO 语句

GOTO 语句把控制无条件转移到相应标号行处，接着开始执行该行的内容，GOTO 语句的格式为：

GOTO　LABEL(标号)

其中的标号(LABEL)为循环体中的独立的标号行，标号行以冒号开始，后面紧跟着标号，冒号与标号之间不得有空格。设置标号时，最好使用助记符，如：BEGIN，：END 等。标号为字符串，但是 DOS 管道操作符(，)，>>及｜、方括号符不能用在标号里，还有一些字符不能作为标号的头一个字符，不同的 DOS 版本有不同的规定。但由阿拉伯数字和英文字母组成的标号，在各种 DOS 版本下运行都没问题。标号的长度应小于或等于 8，即前 8 个字符有效，例如 LABEL12345 与 LABEL123 相同。标号行在 GOTO LABEL 行之前或之后，控制转移到标号行后，执行标号行下面的第一行，不再返回到 GOTO 语句行。如果 GOTO 语句所要转向的标号不存在，则当前批处理程序停止执行，并在屏幕上显示 Label not found 的提示信息。

请看下面名为 LOOP1. BAT 的批处理程序，它将反复执行列目录操作，如果不按 Ctrl-Break 键，将一直循环执行下去。

批处理程序 LOOP1. BAT

```
REM　LOOP1.BAT
：TOP                    标号行
　DIR                   执行 DIR 命令
GOTO　TOP                控制转向标号行，接着执行 DIR 命令
```

批处理程序 LOOP2. BAT 将把一系列文件复制到 A 盘上，全部文件复制完后，文件本身不能自动停止运行。

批处理程序 LOOP2. BAT

```
REM　LOOP2.BAT
ECHO　OFF                使回显处于关闭状态
：TOP                    标号行
```

```
    COPY %1 A:        把指定的文件复制到 A 盘上
    SHIFT             左移实际参数
GOTO  TOP             控制转向标号行,继续执行
```

注意到在上述两个文件里,在 GOTO 语句与标号行之间的内容,每行前面都留有空白,这样做的目的是为了使文件容易阅读。

4.2 IF 语 句

在前一节的例子中,由于没有控制语句,批处理程序运行起来后,不能自动停止,实际上形成了无限循环,这种批处理程序是不实用的。解决这一问题的方法是在批处理程序里加上 IF 语句,例如在上节里的第一个例子里加上 IF 语句后,构成批处理程序 LOOP3.BAT:

批处理程序 LOOP3.BAT

```
REM  LOOP3.BAT
ECHO  OFF
:TOP
   DIR %1
   IF %1==STOP  GOTO  END
   SHIFT
GOTO  TOP
:END
```

运行此文件的结果为:

```
C>LOOP3 *.BAK  STOP

C>REM  LOOP3.BAT
C>ECHO  OFF

Volume  in  drive  C  is  Daqing
Directory  of  C:\BAT

LOOP1  BAK  39  10-12-92  4:16P
LOOP3  BAK  128  10-12-92  4:50P
    2  File(s) 2101248  bytes  free
```

从上面的例子可以看出,IF 语句是在条件满足之后,才能执行的命令。IF 语句在批处理程序里的格式为:

```
IF [NOT] Conditon  command
```

其中:

conditon 是 DOS 布尔条件表达式。

如果条件为真，批处理程序将执行后面的 DOS 命令。若检验的条件为假，则跳过条件后面的命令而执行下一行。条件后面可跟任何 DOS 命令。

IF 语句中的 ERRORLEVEL 出口码，在检验为真值时，并不是等于，而是大于或等于；例如 IF ERRORLEVEL 5，所有在 5 至 255 之间的任何十进制整数都为真。

当比较两个字符串时，string1 和 string2 所对应的字符必须完全相同（字母的大写或小写也必须相同），才能取真值。字符串中不得包括分隔符（如逗号、分号、等号或空格等）。

如果在指定的驱动器上找到了文件标识符所规定的文件，则取真值。文件标识符不可为路径名。

条件语句在检验实际参数时，有一点应该注意，下面的例子可说明这一点。

批处理程序 LOOP4. BAT

```
REM   LOOP4.BAT
ECHO   OFF
:TOP                              标号行
   IF %1== GOTO   END             实际参数用完时，结束批处理程序的执行
   SHIFT                          左移实际参数
   ECHO   STILL   RUNNING         显示信息
GOTO   TOP                        转移到标号行 TOP
:END                              标号行
```

该程序执行时，DOS 把第一个实际参数赋给%1，在执行第二次循环时，SHIFT 把第二个实际参数赋给%1，以此类推，文件继续执行。由于 IF 语句中==右侧的第一个字符为空格，当实际参数用完时，应从条件语句转向标号 END，从而结束该程序的运行。实际上 IF 语句此时获得的是假值，不能结束程序的运行。而必须人为地终止它。运行时屏幕显示为：

```
C>LOOP4 1 2 3

C>REM LOOP4.BAT

STILL RUNNING
STILL RUNNING
Syntax error
STILL RUNNING
Syntax error
STILL RUNNING
Syntax error
STILL RUNNING
Syntax error
STILL RUNNING
```

```
Terminate batch job(Y/N)? Y
C>
```

解决这一问题的方法是把等号左侧的形式参数与等号右侧的空格分别用括号括起来，即：

```
IF (%1)==( ) GOTO   END
```

当实际参数没有完时，检验结果为假值，即：

(实际参数)==( )为假。

将执行下一行的 SHIFT 命令。当实际参数用完时，检验结果为真值，即：

```
( )==( )
```

程序将执行 GOTO 命令，转向标号 END。结束批处理程序的执行。

当条件语句里有 NOT 时，则检验结果取反，即当〈Condition〉条件为假时，则 NOT 〈Condition〉条件便为真，例如：

```
IF  NOT  EXIST  A：%1  COPY  B：%1  A：
```

在这个条件语句中，将要执行的磁盘操作是把满足规定的文件从驱动器 B 复制到驱动器 A。在执行 COPY 命令之前，首先须查明文件是否已存在于驱动器 A。如不存在，则条件为真，执行 COPY 命令；如果文件存在，则条件为假，不执行 COPY 命令。下面的例子对 NOT 的用法作了进一步说明。

批处理程序 SHOWNOT. BAT

```
REM  SHOWNOT.BAT
IF EXIST *.BAK ECHO There are backup files
                                        如果 *.BAK 文件存在，则显示信息
IF  EXIST *.BAK  DEL *.BAK              如果 *.BAK 文件不存在，则显示信息
IF NOT EXIST *.BAK ECHO There are no backup files
                                        如果 *.BAK 文件存在则删除
```

下面的批处理程序，没有任何实用价值，但是却清楚地说明了怎样同时使用几个 IF 语句，来克服字符串比较时易发生的问题。

批处理程序 BIGIF. BAT

```
ECHO OFF
REM BIGIF.BAT
IF (%1)==( )GOTO NONE       如果不输入实际参数，转向标号 NONE
IF %1==A GOTO ALPHA         进行字符串比较时，等号两端相同字母的大小写也必需相同
IF %1==a GOTO ALPHA         如果实际参数为 1，显示信息后即退出批处理程序，在字符
IF %1==1 ECHO 1 Entered     串比较时使用数字，可避免字母的大小写问题
IF %1==I GOTO END
IF %1==2 ECHO 2 Entered     如果实际参数为 2，显示信息后
IF %1==2 GOTO END           即退出批处理程序
GOTO INVALID                如果批处理程序运行到这里，说明输入了不正确的实际参数，
```

```
                                    因此文件转到错误处理部分
:NONE                               由于没输入实际参数,批处理程序
ECHO  MUST  START  WITH             显示提示信息后退出
VARIABEL  AFTER  NAME
GOTO END
:ALPHA                              用户使用的实际参数是字母
ECHO MUST START WITH A NUMBER
GOTO END                            (A 或 a),这是批处理程序不需要的,因此退出批处理程序
:INVALID                            用户没有输入正确的实际参数,显示
ECHO THE OPTION YOU                 信息后,退出批处理程序
SELECT IS INVALID
GOTO END
:END                                标号行
```

输入不同的实际参数时,该文件的运行结果为:

```
C>BIGIF

C>ECHO OFF
  MUST START WITH VARIABEL AFTER NAME

C>BIGIF A

C>ECHO OFF
  MUST START WITH A NUMBER

C>BIGIF 7

C>ECHO OFF
THE OPTION YOU SELECTED IS INVALID CHECK YOUR MANUAL

C>BIGIF 2

C>ECHO OFF
  2 Entered

C>
```

批处理程序 BACKUP1.BAT 将根据输入的实际参数来决定怎样执行 BACKUP 命令,如果输入了错误的实际参数,或者没有输入实际参数,它将提示用户输入正确的实际参数。

批处理程序 BACKUP1.BAT

```
ECHO  OFF
```

```
REM   BACKUP1.BAT
IF (%1)==(1) CD\                          如果输入的实际参数为1,则返回根
IF (%1)==(1) BACKUP C: A: /S              目录并对C盘所有文件作备份,然
IF (%1)==(1) GOTO END                     后退出批处理程序
IF (%1)==(2) CD\                          如果输入的实际参数为2,则返
IF (%1)==(2) BACKUP C: A: /M              回根目录,并对C盘上自上次备
IF (%1)==(2)  GOTO  END                   份以来已作了修改的文件作备份,然后退出批处理
                                          程序
IF (%1)==(3) CD\                          如果输入的实际参数为3,则
IF (%1)==(3) BACKUP C: A:  /M/A           返回根目录,并对C盘上自
IF (%1)==(3) GOTO END                     上次备份以来已作了修改的文件作备份,然后加到
                                          A盘上已有文件里
REM IF %1 Valid,will have jumped to end by now
REM Display error message
ECHO YOU ENTERED AN INVALID PARAMETER
                                          对于没有输入或输入了无效实际参数的处理
ECHO Enter 1 for a full backup
ECHO Enter 2 for a incremental backup
ECHO Enter 3 for an appended
Incremental backup
GOTO END
:END                                      标号行
```

4.3 EXIST 语句

EXIST 语句可与 IF 语句结合起来使用,用来检验当前子目录或当前磁盘上某些文件是否存在,例如:

```
IF  EXIST  PRO.DAT  ECHO  you  can find  it
```

再如:

```
IF  NOT  EXIST  PRO.DAT  ECHO  you  can not find it
```

批处理程序 IF1.BAT 说明了 EXIST 语句的功能和用法

批处理程序 IF1.BAT

```
ECHO  OFF
REM  IF1.BAT
IF  EXIST  %1  ECHO %1  EXISTS !!!    检查名为%1的文件是否存在,如存在则显示信息
IF  EXIST  %2  ECHO %2  EXISTS!!!     检查名为%2的文件是否存在,如存在则显示信息
IF  EXIST  %3  ECHO %3  EXISTS!!!     检查名为%3的文件是否存在,如存在则显示信息
```

执行此文件时,在 DOS 提示符下键入:

```
C>IF1 *.BAK CHARPTER1.DOC
```

在屏幕上显示出该文件的执行结果为：

```
C>ECHO  OFF
*.BAK  EXISTS !!!!

CHARPTER1.DOC  EXISTS !!!!
C>
```

这里用实际参数*.BAK,CHARPTER1.DOC 分别代替了形式参数%1 和%2。由于只输入了两个实际参数，所以形式参数%3 无对应的实际参数，但是该文件能够正常运行。

批处理程序 IF2.BAT 与 IF1.BAT 不同，它是个实用性很强的文件，它可以用来清除临时文件。

批处理程序 IF2.BAT

```
ECHO  OFF
REM  IF2.BAT
IF  EXIST *.OBJ  ECHO *.OBJ  files  being  erased
                检查当前目录中是否在*.OBJ 文件，如存在则显示信息
IF  EXIST *.OBJ  DEL *.OBJ
                如果*.OBJ 文件存在，则删除。使用 IF 语句是为了避免出现 file  not  found
                提示信息的出现
IF  EXIST *.BAK  ECHO *.BAK  files  being  erased
                检查当前目录中是否存在*.BAK 文件，如存在则显示信息
IF  EXIST *.BAK  DEL *.BAK
                如果*.BAK 文件存在则删除
ECHO  Finished  erasing
                告诉用户批处理程序执行完毕
```

在 IF2.BAT 中，含有 ECHO 的 IF 语句与含有 DEL 的 IF 语句，顺序不能改变。因为若先执行 DEL，会把文件删除，在再执行含有 ECHO 的 IF 语句时，会得到假值，运行结果和编写 IF2.BAT 的目的不符。在当前目录存在*.OBJ 文件，不存在*.BAK 文件时，IF2.BAT 的运行结果为：

```
C>IF2

ECHO  OFF
*.OBJ  files  being  erased
Finished  erasing

C>
```

4.4 ERRORLEVEL 语句

前面曾讲到 DOS 环境和环境变量的设置方法。一般程序运行时，都会得到环境变量设置的完整副本，在程序运行过程中，对环境所作的一切操作都是针对副本，而不是对环境设置的原件进行的。程序运行结束时，环境副本也随着消失。

从 DOS2.0 开始提供了 ERRORLEVEL 语句，ERRORLEVEL 不是环境变量。程序或 DOS 命令执行时对 ERRORLEVEL 值的改变，在程序或 DOS 命令结束后仍然存在。若一个程序或 DOS 命令执行成功，则返回的 ERRORLEVEL 的值为 0，否则为 1 至 255 之间 ASCII 值的任意整数，其值只占一个字节的内存空间，例如执行完 BACKUP 命令时，根据命令执行的情况，返回的 ERRORLEVEL 出口码分别为：

0　备份成功
1　未找到备份文件
2　因共享冲突未备份某些文件
3　用户按了 Ctrl-Break 键
4　因错误而中止备份命令的执行

ERRORLEVEL　主要用在 IF 语句中：

```
IF  ERRORLEVEL #  Command IF  NOT  ERRORLEVEL #  Command
```

值得注意的是 IF 语句中检验的 ERRORLEVEL 值是大于或等于而不是等于，例如语句 IF ERRORLEVEL 5 检验的是 ERRORLEVEL>=5 而不是 ERRORLEVEL=5，因此对于 5 至 255 之间的任何整数，此语句检验结果都是真值。

DOS3.0 版本之前，只有 BACKUP 和 RESTORE 支持 ERRORLEVEL，能够根据命令的执行情况，返回相应的 ERRORLEVEL 值。从 DOS 3.0 开始，增加了一些支持 ERRORLEVEL 的命令。目前，一些编译程序也开始支持 ERRORLEVEL。重新启动系统及成功地执行一个 DOS 命令或一应用程序，都把 ERRORLEVEL 的值设置为零。

批处理程序 IF3.BAT 说明了 ERRORLEVEL 的用法。程序中的 XCOPY 命令所要复制的文件不存在，将把 ERRORLEVEL 值设置为 1：

批处理程序 IF3.BAT

```
ECHO  OFF
REM  IF3.BAT
XCOPY C:Daqing.Kr D:
IF  ERRORLEVEL 1  ECHO  FILE  NOT    FOUND  TO  COPY,TRY AGAIN
                          如果 ERRORLEVEL 值大于或等于 1，显示错误信息
IF  ERRORLEVEL 1 GOTO  END
                          如果 ERRORLEVEL 值大于或等于 1，转向标号行 END
IF  ERRORLEVEL  0  ECHO  COPY  Successful
                          如果 ERRORLEVEL 值大于或等于零，显示文件复制成功信息，因为
                          上行已把大于或等于 1 的 ERRORLEVEL 值排除，因此只有 ER-
```

```
                                    RORLEVEL 0 能到达此行
: END                               标号行 END
```

运行结果为：

```
9C : >if3

C : >ECHO OFF
File not found - DAQING. KR
        0 File(s) copied
FILE NOT FOUND TO COPY, TRY AGAIN
C : >
```

4.5 FOR 语 句

在批处理程序中，FOR 语句重复执行 DOS 命令，即对一个或一组文件，重复执行一个 DOS 命令。FOR 语句有两种形式，第一种形式为：

```
FOR %%h IN (CHAPTER1. BAK CHAPTER2. BAK) DO ERASE %%h
```

第二种形式是使用文件通配符：

```
FOR  %%h IN (*. BAK) DO ERASE %%h
```

一般形式：

```
FOR %%variable IN (file set) DO command
```

FOR 语句必须处在同一行，而且不得嵌套，即一个 FOR 语句中只能包括一个 FOR。变量%%variable 只能为单个字符，即%%A—%%Z 之中的一个。同一个 FOR 语句中，前后变量的字符必须相同。某些 DOS 版本允许用%%0—%%9 作为变量。这样极易和批处理程序的形式参数相混淆，为避免出错，应尽量用%%A—%% Z 作为 FOR 语句的变量。在 DOS 提示符下可直接使用 FOR 语句，用法的区别在于必须把%%variable 写成%variable。对于不支持文件通配符的 DOS 命令，在屏幕上直接使用 FOR 语句是十分有用的，例如准备使用 TYPE 命令把三个文件 File1，File2，File3 显示在屏幕上，必须键入命令：

```
TYPE  File1 | MORE
TYPE  File2 | MORE
TYPE  File3 | MORE
```

这样麻烦费时，在屏幕上直接使用 FOR 语句可以完成同样的任务，简单省时：

```
C>FOR %h IN (File?) DO TYPE %h | MORE
```

上述语句中，假定当前目录中文件名前四个字符是 File 的文件只有三个。

FOR 语句中可以包括逻辑检验，例如：

```
C>FOR %j IN (*.*) DO IF EXIST A:%j DEL %j
```

该语句将删除C盘上同时存在于A盘上相同的文件。

少数DOS版本允许FOR语句的变量为多个字符，其作用与单个字符相同。对于不支持多个字符作为变量的DOS版本，如果变量多于一个字符，将出现错误信息。使用单个字符作为FOR语句中的变量，任何DOS版本都可接受。

批处理程序FOR1.BAT中使用%%LIST作为变量，由于所用的DOS版本不支持多个字符变量，因而运行过程中出现了错误信息。

批处理程序 FOR1.BAT

```
REM FOR1.BAT   注释文件名
FOR %%LIST IN (*.DOC) DO COPY %%LIST   D:
```

使用了多个字符作为变量。

运行结果为：

```
C>FOR1

C>REM FOR1.BAT

C>COPY % LIST IN (*.DOC) DO COPY % LIST D:
Invalid number of parameters
```

批处理程序FOR-BAK.BAT将从当前目录上删除所有扩展名为.BAK的文件。

批处理程序 FOR-BAK.BAT

```
REM   FOR-BAK.BAT                    注释文件名
DIR   *.BAK                          在屏幕上列出所有扩展名为.BAK的文件
FOR %%h IN (*.BAK) DO DEL %%h利用FOR语句，删除文件。当然可以用Del *.BAK一
                                     次性删除，这里只是说明FOR语句的用法
DIR   *.BAK                          执行DIR命令
```

执行后屏幕上的显示为：

```
C>FOR-BAK

C>REM   FOR-BAK.BAT

C>DIR *.BAK

Volume in drive C has no lable
Directory of C:\WORD\BOOK

BATCH      BAK  10624  7-26-1993 1:20P
CHAPTER4   BAK  4480   7-26-1993 2:50P
SHOWECHO   BAK  1009   7-26-1993 2:06P
```

```
ECHOMESS  BAK  165     7-26-1993 2：15P
        4  File(s)  4315136  bytes  free

C>FOR %h IN (*.BAK) DO DEL %h

C>DEL  BATCH.BAK

C>DEL  CHAPTER4.BAK

C>DEL  SHOWECHO.BAK

C>DEL  ECHOMESS.BAK

C>

C>DIR *.BAK

Volume  in  drive  C  has  no  lable
Directory  of  C：\WORD\BOOK

File  not  found
```

批处理程序 FOR2.BAT 利用 FOR 语句复制文件

批处理程序 FOR2.BAT

```
REM  FOR2.BAT  注释文件名
FOR %%L IN (*.DOC) DO COPY %%L D：
                        把文件 *.DOC 复制到 D 盘
```

执行结果为：

```
C>FOR2

C>REM  FOR2.BAT

C>FOR %L IN (*.DOC) DO COPY %L D：

C>COPY  CHAPTER1.DOC  D：
        1  File(s)  copied

C>COPY  CHAPTER3.DOC D：
        1  File(s)  copied

C>
```

注意到，由于回显处于 ON 状态，执行的命令都显示在屏幕上，当%%variable 显示在屏幕上时，只显示一个百分号(%)，这与在屏幕上(不是在批处理程序里)直接执行 FOR 语句是一致的。

关于 FOR 语句的使用方法，归纳如下：

(1) 为了在各种 DOS 版本下都能运行，%% variable 的变量必须为单个字符，应避免使用%%0—%%9 作为变量。

(2) FOR %%variable IN (File set) command %%variable 其中 File set 必须放在括号内，文件的个数没有限制，只要不超过 DOS 命令行 127 个字符的限制即可，文件之间用空格或逗号作为分隔符分开，文件名可以使用通配符。

(3) FOR 语句中后面的变量有时可省略，如：

```
FOR %%a  IN (*.BAK) DO  DIR
```

(4) FOR 语句的结尾如果执行另一个批处理程序，执行结束后，不再返回 FOR 语句。

4.6 批处理程序出错的处理方法

到目前为上，我们已经介绍了一些专用的批处理语句，以及建立和使用批处理程序的方法。在建立和使用批处理程序过程中，有时出现错误，导致批处理程序不能正常执行。最常见的是拼写错误，特别是在利用 COPY CON 命令从键盘上建立批处理程序时，应特别仔细。

在批处理程序执行过程中，如果出现了 File not found 的提示信息，而批处理程序并没有调用任何文件，可能是在 ECHO 语句中使用了管道操作符(<,>,|,>>)的缘故。如果在一个语句中有管道操作符和其它 DOS 命令，总是优先执行管道操作，然后再执行 DOS 命令，例如在批处理程序中有语句：

```
ECHO TYPE<TO REDIRECT INPUT
```

DOS 将认为 TO 是一个文件，并把它的内容通过 TYPE 命令显示在屏幕上，如果 TO 文件不存在，就会出现 File not found 提示信息，如果 TO 文件存在，ECHO 将显示 redirect input。批处理程序如果含有语句：

```
ECHO TYPE>TO redirect output
```

ECHO 将不显示任何内容，DOS 将在当前目录中建立一个名为 TO 的文件，如果当前目录已存在文件 TO，上述语句会把 TO 文件原有的内容冲掉。

在批处理程序的执行过程中，有时会出现 Bad command or file name 的提示信息。产生这种错误的多数原因是 DOS 命令拼写错误，也可能是企图进入一个不存在的目录。如果在批处理程序里有下列语句：

```
TYPE | TO pipe output
```

DOS 会认为 TO 是一个 DOS 命令，而实际上并没有名为 TO 的 DOS 命令，也会出

现 Bad command or file name 的错误信息。

Intermediate file error during pipe 错误信息的出现是由于在批处理程序的最后一行没有回车。

标号前面必须有冒号(:),否则会出现 Label not found 错误信息。如果批处理程序里含有两个相同的标号,会造成无休止地循环,因为 DOS 总是首先从文件的开头寻找标号,如有两个相同的标号,DOS 永远不会转移到第二个标号。

一个批处理程序如果不能正确执行,且无任何显示,特别是使用了多个 IF 语句时,很容易产生逻辑错误,此时应打开回显,然后重新运行批处理程序,这样文件中的每个命令在执行前会显示在屏幕上,很容易查出问题的所在。

如果批处理程序执行过程中,想暂停它的运行,可按 PAUSE 键,之后按任一键(shift 除外),文件将继续执行。如果按 Ctrl-C(),DOS 信息将出现在屏幕上:

Dou you want to terminate batch job (Y/N)?

可根据情况回答 YES 或 NO。

第 5 章　批处理程序中子程序的应用

用程序设计语言编写计算程序时，把要重复执行多次的计算过程编成一段程序，这样的程序称为子程序。通常把调用子程序的程序段称为主程序。在主程序中设置调用子程序语句，当执行到调用子程序语句时，控制即转向子程序，执行完子程序后，从子程序返回主程序，继续执行下一个语句。和程序设计语言一样，从 DOS3.3 开始，批处理程序开始支持子程序。

5.1　批处理程序中的子程序

批处理程序支持子程序是指在一个批处理程序运行中可以调用另一个批处理程序，当被调用的批处理程序运行结束时，即返回到调用它的批处理程序，并继续执行后面的命令。

DOS3.3 之前的 DOS 版本，批处理程序在运行过程中，也可以调用另一个批处理程序，但是被调用的批处理程序运行结束时，控制不会返回到调用它的批处理程序。因此在 DOS3.3 之前，批处理程序文件是不支持子程序的。下面两个批处理程序的运行结果说明了这一点。文件 SUB1.BAT 为主程序，由它调用子程序 SUB2.BAT。

批处理程序 SUB1.BAT

```
REM SUB1.BAT
ECHO OFF
REM This is the main program
REM It will call the subroutine
REM on the next line
SUB2                                    调用子程序 SUB2，当 SUB2 执行结束时，不返回
                                        SUB1。
ECHO If control returns to SUB1.BAT     显示信息
ECHO you will see this message
```

批处理程序 SUB2.BAT

```
REM SUB2.BAT
ECHO This is the last line in SUB2.BAT
```

主程序 SUB1.BAT 调用 SUB2.BAT 的运行结果为：

```
C>sub1

C>REM SUB1.BAT
```

```
C>ECHO OFF
This is the last line in SUB2.BAT

C>
```

可以看出，主程序 SUB1.BAT 中最后两个 ECHO 语句并没有执行，说明调用子程序 SUB2.BAT 并执行后，没有返回到主程序 SUB1.BAT 中去。实际上，对于低于 3.3 的 DOS 版本，批处理程序虽然不支持子程序，即在一个批处理程序的运行过程中，启动第二个批处理程序，执行完后，控制不能返回第一个批处理程序，但可以 COMMAND.COM所提供的功能来使用子程序。

COMMAND.COM 实际上是一个程序，要运行 COMMAND.COM，只要在 DOS 提示符下键入：

```
A>COMMAND <Enter>

Microsoft(R) MS-DOS(R) Version 5.00 from AST RESERCH INC.
        (C)Copyright Microsoft Corp 1981-1991
A>
```

由于上述命令不带任何参数，所起的作用是把辅助命令处理程序调入内存，并建立了一个 DOS 环境的副本。如果用 SET 命令来修改 DOS 环境区，所修改的只是 DOS 环境的副本而不是原件，当回到主命令处理程序时，辅助命令处理程序建立的环境副本自动消失。COMMAND.COM 有若干参数可供选择，其中与批处理程序有关的是 [/C String]。这项参数中的 String(字符串)如果是 DOS 命令或批处理程序名，则该命令或批处理程序将被解释执行，与在提示符下输入 DOS 命令一样。例如：

```
A>COMMAND/C dir b:
```

将把辅助命令处理程序调入内存，并执行 dir b:命令。再如：

```
A>COMMAND/C batch file name
```

即把辅助命令处理程序调入内存后，执行批处理程序。对于低于 DOS 3.3 的 DOS 版本，只要在批处理程序内把 COMMAND.COM 与 EXIT 命令结合起来使用，就可以调用子程序，并且能使子程序执行结束后，返回主程序并执行下一个语句。主程序中调用子程序的语句为：

```
COMMAND/C batch file name
```

参数/C 告诉 COMMAND.COM 去执行后面的批处理程序，为了使被调用的批处理程序执行完毕之后，控制能够返回到主程序，子程序(被调用的批处理程序)的最后一个语句必须是 EXIT。批处理程序 SUB3.BAT 和 SUB4.BAT 说明了这种用法。主程序 SUB3.BAT 在运行过程中调用了子程序 SUB4.BAT，SUB4.BAT 执行完毕后，返回 SUB3.BAT 并继续执行后继语句。

批处理程序 SUB3. BAT

REM SUB3. BAT	注释文件名
ECHO OFF	关闭 ECHO
REM This is the main program	注释行
REM It will call the subroutine	注释行
REM on the next line	注释行
COMMAND/C SUB4	利用辅助命令处理程序调入子程序 SUB4 这种方法的作用与 DOS3. 3 或高于 DOS 3. 3 版本所支持的 Call 命令是完全一样的
ECHO If control returns to SUB3. BAT You will	
ECHO see this message	显示信息

批处理程序 SUB4. BAT

REM SUB4. BAT	
ECHO This is the next TO last	
ECHO line in SUB4. BAT	
EXIT	执行完 SUB4. BAT 后,EXIT 命令使控制返回 SUB3. BAT。从 DOS3. 3 开始,不需此命令。

运行后的屏幕显示为:

```
C>sub3

C>REM SUB3. BAT

C>ECHO OFF

C>REM SUB4. BAT

C>ECHO This is the next TO last
This is the next TO last

C>ECHO line in SUB4. BAT
line in SUB4. BAT

C>EXIT
If control returns to SUB3. BAT
you will see this message

C>
```

可以看出,执行完 SUB4. BAT 后,返回了主程序 SUB3. BAT,并继续执行 COMMAND/C SUB4 的后继语句。如果所用的是 DOS3. 3 或更高的版本,用命令行:

Call SUB4

代替 COMMAND/C SUB4 即可，同时应把 SUB4. BAT 中的最后一行 EXIT 去掉。

5.2 批处理程序中的子程序应用实例

批处理程序中子程序的功能很强，用下面的实际例子说明。

5.2.1 利用子程序复制修改过的文件

一些软件，如电子工作表软件 Lotus release 3，文字处理软件 Microsoft Word 等，所建立的文件，从硬盘上调入内存，修改后（或不修改）并重新以相同的文件名存盘后，驻留盘区原文件的文件名不变，扩展名自动变为. BAK，这样可以把重新存盘的文件与原文件区分开，为制作已修改文件的备份及删除修改前的文件提供了方便。

例如硬盘上存有文件 BOOK. BAK，ACCOUNT. BAK，TOTALS. BAK，还有 BOOK. DOC，ACCOUNT. CMP 和 TOTALS. WK，说明前三个扩展名为. BAK 的文件是后三个文件修改前的版本，现在用批处理程序 TOA. BAT 及其子程序 TO. BAT，把三个文件修改后的版本复制到软盘上。

批处理程序 TOA. BAT

```
@ ECHO OFF
REM TOA.BAT
IF (%1)==( ) GOTO END                  如果没有输入实际参数(准备复制的文件名)，退出批处
                                       理程序
:START                                 循环体的起始标号
    SET FILE=%1                        把准备复制的文件名赋给环境变量 FILE，储存于 DOS
                                       环境里
    SET DRIVE=A:                       把目标软盘驱动器名赋给环境变量 DRIVE，储存于
                                       DOS 环境里
    CALL TO.BAT                        调用子程序 TO.BAT，来完成文件复制工作
    IF ERRORLEVEL 1 GOTO END           如果文件 TO.BAT 在复制过程中出现错误，则退出
    SHIFT                              左移实际参数
    IF (%1)==( ) GOTO END              如果实际参数已用尽，则结束批处理程序的运行
GOTO START                             转向循环体的起始标号
:END                                   标号行
ECHO FINISHED                          告诉用户批处理程序运行结束
SET DRIVE=                             使环境变量复原
SET FILE=
```

运行批处理程序 TOA. BAT 只要输入：

```
C>TOA BOOK ACCOUNT TOTALS
```

TOA. BAT 运行时，首先检查作为实际参数的文件名是否输入，如果一个也没有输

入则停止运行。如果输入内容与上面命令行相同，则把第一个文件名 BOOK 赋给环境变量 FILE，把字符串 A：赋给环境变量 DRIVE。然后调用子程序 TO.BAT，由于是在 DOS 4.0 下运行 TOA.BAT，因此直接使用了 CALL 语句，TO.BAT 执行结束即返回到 TOA.BAT 中，然后由 SHIFT 语句把下一个文件名赋给形式参数%1，在循环体内继续循环，这样不断执行下去，直到把最后一个文件复制完为止。

子程序 TO.BAT，它以@ECHO OFF 开始，这是因为调用批处理程序时，总是恢复 DOS 的缺省状态即回显处于 ON 状态，尽管在主程序的开头已有语句@ECHO OFF，但是调用子程序并开始执行时，会把 ECHO 重新置于 ON 状态，为了不使所执行的命令显示在屏幕上，子程序 TO. BAT 应该以@ECHO OFF 为起始行。接着对扩展名分别为.DOC，.CMP 和.WK 的文件进行复制。由于所要复制的文件扩展名都在子程序里，因此在执行 TOA.BAT 命令行的实际参数中，不必包括扩展名。然后检查文件是否存在，如存在则复制到 A 盘上，并用 DOS 管道，把 One file(s) copied 的信息输出到 NUL，这样在屏幕上就不会显示出来。接着用两个 IF 语句检验复制是否成功：

```
IF EXIST %FILE%.DOC
IF NOT EXIST %DRIVE% %FILE%.DOC ECHO COPY ERROR
```

只有当两个 IF 语句都为真时，才能说明出现复制错误，此时应听到响铃声。在这里使用两个 IF 语句而没有用 ERRORLEVEL 值来检验是否复制成功，因为某些 DOS 版本的 XCOPY 命令不能正确地返回 ERRORLEVEL 出口码，有时即使复制失败，返回的 ERRORLEVEL 出口码仍为零。两个 IF 语句检验一个文件是否在 A 盘 C 盘都存在，可以保证检验结果的可靠性。下面是子程序 TO.BAT。

批处理程序 TO.BAT

```
@ECHO OFF
REM TO.BAT
:DOC
IF EXIST %FILE%.DOC XCOPY %FILE%.DOC %DRIVE%>NUL
IF EXIST %FILE%.DOC IF NOT EXIST %DRIVE%%FILE%.DOC
   ECHO COPY ERROR
IF EXIST %FILE%.DOC IF NOT EXIST %DRIVE%%FILE%.DOC
   GOTO ERROR
IF ERRORLEVEL 1 GOTO ERROR
IF EXIST %FILE%.DOC FOR %%J IN (%FILE%.DOC) DO
   ECHO COPIED.....%%J TO %DRIVE%
   :CMP
        ... ...
        (处理*.CMP 文件程序段起始标号行，该程序段用来处理.CMP 文件。整个过程与处
    理.DOC 文件的程序段完全相同，对于其它文件，程序段都是相同的，这里略去)
:ERROR
ECHO XCOPY OF FILE %FILE% WAS NOT COMPLETED.
ECHO BATCH FILE TERMINATING
```

```
ECHO FILES REMAINING TO BE COPIED ARE LISTED BELOW.
ECHO A MAXIMUM OF NINE ARE LISTED
ECHO %FILE%, %2, %3, %4, %5, %6, %7, %8, %9
ECHO
GOTO END
:END
REM FINISHED COPY
```

5.2.2 利用子程序把软盘上的文件复制到硬盘上的不同子目录中

根据用户要求把存在于软盘上的大量文件复制到硬盘上不同的目录里，是经常要做的工作，在实现时利用批处理程序及其子程序，可以提高工作效率。

假设准备把软盘上扩展名为.DOC，.CMP，.WK 以及.ALL 的各类文件分别复制到硬盘不同的子目录内。首先用文件 GET.BAT 把软盘上的所有文件复制至 C：\A 子目录里：

批处理程序 GET.BAT

```
C:                      保证计算机当前工作驱动器是 C
CD\A                    进入 A 子目录
XCOPY A:                把 A 盘所有文件复制到 A 子目录
ECHO Y | del A:         删除 A 盘上所有文件。
```

为了自动回答命令 del A：产生的 DOS 信息

"All files in the directory will be deleted,Are you sure?",

GET.BAT 最后一行使用了"ECHO Y |"是为了自动提供了字母 Y，文件执行到这里就不必等待用户回答而可以不间断地执行。如果有多张软盘，应运行 GET.BAT 多次，直到把所有文件都复制到 C：\A 子目录里为止。文件 FORMA.BAT 是主程序，它首先把文件名赋给环境变量 FILE，把文件将要进入的子目录赋给环境变量 TO。

批处理程序 FORMA.BAT

```
@ECHO OFF
REM ==========================
SET FILE=*.DOC                  设置环境变量
SET TO=C:\WORD\NORMAL
CALL SUBROUTINE                 调用子程序，把文件复制到指定的子目录里
IF ERRORLEVEL 1 GOTO END        如果复制失败，退出批处理程序
REM ==========================
    ... ...
```

（对于.CMP 等其它文件，只是重复上述两个 REM 语句之间的过程，区别是两个环境变量的值不同）

```
:END
```

```
SET FILE=                    使环境变量复原
SET TO=
ECHO ^G^G                    响铃,通知用户文件执行完毕。
```

文件复制工作由子程序 SUBROUTINE.BAT 完成的。它首先检查准备复制的文件是否存在,如存在,则执行 XCOPY 命令;如不存在,退出批处理程序。

批处理程序 SUBROUTINE.BAT

```
@echo off
echo Testing for..... %file%
if not exist %file% goto end
xcopy %file% %to% > nul
if errorlevel 1 echo
if errorlevel 1 echo * * * WARNING * * * Problem copying %file%
if errorlevel 1 goto end
for %%j in (%file%) do echo Moved..... %%j to %to%
                                    如果文件执行到这里,说明文件复制顺利。
del %file%
:end
exit
```

第 6 章　用批处理程序配置计算机系统

本章要介绍的两个重要文件一般是由用户自己建立的，很大程度上关系到计算机的系统配置，是保证操作系统能够在计算机上正常工作的基本条件。这两个文件是：CONFIG. SYS 文件和 AUTOEXEC. BAT 文件。此外还要向读者介绍驻内存驻留程序的概念及用途。

6.1　计算机系统引导

一打开计算机，BIOS 就开始工作，首先要进行加电自检——POST(Power-On Self Test)。POST 将进行内存自检，并初始化某些芯片和磁盘驱动器以及其他标准设备。如果加电自检过程由于某种原因失败，计算机就可能完全不能工作，会向用户提供一条错误信息。这种情况可能相当麻烦。计算机主机芯片有问题或者键盘接点有故障，都可能导致加电自检过程的失败。新购置的计算机，计算机的系统配置信息储存在 CMOS 内存芯片中，如果此芯片中的信息丢失，或是同用户的设置相抵触，就会导致上述过程的失败。启动计算机时一旦发生这种情况，可以用以下方法来解决：

- 如果在计算机扩充内存、改换插件或是改变配置后出现上述问题，应该用系统软盘重新启动系统，重新进行配置。
- 如果没有改变计算机的配置，出现了上述的问题，最好也用系统设置软盘重新启动一下。因为一些诸如电源故障等原因所引起的加电自测失败也会导致计算机丢失系统设置信息。

一般来说，计算机提供的错误信息会告诉用户，重新启动系统设置程序是否会有效地解决问题。

- 如果问题不是由系统配置引起的，按 F1 键，一次不行可以多试几次。
- 也可以先关闭计算机，30 秒后再打开。有时是由于某些随机产生的问题影响计算机工作。关闭一段时间，系统可能正常工作。

如果上面几个办法均没有成功，应该根据提示信息查找原因，或是考虑修理机器的软硬件。

加电自检(POST)以后，引导记录被装入内存，并开始执行。此时计算机开始引导 DOS。这个过程叫做自引导指令过程。这个过程的任务就是读磁盘。如果 A 驱动器中是 DOS 系统盘，将从这个驱动器上读取 DOS。如果 A 驱动器中没有 DOS 盘，那么自引过程将从第一个硬盘，通常是 C 盘上读取 DOS。一旦它从磁盘上找不到 DOS 系统或是磁盘出现故障，用户便会得到这样的提示信息：

Not-system disk(没有系统盘)

随后计算机会提示：

Insert a system disk in drive A（插入系统盘）

或者会自动引导到功能有限的 Basic 系统。究竟是前者还是后者取决于计算机的类型。

读者也许会问为什么操作系统不直接固化进计算机里呢？有些计算机确实固化了操作系统。问题是 DOS 版本在不断地更新，用户在每一次新版 DOS 出现之时都去修改固化的 DOS 是相当麻烦的，所以 DOS 固化并不现实。

IBM PC 及其兼容机的操作系统由三个基本程序组成的，即：IBMBIO. COM，IBMDOS. COM 和 COMMAND. COM。IBMBIO. COM 和 IBMDOS. COM 是隐含文件。如果用的是非 IBM DOS，上述文件通常被称作：SYSBIO. COM，SYSDOS. COM 和 COMMAND. COM，前两个文件用 dir 命令看不到。IBMDOS. COM 和 SYSDOS. COM 用于输入输出操作，COMMAND. COM 是命令处理程序。

引导程序将 IBMBIO. COM 和 IBMDOS. COM 两个隐含文件读至内存中，并将控制传给 IBMBIO. COM，它负责将硬件设备初始化，这时计算机所有驱动器的指示灯都会亮，点阵打印机的打印头移到左边。

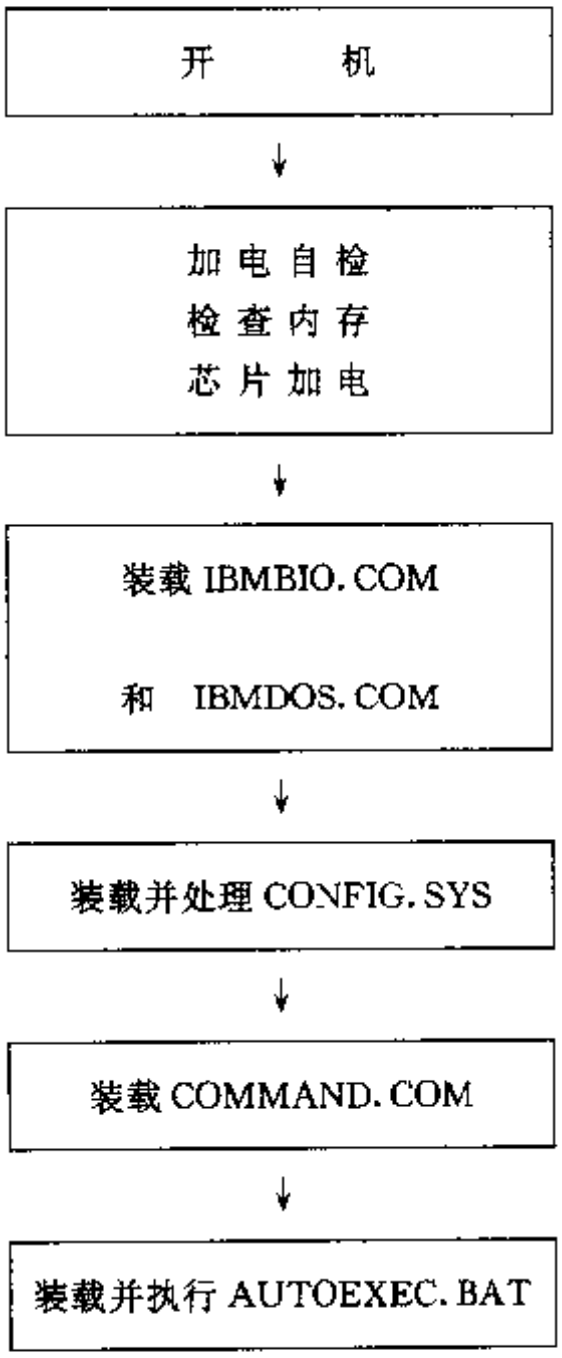


图 6-1　开机后计算机执行步骤

IBMBIO.COM 还要执行解释 CONFIG.SYS 并设置系统环境的任务，CONFIG.SYS 是 CONFIGure the SYStem（配置系统）的缩写。同 AUTOEXEC.BAT 文件一样，CONFIG.SYS 文件必须在引导盘的根目录中。有所不同的是 CONFIG.SYS 文件所包含的是些独特的命令，其中大多数命令是 CONFIG.SYS 文件的专用命令。之所以要把这些命令介绍给读者，是因为它们虽然是 CONFIG.SYS 文件所独有的命令，但同批处理命令很相似。

在系统环境设置完成之后，控制被传递给 IBMDOS.COM。这个文件所负责装载的程序控制着信息在计算机和其它硬件设备的输入输出，如驱动器和打印机等。IBMDOS.COM 完成任务后，COMMAND.COM 开始工作后，要寻找一个特殊文件—AUTOEXEC.BAT。这个文件必须放在根目录中才会被找到。一旦找到就开始执行。这个特殊批处理程序具有优越的条件，可以一次性迅速处理许多命令，比如设置日期、时间和装载内存驻留软件。

图 6-1 是开机后计算机所执行步骤的流程图；

图 6-2 显示的是计算机运行程序时 RAM 的设置。

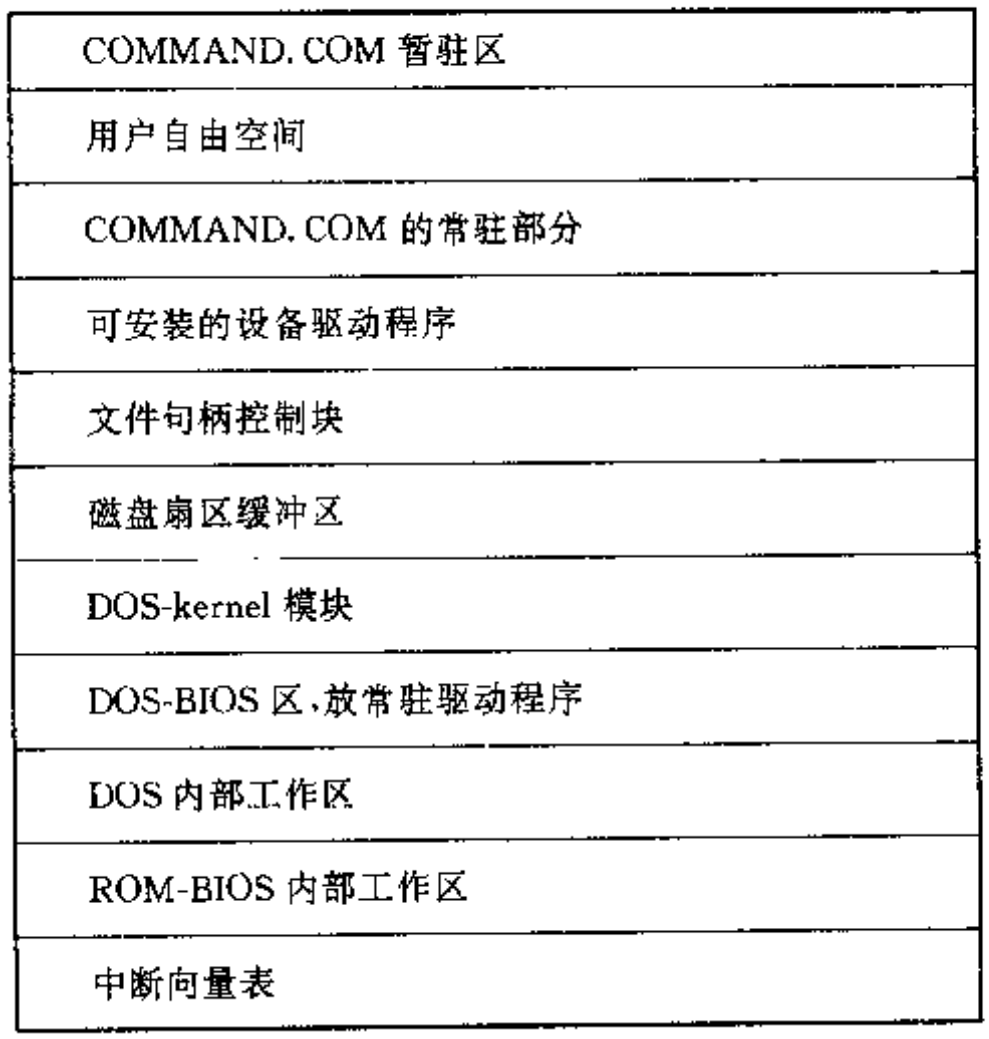


图 6-2　DOS 内存映象

6.2　系统配置文件 CONFIG.SYS

系统配置文件 CONFIG.SYS 是用来控制计算机系统配置的，是一个 ASCII 文件。系统配置文件同批处理程序很相似，通过文件中的一系列命令来控制内存的分配，为了使

计算机在执行各项不同的工作时协调一致，用户应该建立这个文件。如果在引导盘根目录里找不到这个文件，操作系统就跳过这一步去寻找并装载 COMMAND.COM 文件。

CONFIG.SYS 同批处理程序一样，必须是 ASCII 文件。建立 CONFIG.SYS 文件时，每一行只允许有一个命令。同批处理程序不一样的是，CONFIG.SYS 文件所使用的命令只能用于该文件中，而不适用于其他任何文件。原因很简单，CONFIG. SYS 程序是在 COMMAND.COM 文件调入内存之前执行的，执行后内存的使用作出了某些规定。一旦 COMMAND. COM 文件调入后，便不可能修改 CONFIG.SYS 所设定的计算机配置。由于 COFIG.SYS 不是批处理程序，所以它不能使用批处理命令。

在 DOS3.0 版本中，CONFIG.SYS 程序中有 9 个子命令，其中大多数同样适用于低版本 DOS。这些子命令大多可以以任意的顺序出现于 CONFIG.SYS 程序中。这 9 个命令中，最常用的只有三个：BUFFERS(缓冲区数目)=；FLLES(文件数)=；DEVICE(设备)=。下面逐一讨论 CONFIG.SYS 程序中的 9 个子命令。

6.2.1 CONFIG.SYS 程序的子命令

BREAK=ON/OFF(中断)

通常情况下，操作系统在进行标准输出输入操作，比如打印或访问通讯设备时，要检查系统是否可以接受 Ctrl-break。缺省设置是 BREAK=OFF，如果用户设置 BREAK=ON，那么操作系统会在用户需要时用 Ctrl-break 来中断当前程序的执行。实际上设置 BREAK=OFF 是多余的，因为缺省时，DOS 会自动按 BREAK=OFF 处理。所以我们向读者建议，在 CONFIG.SYS 程序中设置 BREAK=ON。

BREAK 命令可以从键盘上键入和执行，也可以在批处理程序中执行，就是说可以在 CONFIG.SYS 程序以外执行，这样的命令在 CONFIG.SYS 程序命令中只有少数几个。在 CONFIG.SYS 程序之外使用该命令时的语法格式同在 CONFIG.SYS 中使用时一样。从键盘上键入 BREAK 或是在批处理程序中执行 BREAK，计算机将会显示该命令的当前状态，而在 CONFIG.SYS 中却不会。

BUFFERS=#(缓冲区)

DOS 在内存中建立一个特殊的存储空间，用来存放从磁盘中读取的磁盘信息，这个区域就叫做缓冲区。如果计算机在运行程序时需要反复调用同样的磁盘信息，DOS 可以迅速从缓冲区中提供这些信息，比从磁盘上读取快得多。在某种程度上，计算机所设置的缓冲区越大，磁盘访问系统也就越快。另一方面，由于缓冲区占用的是内存空间，它所占空间越大内存剩余空间便越小，可用于其他程序的空间也就越小。

缓冲区不只是在读磁盘时可以起到节省时间的作用，在进行磁盘写入时同样可以节省时间。DOS 在向磁盘某一扇区写入一个记录之前，必须先读这个扇区。如果缓冲区足够大，就可以在读写磁盘信息时节省时间。在 BUFFERS=# 命令中“#”可取 1—99 中的任一数字。BUFFERS 命令的缺省值是 2—15 中的任一数，这取决于计算机操作系统版本以及系统配置情况。每设置一个缓冲器数值，DOS 将分配给它 512 字节的内存空间，这个数值正好同磁盘每个扇区的字节数相同。再加上内部一些消耗，每个缓冲区要占去 528 字

节的空间。设置缓冲区的值为 10，缓冲区则为 5280 字节，即 5K 多一点。

比较大的缓冲区可以极大地提高计算机的运行速度。尤其是含有大量数据信息的程序，需要一遍又一遍反复地从磁盘上读取相同的数据，这时大容量的缓冲区和高速缓冲程序的作用比较明显。但对于有些软件来说，缓冲区的作用可能不是那么大。比如 Lotus 1-2-3，运行时把工作表的内容读到内存中，并在内存中存放全部数据，不需要从磁盘上反复读取处理同样的信息，这时缓冲区的用处就不大了。缓冲区的另一个作用是，DOS 每次从磁盘上读取文件的时候，先要寻找文件所在的目录并存入内存，有些目录文件相当多，有些目录有着相当复杂的子目录结构，这时缓冲区的作用就相当明显了。

内存驻留程序件的存在，使得内存空间非常有限。如果缓冲区设置过大，就可能因为占去的内存空间太多而降低程序的执行速度。在实际应用中，许多商用程序要求计算机缓冲区数目设置为 20。用户如果想获得一个比较理想的缓冲区大小配置，只有根据实际情况花时间去摸索。通常情况下，除非拥有一个高速缓冲程序，计算机的缓冲配置不妨设为 20。用户应特别注意，有些版本的 DOS，如果缓冲区设置太大，会极大地降低 DOS 的运行速度，这是不可取的。

DOS4.0 在缓冲区的应用方面有了很大的改进。在 DOS4.0 以上操作系统环境下，DOS 可以在找到目标扇区之前，连续读取 11 个磁盘扇区，把这些额外的信息存入缓冲区。这一过程叫先行查找。早期的 DOS 版本最多仅可以设置缓冲区为 99，在当时这个数目已经足够了，如果再多就会大大影响计算机的运行速度。由于机器内存的大大增多，DOS4.0 允许设置的缓冲区数目为 999，并且可以将这些缓冲区通过 a/x 开关功能放到 DOS 扩展存储器中。

DOS BUFFERS 命令实际上就是一个简单的高速缓冲程序。如果用户的计算机打算装载其它高速缓冲程序，应该把计算机的缓冲区数目设置得低一些。这样做可以避免在硬盘上建立两个高速缓冲程序。一般在装载这种软件的时候，它会提示用户应该怎样设置计算机的缓冲区。用户应当按照提示在自己计算机的 CONFIG.SYS 文件中设置适当的缓冲区数目，不然系统会自动按缺省值 15 来处理。

既然 DOS BUFFERS 本身就是缓冲程序，为什么还有必要使用别的缓冲程序？这是因为 DOS 从磁盘上读取数据的时候，先要在缓冲区中查找内存中是否有相同的数据。所采用的方法很简单，过程却很麻烦，从第一个缓冲区开始，一个一个连续查找，直至找到为止，即使缓冲区中根本没有所需要的数据，也会查遍所有的缓冲区。如果 DOS 每次读取的都是不相干的数据信息，缓冲区不但起不到它应有的作用，反而会影响计算机的运算速度。这也是 DOS 缓冲区数目不宜设置过高的原因之一。

高级高速缓冲程序可以避免上述问题。大多数高速缓冲程序建立的缓冲空间多达几兆字节，DOS 根本无法寻找所有的空间。为便于 DOS 查找磁盘信息，缓冲程序根据自己所含有的信息建立了一个索引。当 DOS 需要读取信息时，只要检查一下索引，就可马上知道是否存在所需要的信息，在什么区域，这就大大简化了缓冲过程。

COUNTRY(国家)=#

COUNTRY=# 子命令是用于控制时间和日期的显示格式。字符"#"代表某一国代

码，表 6-1 给出了所有国家的代码，并附有相应的 KEYBxx 缺省值。DOS 默认状态是美国格式。

表 6-1 COUNTRY 命令的国家代码

国 家		代 码	KEYBxx
United states	美 国	001	U S
Canada（法语）	加拿大	002	C F
Latin America	拉 美	003	L A
Netherlands	荷 兰	031	N L
Belgium	比利时	032	B E
France	法 国	033	F R
Spain	西班牙	034	S P
Italy	意大利	039	I T
Switzerland	瑞 士	041	SF,SG
United Kingdom	英 国	044	U K
Denmark	丹 麦	045	D K
Sweden	瑞 典	046	S V
Norway	挪 威	047	N O
Germany	德 国	049	G R
Australia（英语）	澳大利亚	061	----
Japan	日 本	081	----
Korea	韩 国	082	----
People's Republic of China	中 国	086	----
Taiwan	台 湾	088	----
Asia（英语）	亚 洲	099	----
Portugal	葡萄牙	351	P O
Finland	芬 兰	358	S U
Arabic	阿拉伯	785	----
Hebrew	以色列	972	----

DEVICE= 格式

每台个人计算机都有许多外部设备，最常见的有：软盘驱动器、硬盘、打印机、绘图仪、鼠标器等。每台设备都需要一个设备驱动程序来保持同计算机的通讯联系，每台设备的驱动程序可能都不一样。设备驱动程序一般是由设备制造商提供的，用于特定的硬件设备同计算机 DOS 保持通讯。设备驱动程序一般都以.SYS 为扩展名，当然，.EXE 和.COM也可以做其扩展名。DOS 不是直接去控制计算机附属的硬件设备，而是通过特定的设备驱动程序来控制它们的。只要有适当的设备驱动程序，计算机便可以控制各种各样的硬件设备。

设备驱动程序只有写入 CONFIG.SYS 文件后才能同 DOS 联系起来。DEVICE 子命令的完整格式是：DEVICE= [d：][path]\[filename]
[.ext]

[d：]：　　驱动程序所在盘驱动器标识符
[path]：　　驱动程序所在路径
[filename]：　驱动程序的文件名

在 CONFIG.SYS 文件中用 DEVICE 子命令时，如果没有指定路径，那么 DOS 将只在引导盘根目录中寻找设备驱动程序。如果设备驱动程序在磁盘的某一子目录中，那么在 DEVICE 子命令中就应给出完整的路径名。比如，在 C 盘 UCDOS 子目录中有一个叫 M1724.SYS 的打印机驱动程序，在 CONFIG.SYS 文件中就应有这样的命令：

DEVICE=C：\UCDOS\M1724.SYS

一旦设备驱动程序同 DOS 联系起来，它便成为 DOS 的一部分，这样每次启动 DOS 时 M1724.SYS 打印机驱动程序都会装入内存。

FCBS=(文件控制块)

FCBS 命令用来确定 DOS 一次可以同时打开的文件控制块数目。格式是：

FCBS=m,n

m 所规定的是 FCBS 可以同时打开的文件总数，缺省值是 4，数值设置的范围是 1—205；n 所规定的是可以不被关闭的文件数，也就是说，如果程序试图利用文件控制块打开的文件超过 m 所规定的范围，那么可以不被自动关闭的文件数为 n，缺省值是 0，取值范围是：m—255。

FILES=#

FILES 命令用来控制整个系统一次可同时打开文件的数目。缺省缺值是 8。对于许多应用程序来说，这个数值远远不够。每一过程可一次性打开文件的最高数为 20。这 20 个文件包括 DOS 为标准 I/O 设备预留的 5 个文件：标准输入、标准输出、标准错误、辅助输出输入和标准打印机等 5 个文件。有些重要的应用程序，比如 dBase，需设置 FLLES 的值为 20。还有些程序具有暂时退出当前运行程序进入另一个应用程序的功能，比如，Lotus 1-2-3 软件中的“/system”功能，Wordstar 中的“Run a program”功能，Basic 中的“shell”功能，以及 Microsoft word 中的“Escape transfer run” 功能。另外，FILES 命令还允许有“子处理”。外壳进程允许 DOS 在暂时进入另外的应用程序时再打开 20 个文件。实际上，虽然 FILES 命令允许打开最多文件数为 20，利用“外壳进程”DOS 最多可同时打开 40 个文件。

如果用户设置 FILFS 值大于缺省值 8，那么超过 8 的部分每个文件管理字要占用 48 字节的内存。可设置 FILES 的数目最大为 99。一般来说，可设置 FILFS 值为 20。如果需打开更多的文件，计算机中的提示信息将会告诉用户：No free file handles，这时必须修改 CONFIG.SYS 文件中的 FILES 命令，增加 FILES 数目，然后重新引导 DOS。由于超过 8 个之后每个文件只需 48 字节的内存，不会占用太多的内存。

LASTDRIVE=#

这个命令是 DOS3.0 中增加的，用于规定 DOS 可拥有多少个磁盘驱动器。“#”代表

系统中的最后一个磁盘驱动器的盘符，比如对于 XT 系列机来说，最后一个磁盘驱动器是 C。此命令只设置上限，DOS 会自动跟踪实际设置的磁盘驱动器盘符。请注意，在 LASTDRIVE 命令中，代表驱动器的字母后没有冒号。要提醒用户的是，在所限定的磁盘驱动器中，必须包括那些虚设的驱动设备。包括：RAM 磁盘，非常规的磁盘驱动器，设备驱动程序所支持的磁带驱动器，以及在硬盘上划分的逻辑磁盘驱动器等，此外还有 Subst 和 Join 命令建立的“伪”驱动器。

如果不做设置，LASTDRIVE 的缺省值是 E。不过，某些软件可以不顾 LASTDRTVE 的设置而增加更多的驱动器。最典型的例子是，计算机联网软件可以定义多于 LASTDRIVE 所设置的驱动器。

SHELL=

前面讲了，DOS 命令处理程序是 COMMAND. COM 文件。在实际应用中，可以自行开发一个命令处理器来取代 COMMAND. COM，比如 Command. Plus，4DOS，Unix 等。在 CONFIG. SYS 文件中，可以用 SHELL 另外指定一个命令处理器来取代 COMMAND. COM。同设备驱动程序一样，用户定义的命令处理程序不在根目录中，在 SHELL 命令中必须指明它所在目录的全称路径。SHELL 命令也可用作扩展 DOS 环境。

有些硬件设备，要求从软盘上引导 DOS，不能从硬盘上引导。比如早期的 Bernoulli boxes。对于这类设备，必须把引导软盘始终放在 A 驱动器中，以便从 A 盘上引导 DOS。

操作系统占去了相当一部分内存。用户在实际应用中，经常会遇到内存不够用的情况，这或许是因为用户的计算机内存少于 640K，也可能是内存中的驻留软件过多，这是个很难对付的问题。用户会因为内存不够用而不能运行某些应用程序，也会因内存占用过多而影响其它应用程序的运行。DOS 本身可以解决这个问题。DOS 可以使自身的一部分暂时驻留内存，如果某个应用程序需要这一部分内存空间，就会自动覆盖 DOS 的暂驻部分占用的内存。

退出覆盖 DOS 暂驻部分的应用程序后，DOS 需要把 COMMAND. COM 文件重新读入内存。所以 COMMAND. COM 文件同其它两个系统文件不一样，不是隐含文件。一般来说，DOS 将仍从原来的引导盘上重新装载。可以在引导盘 CONFIG. SYS 文件中设置这样的命令：

SHELL=C：\COMMAND. COM

用这样的方法，可以迫使 DOS 从硬盘或是其它任何地方重新装载。很多计算机病毒专门侵蚀 COMMAND. COM 文件，根目录中的 COMMAND. COM 文件尤其容易被病毒感染。为了有效地保护 COMMAND. COM 文件，可对 SHELL 命令做一下修改：

SHELL=C：\DOS\COMMAND. COM

修改以后，需要将 COMMAND. COM 文件存放在 C：\DOS 子目录中。当然，这时在根目录中已不需要 COMMAND. COM 文件了。有趣的是，许多用户仍旧在根目录中保留 COMMAND. COM 文件，以此来迷惑计算机病毒。其实，这种做法只能迷惑一些简单的病毒。如果硬件设备要求从软盘上引导 DOS，可以将 COMSPEC 变量指向 C 驱动器，以

使 DOS 可以再一次从硬盘上装载 COMMAND.COM。

如果使用的是 DOS 2.X 版本，SHELL 命令将因 DOS 的一些固有缺陷发生错误而不能正常工作。通常情况下，第一次使用 SHELL 命令不会有什么问题，但以后 SHELL 命令可能再不起作用了。解决这个问题的最好办法，当然是把操作系统换成 DOS3.0 以上版本。此外也可以采取以下步骤，从 DOS 2.X 的引导盘上重新装载 COMMAND.COM：

(1) 正常引导 DOS。

(2) 把 COMMAND.COM 文件拷贝到要重新装载的磁盘上，但不必在每次重新装载时都进行这一步。

(3) 进入 COMMAND.COM 文件所在盘。比如想从 C 盘上重新装载 COMMAND.COM，就进入 C 盘。

(4) 执行命令 COMMAND C：。

如果想从其它驱动器磁盘上重新装载 COMMAND.COM，就采用相应的驱动器名称。要是把 COMMAND.COM 存放在某一子目录中，需执行：

COMMAND C：\子目录

以上的方法虽然可行，但不如更换 DOS 版本有效。

在 DOS 提示符下不能运行 SHELL 命令。根据 SHELL 的设置找不到 COMMAND.COM的所在，只有 COMSPEC 指出了 COMMAND.COM的路径。在 DOS 提示符下用 COMSPEC 命令可以改变 DOS 寻找 COMMAND.COM 的路径：

SET COMSPEC=C：\COMMAND.COM

或是：

COMSPEC=C：\COMMAND.COM

同 SHELL 命令一样，COMSPEC 命令在 DOS2.X 下不能正常工作。除了上述的作用之外，SHELL 命令还可用于扩展 DOS 环境。以后将详细介绍。

STACKS=(堆栈)

STACKS 命令用来控制堆栈数目和大小。命令格式是：

STACKS=m,s

m 为堆栈数，可以是 0，也可以是 8—64 之间任一数值，缺省值是 0。s 定义的是每个堆栈的大小，取值可以是 0，或 32—512 之间任一数值，缺省值是 0。STACKS 命令的缺省状态是“0,0”，表示 DOS 不需要堆栈支持。这种状态为大多数系统所采用。当 DOS 需要堆栈支持，而它又没有被设置时，计算机会给出错误信息加以提示。设置：STACKS=512，意味着通知 DOS 装入 6 个 512 字节的堆栈。

DOS4.0 在 CONFIG.SYS 文件中新增加了两个命令。这两个命令只适用于 DOS4.0 以上版本：

INSTALL=

INSTALL 允许不通过 AUTOEXEC.BAT 文件，而直接从 CONFIG.SYS 装载某些

内存驻留程序。在 CONFIG. SYS 中，不管各种命令的排列顺序如何，用 INSTALL 来装载的程序，总是先于其他内存驻留程序。以下几个文件是由 INSTALL 支持的驻内存软件：

- FASTOPEN. EXE
- KEYB. COM
- NLSFUNC. EXE
- SHARE. EXE

以 SHARE. EXE 为例，说明一下如何在 CONFIG. SYS 中运用 INSTALL 命令装载内存驻留程序：

INSTALL=C：\DOS\SHARE. EXE

CONFIG. SYS 文件在 AUTOEXEC. BAT 之前进行解释，这时还没有定义明确 DOS 路径。所以，命令中需给出要装载文件的路径名全称，包括文件名的扩展名。

SWITCHES=

CONFIG. SYS 中的 SWITCHES=/K 命令行将通知 DOS，即使用户已安装了增强功能的键盘也应使用常规键盘的功能。命令的用途是，用户安装了增强功能键盘，而应用程序又不识别增强功能的键盘，在这种情况通过这条命令使 DOS 仍启用常规键盘。

表 6-2 中列出了 CONFIG. SYS 的专用命令，表 6-3 为常见的 CONFIG. SYS 设置情况。

表 6-2　CONFIG. SYS 文件子命令一览表

命　令	含　义
BREAK=ON/OFF	控制 DOS“Ctrl-Break”
BUFFERS=#	控制磁盘缓冲区
DEVICE=	保持 DOS 与外部设备的联系
COUNTRY=	控制 DOS 如何显示日期时间
DEVICEHIGH	向扩展存储器中装载设备驱动程序
DOS=HIGE/LOW	DOS=HIGH 在扩展存储器中装载大部分 DOS
DOS=UMB/NOUMB	UMB 允许驻内存软件占用扩展存储器
FCBS	确定 DOS 可同时打开的文件控制块数目
FILES	控制系统一次可打开的文件数
INSTALL	在 CONFIG. SYS 文件中装载某些驻内存软件
LASTDRIVE=	设定 DOS 可访问的最后一个磁盘驱动器
REM	在 CONFIG. SYS 中起注释作用
SHELL=	扩充环境，定义一个命令处理程序代替 COMMAND. COM
STACKS	确定堆栈数和大小
SWITCHES= /K	通知 DOS 仍使用常规键盘

表 6-3 常见的 CONFIG.SYS 设置情况

在引导盘根目录中建立 CONFIG.SYS 文件
在 CONFIG.SYS 文件中一般应有以下命令： BREAK=ON BUFFERS=20 FILES=20 LASTDRIVE=E
最常用命令： BREAK,BUFFERS,DEVICE,SHELL
最不常用命令： FCBS
用 COUNTRY= 改变国家设置，默认状态为美国

6.2.2 设备驱动程序

编写设备驱动程序是一项既复杂又困难的工作，大多数内存驻留程序如果被写成设备驱动程序，就可能会更有效地发挥作用，但即使是比较有能力的软件人员也很难胜任这项工作。

有两个设备驱动程序已成为 DOS 的一部分，应该引起用户的特别注意。这两个设备驱动程序是：ANSI.SYS 和 VDISK.SYS。下面分别介绍：

ANSI.SYS

这个驱动程序最先出现在 DOS2.0 版本中，以后一直成为 DOS 必不可少的部分。ANSI.SYS 设备驱动程序增强了 DOS 对键盘功能的充分利用和屏幕视频效果的处理。它的编写过程很复杂，本书不做过多的叙述，然而它的基本功能却很明了，就是支持增强功能键盘和视频，ANSI.SYS 如同一个简单的键盘宏程序，其语法功能要比大多数驻内存软件强得多，它可以给任何一个键定义其他键所具有的功能。比如，我们可以用键盘宏程序来定义 CTRL-A()来记录 AUTOEXEC.BAT 批处理程序，ANSI.SYS 可以执行同样的功能。ANSI.SYS 还可以定义专用键来控制屏幕，比如把屏幕上的 DOS 提示符变成其它形态。总之，利用 ANSI.SYS 设备驱动程序，可以尽可能有效地利用键盘和变换屏幕视频效果。

VDISK.SYS

VDISK.SYS 由 DOS3.0 首先采用，但它的源码首先出现在 DOS2.0 中。VDISK.SYS的作用是，在内存中划出一部分空间来模拟软盘驱动器，这个驱动器叫 RAM 磁盘。用户可以同使用其他软、硬盘驱动器一样在 RAM 盘上存取信息。RAM 盘的优点是存取速度极快，为读写一般磁盘驱动器速度的两倍。但是，一旦电源被切断，或是关闭计算机，RAM 盘上的数据将全部丢失，无法保存。显然，RAM 盘要占用一定的内存空间，减少应用程序的可用内存空间，降低计算机的运行速度。

DISPLAY.SYS,DRIVER.SYS 和 PRINTER.SYS

这三个设备驱动程序是DOS3.3中新增加的。DRIVER.SYS允许将外部磁盘驱动器定义为一个逻辑驱动器。DRIVER.SYS的基本作用是在只有3.5英寸磁盘驱动器的微机上，定义一个5.25英寸的逻辑驱动器。其它两个设备驱动程序的作用是，使DOS可以控制不同国家生产的打印、显示设备。

CEMM.SYS

CEMM.SYS首先出现在COMPAQ DOS3.2中。仅适用于COMPAQ DOS，其它版本的DOS也含相似的设备驱动程序。CEMM.SYS允许用户将多于640K的内存空间作为RAM虚拟磁盘，或是做为扩展内存。

6.3 AUTOEXEC.BAT文件

计算机启动后，DOS最后要在根目录中寻找并执行AUTOEXEC.BAT自动批处理程序。引导或重新引导计算机时，只要在引导盘根目录中有AUTOEXEC.BAT文件，DOS就会自动执行这个批处理程序。AUTOEXEC.BAT具有得天独厚的优越条件，有些指令用户在每次启动计算机时都要执行，而且直至下一次启动为止只执行一次。比如，设置路径，装载驻内存软件，建立计算机抗病毒系统等等，每次开机及重新引导计算机时都要重复执行，如果每次都由键盘输入，就太麻烦了。利用AUTOEXEC.BAT文件可以圆满而准确无误地完成这项任务。AUTOEXEC.BAT文件是个批处理程序，所以具有批处理的一切功能。它本身并没有特殊的命令，所含所有命令都是普通的DOS命令和批处理命令。

一旦DOS在引导盘根目录中找不到AUTOEXEC.BAT文件，DOS系统引导的最后一项是向用户显示当前日期、时间，并提示用户输入新的日期、时间。如果在根目录中找到了AUTOEXEC.BAT文件，而不会向用户提示时间、日期，需要说明的是，上述过程也有例外的情况。上一节介绍了CONFIG.SYS子命令SHELL，如果在CONFIG.SYS中建立：

```
SHELL=C:\DOS\COMMAND.COM
```

而C盘根目录确实存在COMMAND.COM，那么DOS不会考虑根目录中有无COMMAND.COM文件，也不管目录中是否存在AUTOEXEC.BAT文件，而是直接显示DOS版本，并且既不提示日期时间也不执行AUTOEXEC.BAT文件。类似这样的情况，在实践中还会遇到。要避免这种情况只需将SHELL命令改为：SHELL=C:\DOS\COMMAND.COM/p。老式的计算机没有内部时钟，需要在AUTOEXEC.BAT中设置日期和时间命令，以便每次引导时输入日期、时间。对于这些老式计算机，制造商们曾推出一种装有系统时钟的插件板。使用这类装有插件板的计算机，需要在AUTOEXEC.BAT文件中执行一个程序，以便将系统时钟传给DOS，也可在CONFIG.SYS文件中建立一个设备驱动程序。目前市场上销售的计算机大都有内部时钟，并可同DOS自动通讯。

AUTOEXEC.BAT可以含有一些形式参数。但某些DOS版本不允许用户在AUTOEXEC.BAT文件中使用形式参数。如果使用了，计算机会自动锁机。

6.3.1 建立 AUTOEXEC.BAT 文件

大多数系统配置是由 CONFIG.SYS 文件来完成的，也有一些需要 AUTOEXEC.BAT文件来配置装载，这一节将介绍这些配置。

1. PATH=

DOS 承认四种命令：

(1) DOS 内部命令

(2) 可执行文件(.EXE)的文件名

(3) 命令文件(.COM)的文件名

(4) 批处理程序(.BAT)的文件名

如果每当用户输入一个命令后，DOS 将首先检查它是否是内部命令，如果是则执行，如果不是则在当前目录中检查它是不是.COM 文件，如果是则执行，不是则继续在当前目录中检查它是不是.EXE 文件。最后看它是否是.BAT 文件。如果在当前目录中 DOS 找不到同输入命令名称一致的以上四种文件，DOS 就要在其它路径中按照.COM 文件、.EXE文件、.BAT 文件这个顺序寻找并执行。如果找不到，DOS 将提示错误信息：

Bad command or file name(错误命令或文件名)

DOS 只在 Path 命令所设置的路径中寻找文件。Path 命令的格式是：

PATH=C：\;目录 1;目录 2;…目录 n

在 PATH 命令中，如果在路径名前没有驱动器标识符，那么 DOS 将采用缺省驱动器。在以下的 PATH 命令中：

PATH=C：\SYS;\DOS;\CCED

DOS 将只在缺省(当前)驱动器(如 A 驱动器)上承认 PATH 命令所设置的路径。因此向用户建议，用 PATH 命令设置路径的，最好用全称路径名：

PATH=C：\;C：\SYS;C：\DOS;C：\CCED;

设置过多的路径，则许多路径都得不到 DOS 认可。因为 DOS 限制 PATH 命令的长度在 127 个字符之内，超过 127 字符的路径 DOS 将不予承认。这个问题在 DOS3.0 之前无法解决。DOS3.0 以上版本，DOS 仍要求 PATH 命令不超过 127 个字符长度，但增加了 Substitute(替代)命令。这个命令允许用一个代表驱动器盘符的字母来替代一个子目录名称。如：

SUBST D： C：\SYS1\SYS2\SYS3

上面这个表达式的意义是，在 PATH 命令中可以用“D” 来替代“C：\SYS1\SYS2\SYS3”。如果原来的 PATH 命令表达式为：

PATH=C：\;C：\SYS1\SYS2\SYS3

现在可以简化为：

```
PATH=C:\;D:\
```

这样一来PATH命令表达式变得既简短又容易读懂。通常情况下，在AUTOEXEC.BAT程序执行SUBST命令时，还没有设置任何路径。因此，SUBST.EXE文件应该在根目录中。否则，用户在执行SUBST命令时应首先进入含有SUBST.EXE程序的子目录，或是给出SUBST.EXE所在目录的完整名称。例如，SUBST.EXE文件所在目录是C:\SYS，在AUTOEXEC.BAT文件中建立：

```
CD\SYS.
SUBST D: C:\SYS\Level1\level2
```

或是：

```
C:\SYS\SUBST D: C:\SYS\Level1\Level2
```

也可以在设置了路径以后再命名SUBST命令。在DOS提示符下，从键盘上直接输入：PATH，DOS将显示当前设置的路径；如果输入：PATH；DOS将取消当前设置的所有路径，这将限制DOS只在当前目录中寻找程序文件名。如果错误地建立了一个路径，DOS只有在需要查找这个目录时才能发现错误。如果所设置的路径实际并不存在，DOS将自动忽略。

值得注意的是，在运用DOS SUBST命令时，用于替代路径名的字母，一般是当前确实存在的驱动器或逻辑驱动器的字母。比如，硬盘划分为C，D，E三个分区，那么输入下面的命令DOS将不予理采：

```
SUBST F: C:\SYS
```

在CONTIG.SYS文件中写入LASTDRIVE=n，SUBST命令将承认n以前的所有字母。一旦采用了某个驱动器字母来替代路径名，将无法继续使用那个驱动器。在AUTOEXEC.BAT中写入命令：

```
SUBST D: C:\CCED
```

之后从键盘上输入：D:〈Enter〉，实际进入的是C:\CCED这个目录。即使用PC工具等软件 同样无法进入实际上的D盘。PATH命令是个可设置变量。同其它环境变量不同的是，不必键入“SET”同样可以设置路径。和其它环境变量一样，PATH可以在批处理程序中调用。因此，可以在批处理程序中用%PATH%建立E驱动器的路径：

```
PATH=%PATH%;E:\
```

这条命令只能用在批处理程序中，从键盘直接输入将无效。

2. PROMPT=

利用DOS的PROMPT（提示）命令可显示许多用户需要的重要信息。这个命令主要是在AUTOEXEC.BAT文件中使用。缺省时，DOS提示符是“C>”，C是缺省驱动器。当然，用A驱动器引导时缺省值为“A”。直接键入：PROMPT，无论当前DOS提示符是什么，都将变为：C>或D>，E>等等。PROMPT命令中可包括所有可显示的字符串。有些字符需要用特殊的代码来表示。表6-4给出了这些特殊的代码。应当知道，用PROMPT

命令设置的任何提示符都被存入环境区，同路径名以及其它可设置变量一起显示。用户所设置的提示符越长，所占环境区越大，再加上路径名和其它可设置变量，有时就不得不扩大环境空间。

表 6-4 PROMPT 命令中的元字符

命令	屏幕显示结果
$$	美元符
$t	当前时间
$d	当前日期
$p	当前目录
$v	DOS 版本
$n	当前驱动器
$g	>
$l	<
$b	|
$q	=
$h	退回一格(将删除前一个字符)
$e	包含一个换码键
$	回 车

最常用的是 PROMPT＝PG，使 DOS 同时显示驱动器名和当前目录名。为了显示当前目录，DOS 必须首先读磁盘。若想把当前驱动器切换成软盘驱动器，却忘了在驱动器中插入磁盘或没有关闭小门，DOS 仍试图读这个驱动器中的磁盘，这种尝试失败后 DOS 会提示这样的错误信息：Abort，Ignore，or Retry？选择 Abort(放弃)则 DOS 仍试图显示用户要求的提示符，从而导致读盘再次失败。如果要选择 Abort，只有把磁盘插入驱动器关上小门，否则只好重新启动计算机。可以选择 Ignore(忽略)(有些 DOS 版本是 Fail)。选择 Fail(失败)，一次就可以摆脱错误，但如选择 Ignore，就必须试好几次。这一过程如下：

```
C：\BAT\LIU>b：

Not ready reading drive B
Abort, Retry, Fail? a

Not ready reading drive B
Abort, Retry, Fail? r

Not ready reading drive B
```

```
Abort, Retry, Fail? f

Not ready reading drive B
Abort, Retry, Fail? f
Current drive is no longer valid>c :

C : \BAT\LIU>
```

VERIFY=on/off

用户希望DOS应该具有检查核实的功能，查看已经读入磁盘的数据同原来的数据是否一致。如果有出入，则DOS应重新处理这些数据，或是用提示信息通知用户。实际上DOS在核对数据时并不是回去重新读取数据。DOS在向磁盘写入数据信息的同时，也包含了一个叫做CRC(Cyclical Redundancy Check)循环冗余检查的检查和。由于在磁盘上同时存在数据和CRC，DOS就具有两份信息。

DOS从磁盘上读取数据的时候，同时要计算CRC并同从磁盘读取的CRC进行比较。如果不一致，DOS便认为读取的数据有误。在这种情况下，DOS会反复比较几次，如果还是不对，DOS将给出这样的错误信息：

```
Abort, Ignore, or Retry?
```

当VERIFY=on时，DOS不仅仅向磁盘上写入数据，而且在写入数据后，DOS还要读这些数据，并计算新的CRC。如果新的CRC与原来同数据存储在一起的CRC一致，DOS就认为读取的数据正确。DOS并不比较磁盘数据和内存中的数据，而是比较检查和。

设置VERIFY=on，DOS在写入操作后将部分地检查磁盘上的数据。通过这一过程，DOS可能会发现某些错误，但不可能发现全部错误。但是另一方面，设置VERIFY=on会减缓磁盘操作过程。DOS首先要向磁盘中写数据，而后必须等磁盘的数据区转回磁头下面，再读这些数据，计算CRC，同以前的CRC进行比较。这个复杂的过程将花费很多时间。

VERIFY=on虽能找出一部分错误，却使整个操作过程变得很慢。那么到底是选择VERIFY=on，还是VIRIFY=off呢？。权衡利弊，我们建议选择VERIFY=on。键入“VERIFY”会显示当前VERIFY状态。

3. MODE

DOS的MODE.COM程序是用来设置PC机I/O设备参数，比如，一个台式打印机的波特率是4800bps，则应该设置这样的命令：

```
MODE LPT1 : 4800
```

如果不改变打印机的波特率，就可以将上面这个命令写入AUTOEXEC.BAT文件中。系列设备，如Star系列打印机、Roland系列绘图仪，其波特率是一样的。因此，用户可以在AUTOEXEC.BAT文件中设置这些设备的波特率。在运用MODE命令时，需注意MODE.COM所在的目录路径。最好是在AUTOEXEC.BAT文件中的PATH命令之后

再设置 MODE 命令；否则，应在 MODE 命令中写出完整的路径名称：

```
C：\DOS\MODE LPT1：4800
```

6.4 内存驻留程序

内存驻留程序就是驻留在内存中的软件，除非关机、重新引导 DOS，或是把它从内存中清除才会从内存中消失。同其它软件不同，驻内存软件在装载时要按一定的方法来执行。一般来说驻内存软件只装载一次。如果试图再次装载内存中已存在的软件，会带来一系列麻烦，另外，向内存中装载两个以上内存驻留程序时，必须按一定的顺序进行。

基于上述原因，用 AUTOEXEC.BAT 文件来装载驻内存软件就显得非常方便了。在 AUTOEXEC.BAT 文件写入的顺序，每次启动或重新引导时，它会自动按照固定顺序加载内存驻留程序进行，用户不必担心出错，也没有必要记住这个顺序。

表 6-5 常见设备驱动程序

驱动程序	功　　能
ANSI.SYS	使 DOS 具有增强功能的屏幕显示和键盘
DISPLAY.SYS	增设外部驱动器
DRIVER.SYS	显示其它国家的字符设置
EMM386.EXE	将扩充存储区变为扩展存储器，并允许在总存储器中装载驻内存软件
PRINTER.SYS	打印其它国家字符
VDISK.SYS	将一部分内存设置成模拟磁盘

6.5 计算机常规配置

前面介绍了决定计算机配置的两个重要文件：CONFIG.SYS 文件和 AUTOEXEC.BAT文件。许多用户的计算机配置是单一的，不需要变化，有一套系统配置文件就足够了。但是对于相当一部分用户来说，在计算机上要运行许多不同的程序，这些应用程序可能对计算机的配置有不同的要求。仅有一种配置不能满足用户的实际需要。举个例子说，有些应用软件比如 Lotus 2.2 需要扩展存储器，那么在用计算机的 CONFIG.SYS 文件中必须配置一个扩展存储器模拟器，而另外一些应用程序比如 Lotus 3.1 却需要扩充存储区，在启动系统时用户必须跳过扩展存储器。

6.5.1 拷贝不同的配置文件

为了适应不同的计算机配置，可以采取这样的办法：建立不同的配置文件(AUTOEXEC.BAT 文件和 CONFIG.SYS 文件)，再建立一个批处理程序用于控制在计算机引导时选择不同的配置文件，也可以在需要时利用这个批处理程序来重新引导系统，以获得相应的配置。在本书附盘中，有个 BOOT.COM 程序，利用这个程序，可以重新引导系

统。用户可以参照下面这个批处理程序，根据自己系统的软硬件资源建立相应的 CONFIG. SYS 和 AUTOEXEC. BAT 文件，有选择地进行配置。下面这个批处理程序提供选择三种形式来配置系统：

批处理程序 AUTOBOOT. BAT

```
ECHO OFF                          不显示命令行
REM AUTOBOOT. BAT                 注释
REM First test for missing parameter and branch
                                  注释
IF (%1)=( ) GOTO NOTHING          如果不输入实际参数，则转到标号 NOTHING
REM NOW test for proper parameters
                                  注释
If found,copy files and branch
IF %1==001 COPY AUTOEXEC. 001 AUTOEXEC. BAT
                                  如果输入 001，则选用 AUTOEXEC. 001 文件
IF %1==001 copy CONFIG. 001 CONFIG. SYS
                                  如果输入 001,则选用 CONFIG. 001 文件
IF %==001 GOTO BOOT               重新引导 DOS。
IF %1=002 COPY AUTOEXEC. 002 AUTOEXEC. BAT
                                  如果输入 002，则选用 AUTOEXEC. 002 文件
IF %1==002 COPY CONFIG. 002 CONFIG. SYS
                                  如果输入 002,则选用 CONFIG. 002 文件
IF %1==002 GOTO BOOT              重新导 DOS。
IF %1==003 COPY AUTOEXEC. 003 AUTOEXEC. BAT
                                  如果输入 003,则选用 AUTOEXEC. 003 文件
IF %1==003 COPY CONFIG. 003 CONFIG. SYS
                                  如果输入 003,则选用 CONFIG. 003 文件
IF %1==003 GOTO BOOT              重新引导 DOS。
REM Must be wrong parameter. Branch to message
                                  如果进行到这里，肯定输入了错误的实际参数
GOTO WRONG
: BOOT
BOOT
REM Since Computer reboots,rest of this batch file not executed
                                  注释,通知用户要重新启动计算机
: NOTHING
ECHO NO Parameter entered after AUTOBOOT
                                  在执行 AUTOBOOT 的命令行上没输入实际参数
: GOTO MESSAGES                   转向标号 MESSAGES
: WRONG
ECHO Wrong parameter entered after autoboot
                                  提示用户输入了错误的实际参数
```

```
GOTO MESSAGES                    转向标号 MESSAGES
:MESSAGES
ECHO Enter 001 for no memory resident software
ECHO Enter 002 for writing set of memory resident
     software
ECHO Enter 003 for data base set of  memory  residant software
                    给用户提供信息
```

从上面这个批处理程序中可以看出，通过不同的 AUTOEXEC. BAT 和 CONFIG. SYS文件，可以有选择地进行系统配置。

并非所有的应用程序都需要重新配置计算机系统。在运行一个程序之前，一般要先检查一下当前的配置是否适合于程序的运行，如果合适，就没有必须进行重新配置。如果不适合程序的运行，就应根据具体情况作一些修改，然后重新启动系统。通过下面的例子来介绍一下相应的批处理程序。

例如有这样两套系统配置程序：

文件 CONFIG. 001：

```
FILES=20
BUFFERS=5
```

文件 AUTOEXEC. 001

```
PROMPT=$P$G
PATH=C:\;C:\DOS
STARTAPP.BAT
```

使用这两个配置文件设置环境，给内存留下了相当多的剩余空间。从 CONFIG. 001 的内容可以看到，文件所设置的缓冲区和一次性打开的文件数都不多，而且不装载任何设备驱动程序。在 AUTOEXEC. 001 中也没有装载任何内存驻留软件。

文件 CONFIG. 002

```
FILES=40
BUFFERS=40
DEVICE=C:\MOUSE\MOUSE.SYS
DEVICE=C:\RAMDISK\RAMDISK.SYS /1000
DEVICE=C:\TOOLS\CACHE.SYS /2000
DEVICE=C:\NETWORK\PROGRAM.SYS
```

文件 AUTOEXEC. 002

```
PROMPT=$P$G
PATH=C:\;...;C:\DOS
CD\SUPERDO
SUPERDO
CD\SUPERTOO
SUPERTOO
```

```
CD\
STARTAPP.BAT
```

在这两个配置文件中，增加了一些对用户来说非常重要的环境配置。在 CONFIG.002文件中所设置的缓冲区和系统一次可打开文件数都比 CONFIG.001 中设置的要多，而且增加了几个设备驱动程序，AUTOEXEC.002 中加载了两个内存驻留软件。这样一来就会占据较多的内存。运行需要占用较大内存空间的软件也许就不可能了。

下面的两个批处理程序就是为适应不同的应用程序对计算机配置的不同要求而编写的，所支持的两个应用程序分别是：BIGAPP 和 TINYAPP。01.BAT 文件负责配置 BIGAPP 所需的环境，并且运行这个程序；02.BAT 则负责 TINYAPP。

批处理程序 01.BAT

```
@ECHO OFF                                    关闭显示
REM 01.BAT                                   注释行
REM Runs BIGAPP, a Program That
REM Needs a Lot of Memory
IF EXIST FLAG.001 GOTO OK                    若标志文件 FLAG.001 存在，说明当前设置恰
                                             当，转向标号 OK
IF EXIST FLAG.002 GOTO 002                   若标志文件 FLAG.002 存在，说明当前设置
                                             不恰当，转向标号 002
REM At This Point, Flag Does Not Exist       通知用户，若标志文件不存在，会生成一个标
REM Will Assume Files Wrong                  志文件，并拷贝适当的配置文件，重新引导系
REM Will Create Flag File, Copy Over         统生成一个 0 长度的文件，表示当前配置
REM Proper Files and Reboot                  CONFIG.002 不恰当
TYPE NOFILE > FLAG.002

:002                                         若当前配置不适当，
COPY CONFIG.001 CONFIG.SYS                   则将 CONFIG.001 和
COPY AUTOEXEC.001 AUTOEXEC.BAT               AUTOEXEC.001 拷贝成当前配置
REN FLAG.002 FLAG.001                        更换标志文件文件名
REM Create Marker File                       注释行
TYPE NOFILE > START.001                      生成一个 0 长度文件，通知 AUTOEXEC.
                                             BAT 文件，有程序等待运行
BOOT                                         重新引导 DOS
:OK                                          若当前配置适当，运
CD\BIGAPP                                    行 BIGAPP
BIGAPP
```

批处理程序 02.BAT

```
@ECHO OFF                                    关闭显示
REM 02.BAT                                   注释行
REM Runs TINYAPP, a Program That Does
```

```
REM Not Need a Lot of Memory
IF EXIST FLAG.002 GOTO OK                    若标志文件 FLAG.002 存在，说明当前设置恰当，转向标号 OK

IF EXIST FLAG.001 GOTO 001                   若标志文件 FLAG.001 存在，说明当前设置不恰当，转向标号 001

REM At This Point, Flag Does Not Exist       通知用户，若标志文件不存在，会生成一个标志文件，并拷贝适当的配置文件，重新引导系统
REM Will Assume Files Wrong
REM Will Create Flag File, Copy Over
REM Proper Files and Reboot

TYPE NOFILE > FLAG.001                       生成一个 0 长度的文件，表示当前配置不恰当
:001
COPY CONFIG.002 CONFIG.SYS                   将 CONFIG.002 和 AUTOEXEC.002 拷贝成当前配置
COPY AUTOEXEC.002 AUTOEXEC.BAT
REN FLAG.001 FLAG.002                        更换标志文件文件名
REM Create Marker File                       注释行
TYPE NOFILE > START.002                      生成一个 0 长度文件，通知 AUTOEXEC.BAT 文件，有程序等待运行

BOOT                                         重新引导 DOS
:OK                                          若当前配置适当，运行 TINYAPP
CD\TINYAPP
TINYAPP
```

用户一般都希望操作过程越简单、速度越快越好。可以看出，上面的每一个批处理程序，都要在运行程序前先检查一下计算机当前的环境配置。如果环境配置合适，就马上运行应用程序；如果不适合，就立即拷贝相应的 CONFIG.SYS 文件和 AUTOEXEC.BAT 文件到当前引导盘根目录中，然后重新引导计算机，再运行应用程序。一般来说这一过程是应该是简单易行的。若要求计算机显示当前环境配置或引导后自动运行相应的应用程序就相对复杂一些，上面的两个批处理程序 01.BAT 和 02.BAT 已给出了圆满的解决办法。

我们可以通过一个实例来进一步说明。假定计算机是由 CONFIG.002 和 AUTOEXEC.002 来配置的。在这个环境下，要运行 BIGAPP 应用程序。这是第一次启用相应的系统，所以标志文件 FLAG.002 并不存在。在 DOS 提示符下键入 01，计算机会执行 01.BAT 批处理程序。重新引导后，DOS 开始处理 CONFIG.SYS 文件，然后是 AUTOEXEC.BAT 文件，在 AUTOEXEC.BAT 文件的最后一项，要执行一个叫 STARTAPP.BAT 的批处理程序，文件的内容如下：

批处理程序 STARTAPP.BAT

```
@ECHO OFF                                    关闭显示
IF EXIST START.001 GOTO 001                  若标志文件 START.001 存在，会提示运行某个程序，转向执行 001：

IF EXIST START.002 GOTO 002                  若标志文件 START.002 存在，会提示运行某个程序，转
```

	向执行 002：
GOTO END	若两个标志文件都不存在，则退出批处理
:001	
DEL START.001	删除标志文件以便下次引导 DOS 时不会自动运行应用程序
1	运行另一个批处理程序，控制不再返回
:002	
DEL START.002	删除标志文件以便下次引导 DOS 时不会自动运行应用程序
2	运行另一个批处理程序，控制不再返回
:END	

从上面的阐述中可以知道，01.BAT 只有在计算机配置不恰当时才重新引导计算机，所以，只有当前环境设置不恰当或是计算机不清楚当前的环境设置时，才会重新启动计算机。这个例子中只涉及两种不同的环境，两种不同的应用程序，在实际应用中可能会遇上更复杂的问题。

6.5.2 软盘引导

为运行不同的应用程序而修改系统配置的方法已非常普及了。除了前面介绍的方法以外，还有一种办法可以改变计算机的配置，即为每一个应用程序制做一张特定的引导软盘。每一张引导软盘上有特定的系统配置文件——CONFIG.SYS 和 AUTOEXEC.BAT，用软盘启动计算机并不影响 DOS 从硬盘上装载内存驻留软件。只要在 AUTOEXEC.BAT 文件中装载内存驻留软件的命令前加入命令行：C：就可以了。也可以在 CONFIG.SYS 文件中用 SHELL 命令，在 AUTOEXEC.BAT 中用 COMSPEC 命令来从硬盘上装载 COMMAND.COM，这样就不必在引导后仍在 A 驱动器中放置引导盘。

6.5.3 有条件地装载内存驻留程序

不是每一次引导计算机都要装载同样的内存驻留程序。如果所需的内存驻留程序不同，就要不断地修改 AUTOEXEC.BAT 文件，给操作带来许多麻烦。为了方便起见，可以采用人机对话的方式，询问用户需要装载哪些内存驻留程序。具体办法是：在 AUTOEXEC.BAT 文件中装载内存驻留程序的命令前加入提示，询问用户如何装载内存驻留程序。可以参照书后磁盘中的 Inkey 程序。下面的批处理程序 AUTOASK.BAT 用于有选择地装置内存驻留软件，供读者参考。

批处理程序 AUTOASK.BAT

@ECHO OFF	注释行
REM AUTOEXEC.BAT File Illustrating	
REM Conditional TSR Loading	
ECHO Do You Want to Load Superkey(Y/N)	是否装载 Superkey
BE ASK "Load TSR ", YN	运行 NORTON 实用程序中的 BE ASK
IF NOT ERRORLEVEL 2 GOTO SKIP1	若回答 Y，则转到标号:skip1

```
   CD\SUPERKEY                                运行 superkey
   KEY
   CD\
 : SKIP1
ECHO Do You Want to Load PrintCache (Y/N)   若回答 Y,则有条
BE ASK "Load TSR ", YN                      件装载 PrintCache
IF NOT ERRORLEVEL 2 GOTO SKIP2
   CD\PCACHE
   PCACHE
   CD\
 : SKIP2                                    若回答 Y,则有条
ECHO Do You Want to Load Sidekick (Y/N)     件装载 SideKick
BE ASK "Load TSR ", YN
IF NOT ERRORLEVEL 2 GOTO SKIP3
   CD\SIDEKICK
   SK
   CD\
 : SKIP3
REM Rest of AUTOEXEC.BAT File Goes Here
```

在这个批处理程序中包含三个内存驻留程序：Superkey ,Printcache 和 Sidekick。可以选择其中一个或几个,也可以一个也不选。

本章详细阐述了计算机配置的两个重要文件 CONFIG. SYS 和 AUTOEXEC. BAT。对这两个文件中应用的一些子命令进行了剖析,并向读者提供了许多可供参考的程序。根据实际需要,还介绍了不同应用程序所需要的不同环境配置方法。所采用的一些批处理程序大都包含在本书后的所附磁盘上,以供参考。

第 7 章　用批处理程序建立菜单系统

微机已成为现代化办公设备，被用来处理大量的日常办公事务，在硬盘上建立菜单系统可以帮助用户简化操作过程，达到节省时间，减少误操作的目的。这一章将向读者介绍怎样利用批处理程序建立用户自己的菜单系统。在建立有效的菜单系统之前，必须合理地划分计算机的硬盘，一个比较完美的硬盘结构对菜单系统的运转是非常重要的。

7.1　合理划分硬盘

一张软盘就好比是一张书桌，由于容量有限，即使桌上摆满物品，也很容易找到所需要的东西。同样的道理，用户把文件拷贝到一张软盘上，即使占满整张软盘，也很容易找到所要的文件。硬盘就不一样了，由于容量巨大，在硬盘中存入相当数量的文件，就很难查找其中的某一个文件。这就好比一个堆满了书籍的书柜，想从中找出一本书来是颇费力气的。计算机用户每年要处理成百上千的程序文件、数据文件。可以想象，如果把这些文件都存放在一个目录中，只是列出这些文件的名字就要花费相当长的时间。在日常生活工作中，人们总是把文件资料加以分类归档，将同一类的文件放在一起存入书柜中的某个抽屉里，许多这样的抽屉就组了一个书柜。计算机的硬盘就好比是一个大书柜。为了便于管理各种程序文件、数据文件，用户可以将硬盘划分成若干个区域，还可以将这些分区再划分成目录、子目录。再把众多文件分类装入这些目录中。这样目录就起到了抽屉的作用。

划分硬盘的另一个作用是，DOS 对根目录中的文件数目有限制，这个数目是相当有限的。比如，DOS3.2 在 AT 系列微机上格式化 20 兆字节的硬盘，文件分配表上只允许存放 512 个文件。如果不利用子目录的话，用户在硬盘上存放的文件数不能超过 512 个。其它的 DOS 版本和其它类型的计算机也有类似的限定，但具体的限定数目不尽相同。由于 DOS 存储子目录的方式不同，这些限定不适用于子目录。

在根目录中划分的目录叫“目录”，下一级目录就叫“子目录”。当然，这些称谓是经常互换的。子目录就好比若干张较小的磁盘，用户可以根据需要在 DOS 限定的范围内，任意用子目录划分硬盘。子目录所占硬盘空间大小也随用户需要而定，一个子目录甚至可以占据整个硬盘空间。每个子目录是作为一个文件储存在它的上一级目录中的。既然是一个文件，子目录就可以存储相当数量的文件而不受 DOS 约束。硬盘上的每个子目录对于 DOS 来说，就好比是一张张软盘。像磁盘一样，DOS 对每个目录都建立了自己的文件结构目录表。在不同的目录中可以存储相同文件名的文件。建立子目录的优点很多，不占很多硬盘空间，适当的目录结构对计算机的运算速度也没有大的影响。由于子目录不像软盘那样受到存储空间的限制，又不像硬盘那样受文件数的限制，所以建立子目录对用户来说是极其重要的。

实际工作中，在子目录中存放的文件数太多也会带来许多麻烦。从 DOS 工作的技术

角度来讲，最好在每个子目录中不要存储太多的程序文件。DOS 在对某个子目录工作时，总是把所有的文件名读入内存，然后再进行分类、查找。如果文件太多，内存不能一次性读完，会做第二次、第三次读盘，这样 DOS 运算速度将大大降低。用户可能会在实际工作中体会到这个问题，比如在一个子目录中存放的文件太多，当用 DIR 命令列文件时，会非常缓慢，往往需要读盘数次。所以，应该限制一个子目录中的文件数。如果确实需要很多文件存在于一个子目录中，用户可以相应地在 CONFIG. SYS 文件中增加缓冲区的数目。

现在介绍一下有关 DOS 的目录命令。

FDISK 用于硬盘的划分，可将硬盘划分成若干个逻辑驱动器。如果磁盘上有数据信息存在的话，将被 FDISK 命令所破坏。

FORMAT，用 FDISK 命令将硬盘分区后，用户必须用 FORMAT 命令来格式化所划分的每个硬盘区域 C：，D：，E：…。软盘格式化时用的也是这个命令。要注意的是，同 FDISK 一样，在使用 FORMAT 命令时，硬盘上的信息将全部丢失。所以在使用这个命令时要非常谨慎。一旦误用了这个命令，也有补救办法。有些通用软件工具，如 Norton Utilities 可以全部或大部分恢复被 FORMAT 命令所破坏的数据。但是如果用 FDISK 命令将原来的硬盘分配打乱，任何工具都恢复不了原来的磁盘数据。

MD(目录名)(或 MKDIR 目录名)是个内部命令，用于建立子目录。用 MD 命令建立的子目录，是当前目录的分支。

RD 命令也是个内部命令，用于删除当前目录中的子目录。如果用户所要删除的目录中有文件，DOS 将提示这样的出错信息：

```
Invalid path, not directory,
or directory not empty
```

每张磁盘都有一个根目录，格式化磁盘时，这个根目录会自动生成。可以把磁盘上的众多子目录看成是一个目录树。DOS 对根目录中的文件数在技术上做出了规定，但根目录中的子目录数却可以随用户的需要而任意增加，而且子目录中还可以套子目录，如此形成了一个庞大的目录树。PATH(路径)命令把路径的长度限制在 127 个字符以内，因此在建立子目录的时候就要考虑这个问题。用户每次所建立的子目录都是当前目录的下一级子目录，以此类推，当要描述最后一级目录的路径名时，必须逐级加上上一级目录的路径名称，直至根目录。如：

```
C：\UCDOS\CCDE\…\CWS
```

这样往往会形成很长的一串名称，但必须在 63 个字符之内。出于方便，也为了避免路径名过长，用户最好是将相互没有联系的目录存放在同一个根目录中，尽量建立同一级目录。如下例：

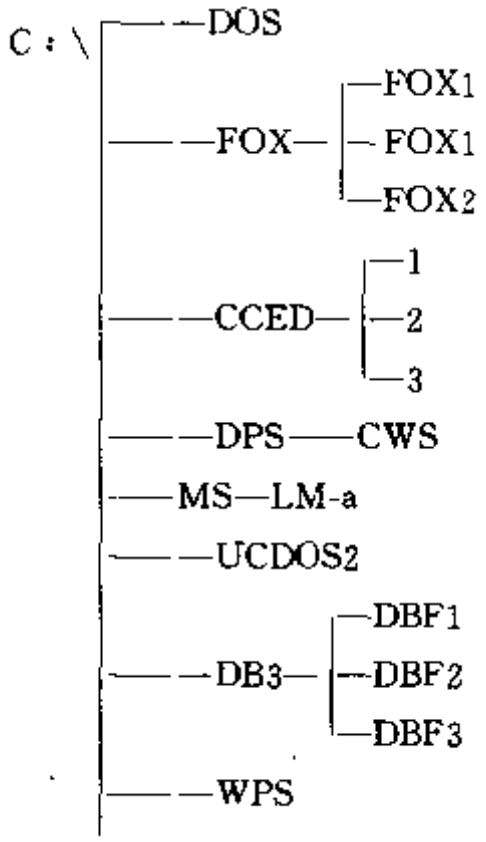


图 7-1

微机用户会注意到，除了根目录以外每个子目录中都会有这样两个文件：

·　　(DIR)

· ·　(DIR)

这两个文件是在建立子目录时自动生成的。用 DIR 命令可以很容易发现这两个文件。这两个文件对 DOS 具有特定的意义，可以简化 DOS 目录操作。两个圆点的文件代表上一级目录。有了这个文件，在子目录中简单地键入“CD. .”便可以退到上一级目录中。连续使用这个命令，可以逐级地一直退到根目录中。一个圆点的文件代表这一级目录中的所有文件。有了它，用户可以用 DOS 通配符 *. *（或 ·）来代替目录中的所有文件。如果想对目录中的所有文件进行拷贝、删除等操作，使用 · 就可以大大简化操作过程。

删除一个目录必先删除目录中的全部文件，但两个特殊的圆点文件例外。如果在删除目录时没有成功，应该首先查看目录中是否仍有文件存在。有时用 DIR 命令检查时目录中没有任何文件，但仍然无法删除目录，这时应该想到目录中可能存在隐含文件。隐含文件一般是由拷贝保护程序生成的，用户为了保护文件不被任意拷贝或是误删除也可以隐含某些文件。删除这些隐含文件用常用的“DEL” 命令不行，必须用“PCTOOLS ”，“ Norton Utilities”一类的通用软件工具，或是由生成这些隐含文件的软件来删除。某些软件包中含有的隐含文件非常重要，一旦被删除整个软件包将无法工作。用 PCTOOLS 等通用软件并不是可以删除所有的文件。在实际工作中，会碰到这样的问题，用“DIR”命令检查目录，发现目录中有许多奇怪的文件，无论用文件删除命令或是 PCTOOLS 都无法清除这些文件；甚至用 PCTOOLS 检查时还无法看到这些奇怪文件。出现这种情况的原因可能是文件分配表发生了混乱。用常规办法是无法删除这些文件的。这里向读者介绍一种办法，用 PCTOOLS 中的“ Find ”功能找出这个子目录的 ASCII 字符，通过变化代码将它们变成一个非目录文件，然后便可以删除这些文件了。

在改变路径时，相同的目录串中可以用“CD”命令加路径名进入目标目录。不同的目

录串之间不能直接改变路径。在图 7-1 所示的目录树中，从 C：\DOS 到 C：\CCED 不能直接进入，必须“CD\”退到根目录 C：\中，再用 CD 命令进入 C：\CCED 中，这两个步骤也可以合二为一，CD\CCED 就可以从 C：\DOS 进入 C：\CCED。

用户应根据自己的需要合理地建立目录结构。一般来说，根目录中除存放必要的系统文件、配置文件以外，应尽量避免放入太多的文件。DOS COMMAND. COM 文件以及两个隐含的系统文件 AUTOEXEC. BAT 文件和 CONFIG. SYS 文件，这些文件一般必须存在于根目录中。除此以外其他的程序文件若没有特殊要求，都应用子目录来管理。不同用途的程序文件最好分别建立相应的目录。一般来说，第一级的各个目录是用来管理各种程序软件的。如图 7-1 所示，第一级子目录中的 CCED，DPS，FOX 等目录分别管理着：图表制作系统软件；中文字处理系统软件；数据库系统软件等。在各个目录中，不同的使用者可以建立自己的子目录，用于存放自己生成的数据文件、文本文件等等，以区别于其它用户的工作文件。但也要注意，建立的子目录越多不一定越好，太多了可能会降低计算机的运算速度。

7.2 菜单技术

所谓菜单就是把计算机中所有或一部分软件资源以表格的形式在屏幕上显示出来，供用户选择。菜单的每一项都有相对应的数字，用户如要使用哪一项程序，只需按一下相应的数字键（并回车）便可运行这个程序，非常简单实用。

本章所介绍的制作菜单的办法有好几种，后面将详细介绍。图 7-2 所给出是一个比较简单的屏幕菜单，这个菜单和图 7-1 所给出的硬盘所含软件资源是相应的。用户想要运行某个程序，可以按相应的数字键，并回车，即可运行此程序。

PRESS	FOR
1	Run FOX
2	Run CCED
3	Run CCDOS4. 23(DPS)
4	Run UCDOS
5	Run Dbase

图 7-2

只有菜单并不能运行任何程序。在上面给出的菜单示例中，如果没有相应的运行程序的批处理程序，在 DOS 提示符下键“1”没有任何作用。菜单上所给出的选择都有相应的一个批处理程序。比如，选择“1”运行“FOX”数据库，那么用户事先必须建立一个可以运行“FOX”数据库的批处理程序“1. BAT”。这样键入“1”才会运行 FOX 数据库程序。在每个与菜单相对应的批处理程序中，除了含有运行相应程序的命令外，文件中的最后一个命令必须使屏幕能够重新显示主菜单。因为每当用户使用完一个应用软件后，可能还要运行

其它软件程序，返回主菜单是必要的。这个命令的格式取决于菜单的形式。

使用菜单的意义不仅在于方便省时，还在于它可以在一定程度上防止误操作。大家都知道，格式化磁盘将破坏该磁盘上的所有信息。对于有些 DOS 版本，如果只键入“FORMAT”命令，而没有跟相应的驱动器标识符，DOS 将格式化全部硬盘。有了菜单就可以避免这样的误操作。

7.2.1 制作菜单

制作菜单的办法很多，每一种都有自己的优缺点。前面讲过，菜单只是在屏幕上列出各项软件的程序清单，实际起作用的是相应的批处理程序。所以，最简单的办法是建立一个含有多个 ECHO 命令的批处理程序。每个 ECHO 命令负责显示一项菜单中的内容。下面是根据图 7-1 的硬盘软件资源所建立的显示菜单的批处理程序：

批处理程序 MENU1. BAT

```
ECHO OFF
REM MENU1. BAT
ECHO
ECHO
ECHO
ECHO
ECHO
ECHO
ECHO
ECHO
ECHO
ECHO
```

PRESS	FOR
1	Run Word Processor
2	Run Spreadsheet
3	Run Data Base
4	Play Games
5	Format a Disk
6	Backup Hard Disk To Floppies

这种方法的优点是简便易行，缺点是比较麻烦，显示速度慢，菜单在屏幕上的显示也不太美观，且需装载 ANSI. SYS 文件来控制屏幕的颜色。

第二种方法也比较简单，利用 DOS 的“TYPE”命令。先将菜单的内容用文本文件的形式存储。然后建立一个批处理程序，用“TYPE”命令显示这个文本文件，下面是示例：

文本文件 MENU2. TXT

```
1. Pun FOX
2. Run CCED
3. Run CCDOS4. 23
4. Run UCDOS
5. Run DBASE
```

批处理程序 MENU2. BAT

```
ECHO OFF
REM MFNU2. BAT
TYPE MENU2. TXT
```

这个办法比第一种办法显示速度快，但也存在第一种办法的一些缺点。

以上两种方法都比较简单。还有一种方法需要用高级程序语言。用户可以运用汇编程序语言来编辑一个可执行文件显示菜单。可执行文件都具有扩展名.EXE，只要在批处理程序中写入可执行文件的文件名，DOS便可以执行这个文件。大多高级语言都可以控制屏幕颜色，而且执行速度很快。关于用高级程序语言制作菜单的技术在这里就不过多地介绍了。

下面向读者们介绍一种快速的屏幕编辑显示软件——Fast-screen。这个程序在本书附盘上。运行Fastscreen的方法很简单，在DOS提示符下，选择正确的磁盘驱动器和目录路径，键入Fastscreen即可运行这个程序。这个程序将把一个文本文件转换成一个小型的命令文件(.COM文件)，用户只要键入这个文件的文件名即可在屏幕上显示文件内容。

可以利用这个程序来编辑，并显示菜单。首先，需要将菜单的内容编辑到一个文本文件中，然后运行Fastscreen程序，它会将这个文本文件转换成.COM文件。要注意的是，在这个文本文件中，每一行不能超过79个字符长度，且每一行必须以回车结束。每个文件不得超过24行。当运行Fastsareen程序的时候，程序会自动要求选择前景颜色，背景颜色，用户可根据需要来选择。最后程序会询问用户要转换成命令文件的文本文件的文件名，得到回答后，Fastscreen会读这个文本文件，然后处理这个文件，产生一个同名的小型.COM文件。完成了这一过程以后，用户键入这个小型命令文件的文件名，屏幕上就会出现原来文本文件中的内容。如果这个文件的内容是一张菜单，屏幕上便会显示这张菜单。另外，这个命令文件将按照用户的设置显示前景及背景颜色。有些读者可能会认为，原来的那个文本文件现在已找不到了，变成了命令文件。这是不对的，原来的文本文件并没有丝毫变动，用户还可以根据需要对它进行修改、重新编辑等操作，然后再生成新的命令文件，来显示修改后的屏幕菜单。

Fastscreen除了可以制作菜单外，还可以在以下几个方面加以应用：

- 应用Fastscreen可以改变光标形状。

  有些应用程序结束返回DOS时，DOS提示符后的光标不能恢复原来的形状。这时可以简单地编一个ASCII文件，这个文件只有一个空白行，通过Fastscreen将这个文件转换成.COM文件。然后只要运行这个.COM文件，重新设置屏幕颜色，会恢复光标的原来形状。

- Fastscreen可以改变屏幕的颜色。

  用户可以很简单地编辑一个文本文件，文件中可不含任何内容。用Fastscreen将它转换成相应的.COM文件；运行Fastscreen时选择适当的前景颜色、背景颜色。这样，只要运行这个命令文件，用户就可以得到所需要的屏幕颜色。用CLS清屏命令可以马上清除当前设置的颜色。有些应用程序在运行后不能恢复屏幕原来的颜色。有了上述办法后，用户就可以随时选择自己喜欢的颜色了。

- 可以在批处理程序中用特殊的颜色显示用户想知道的信息行。具体办法这里不再赘述。

在配单色显示器的计算机上，Fastscreen不能正常工作。如果用户的显示器是单色

的，也不妨先一试试，看 Fastscreen 是否可以工作。

7.2.2 菜单的使用

只在屏幕上显示菜单不能使菜单真正开始工作。菜单上每一项内容必须有一个相应的批处理程序，运行这个批处理程序才能运行用户所需要的应用软件。下面通过一个例子来讲述怎样使菜单工作。

在硬盘上有这样一个菜单：

PRESS	FOR
1	Run Word Processor
2	Run CCED
3	Run DBASE Ⅱ
4	RUN LOTUS 1-2-3
5	RUN GRAPHER

根据这个菜单的内容，采取以下步骤：

- 在硬盘上建立 C：\BAT 目录。
- 在 AUTOEXEC.BAT 文件中设置路径：

  C：\;C：\wordstar;C：\CCED;C：DBASE;C：\lotus;C：\grapher;C：\bat;
- 在 AUTOEXEC.BAT 文件的最后一行设置：

  M.
- 在根目录中建立 M.TXT 文本文件，含有以下内容：

PRESS	FOR
1	Run Word Processor
2	Run CCED
3	Run DBASE Ⅲ
4	RUN LOTUS 1-2-3
5	RUN GRAPHER

- 运行 Fastscreen 生成 M.COM 文件。
- 在 C：\bat 子目录中建立相应的批处理程序：

1.Bat：

```
echo off
echo this is a word processing program, press any key
echo to continue    提示用户
echo off
ws %1               运行 wordstar 程序
M                   编辑完成后返回菜单
```

2. Bat：

```
ccho off              关闭显示
echo this is a table-making program，press any key
echo to continue      提示用户
echo off
pause                 暂停
cced                  运行图表处理程序

M                     使用后返回菜单
```

3. Bat：

```
echo off
echo this program is for data-base processing
echo press any key to continue
pause                 暂停
dbase                 运行 DBASE Ⅱ 数据库程序
M                     返回主菜单
```

4. bat：

```
echo off
echo this is Lotus 1-2-3 2.00 program press any key
echo to continue      提示用户
echo off
pause                 暂停
lotus                 运行程序
M                     返回菜单
```

5. bat：

```
echo off
echo this progarm is for curve-making
echo off
pause                 暂停
grapher               运行程序
M                     返回菜单
```

由于在 AUTOEXEC.BAT 文件中设置了相应的路径，所以在任何目录中上述菜单及批处理程序都可以正常工作。

上面介绍的只是个简单的菜单系统，而且在读者的计算机上也未必可行。读者应根据自己计算机的特点建立合适的菜单系统，并且可以建立一个多级菜单系统，以适应较复杂的情况。

第 8 章　用批处理程序实现文件归档

微型计算机技术发展十分迅猛，当硬盘还没有出现时，一个错误的 DOS 命令，或者是计算机硬件方面的问题，所造成的文件损失至多不会超过一张软盘的容量，也就是 360K 字节。目前一些微机硬盘的容量已超过 200M 字节，如果错误地键入了命令：

C>FORMAT

会把硬盘上的所有文件清除，造成不可估量的损失。突然断电、失火、被盗以及其他意想不到的事情都可能使文件丢失。定期地把一些重要文件备份下来，进行归档，把备份软盘保存在安全的地方，可有效地防止文件丢失。

8.1　定 期 备 份

对于刚刚购买计算机或者准备给计算机配置硬盘的用户来讲，备份是一个新的概念。只使用软盘时，定期制作文件的副本是十分容易的，通常只需把存有源文件的磁盘放进驱动器 A，把格式化后的空盘放进驱动器 B，利用下面的命令：

COPY A：*.* B：

执行该命令后，把源盘上的所有文件都复制到目标盘上。如果用复制磁盘命令：

DISKCOPY A：B：

亦可完成上述工作，执行完 DISKCOPY 命令后，应运行 DISKCOMP 命令，以确保两张软盘内容完全一致。

配有硬盘的用户，在刚刚得到新的系统软件，例如随新购进的计算机一同得到的系统软件时，或者准备把 DOS 版本升级，从硬盘上清除旧的 DOS 版本，安装新的 DOS 版本，安装完毕后，都应立即进行系统软件的备份工作。通常情况下，用户不必修改系统软件。这样把制作好的系统软件备份软盘同原磁盘一起保存在安全的地方。如果硬盘上系统软件中的某个文件丢失或者损坏，遭到病毒感染等，就可以用保存的备份软盘重新安装。

用户建立的文件，比如应用程序或者数据文件，则是另外一种情况。用户自己最清楚这些文件的价值，这些文件一旦损坏，只有用户自己才能修复。某些重要文件，例如一个几万条指令的程序，或者长达几百页的书稿，往往需要几个月，甚至几年时间才能完成。因此在建立这类文件的过程中，应定期地制作备份，每天，每周，每月，定期地把一段时间内完成的内容备份下来，这样可以防止误操作引起的文件丢失。如果用户建立的文件很多，在硬盘上与系统文件混杂在一起，制作定期备份时，需花费大量的时间，才能逐个把用户建立的文件复制到软盘上。由于 DOS 的 COPY 命令速度慢，先读一个文件，然后写盘，再读第二个，…，定期备份所需时间较长。

定期备份是一项经常性的工作，日积月累，所存的备份软盘愈来愈多。假如正在修改一个长度为 300K 的数据库，那么每天工作结束，建立当天的定期备份时，都在软盘上建立了这个数据库的新版本，几乎需要一张 360K 的软盘。进行定期备份时，多数制作备份软盘的软件不能自动把该文件的前一个定期备份从软盘上删除；硬盘上已删除的文件备份软盘上依然存在。已改名的文件，进行定期备份时，同一个文件会以不同的名字存在于软盘上。这样如果从备份软盘上向硬盘恢复文件，硬盘上会出现多余的文件。用批处理程序 BACKUP-1. BAK 可以解决这个问题。

批处理程序 BACKUP-1. BAT

```
ECHO OFF
REM BACKUP-1.BAT                          文件名注释。
CD\                                       确保处于根目录。
DIRSORT EN/S                              运行 Norton Utilities 中 的 DS 程序，把各子目录
                                          内 的文件按扩展名字母顺序 排队。
FILEFIND *.* > C:\BAT\LISTING.TXT         运行 Norton Utilities FF 程序，把所有文件名输
                                          送到文件 LISTING. TXT 中。
```

文件 BACKUP-1. BAT 中，首先进入根目录，然后用 Norton 中的 DIRSORT 程序把所有文件按扩展名字母的顺序排队，再用 Norton 中的 FILEFIND 程序和 DOS 输出重新定向，把文件名输入到文件 LISTING. TXT 里。如果文件 LISTING. TXT 已存在，其内容将被改写，如不存在将自动建立，这样就建立了硬盘上所有文件的清单。在进行备份前，首先运行文件 BACKUP1，备份时，LISTING. TXT 也被备份到软盘上。这样，文件 LISTING. TXT 中的内容总是当前硬盘上文件的清单。用 RESTORE 命令把备份磁盘上的文件恢复到硬盘后，把恢复到硬盘上的文件和 LISTING. TXT 的文件名进行比较，硬盘上存在的文件而 LISTING. TXT 中没有，说明在恢复之前已从硬盘上删除或重新命名，这样可把这些文件从硬盘上删掉。

如果手边没有 Norton Utilities，可以用文件 BACKUP-2. BAT 来完成同样的工作。BACUP-2. BAT 用 CHKDSK 命令列出所有文件后，再输入到 LISTING. TXT 里。

批处理程序 BACKUP-2. BAT

```
ECHO OFF
REM BACKUP-2.BAT
CD\                                       确保处于根目录。
CHKDSK/V > C:\BAT\LISTING.TXT             运行 CHKDSK 命令，并把运行结果输出到文件 LIST-
                                          ING. TXT 里面。
```

文件 BACKUP-2. BAT 运行的快慢取决于 DOS 版本，较低的 DOS 版本会长达 30 分钟。当备份软盘太多时，可以用这种方法把备份软盘上多余的文件删除，然后制作一个完整的备份。

8.2 保存旧文件

对某些关键文件，有时需要进行多次修改，每次修改前的版本最好保存，这样对修改后的版本不满意时，可对修改前的版本重新修改，例如一本书的第 2 章，存在硬盘上原稿的文件名为 CHAPTER2.000，作了第一次修改后的文件名为 CHAPTER2.001，作了第二次修改后的文件名为 CHAPTER2.002，等等。可把这些文件放入硬盘上的一个专用子目录里，例如子目录 C：\OLD。批处理程序 BACKUP-3.BAT 可完成这样的工作。

批处理程序 BACKUP-3.BAT

```
REM BACKUP-3.BAT
IF (%1)==( ) GOTO ERROR          如果作为第一个实际参数的修改前文件名没输入，则转到标号 ERROR。
IF (%2)==( ) GOTO ERROR          如果作为第二个实际参数的修改后文件名没输入，则转到标号 ERROR。
IF EXIST C：\OLD\%2 GOTO EXISTS
                                 如果修改后文件在 OLD 子目录已存在，则转到标号 EXISTS。
COPY %1 C：\OLD\%2/V             从当前目录里把修改后的文件复制到 OLD 子目录里。
GOTO END                         退出批处理程序。
：ERROR                          标号行。
ECHO SOURCE OR TARGET FILENAME NOT ENTERED
                                 提示用户没有输入文件名。
GOTO END                         退出批处理程序。
：EXISTS                         标号行。
ECHO %2 ALREADY EXISTS!          提示用户在OLD 子目录里，修改后 的文件已存在。
ECHO SELECT ANOTHER FILE NAME AND TRY AGAIN
                                 告诉用户如何纠正错误。
：END                            标号行。
```

当 OLD 子目录中文件很多时，如果准备保存，可把它们备份到软盘上，如果已无任何价值，可从硬盘上删除。对于那些十分容易重新建立的文件，可保留最新修改过的版本，而把以前各种版本删除。下面几类文件是很容易重新建立的：

1. .OBJ 文件　运行编译程序时，编译程序先把源程序转化为.OBJ 文件，然后再转化为二进制的.EXE 文件，保留源程序，重新运行编译程序，即可重新建立.OBJ 文件。

2. .BAK 文件　大多数文字处理软件，当把修改后的文件与修改以前的文件同名存盘时，修改前的文件自动以.BAK 为扩展名存盘。

3. .PRN 文件　这种文件是打印电子工作表文件时建立的，如准备重新建立，重新打印工作表文件即可。

4. .$$$和.TMP　由某些商业软件包建立的工作文件，在退出程序时，它们自动被清除，然而重新启动或者磁盘故障可使它们驻留在硬盘上。

对于硬盘上暂时不用的文件，可把它们复制到专用的子目录里，然后把原文件删除。批处理程序 KEEPOLD. BAT 把用户列在执行该文件命令行上的所有文件复制到保存旧文件的专用子目录 C：\KEEPOLD 里面，在执行 COPY 命令之前，该文件首先检验在 KEEPOLD 子目录里是否已存在准备复制的文件，在完成了文件复制后，KEEPOLD .BAT把已复制到 KEEPOLD 子目录里的文件删除。

批处理程序 KEEPOLD. BAT

```
ECHO OFF
REM KEEPOLD. BAT
：TOP                                   循环体的起始标号。
        IF (%1)==( )GOTO END            实际参数(准备复制文件的名)用 尽后或没输入，退出批处
                                        理程序。
        IF EXIST C：\KEEPOLD\%1 GOTO ERROR
                                        如果子目录 KEEPOLD 里有与准备 复制的文件同名的文
                                        件，则转向标号 ERROR。
        COPY %1 C：\KEEPOLD\%1/V
                                        把文件复制到 KEEPOLD 子目录里。
        IF EXIST C：\KEEPOLD\%1 DEL %1
                                        如文件复制成功，则删除已被复 制文件的原件。
        SHIFT                           用下一个文件名取代形式参数%1。
GOTO TOP                                转移到循环体的顶部。
：ERROR                                 标号行。
ECHO %1 ALREADY EXISTS IN C：\KEEPOLD SUBDIRECTORY
                                        告诉用户存在的问题。
ECHO YOU MUST DECIDE WHICH FILE TO KEEP
                                        告诉用户解决问题的方法。
：END                                   标号行。
```

当子目录 C：\KEEPOLD 里面文件足够多时，应把它们备份到软盘上，然后再从硬盘上把这些文件删除，最后还应在硬盘上建立一个所有已备份到软盘上文件名的清单，这样可便于查找。

如果打算将目录 KEEPOLD 所有的文件全部删除，使用下面命令即可：

```
C：\KEEPOLD\DEL *.* <ENTER>
```

某些情况下，子目录 KEEPOLD 中的文件只是部分制作了备份，这样只能删除那些已备份到软盘上的文件，还没有备份的文件必须保留。批处理程序 DELCOPED. BAT 即为此目的编写的。

批处理程序 DELCOPED. BAT

```
@ECHO OFF
REM DELCOPED. BAT
A：                                     进入 A 驱动器。
```

```
CD\                          返回 A 盘根目录。
FOR %%j IN (*.*) DO DEL C:\KEEPOLD\%%j
                             利用 FOR 语句,把 A 盘上已有的文件从 C 盘删除。
C:                           进入 C 驱动器。
CD\KEEPOLD                   进入 KEEPOLD 子目录。
```

利用下面命令,可在硬盘建立已备份到软盘上所有文件名的清单:

```
C:\KEEPOLD\DIRECTORY A:>12.DIR
```

12.DIR 中的文件名 12 是软磁盘号码,应用时用实际软磁盘号码来代替。

第 9 章　增强批处理程序的功能

9.1　抗病毒的批处理程序

每年 3 月 6 日，全世界几乎所有的个人计算机都处于高度戒备状态，以防备“米氏”病毒的袭击。3 月 6 日是意大利著名画家米开朗基罗的生日，以他的名字命名的计算机病毒每年的 3 月 6 日发作。带有此病毒的计算机如果在 3 月 6 日这一天工作，将丢失存储在工作磁盘上的所有信息。由此可见，计算机病毒的危害是相当大的。究竟什么是计算机病毒，如何防止它的传染发作，这一节将从这两个方面向读者作一个简单的介绍。

9.1.1　计算机病毒简介

计算机病毒的概念首先是由弗莱德·恩教授在 1983 年 11 月 3 日的一次计算机安全学术讨论会上提出的，他对此所做的定义是：能够通过修改程序，把自身拷贝进去，进而“传染”其它程序的程序，这便是狭义的计算机病毒概念。广义的计算机病毒概念则从狭义的计算机病毒概念中扩展开来，把能够引起计算机系统故障、破坏计算机数据等的手法统统包括进去。我国一般偏向狭义概念。个人计算机上的常见病毒一般可分为四类：

(1) 系统引导型。这类病毒的特点是，当系统引导时把病毒程序装入内存，在计算机运行过程中能够捕获到 CPU 控制，在得到 CPU 控制权的时候进行病毒传播，在一定条件下发作。

(2) 文件型。可执行文件，包括.EXE 和.COM 文件，最容易被这种病毒感染。每使用一次已感染的程序，病毒就在磁盘上寻找尚未感染的程序，进行传染，消耗了大量的 CPU 时间及磁盘空间，使受感染的计算机工作效率大大降低，最终可能导致死机。

(3) 源码型。源码型病毒在程序被编译之前插入到诸如 FORTRAN，PASCAL 等语言编写的源程序当中，但比较少见。

(4) 入侵型。这类病毒可以侵入到现有程序之中，侵入程序体后很难清除，但这类病毒比较难以制造。

上面简单介绍了什么是计算机病毒，那么究竟如何防止计算机病毒的传染呢？当然最妥当的方法是购买安装一套计算机保护系统，以使计算机百毒不侵。购买安全系统软件一般都比较昂贵，对于一部分用户来说实在没有必要支出这笔开销。这里向用户推荐几种简单易行的办法，利用 DOS 批处理程序来保护微机系统。运用这几种方法可以在一定程度上使微机免遭病毒袭击。由于是利用 DOS 操作系统所提供的功能，所以用户可以不花一分钱。这些方法一般不会给正常的操作带来什么麻烦，这也是较其它安全系统优越的地方。下面就介绍这几种方法。

9.1.2 设置文件为只读方式

DOS 中有一个叫 ATTRIB.EXE 的程序，利用这个程序可以使用户的应用程序只可以读而不能被修改。这个程序可以防止用户自己的误删除，也可以防止某些病毒感染应用程序。当然比较高级的病毒还是能逃过这道关卡。具体使用格式是：

ATTRIB [文件名]+r

这个命令识别 DOS 通配符，所以用户可以一次性处理多个文件：

ATTRIB *.* +r

还可以用"/S"改变当前目录中所有文件的属性(DOS3.3 以上)。应该提醒用户的是，为了防止病毒感染，应该将所有的程序文件，即.EXE 文件，.COM 文件变成只读(Read-only)文件。如果想删除只读文件，要先用下面的命令来解除 Read-only：

ATTRTB 文件名 -r

9.1.3 隐含 COMMAND.COM

所有文件中，最容易被病感染的是 DOS 命令处理器 COMMAND.COM 文件。这是因为 COMMAND. COM 文件是 DOS 最基本的组成部分，每台微机上都要运行这个程序。只要计算机一启动、引导程序就要寻找这个文件。正是由于这一点，病毒只要感染了它，就对磁盘上的其它文件形成了最大的威胁。

一般都将 COMMAND.COM 文件放在引导盘根目录中。DOS3.0 以下的版本只允许 COMMAND.COM 文件存在于根目录中。显然在根目录中被病毒感染的可能性比较大。DOS3.0 以上版本允许将 COMMAND.COM 文件移到某一子目录去，这样可以大大减弱被病毒感染的机会。比如，用户可以将它存入 DOS 文件存放的子目录中。

通过以下步骤便可以照常运行 COMMAND.COM：

(1) 将 COMMAND.COM 拷贝到硬盘上用户用于存储 DOS 文件的子目录 C：\DOS 中。

(2) 把原来的 COMMAND.COM 继续保留在根目录中，即使被病毒感染也没有关系。因为它不会被 DOS 启用。

(3) 在 CONFIG.SYS 文件中写入以下内容：

SHELL=C：\DOS\COMMAND.COM/P

(4) 在 AUTOEXEC.BAT 文件中写入：

SET COMSPEC=C：\DOS\COMMAND.COM

在上面的步骤中，可以用任一个目录名来取替"C：\DOS"，只要在这个目录中确实存在 COMMAND.COM。应该注意的是，在第三个步骤中必须写出扩展名".COM"，也必须写入"/P"，否则，DOS 将不会运行 AUTOEXEC.BAT 文件。

用上面的方法将 COMMAND.COM 隐藏在某一目录中，并不十分安全。在 AU-

TOEXEC.BAT 文件中 COMSPEC 变量已经指出了 COMMAND.COM 的真正所在，病毒仍然可以查寻出它的地址，不管 COMMAND.COM 在什么地方，也不论文件名是什么。尽管如此，用更改文件名的方法在一定程度上还是可以减少病毒的危害。请看下面的两个文件：

文件 CONFIG.SYS

```
SHELL=C：\DOS\COMMAND.COM/P/E：3000
DEVICE=C：\386MAX\386MAX.SYS AUTO
FILES=35
BUFFERS=20
BREAK=ON
LASTDRIVE=N
```

文件 NOTUSED.COM

```
SHELL=C：\DOS\COMMAND.COM/P/E：3000
DEVICE=C：\386MAX\386MAX.SYS AUTO
FILES=35
BUFFERS=20
BREAK=ON
LASTDRIVE=N
```

这两个文件中，后一个与前一个功能是基本一样的，只不过将 COMMAND.COM 换成 NOTUSED.COM。

上面提到过在根目录当中留一份 COMMAND.COM 的拷贝来愚弄病毒。如果病毒按照用户在 CONFIG.SYS 及 AUTOEXEC.COM 两个配置文件中所提供的线索找到了 COMMAND.COM 的真正所在，那么每次启动引导计算机时还是很容易被病毒感染。既然磁盘上有两份 COMMAND.COM。我们就利用其中一份来引导系统，然后利用另一份 COMMAND.COM 再运行一次 COMMAND.COM。这样做并不太麻烦。如果在根目录中的 COMMAND.COM 是假的，在路径设置中就不能设置根目录，否则仍不能避免使用假的备份。有些操作系统允许用户终止 COMMAND.COM，退到 DOS 提示符下，以便重新运行正常的 COMMAND.COM。

9.1.4 检验关键文件

每次引导此导计算机后可以先检验一下某些重要文件，比如.COM，.EXE 等，这些文件最容易被病毒感染。我们知道，程序文件运行次数越多被感染的机会就越大。因此，每次引导系统后检验 COMMAND.COM 是必要的。所谓检验就是用另外一个正常的备份与之进行比较，所以要被检验的文件必须要有一个完全相同的备份，以供比较。下面的这个批处理程序就是用来进行这样的比较的：

批处理程序 TESTCOMM.BAT

```
@ECHO OFF
```

```
REM TESTCOMM.BAT
REM Simple Anti-Viral Program
REM to Test COMMAND.COM
REM I Have COMMAND.COM Stored in
REM My \SYSLIB Subdirectory Rather
REM Than in my Root Directory
REM There is an Identical
REM Copy in my \MISC Subdirectory
REM Under the Name TEST.TXT
FC \SYSLIB\COMMAND.COM \MISC\TEST.TXT
IF ERRORLEVEL 1 GOTO VIRUS
GOTO END
:VIRUS
ECHO COMMAND.COM MODIFIED!
ECHO Find Problem Before Continuing
PAUSE
:END
```

显然 DOS COMP 命令不能进行这种比较，因为它不能返回 Errorlevel 出口码。从上面的批处理程序可以知道，我们可以利用 DOS FC 程序，这个程序支持 Errorlevel，当所比较的两个文件不同时返回的 Errorlevel 出口码为 1。用于进行比较的备份也可以具有不同的文件名。在用户的 AUTOEXEC.BAT 文件中插入上面这一段批处理程序，用户便可以在每次引导时都检验 COMMAND.COM 了，也可以用它来检验其它关键文件。不只是程序文件，数据文件也一样。在 AST/386 微机上，检验一个文件大约要用 10 秒钟时间。因此，对大量的文件进行比较检验没有必要，因为这样做不但会用去相当长的时间，而且会占用相当的磁盘空间，因为每检验一个文件，就需要一个备份。看来用户不必每次引导系统后都检验大量的磁盘文件，只要 COMMAND.COM 不出问题，其它的文件就可以周期性地检验。

9.2 智能型批处理程序

9.2.1 人机对话

通过一个批处理程序可以运行多个应用程序。本节所要介绍的就是怎样利用这个特点实现操作者与计算机之间的“对话”，通过“对话”达到有选择地运行各种应用程序的目的。最简单的人机“对话”通过 pause 语句就可以实现。请看下例：

批处理程序 KILL.BAT

```
ECHO OFF
REM KILL.BAT
DIR %1/W
ECHO Hit Control Break to STOP the erasing of these files
```

```
PAUSE
ERASE %1
```

在这个例子中 pause 语句给用户提供了再一次选择的机会。在本书后附的磁盘上，向读者提供了一个非常有用的程序——CHECK，利用这个程序可以更好更准确地给用户提供选择。这个程序会主动询问用户，并通过用户作出的回答采取行动。

CHECK.COM 程序有 16 个关键词，运用时这 16 个关键词跟在 CHECK 命令之后，通过这些指令可以实现人机对话。表 9-1 给出的是这 16 个关键词的表达式及含义。这 16 个关键词应该灵活运用，要想记住并用好这 16 个关键词必须多练习。表 9-1 给出了 CHECK1.BAT 演示的每一个关键词。

表 9-1 CHECK 的关键词

关键词	语法格式	功能
8087	CHECK 8087	检查数字协处理器是否存在。如果存在，返回的错误级别出口码为 0，否则为 1。
80287	CHECK 80287	
DAY	CHECK DAY	检查日期
DISKSPACE	CHECK DISKSPACE	检查默认的磁盘驱动器磁盘的剩余空间。由于 ERRORLEVEL 的限制，它不能直接提示空间的大小，而是给出以 16K 为单位的块数。如："12"代表 200K (200/16=12.5)
FILEFOUND	CHECKFLEFOUND FILE	检查文件是否存在，若存在返回 0；不存在则返回 1
FILETEXT	CHECK FILETEXT FILE TEXT	检查文件中的文字段。所指文字段必须准确表达，包括字母大小写。若指定文字段存在返回 0，若不存在或有错误则返回 1
FILESIZE	CHECK FILESIZE FILE	给出文件的大小单位为 K。由于 ERRORLEVEL 的最大值是 255，因此大于或等于 255K 的文件将用 255 表示。
KEYBOARD	CHECK KEYBOARD	检查键盘缓冲区。若键盘缓冲区中有击键返回 1，若无则返回 0。
KEYPRESS	CHECK KEYPRESS	返回每个击键的 ASCII 码值，大小写字母返回不同的值。若所击键的是扩展码功能键，光标键等将返回 0。
MEMORY	CHECK MEMORY	通知用户内存空间大小，以 16K 为一个单位。如：640K 为 40(40×16)。但不是所有这些内存都可用。

续表

关键词	语法格式	功能
MODEL	CHECK MODEL	检查 IBM PC 机的型号。其它机种不一定适用。型号是： 号码　机型 255　PC 254　XT 和 Portable PC 253　PCjr 252　AT 249　PC 兼容机
MONTH	CHECK MONTH	返回当前月份，1 月份＝1，2 月＝2…。
TIME	CHECK TIME	检查当前时间，以小时为单位，24 小时为一天。如 1PM 至 1:59PM 返回值为 13
VERSION	CHECK VERSION	返回当前 DOS 版本，只显示整数位，因为 Errorlevel 值只取整数值。
VIDEOCARD	CHECK VIDEOCARD	返回显示器类型。 码值　显示类型 0　MDA 1　CGA 2　EGA
VIDEOMODE	CHECK VIDEOMODE	返回当前显示方式

下面的这个批处理程序显示出每个关键词的功能及用途。

批处理程序 CHECK1. BAT：

```
ECHO OFF
REM CHECK1.BAT
CHECK 8087
IF ERRORLEVEL 1 ECHO No math coprocessor exists
IF ERRORLEVEL 1 GOTO 80287
IF ERRORLEVEL 0 ECHO Math coprocessor exists
IF ERRORLEVEL 0 GOTO 80287
:80287
CHECK 80287
IF ERRORLEVEL 1 ECHO No math coprocessor exists
IF ERRORLEVEL 1 GOTO DAY
IF ERRORLEVEL 0 ECHO Math coprocessor exists
IF ERRORLEVEL 0 GOTO DAY
:DAY
```

检查 8087 数字协处理器。
如果存在则通知用户，不存在则检查 80287 数字协处理器。

检查 80287 数字协处理器，如果有则通知用户，没有则进行下一步。

```
CHECK DAY
IF ERRORLEVEL 31 ECHO 31th of month
IF ERRORLEVEL 31 GOTO DISKSPAC
IF ERRORLEVEL 30 ECHO 30th of month
IF ERRORLEVEL 30 GOTO DISKSPAC
IF ERRORLEVEL 29 ECHO 29th of month
IF ERRORLEVEL 29 GOTO DISKSPAC
IF ERRORLEVEL 28 ECHO 28th of month
IF ERRORLEVEL 28 GOTO DISKSPAC
IF ERRORLEVEL 27 ECHO 27th of month
IF ERRORLEVEL 27 GOTO DISKSPAC
IF ERRORLEVEL 26 ECHO 26th of month
IF ERRORLEVEL 26 GOTO DISKSPAC
IF ERRORLEVEL 25 ECHO 25th of month
IF ERRORLEVEL 25 GOTO DISKSPAC
IF ERRORLEVEL 24 ECHO 24th of month
IF ERRORLEVEL 24 GOTO DISKSPAC
IF ERRORLEVEL 23 ECHO 23th of month
IF ERRORLEVEL 23 GOTO DISKSPAC
IF ERRORLEVEL 22 ECHO 22th of month
IF ERRORLEVEL 22 GOTO DISKSPAC
IF ERRORLEVEL 21 ECHO 21th of month
IF ERRORLEVEL 21 GOTO DISKSPAC
IF ERRORLEVEL 20 ECHO 20th of month
IF ERRORLEVEL 20 GOTO DISKSPAC
IF ERRORLEVEL 19 ECHO 19th of month
IF ERRORLEVEL 19 GOTO DISKSPAC
IF ERRORLEVEL 18 ECHO 18th of month
IF ERRORLEVEL 18 GOTO DISKSPAC
IF ERRORLEVEL 17 ECHO 17th of month
IF ERRORLEVEL 17 GOTO DISKSPAC
IF ERRORLEVEL 16 ECHO 16th of month
IF ERRORLEVEL 16 GOTO DISKSPAC
IF ERRORLEVEL 15 ECHO 15th of month
IF ERRORLEVEL 15 GOTO DISKSPAC
IF ERRORLEVEL 14 ECHO 14th of month
IF ERRORLEVEL 14 GOTO DISKSPAC
IF ERRORLEVEL 13 ECHO 13th of month
IF ERRORLEVEL 13 GOTO DISKSPAC
IF ERRORLEVEL 12 ECHO 12th of month
IF ERRORLEVEL 12 GOTO DISKSPAC
IF ERRORLEVEL 11 ECHO 11th of month
```

设置 ERRORLEVEL，使之同日期相对应，然后执行一系列的 ERRORLEVEL 测试来显示当前日期。注意一旦显示了当前日期，程序会自动执行下一步，而不是一个接一个地去执行 ERRORLEVEL 测试。

```
IF ERRORLEVEL 11 GOTO DISKSPAC
IF ERRORLEVEL 10 ECHO 10th of month
IF ERRORLEVEL 10 GOTO DISKSPAC
IF ERRORLEVEL 9 ECHO 9th of month
IF ERRORLEVEL 9 GOTO DISKSPAC
IF ERRORLEVEL 8 ECHO 8th of month
IF ERRORLEVEL 8 GOTO DISKSPAC
IF ERRORLEVEL 7 ECHO 7th of month
IF ERRORLEVEL 7 GOTO DISKSPAC
IF ERRORLEVEL 6 ECHO 6th of month
IF ERRORLEVEL 6 GOTO DISKSPAC
IF ERRORLEVEL 5 ECHO 5th of month
IF ERRORLEVEL 5 GOTO DISKSPAC
IF ERRORLEVEL 4 ECHO 4th of month
IF ERRORLEVEL 4 GOTO DISKSPAC
IF ERRORLEVEL 3 ECHO 3th of month
IF ERRORLEVEL 3 GOTO DISKSPAC
IF ERRORLEVEL 2 ECHO 2th of month
IF ERRORLEVEL 2 GOTO DISKSPAC
IF ERRORLEVEL 1 ECHO 1th of month
IF ERRORLEVEL 1 GOTO DISKSPAC
: DISKSPAC
CHECK DISKSPACE
IF ERRORLEVEL 255 ECHO More than 4080K free on hard disk
IF ERRORLEVEL 255 GOTO FILEFOUN
IF ERRORLEVEL 200 ECHO More than 3200K and less
                  than 4080K free on hard disk
IF ERRORLEVEL 200 GOTO FILEFOUN
IF ERRORLEVEL 150 ECHO More than 2400K and less than 3200K free on hard disk
IF ERRORLEVEL 150 GOTO FILEFOUN
IF ERRORLEVEL 100 ECHO More than 1600K and less than 2400K free on hard disk
IF ERRORLEVEL 100 GOTO FILEFOUN
IF ERRORLEVEL 50 ECHO More than 800K and less than 1600K free on hard disk
IF ERRORLEVEL 50 GOTO FILEFOUN
IF ERRORLEVEL 10 ECHO More than 160K and less than 800K free on hard disk
IF ERRORLEVEL 10 GOTO FILEFOUN
ECHO Less than 160K free
GOTO FILEFOUN
: FILEFOUN
CHECK FILEFOUND NO-FILE. TXT
IF ERRORLEVEL 1 ECHO NO-FILE. TXT missing
```

设置 ERRORLEVEL 使这同每个代表不同盘剩余空间的数值相对应。然后执行一系列 ERRORLEVEL 测试，直至显示出剩余的磁盘空间。

检查 NO-FILE. TXT 是否存在并设置相应的 ERRORLEVEL 值。然后执行一系列 ERRLR-LEVEL 测

```
IF ERRORLEVEL 1 GOTO FILEFOU2
IF ERRORLEVEL 0 ECHO NO-FILE. TXT found
IF ERRORLEVEL 0 GOTO FILEFOU2
: FILEFOU2
CHECK FILEFOUND CHECK1. BAT
IF ERRORLEVEL 1 ECHO CHECK1. BAT missing
IF ERRORLEVEL 1 GOTO FILESIZE
IF ERRORLEVEL 0 ECHO CHECK1. BAT found
IF ERRORLEVEL 0 GOTO FILESIZE
: FILESIZE
CHECK FILESIZE CHECKERR. BAT
IF ERRORLEVEL 255 ECHO File was 255K or larger
IF ERRORLEVEL 255 GOTO FILETEXT
IF ERRORLEVEL 200 ECHO File was 200-254k
IF ERRORLEVEL 200 GOTO FILETEXT
IF ERRORLEVEL 150 ECHO File was 155-199k
IF ERRORLEVEL 150 GOTO FILETEXT
IF ERRORLEVEL 100 ECHO File was 100-149k
IF ERRORLEVEL 100 GOTO FILETEXT
IF ERRORLEVEL 90 ECHO File was 90-99K
IF ERRORLEVEL 90 GOTO FILETEXT
IF ERRORLEVEL 80 ECHO File was 80-89K
IF ERRORLEVEL 80 GOTO FILETEXT
IF ERRORLEVEL 70 ECHO File was 70-79K
IF ERRORLEVEL 70 GOTO FILETEXT
IF ERRORLEVEL 60 ECHO File was 60-69K
IF ERRORLEVEL 60 GOTO FILETEXT
IF ERRORLEVEL 50 ECHO File was 50-59K
IF ERRORLEVEL 50 GOTO FILETEXT
IF ERRORLEVEL 40 ECHO File was 40-49K
IF ERRORLEVEL 40 GOTO FILETEXT
IF ERRORLEVEL 30 ECHO File was 30-39K
IF ERRORLEVEL 30 GOTO FILETEXT
IF ERRORLEVEL 20 ECHO File was 20-29K
IF ERRORLEVEL 20 GOTO FILETEXT
IF ERRORLEVEL 10 ECHO File was 10-19K
IF ERRORLEVEL 10 GOTO FILETEXT
IF ERRORLEVEL 9 ECHO File was 9K
IF ERRORLEVEL 9 GOTO FILETEXT
IF ERRORLEVEL 8 ECHO File was 8K
IF ERRORLEVEL 8 GOTO FILETEXT
```

试来显示最终结果。

检查指定文件的大小，并设置相应的 ERRORLEVEL 值，最后执行一系列的 ERRORLEVEL 测试，来显示文件的大小。

```
IF ERRORLEVEL 7 ECHO File was 7K
IF ERRORLEVEL 7 GOTO FILETEXT
IF ERRORLEVEL 6 ECHO File was 6K
IF ERRORLEVEL 6 GOTO FILETEXT
IF ERRORLEVEL 5 ECHO File was 5K
IF ERRORLEVEL 5 GOTO FILETEXT
IF ERRORLEVEL 4 ECHO File was 4K
IF ERRORLEVEL 4 GOTO FILETEXT
IF ERRORLEVEL 3 ECHO File was 3K
IF ERRORLEVEL 3 GOTO FILETEXT
IF ERRORLEVEL 2 ECHO File was 2K
IF ERRORLEVEL 2 GOTO FILETEXT
IF ERRORLEVEL 1 ECHO File was 1K
IF ERRORLEVEL 1 GOTO FILETEXT
IF ERRORLEVEL 0 ECHO File was 0K or did not exist!
IF ERRORLEVEL 0 GOTO FILETEXT
:FILETEXT
CHECK FILETEXT CHECK1.BAT 'CHECK'
IF ERRORLEVEL 1 ECHO Text not found in
        CHECK1.BAT or error encountered
IF ERRORLEVEL 1 GOTO FILETXT2
IF ERRORLEVEL 0 ECHO Text found in CHECK1.BAT
IF ERRORLEVEL 0 GOTO FILETXT2
:FILETXT2
CHECK FILETEXT CHECKERR.BAT 'NOW IS THE TIME FOR ALL GOOD MEN'
IF ERRORLEVEL 1 ECHO Text not found in CHECKERR.BAT or error encountered
IF ERRORLEVEL 1 GOTO KEYBOARD
IF ERRORLEVEL 0 ECHO Text in CHECKERR.BAT found
IF ERRORLEVEL 0 GOTO KEYBOARD
:KEYBOARD
CHECK KEYBOARD
IF ERRORLEVEL 1 ECHO Keystroke waiting
IF ERRORLEVEL 1 GOTO KEYPRESS
IF ERRORLEVEL 0 ECHO Keystroke not waiting
IF ERRORLEVEL 0 GOTO KEYPRESS
:KEYPRESS
CHECK KEYPRESS
IF ERRORLEVEL 122 ECHO z Pressed
IF ERRORLEVEL 122 GOTO MEMORY
IF ERRORLEVEL 121 ECHO y Pressed
IF ERRORLEVEL 121 GOTO MEMORY
```

检查在 CHECK1.BAT 中是否有"CHECK"这个词，并在 ERRORLEVEL 中存储结果。然后用一系列 ERRORLEVEL 测试来显示结果。

测试键盘缓冲区中是存储有击键。通过一系列 ERRORLEVEL 测试来显示结果。检查哪个键已敲过，然后设置相应的 ERRORLEVEL 值。进行一系列 ERRORLEVEL 测试显示结果。

```
REM TESTS CONTINUE IN THIS FASHION
ECHO OTHER KEY PRESSED
: MEMORY
CHECK MEMORY
IF ERRORLEVEL 40 ECHO At least 640K RAM available
IF ERRORLEVEL 40 GOTO MODEL
IF ERRORLEVEL 39 ECHO 624K available
IF ERRORLEVEL 39 GOTO MODEL
IF ERRORLEVEL 38 ECHO 608K available
IF ERRORLEVEL 38 GOTO MODEL
IF ERRORLEVEL 37 ECHO 592K available
IF ERRORLEVEL 37 GOTO MODEL
IF ERRORLEVEL 36 ECHO 576K available
IF ERRORLEVEL 36 GOTO MODEL
IF ERRORLEVEL 35 ECHO 560K available
IF ERRORLEVEL 35 GOTO MODEL
IF ERRORLEVEL 34 ECHO 544K available
IF ERRORLEVEL 34 GOTO MODEL
IF ERRORLEVEL 33 ECHO 528K available
IF ERRORLEVEL 33 GOTO MODEL
IF ERRORLEVEL 32 ECHO 512K available
IF ERRORLEVEL 32 GOTO MODEL
IF ERRORLEVEL 31 ECHO 496K available
IF ERRORLEVEL 31 GOTO MODEL
IF ERRORLEVEL 30 ECHO 480K available
IF ERRORLEVEL 30 GOTO MODEL
ECHO Less than 480K available
: MODEL
CHECK MODEL
IF ERRORLEVEL 255 ECHO PC
IF ERRORLEVEL 255 GOTO MONTH
IF ERRORLEVEL 254 ECHO XT or Portable PC
IF ERRORLEVEL 254 GOTO MONTH
IF ERRORLEVEL 253 ECHO PCjr.
IF ERRORLEVEL 253 GOTO MONTH
IF ERRORLEVEL 252 ECHO AT
IF ERRORLEVEL 252 GOTO MONTH
IF ERRORLEVEL 250 ECHO Unknown machine
IF ERRORLEVEL 250 GOTO MONTH
IF ERRORLEVEL 249 ECHO PC Convertible
IF ERRORLEVEL 249 GOTO MONTH
```

检查 RAM 大小，设置相应的 ERRORLEVEL 值，并显示结果。

检查计算机的型号，设置 ERRORLEVEL 值。进行一系列的 ERRORLEVEL 测试来显示结果。

```
ECHO Unknown machine
:MONTH
CHECK MONTH
IF ERRORLEVEL 12 ECHO December
IF ERRORLEVEL 12 GOTO TIME
IF ERRORLEVEL 11 ECHO November
IF ERRORLEVEL 11 GOTO TIME
IF ERRORLEVEL 10 ECHO October
IF ERRORLEVEL 10 GOTO TIME
IF ERRORLEVEL 9 ECHO September
IF ERRORLEVEL 9 GOTO TIME
IF ERRORLEVEL 8 ECHO August
IF ERRORLEVEL 8 GOTO TIME
IF ERRORLEVEL 7 ECHO July
IF ERRORLEVEL 7 GOTO TIME
IF ERRORLEVEL 6 ECHO June
IF ERRORLEVEL 6 GOTO TIME
IF ERRORLEVEL 5 ECHO May
IF ERRORLEVEL 5 GOTO TIME
IF ERRORLEVEL 4 ECHO April
IF ERRORLEVEL 4 GOTO TIME
IF ERRORLEVEL 3 ECHO March
IF ERRORLEVEL 3 GOTO TIME
IF ERRORLEVEL 2 ECHO February
IF ERRORLEVEL 2 GOTO TIME
IF ERRORLEVEL 1 ECHO January
IF ERRORLEVEL 1 GOTO TIME
:TIME
CHECK TIME
IF ERRORLEVEL 23 ECHO After 11PM
IF ERRORLEVEL 23 GOTO END
IF ERRORLEVEL 22 ECHO 10PM - 11PM
IF ERRORLEVEL 22 GOTO END
IF ERRORLEVEL 21 ECHO 9PM - 10PM
IF ERRORLEVEL 21 GOTO END
IF ERRORLEVEL 20 ECHO 8PM - 9PM
IF ERRORLEVEL 20 GOTO END
IF ERRORLEVEL 19 ECHO 7PM - 8PM
IF ERRORLEVEL 19 GOTO END
IF ERRORLEVEL 18 ECHO 6PM - 7PM
IF ERRORLEVEL 18 GOTO END
```

设置 ERRORLEVEL 值使之与每个月相对应。进行 ERRORLEVEL 测试显示相应月份

```
IF ERRORLEVEL 17 ECHO 5PM - 6PM
IF ERRORLEVEL 17 GOTO END
IF ERRORLEVEL 16 ECHO 4PM - 5PM
IF ERRORLEVEL 16 GOTO END
IF ERRORLEVEL 15 ECHO 3PM - 4PM
IF ERRORLEVEL 15 GOTO END
IF ERRORLEVEL 14 ECHO 2PM - 3PM
IF ERRORLEVEL 14 GOTO END
IF ERRORLEVEL 13 ECHO 1PM - 2PM
IF ERRORLEVEL 13 GOTO END
IF ERRORLEVEL 12 ECHO Noon - 1PM
IF ERRORLEVEL 12 GOTO END
IF ERRORLEVEL 11 ECHO 11AM - Noon
IF ERRORLEVEL 11 GOTO END
IF ERRORLEVEL 10 ECHO 10AM - 11AM
IF ERRORLEVEL 10 GOTO END
IF ERRORLEVEL 9 ECHO 9AM - 10AM
IF ERRORLEVEL 9 GOTO END
IF ERRORLEVEL 8 ECHO 8AM - 9AM
IF ERRORLEVEL 8 GOTO END
IF ERRORLEVEL 7 ECHO 7AM - 8AM
IF ERRORLEVEL 7 GOTO END
IF ERRORLEVEL 6 ECHO 6AM - 7AM
IF ERRORLEVEL 6 GOTO END
IF ERRORLEVEL 5 ECHO 5AM - 6AM
IF ERRORLEVEL 5 GOTO END
IF ERRORLEVEL 4 ECHO 4AM - 5AM
IF ERRORLEVEL 4 GOTO END
IF ERRORLEVEL 3 ECHO 3AM - 4AM
IF ERRORLEVEL 3 GOTO END
IF ERRORLEVEL 2 ECHO 2AM - 3AM
IF ERRORLEVEL 2 GOTO END
IF ERRORLEVEL 1 ECHO 1AM - 2AM
IF ERRORLEVEL 1 GOTO END
IF ERRORLEVEL 0 ECHO Midnight - 1AM
IF ERRORLEVEL 0 GOTO END
:END
```

CHECK1.BAT 的运行结果如下：

```
C>check1
NO math coprocessor exists
1th of month
```

```
More than 2400K and less than 3200K free on hard disk
No-FLIE. TXT missing
CHECK1. BAT found
File was 10-19K
Text found in CHECK1. BAT
Text not found in CHECKERR. BAT or error encountered
Keystroke not waiting
OTHER KEY PRESSED
At least 640K RAM available
AT
April
9PM-10PM
C>
```

有关这些关键词的运用就介绍到这里。只要注意运用这些关键词，就会发现它们是非常有用的。很多文字处理软件，如“WS”，“CWS”编辑生成一个文本文件后会自动产生一个 *.BAK 文件做为备份。这些备份文件往往没有用，却占据相当多的磁盘空间。可以利用 CHECK.COM 建立一个批处理程序，检查是否有 *.BAK 文件存在，如果有则删除掉。又如，有些版本的应用程序对计算机的类型、显示器型号有特别要求，运用 CHECK.COM 程序就可以知道自己的机器是否适合于运行这些应用程序。某些计算机病毒是定期发作的，可以建立一个批处理程序，每次关机时自动检查当前日期，如果正好是病毒发作日期的前一天便自动改变日期，以免病毒发作。CHECK.COM 程序的应用还不只这些，需要广大用户在实际工作中去发掘。

除了 CHECK.COM 以外，本书后附的磁盘上有一个 INKEY.COM 程序，这个程序提供了比 CHECK 更优越的人机对话条件。根据用户键盘的类型不同，本书附盘上提供了不同种类的 INKEY 程序。

INKEY 的命令格式是：

INKEY [提示]

在批处理程序中使用了 INKEY，显示器上便显示所提示的内容，只要按任一键便会继续执行下面的命令。INKEY 所设置的 Errorlevel 值同用户所要按的键一一对应。INKEY 后面所跟的提示内容的长度在 80 字符以内。除了表示 DOS 管道的四个符号：<，|，>，>>以及 $ 以外，Inkey 后面可以跟任何字符及字符串，包括 ANSI.SYS 光标设置和显示属性命令。这为充分发挥用户的想象力提供了条件。依靠这些条件，用户可以在批处理程序编程方面充分发挥作用。

对于标准 ASCII 字符 Ctrl-A 到 Z，Inkey 返回的是每一键的正常 ASCII 码。从 ASCII 码表上可以查到这些值。在表中 ASCII 码 65—90 所对应的是 CtrlA-Z，这一组组合忽略字母的大小写，INKEY 将所有的小写击键均转化成大写。对于非 ASCII 键(如功能键)，可以从表 9-2 和表 9-3 中查到它相应的 Errorlevel 值，这也取决于键盘类型。如果在表中查不到，可以运行一下 INKEY，然后在 CHECKERR.BAT 中测试一下 Errorlevel 值。

通常情况下，Ctrl-C 和 Ctrl-Break 一样会中断当前程序的执行，计算机会这样提示用户：Terminate batch job(Y/N)?但是在 Inkey 程序运行过程中，Ctrl-C 同其它的字符一样不会中断批处理程序的执行。这是因为 Inkey 不覆盖键盘缓冲区。

批处理程序 2-2. BAT 是根据 2. BAT 修改而来的，用来装载鼠标器驱动程序。

批处理程序 2-2. BAT

```
ECHO OFF                                    关闭文件回显
REM 2-2. BAT                                注释文件名
INKEY-E Do you want to load your mouse driver?
                                            运行"Inkey"给出提示屏幕
cd\123                                      改变路径到 Lotus
IF ERRORLEVEL 89 IF NOT ERRORLEVEL 90 123MOUSE
                                            如果用户回答"yes"则运行装载鼠标器驱动程序
123
MENU
```

运行 INKEY 时，Errorlevel 值是由 INKEY 来设置的，而有些应用程序在运行时也要设置 Errorlevel 值，势必要改写由 INKEY 所设置的 Errorlrevl 值。遇到这种情况，就需要将 Errorlevel 值存储到环境中去。下面的这个批处理程序就是用来执行 Errorlevel 的多重测试的。

批处理程序 2-3. BAT

```
ECHO OFF
REM 2-3. BAT
REM Set default value to environment        注释
SET MOUSE=NO                                设置一个环境变量
INKEY-E Do you want to load your mouse driver?
                                            运行 INKEY-E 以提示用户，"是否要装载鼠标器驱动
                                            程序
CD\123                                      改变当前目录到 C：\123
REM Change default value only if Y pressed --注释
IF ERRORLEVEL 89 IF NOT ERRORLEVEL 90 SET MOUSE=YES
                                            如果用户选择"Y"，则在 Mouse 环境变量中存入"Yes"，
                                            这是因为 POPDROP 将重新设置 ERRORLEVEL
IF %MOUSE%==YES POPDROP U                   如果用户选择"Yes"，则运行 POPDROP 来设置一个内
                                            存标记，以使鼠标器驱动程序卸载
IF %MOUSE%==YES 123MOUSE                    如果用户选择"Yes"，则运行 123mouse
123                                         运行 Loutus
IF %MOUSE%==YES POPDROP D                   如果用户选择"Yes"则运行 POPDROP 退出 123mouse
MENU
```

INKEY 是个高度灵活的人机对话工具，在本书后附的磁盘中有三个 INKEY 程序，适用于一般键盘、增强型键盘、旧式键盘。

Answer 程序给人机对话提供了另外一种方式。Answer 允许用户提出一个问题，然后把所得到的回答连同 Answer 的变量一起放入环境中。类似的提问及变量设置可以做多次，但要求环境适合。下面的批处理程序演示了这个过程。

批处理程序 ANSWER1. BAT

```
@ECHO OFF
REM This file demonstrates ANSWER.COM          注释
SET                                            显示当前环境设置。
ANSWER Enter the drive with the file to copy
                                               运行 Answer.com，提示用户。Answer 将对用户
                                               的回答作出反应，并将它以 Answer 的名称放入
                                               环境中
SET                                            显示当前环境设置内容。
SET DRIVE=%ANSWER%                             将 Answer 环境变量的内容存入另一个变量中，
                                               由于 Answer.com 每次运行时都用相同的名称，
                                               所以如果用户打算选择多种回答，就一定要进行
                                               这一步。注意批处理程序在响应用户回答时并不
                                               进行错误核查。
ANSWER Enter the file name to copy             再次运行 Answer 以获取另一个回答。
SET                                            显示当前环境设置内容。
SET FILE=%ANSWER%                              将用户的回答存入
                                               环境变量 FILE 中
ANSWER Enter destination                       再次运行 Answer 以获取又一个回答。
SET                                            显示当前环境设置内容。
SET TO=%ANSWER%                                将用户的回答存入
                                               环境变量 TO 中
COPY %DRIVE%%FILE% %TO%                        以用户所作出的三个反应来建立一个命令。
```

ANSWER 程序也有些让用户感到不方便的地方。ANSWER 命令后面的提示行中，如果插入一个空格，它也要求用户对它的含义做出反应，这是无法避免的。还有，当屏幕显示 ANSWER 提示行时，光标的位置并没有做相应的下移，因此当另有其它的信息或是程序内容要显示时，就会同 ANSWER 提示行参杂在一起，效果非常不好。所以，一旦 ANSWER 所提供的信息被用户得知以后，用户应该立即用 CLS 命令清屏。ANSWER 对 DOS 版本也有苛刻的要求，由于它一直没有升级，所以只有 DOS2.0 和 DOS4.0 才适合于 ANSWER，目前比较遍及的 DOS3.3 版本，ANSWER 并不适用。

9.2.2 获得批处理信息的程序

在运行某些批处理程序过程中，有时需要调用其它信息。本书附盘上的“GET.EXE”程序可以到达这个目的。这个程序对于批处理来说非常有用，它不但通过 DOS Errorlevel 获得信息，而且还可以将同样的信息放入环境变量 GET 中。在批处理程序中用户可以用%GET%来调用这个变量。

GET 有 15 个参数，运用时在 GET 后面跟随这 15 个可选参数之一，可以帮助用户执行各种各样的查询任务。在下表中将介绍这 15 个参数的用途及格式等。

表 9-2　GET. EXE 的 15 个参数

参数	功　　能
7	检查数字协处理器，返回 1 表示存在，返回 0 表示不存在。返回值同时被赋给 GET 环境变量和 ERRORLEVEL。
A	检查 ANSI. SYS 文件是否装载。如果的确装载返回值为 1，反之为 0。返回值被同时赋给 GET 环境变量及 ERRORLEVEL。
B	清理屏幕，并改变屏幕前景颜色和背景颜色。用户有必要试验一下颜色设置的顺序。所选择的设置会自动存入环境变量及 ERRORLEVEL。
C	使 GET 获得一个单独的字符。在标准测试时，ASCII 码值的十进制码会被放入 ERR-OR-LEVEL 中。由于环境变量都是大写的，字符将以大写的形式放入环境变量中，这使用户的检查减少一半。在这个命令的后面，用户可以加上提示，以表示应该换什么键。
D	获取 DOS 版本(当前)，将 DOS 版本的整数部分存入 GET 环境变量中。在 ERRORLEVEL 中的存放按这样的方式：整数部分×10＋分位数×10。如 DOS3. 2 在 ERRORLEVEL 中即为 32。
E	将环境中所剩余的字节数存入 ERRORLEVEL 和环境变量 GET 中。也可以选择存储单位为 bytes/10。
F	将指定文件的大小以 16 字节为单位存入 GET 环境变量中，以千字节为单位存入 ERROR-LEVEL 中，也可以选择存储 1/10 字节。
K	将磁盘剩余空间的大小以千字节为单位存储在 GET 环境变量及 ERORRLEVEL 中。也可以选择以 1/10 字节为单位。
M	将内存的剩余空间大小以千字节单位存入 GET 环境变量及 ERRORLEVEL 中。也可以选择以 1/10 字节为单位。
N	只接受“Y”(是)或“N”(不是)，按其它任何键都引起计算机响铃。
P	检查打印机。如果存在返回 1，反之则返回 0。返回值被送给 GET 环境变量及 ERRORLEV-EL。
S	向用户提示一个字符串，并将字符串存入 GET 变量中(不必大写)，将字符串长度存入 ER-RORLEVEL 中。同参数“C”和“N”一样，用户可以增加可选择的提示。
T	参数 C 的一种特殊形式。除了提示部分其余同“C”的功能一致。在 T 参数下，将一个文件认为一个做为提示的字符串。文件的大小不能超过 4K，文件中不包含回车符或其它特殊字符。GET 在显示这个文件的时候一行接一行不停顿，并且一遍遍地重复，直到用户按任意一个键；这个键将会被存入 GET 环境变量中，它的十进制等效值将被存入 ERRORLEVEL 中。

续表

参数	功　能
V	设置显示码。在 GET. DOC 中没有给出这些码的清单，需要用户自己去试验。用户所用的数码会被存入 GET 环境变量和 ERRORLEVEL 中。请注意，用这个参数的一定要慎重，必须要先弄清楚运行的结果。
Y	向 GET 环境变量返回驱动器标识符及当前子目录。向 ERRORLEVEL 中返回目录层数。也可以选择向 ERRORLEVEL 中优先返回驱动器标识符。在本书后附的磁盘中还有一个同 GET. EXE 相应的文件"GET. DOC"，这个文件向读者详细介绍了上述 15 个参数的用途。

9.2.3　设置 ERRORLEVEL 值程序

不管是运行某些应用软件还是建立用户自己的批处理程序，都要碰到重新设置 ERRORLEVEL 的问题。"SETERROR"就是这样一个可以设置 ERRORLEVEL 的程序。它的格式是：

SETERROR #.

#表示 0—255 之间的任一整数。程序在将 ERRORLEVEL 设置到用户指定的数字后会自动退出。用户可在任何一个批处理程序的开头加入这样的命令行：SETERROR 0。(在@ECHO OFF 之后)，以此作为错误陷井，确保批处理程序开始运行时的 ERRORLEVEL 值为零。这个程序也附在书后磁盘上。

9.2.4　显示环境大小程序

这个程序可以告诉用户当前环境区所占空间的大小，以及还有多少剩余环境空间。用户只要键入：

SHOWENVI

就可以得到上述结果。

9.2.5　NORTON 批处理增强器

NORTON UTILITIES 是目前比较流行的通用软件。它是由美国人 Peter Noron 编写的，具有很强的功能。这里只介绍一下在这种软件的高级本版中出现的批处理增强器——BE(Bacth Enhancer)。BE 在 NORTON 4.0 以上版本中才能找到。BE. EXE 含有多个小程序，用户可以用 BE 同相应关键词的组合来访问 BE. EXE 中的多个不同的程序以实现特定的功能。比如，BE Beep 会引起计算机鸣响。而在早期的版本中，这个功能由 BEEP. COM 来完成。下面的这批处理程序 NORTON. BAT，向用户演示了 BE 各种功能的应用。表 9-5 、表 9-6 分别给出了 BE 的每个关键词及其功能和 ASK 的选择项。

批处理程序 NORTON. BAT

```
@ECHO OFF
REM NORTON. BAT
```

```
REM This batch file illustrates      注释
REM the Norton Batch Enhancer
CLS                                  清屏
BE ASK "Do you want to hear the computer beep (y/n) " ny
                                     用于从用户获得执行信息
CLS
REM The above line with set ERRORLEVEL to 2 for a y or Y
REM and 1 for a n or N               注释
IF ERRORLEVEL 2 BE BEEP              如果选择"y"则鸣铃
BE ASK "Do you want to hear a high pitch beep (y/n) " ny DEFAULT=Y TIMEOUT=2
                                     提示用户缺省状态为"y"
                                     Timeout=2 显示 2 秒钟后如没有选择将自动
                                     按"Y"执行
CLS
REM Notice that this will time out after two seconds
REM with a default answer of y       注释
IF ERRORLEVEL 2 BE BEEP /F5000 /D18      如果回答"Y "则鸣铃
REM The tone will sound at 5K Hertz for 1 second
REM Notice that duration is specified in 1/18 second intervals      注释
CLS
REM Now let's draw boxes on the screen      注释
BE BOX 0, 0, 5,15,DOUBLE,BOLD,BLUE   从(0,0)到(5,15)象元在屏幕上画一条兰粗的双线
BE BOX 5,15,10,30,DOUBLE,BOLD,BLUE   画线
BE BOX 10,30,15,45,DOUBLE,BOLD,BLUE  画线
BE BOX 15,45,20,60,DOUBLE,BOLD,BLUE  画线
BE ROWCOL 23,0                       设置光标位置
BE ASK "Press Any Key To Continue "TIMEOUT=10
                                     10 秒钟后继续执行
CLS
BE BOX 0,10, 5,25,SINGLE,BOLD,WHITE      在屏幕上画方框,一次画一条
BE BOX 3,13, 8,28,SINGLE,BOLD,WHITE
BE BOX 6,16,11,31,SINGLE,BOLD,WHITE
BE BOX 9,19,14,34,SINGLE,BOLD,WHITE
BE BOX 12,22,17,37,SINGLE,BOLD,WHITE
BE BOX 15,25,20,40,SINGLE,BOLD,WHITE
BE BOX 18,28,23,43,SINGLE,BOLD,WHITE
BE ROWCOL 23,0                       设置光标
BE ASK "Press Any Key To Continue "TIMEOUT=10
                                     提示用户
CLS
BE PRINTCHAR =,80                    打印一个字符 80 次
ECHO Warning: Something is about to happen
```

```
                                                    提醒用户
ECHO This is just a test                            提示用户
ECHO If it were a real warning
ECHO instructions would follow
BE PRINTCHAR =,80
BE ROWCOL 23,0                                      设置光标
BE ASK "Press Any Key To Continue "TIMEOUT=10
                                                    提示用户
CLS
BE ROWCOL 0,0,"Cursor is at top left of the screen"
BE ROWCOL 10,40,"Cursor is in the middle of the screen"
BE ROWCOL 22,79,"X"
BE ROWCOL 23,0,"X was printed at bottom right"
                                                    设置光标,并在屏幕上给出提示
BE ROWCOL 23,0                                      设置光标
BE ASK "Press Any Key To Set Screen Colors and Exit "TIMEOUT=10
                                                    提示用户
CLS
BE SA BOLD WHITE ON BLUE                            设置屏幕颜色
```

表 9-3

关键词	功　　能
ASK	向用户读取一个字节的信息,然后赋给 ERRORLEVEL,ASK 是一个测试 ER-ROR-LEVEL 的比较好的工具。语法格式为: BE ASK"提示内容",相关键,DEFAULT(缺省)=相关键,TIMEOUT=#,ADJUST=n,color(颜色)。如 BEASK"Files will be deleted! Continue?"(Y/N) ny,DEFAULF=n,TIM-EOUT=5,ASK 后的选择参数在表 8-9 中列出。
Beep	使计算机鸣叫。其后可以跟控制音调的多项可选参数。这些参数包括:持续时间、频率、重复次数、重复间隔时间等。
Box	在屏幕上画方框。语法结构为:BE BOX top 顶端,left(左),bottom(底边),right(右),line type(画线类型),及其颜色。结合批处理增强器的 Rowcol 用户可以画一方框后,设置光标位置至框中,并可在框中定改文本内容。
Printchar	使某一字符重复出现,次数可以选择,以便在批处理程序中画线。
Rowcol	在屏幕的特定位置(行列)上设置光标位置。
SA	直接用英文设置屏幕颜色,如 red (红色)、black (黑色)等。除非在 CONIG. SYS 中装载 ANSI. SYS 文件,否则用 DOS"CLS" 命令就可以清除屏幕颜色。
Window	类似于 BOX,所不同的是它可以在一个窗口中设置另一个颜色不同的窗口。使用这命令时间请参考 BOX 命令的用法。

表 9-4　ASK 程序的选择项

Adjust	增加特定的值给 ERRORLEVEL(由 ASK 返回)。可以向用户询问多个问题，并将 用户的回答做为 ERRORLEVL 值传给批处理程序。
Color	改变提示行颜色。
Default	设置批处理程序默认值，若用户直接回车或超时就执行默认值。
Keys	将 ASK 所设置的 ERRORLEVEL 值同其它程序所设置的 ERRORLEVEL 值区别开。它所规定的键只有 ASK 接受。ASK 将 ERRORLEVEL 值 1 赋给第一个键，2 给第二个键。如此类推。如果用户按了其它的键，计算机只是鸣叫不会有任何运行结果。ASK 对大小写没有什么不同的反应。
Prompt	ASK 提供给用户的提示信息。
Timeout	设置停顿时间。如果从提示开始到下一次击键所用时间超过所设置时间，ASK 会自动按缺省状态处理，并将控制器还给批处理程序。时间设置的单位为秒，标准是机器中的时钟，也取决于处理器的速度。

根据上表中的 Window 功能，用户可以在屏幕上展现各种各样的窗口。下面的这个批处理程序便向读者演示了窗口的设置方法。

批处理程序 NORTON2. BAT

```
@ECHO OFF
REM NORTON2. BAT
REM Window demonstration                          注释
CLS
BE WINDOW 2,2,22,78, BLUE, SHADOW, ZOOM           在(2,2)和(22,78)象元间设置一个窗口
BE ROWCOL 12,30, "This is a window"               设置光标位置，并在屏幕上显示
BE ROWCOL 20,4, "press any key"
BE ASK " ", TIMEOUT=10                            没有提示行，停留时间为 10 秒
CLS
BE SA WHITE, RED > NUL                            将屏幕背景变为红色，将字符显
                                                  示变为白色，并将这个命令送给 NUL
BE WINDOW 8,20,13,60, RED, ZOOM                   设置一个红色窗口
BE ROWCOL 10,25,"A simulated error has occured", BLINKING WHITE
                                                  设置光标并写在屏幕上
BE ROWCOL 12,25, "Press any key to abort", WHITE
BE ASK " ", TIMEOUT=10                            提示用户
CLS
```

在运用 ASK 时，可以同时向用户提问多次，并将这些问题归到单一的 ERRORLEVEL 值中。ASK 程序的格式为：

```
ASK"Start which program? →",WSDQ
```

双引号中的内容是询问用户启动哪个程序，后面的 WSDQ 分别代表 Word process-

ing(文字处理),Spreadsheet(Lotus 1-2-3),Database(数据库)和 Quit(退出)。用户选择 W,ASK 程序设置 ERRIRKEVEL 为 1;如果选择 S,D 或 Q,ERRORLEVEL 分别被设置为 2,3 或 4。WSDQ 四项是建立批处理程序时规定的。下面的例子清楚地说明了 ASK 程序的用法和功能。

批处理程序 SHOWASK. BAT

```
ECHO OFF
REM SHOWASK.BAT
: START
CLS
ECHO Word processing(WORD STAR)
ECHO Spreadsheet(Lotus 1-2-3)
ECHO Database(Q&A 3.0)
ECHO .
ECHO Quit
ECHO .
BE ASK "Start which program? →",WSDQ
IF ERRORLEVEL 4 GOTO QUIT
IF ERRORLEVEL 3 GOTO DATABASE
IF ERRORLEVEL 2 GOTO SPREADSHEET
IF ERRORLEVEL 1 GOTO WORDPROC
GOTO START
: WORDPROC
CD C : \WORD
WORD
CD\NORTON
GOTO START
: SPREADSHEET
CD C : \LOTUS
123
CD\NORTON
GOTO START
: DATABASE
CD C : \QA
QA
CD\NORTON
GOTO START
: QUIT
EXIT
```

执行 SHOWASK. BAT 后的屏幕显示为:

```
Word processing(WORD STAR)
Spreadsheet(Lotus 1-2-3)
Database(Q&A 3.0)
.
Quit
.
Start which program? →
```

文件 SHOWASK.BAT 中，根据用户的选择，设置 ERRORLEVEL 值，控制程序的流向。用户选择项中不得使用专用键，例如功能键、退格键、空格键、删除及插入键等，一般应使用字母键和大键盘区的数字键。

ASK 程序有 6 个选择项，NORTON1.BAT 演示了部分选择项的用法。

批处理程序 NORTON1.BAT

```
@ECHO OFF
REM NORTON1.BAT
REM Illustrate the ability of Ask to
REM adjust the ERRORLEVEL value to
REM return more than one response.          注释
CLS
BE ASK "Do you want option A, B or C ", abc
                                            用户选择 A,B 或 C,则分别把 ERRORLEVEL 设置
                                            为 1, 2 或 3
CLS
IF ERRORLEVEL 3 GOTO THREE                  若 ERRORLEVEL≥3 则执行 THREE,ASK
                                            中设置 ERRORLEVEL 的最大值为 3
IF ERRORLEVEL 2 GOTO TWO                    若 ERRORLEVEL≥2,则执行 TWO,
IF ERRORLEVEL 1 GOTO ONE                    若 ERRORLEVEL≥1,则执行 ONE
: THREE                                     THREE 标号行
BE ASK "Do you want option 1 or 2 ",12,ADJUST=10
                                            提问用户问题,并将用户的选择增
                                            值 10(ERRORLEVEL)
CLS
GOTO NEXT                                   转向至 NEXT
: TWO
BE ASK "Do you want option 1 or 2 ",12,ADJUST=20
                                            提问用户一个问题,并将选择增值 20
CLS
GOTO NEXT                                   转向
: ONE
BE ASK "Do you want option 1 or 2 ",12,ADJUST=30
                                            提问用户一个问题,并将选择增值 30
```

```
*LS
GOTO NEXT                          转向
:NEXT
IF ERRORLEVEL 32 GOTO A2           ERRORLEVEL 测试,决定用户选择的结果。
IF ERRORLEVEL 31 GOTO A1
IF ERRORLEVEL 22 GOTO B2
IF ERRORLEVEL 21 GOTO B1
IF ERRORLEVEL 12 GOTO C2
IF ERRORLEVEL 11 GOTO C1
:A2
ECHO A2 SELECTED                   显示运行结果
GOTO END
:A1
ECHO A1 SELECTED
GOTO END
:B2
ECHO B2 SELECTED
GOTO END
:B1
ECHO B1 SELECTED
GOTO END
:C2
ECHO C2 SELECTED
GOTO END
:C1
ECHO C1 SELECTED
GOTO END
:END
```

第 10 章　批处理程序应用技巧

应用批处理技术，可以解决很多应用方面的实际问题，例如设置口令字、定期自动执行处理程序、控制打印机等。读者阅读本章后，可以从中得到启发，掌握编写批处理程序的方法和技巧，解决日常工作中所遇到的一些技术问题。

10.1　暂时访问 DOS

很多应用软件在本身已调入内存的情况下，允许暂时退出，访问 DOS。例如 Lotus 1-2-3 主命令菜单中的参数/S，可以在 Lotus 1-2-3 已调入内存的情况下，暂时返回 DOS。在暂时返回 DOS 的情况下，用户有可能把另一种软件调入内存并运行，由于内存有限，其运行速度会很慢。也可能由于剩余的内存空间太小，根本无法调入。例如，Lotus 1-2-3 调入内存之前，执行 CHKDSK 命令：

```
C：\LOTUS>chkdsk
Volume 481009-001 created Jan 1 1993 10：04a

2 13 09449 bytes total disk space
  1421312 bytes in 46 hidden files
     34816 bytes in 13 directories
  9684992 bytes in 566 user files
10 168320 bytes available on disk
   654336 bytes total memory
   598112 bytes free

C：\LOTUS>
```

可用的内存为 598112 字节，调入 Lotus 1-2-3 后，用 1-2-3 命令主菜单里/S 参数，暂时访 DOS，此时调用了辅助命令处理程序，再执行 CHKDSK 命令，屏幕显示：

```
Microsoft(R) MS-DOS(R) Version 3.30
            (C)Copyright Microsoft Corp 1981-1987

C：LOTUS>chkdsk
Volume 481009-001 created Jan 1,1993 10：04a

2 13 09440 bytes total disk space
  1421312 bytes in 46 hidden files
```

```
     34816 bytes in 13 directories
   9684992 bytes in 566 user files
  10168320 bytes available on disk

  654336 bytes total memory
  415024 bytes free

C:\LOTUS>
```

空闲的内存为 415024 字节，比执行第一次 CHKDSK 命令时减少了 183088 字节，这是 Lotus 1-2-3 占用的内存量，此时如果试图把文字处理软件 WP(WORD PERFECT)装入内存：

```
C:\LOTUS>wp
Program too big to fit in memory

C:\LOTUS>
```

目前内存空间已容纳不下 WORD PERFECT，所以无法调入。为了避免上述问题的发生，可以用批处理程序来启动类似 Lotus 1-2-3 这样的应用软件。批处理程序 LOTUS.BAT，是用来启动 Lotus 1-2-3 的。在启动之前，把通常使用的 DOS 提示符改为“Type EXIT to return to Lotus $-$p$g，用户在利用主命令菜单中的/System 命令暂时访问 DOS 时，提示符将提示用户 Lotus 1-2-3 仍在内存里，并告诉用户如何返回 Lotus 1-2-3。程序运行结束时，批处理程序把 DOS 提示符恢复为 pg。

批处理程序 LOTUS.BAT

```
@ECHO OFF
REM LOTUS.BAT
C:                                  确保在C驱动器的根目录。
CD\123                              进入 Lotus 子目录。
PROMPT=TYPE EXIT to return to Lotus $-$P$g
                                    改变提示符，用户暂时返 回 DOS 时，将显示该提示符，
                                    提示用户如何返回 1-2-3。
123                                 启动 Lotus 1-2-3。
PROMPT=$P$g                         退出 Lotus 1-2-3 时，把提示符复原。
```

运行 LOTUS.BAT，可以把 Lotus 1-2-3 调入内存，并在暂时返回 DOS 时，告诉用户如何回到 Lotus 1-2-3 中去，然而这无法阻止用户把其他的软件调入内存。

利用 EXIT 命令构成批处理程序 LOTUS1.BAT，可以防止在应用程序内部启动辅助命令处理程序 COMMAND.COM 暂时退出并访问 DOS 的情况下，调入其应用程序。EXIT 命令应放在文件的开头。如果在主命令处理程序下运行这样的批处理程序，启动 Lotus 1-2-3，EXIT 命令不起任何作用。如果在启动了辅助命令处理程序(例如 Lotus 1-2-3 主菜单中的/S 命令)，暂时访问 DOS 的情况下，执行文件 LOTUS1.BAT，企图重新把

Lotus 1-2-3 调入内存，首先遇到的是 EXIT 命令，它将终止辅助命令处理程序，立即返回启动命令辅助处理程序的应用程序中。

批处理程序 LOTUS1. BAT

```
@ECHO OFF
REM LOTUS1.BAT
EXIT        首次运行该程序时，EXIT 命令不起任何作用。在 Lotus 1-2-3
            运行过程中暂时返回 DOS 时，如果企图利用该文件重新调用 1-2-3，
            由于 EXIT 命令的执行，可以阻止 1-2-3 的重新调入，
            并终止辅助命令处理程序，返回退出之前的工作表文件中，
            EXIT 以后的命令也不再执行。
C:
CD\123
PROMPT=Type EXIT to return to Lotus $-$p$g
123
PROMPT=$p$g
```

在某一应用软件已装入内存的情况下，例如文字处理软件 Mi-crosoft Word 已经调入并执行，利用 Escape Library Run 命令，启动辅助命令处理程序，暂时返回 DOS，此时在 DOS 提示符下键入：

```
C:\Lotus1
```

要调入 Lotus 1-2-3 是不可能的，因为 LOTUS1. BAT 中的 EXIT 命令将中止辅助命令处理程序，立即返回中断执行的 Microsoft Word 中，而且不再执行 LOTUS1. BAT 中 EXIT 后面的命令。应用这种方法编写的批处理程序来调入允许从内部暂时访问 DOS 的应用程序时，可给用户提供返回程序的正确方法，并防止在暂时返回 DOS 时调用其他应用程序。

10.2 路径与子目录问题

10.2.1 进入预定子目录

一个复杂的、嵌套了多层的目录系统，经常需要从当前目录进入其他目录，工作完毕后再返回当前目录，完全靠手动改变目录是很繁琐的。

目前流行的一些 PCTOOLS 软件包，一般都包括改变目录程序，例如 Norton 实用程序中的 NCD(Norton Change Directory)程序，在执行时，把目录结构显示在屏幕上，此时所在的当前目录是加亮显示的。用户把光标移到准备进入的目录，按回车键，然后再接 ESC 键或者 F10 键，就进入了预定的子目录。

如果目录结构简单，嵌套层次少，编写一些专用的批处理程序，通过运行这些程序进入预定的子目录也是一个较好的方法。例如只用一条命令：

```
CD\C\TC\EXAMPLE
```

组成一个批处理程序，每当运行这个文件，总是进入这条命令指定的目录中。

如果需要经常从当前目录进入多个不同的子目录，问题就较为麻烦。用批处理程序可以部分地解决这一问题。建立这样一个批处理程序，它总是把当前目录作为主目录，运行这个程序文件时，能够自动建立一个返回主目录的文件，例如，首先建立文件 RETURN，它只有一条命令：

```
CD ^Z
```

其中 ^Z 是文件结束符。在批处理程序 RETURN1.BAT 运行时，建立文件 RETURNTO.BAT。

批处理程序 RETURN1.BAT

```
@ECHO OFF
REM RETURN1.BAT
COPY C:\BAT\RETURN C:\BAT\RETURNTO.BAT        把文件 RETURN 的内容
                                              复制到 RETURNTO.BAT 里。
CD >> C:\BAT\RETURNTO.BAT                     把当前子目录加进文件
                                              RETURNTO.BAT。
```

如果当前子目录为 CD\D\LOTUSFIG\0-9，RETURN1. BAT 执行后，RETURNTO.BAT将含有下列内容：

```
CD C:\D\LOTUSFIG\0-9
```

这是由于文件 RETURN 中 CD^Z 没有回车键，在执行 RETURN1.BAT 过程中，CD 命令的执行结果由输出转向符>>连接到 RETURNTO.BAT 的后面。RETURN1.BAT 的作用是把当前目录作为主目录，并且建立文件 RETURNTO.BAT，RETURNTO.BAT用于返回主目录。从不同的目录执行 RETURN1.BAT，就把本次的当前目录作为主目录，从主目录进入其它目录后，准备返回时运行 RETURNTO.BAT 即可。每次执行 RETURN1.BAT，RETURNTO.BAT 的内容都被改写。由于本例中，文件 RETURNTO.BAT 和 RETURN 所在的子目录为 C:\BAT，环境中的 PATH 语句中必须包括 C:\BAT 子目录。这样在任何目录里都可以运行它们。

批处理程序 RETURN2.BAT 提供了另一种返回主目录的方法，首先把进入主目录的路径赋于环境变量 HOME：

```
SET HOME=C:\D\LOTUSFIG\0-9
```

批处理程序 RETURN2.BAT：

```
CD %HOME%                    调用环境变量 HOME，进入 HOME 指定的目录。
```

文件 RETURN2. BAT 的缺点是用户必须把进入主目录的路径事先赋给环境变量。批处理程序 RETURN3.BAT 利用形式参数，自动把进入主目录的路径赋给环境变量。在运行 RETURN3.BAT 时，必须输入作为实际参数的主目录路径；如果只键入文件名而不输入实际参数，RETURN3. BAT 的运行结果为进入已存在环境变量 HOME 中的目录。

批处理程序 RETURN3. BAT

```
@ECHO OFF
REM RETURN3. BAT
IF (%1)==( ) GOTO GOHOME          如不输入准备进入的目录路径，
                                  则进入当前环境变量中存的目录中。
CD\%1                             进入用户输入的目录。
SET HOME=%1                       把输入的路径存入环境变量。
GOTO END                          退出批处理程序。
:GOHOME                           标号行。
CD %HOME%                         如不输入路径，则进入当前 HOME 中
                                  所存的目录。
:END
```

10.2.2 修改路径

许多用户习惯于把所有外部命令或者某些应用程序所在子目录的路径放在 DOS 路径命令中，DOS 通过 PATH 指定的目录结构查找文件所在的子目录。PATH 命令属于文件 AUTOEXEC. BAT 中的一条命令。如果准备修改 PATH 中指定的路径，必须编辑文件 AUTOEXEC. BAT，然后重新启动，修改后的 PATH 命令才能起作用。

如果目录结构复杂，子目录很多，DOS 规定 PATH 命令语句的长度又不得超过 127 个字符，这样把所有子目录所在路径全部包括在一个 PATH 命令语句中是不可能的，这就需要两条 PATH 语句。再者，有时需要对 PATH 命令语句进行临时修改，用完后再复原，这类问题可以用批处理程序解决，步骤如下：

(1) 把当前 PATH 命令语句的路径赋给一个环境变量，存在 DOS 环境区中。

(2) 用新的 PATH 语句取代原来的 PATH 语句。

(3) 利用保持原 PATH 语句的环境变量恢复原 PATH 语句。

批处理程序 PATH1. BAT

```
ECHO OFF
REM PATH1. BAT
IF (%1)==( ) GOTO NOPATH          如果不输入新路径，则转向标号 NOPATH
SET OLDPATH=%PATH%                把原 PATH 指定的路径保持在环境变量
                                  OLDPATH 里。
SET PATH=%1                       设置新路径。由于 DOS 把实际参数中的
                                  分号当作分隔符，只能输入一个子目录
                                  作为实际参数。
GOTO END
:NOPATH                           如果不输入新的路径，则转到这里。
ECHO NO PATH SPECIFIED            提示用户没输入路径。
:END
```

执行 PATH1. BAT，需同时键入文件名和路径。

批处理程序 PATH2.BAT 把路径复原。

批处理程序 PATH2.BAT

```
ECHO OFF
REM PATH2.BAT
SET PATH=%OLDPATH%                    恢复原路径。
```

批处理程序 PATH3.BAT 把子目录加到原 PATH 命令语句的后面。如要恢复原路径，仍需运行 PATH2.BAT。

批处理程序 PATH3.BAT

```
ECHO OFF
REM PATH3.BAT
IF (%1)==( ) GOTO NOPATH
SET OLDPATH=%PATH%
SET PATH=%PATH%;%1                    给当前路径增加新内容
GOTO END
:NOPATH
ECHO NO PATH SPECIFIED
:END
```

批处理程序 PATH4.BAT 可以在三个 PATH 语句中进行选择。

批处理程序 PATH4.BAT

```
ECHO OFF
REM PATH4.BAT
IF (%1)==( )GOTO NOPATH
IF %1==1 PATH=C:\WORD                 如果输入的实际参数是 1，
IF %1==1 GOTO END                     则选用第一个 PATH 命令语句。
IF %1==2 PATH=C:\LOTUS                如果输入的实际参数是 2，
IF %1==2 GOTO END                     则选用第二个 PATH 命令语句。
IF %1==3 PATH=C:\DBASE                如果输入实际参数是 3，
IF %1==3 GOTO END                     则选用三个 PATH 命令语句。
GOTO ERROR                            如果输入了不正确的实际参数
                                      (不是 1,2 或 3)，则转到这里。
:ERROR                                标号行。
ECHO ONLY 1,2,OR 3 ARE                提示用户，输入正确的实际参数。
VALID PATH SELECTIONS
:GOTO END
:NOPATH
ECHO NOPATH SELECTED
:END
```

利用批处理程序，可以部分地删除 PATH 语句中指定的路径。在批处理程序内部可

以调用环境变量%PATH%，但是不能直接修改 PATH 指定的路径。由于 PATH 语句指定的路径中各个子目录是由分号隔开的，例如 PATH=C：\;\DOS;\BAT;\NORTON。如果键入命令：

C>Batchname %PATH%

希望把 PATH 指定路径中的子目录作为实际参数使用，以便修改路径。不幸的是 DOS 不允许在提示符下执行批处理程序同时调用环境变量。在批处理程序内部调用子程序时，是可以做到这一点的，这样可根据需要删除路径中的子目录。

要完成这一工作，需要两个批处理程序。文件 EDITPAT1.BAT 是主程序，它调用子程序 EDIT1.BAT。在执行 EDITPAT1.`BAT 并键入文件名的同时，必须输入要求保留 PATH 语句子目录的个数，这个作为实际参数的数字应小于当前 PATH 语句中所含子目录的个数，这样就从后面把多余的子目录去掉，实现了部分删除路径中子目录的目的。程序 EDITPAT1.BAT 最多只能在路径中保持 8 个子目录，如果使用 SHIFT 命令，则路径中子目录的个数不受限制。对文件 EDITPAT1.BAT 稍加修改，可以从当前 PATH 语句的前面删除子目录。

批处理程序 EDITPAT1.BAT

```
@ECHO OFF
REM EDITPAT1.BAT
REM Keep first n elements of PATH
IF (%1)==( ) GOTO ERROR          如果不输入路径中准备保留子目录的个数，则转到处
                                 理错误部分。
IF %1==1 GOTO OK                 输入的实际参数只能是 1 至 8 之中的一个整数。
IF %1==2 GOTO OK
IF %1==3 GOTO OK
IF %1==4 GOTO OK
IF %1==5 GOTO OK
IF %1==6 GOTO OK
IF %1==7 GOTO OK
IF %1==8 GOTO OK
GOTO ERROR                       执行到这里，说明输入了错误的实际参数
:ERROR
ECHO Number of elements to retain not specified
ECHO or specified incorrectly
GOTO END
:OK                              执行到这里，说明程序执行正确。
EDIT1 %1 %PATH%                  调用程序 EDIT1.BAT，并且 PATH 中的各个子目录作
                                 为实际参数传递给程序 EDIT1.BAT。由于没有使用
                                 CALL 命令，执行完 EDIT1. BAT 后，不再返回 EDIT-
                                 PAT1.BAT 中。
:END
```

批处理程序 EDIT1. BAT

```
@ECHO OFF
REM EDIT1.BAT
REM Reconstruct first N elements of PATH
REM N assumed to be less
REM than or equal to eight
PATH=%2                          把环境变量 PATH 中的第一个子目录赋给 PATH。
IF %1==1 GOTO SHOWPATH           如果%1 对应的实际参数为 1,则程序转向标号 SHOW-
                                 PATH。
PATH=%PATH%;%3                   把第二个子目录加进 PATH 命令语句中。应注意,执行该
                                 行之前,PATH 只含有一个子目录。
IF %1==2 GOTO SHOWPATH           如果只要求 PATH 指定两个子目录,则转向标号 SHOW-
                                 PATH。
PATH=%PATH%;%4
IF %1==3 GOTO SHOWPATH
PATH=%PATH%;%5
IF %1==4 GOTO SHOWPATH
PATH=%PATH%;%6
IF %1==5 GOTO SHOWPATH
PATH=%PATH%;%7
IF %1==6 GOTO SHOWPATH
PATH=%PATH%;%8
IF %1==7 GOTO SHOWPATH
PATH=%PATH%;%9
:SHOWPATH
ECHO NOW PATH Set to %PATH%
```

批处理程序 EDITPAT1. BAT 和 EDIT1. BAT,只能部分地从 PATH 命令语句的后边删除子目录,应用时有一定的局限性。程序 EDITPAT2. BAT 可以部分删除 PATH 命令语句中任何位置的子目录,它调用程序 EDIT2. BAT,EDIT2. BAT 中使用了 NORTON 实用程序中的 ASK 命令来设置 ERRORLEVEL 值,根据 ERRORLEVEL 值,决定命令语句中子目录的取舍。

批处理程序 EDITPAT2. BAT

```
@REM EDITPAT2.BAT
@REM Selectively delete Subdireetories
@EDIT2 %PATH%                    调用程序 EDIT2。环境变量 PATH 指定的路径中的各个子目录作为
                                 实际参数传递给 EDIT2 中的形式。由于没有使用 CALL 命令,控制
                                 不再返回 EDITPAT2. BAT。当前 PATH 中作为分隔符的分号,在传
                                 递实际参数时被消除,因此在程序 EDIT2 执行时,应在各子目录之
                                 间加上分号。
```

批处理程序 EDIT2. BAT

```
@ECHO OFF
REM Selectively Delete Path Subdirectories

PATH=;                                  置路径为空白。
:TOP                                    循环体的起始标号。
    IF (%1)==( ) GOTO END
    C:\NORTON\BE ASK"keep %1 in PATH(Y/N)"NY
                                        询问用户是否准备在PATH中保持某个子目录。
    IF ERRORLEVEL 2 IF NOT              把某个子目录加到已有路径的后面。
    (%PATH%)==( ) PATH=%PATH%;%1
    IF ERRORLEVEL 2 IF                  设置PATH中的第一个子目录。
    (%PATH%)==( ) PATH=%1
    SHIFT                               把所有实际参数左移一个位置。
    GOTO TOP                            转到循环体的起始标号。
:END
ECHO NOW PATH SET TO %PATH%
```

由于在EDIT2.BAT的开头,将PATH置为空路径,调用BE ASK命令时,必须指明该命令所在的子目录。

10.3 自动配置系统

计算机的系统配置是由CONFIG.SYS与AUTOEXEC.BAT完成的。系统配置的工作,不可能是一劳永逸的。某些应用软件,比如Lotus 2.2,运行时需要用扩展内存。再如微机联网等,也对系统配置有一定要求。一成不变的系统配置不可能满足这些要求。如果用户进行大量的编程工作,或者使用大型数据库,都必须修改上述两个文件,使系统配置满足需要。

本书第5章介绍了一种方法,即根据需要选择不同的CONFIG.SYS和AUTOEXEC.BAT来启动计算机,以满足不同的要求。第5章对这种方法已有说明,这里不在重复。目前很多较大型的应用程序,例如文字处理软件WORD PERFECT甚至磁盘操作系统(例如4.0以上的DOS版本)本身都附带有安装程序。在硬盘上安装这类应用程序时,一般由安装程序自动修改CONFIG.SYS和AUTOEXEC.BAT,以得到合适的系统配置。

这里我们只介绍一个应用软件安装程序修改AUTOEXEC.BAT。首先安装程序里应包括两个批处理程序,一个批处理程序包括AUTOEXEC.BAT准备增加的内容,另一个用于对现有的AUTOEXEC.BAT进行修改。批处理程序ADD.BAT包括准备安装的新程序所在的子目录,另一个批处理程序DOADD.BAT修改当前的AUTOEXEC.BAT。

批处理程序ADD.BAT

```
@ECHO OFF
```

```
REM ADD.BAT
REM Appends commands to an AUTOEXEC.BAT file
PATH=C:\NEWPROGRAM;%PATH%          把新程序所在子目录加到当前路径里。
```

程序 DOADD.BAT 利用 REN 命令把当前的 AUTOEXEC.BAT 更名为 AUTOEXEC.OLD，然后用 COPY 命令把 AUTOEXEC.BAT 与 ADD.BAT 合并，组成新的 AUTOEXEC.BAT。这样就把新的路径语句加到了 AUTOEXEC.BAT 文件的最后，完成对文件 AUTOEXEC.BAT 的修改。

批处理程序 DOADD.BAT

```
@ECHO OFF
REM DOADD.BAT
CLS
ECHO---UPDATING AUTOEXEC.BAT---
REN AUTOEXEC.BAT *.OLD          把当前的 AUTOEXEC.BAT 更名为 AUTOEXEC.
                                OLD 保存起来。
COPY AUTOEXEC.OLD+ADD.BAT AUTOEXEC.BAT
                                把新的 PATH 命令语句加到 AUTOEXEC.BAT 中。
```

如果准备把 PATH 语句加到 AUTOEXEC.BAT 的开头，只要把 COPY 语句改为：

```
COPY ADD.BAT+AUTOEXEC.OLD AUTOEXEC.BAT
```

即可。

如果原来的 AUTOEXEC.BAT 的最后一行只是一个批处理程序名（没有使用 CALL 语句来调用），执行该程序后，控制无法返回 ATUTOEXEC.BAT 中，新增加的内容又是原 AUTOEXEC.BAT 文件的最后一行，新增加的内容永远得不到执行。这是应该注意的一点。

10.4 查找文件

批处理程序 FILEFIND.BAT，用于在硬盘上查找文件。文件中的 CHKDSK/V 命令，使用了参数/V，将列出所有文件及其所在子目录，FIND 命令用来查找指定的文件，要查找的文件名应以大写的形式输入。不必输入文件名的全称，只要指定的文件名能与其他文件名区别开来即可，但是不能使用通配符。由于使用了命令 CHKDSK，该程序速度较慢。

批处理程序 FILEFIND.BAT

```
ECHO OFF
REM FILEFIND.BAT
IF (%1)==( ) GOTO ERROR
CHKDSK/V | FIND"%1"             用 CHKDSK/V 命令列出所有文件，用 FIND 命令查找指定
                                的文件。
GOTO END
:ERROR
```

```
ECHO Must enter filename to match
GOTO END
: END
```

目前许多通用软件都可用于在磁盘上寻找文件，比如 Norton 和 Mace 实用程序中都包含有一个与 FILEFIND. BAT 功能相同的程序，与 FILEFIND. BAT 相比，这些程序执行速度快，功能强。例如 Norton 实用程序里的 FF(File Find)，在启动时，可根据需要选择不同的参数输入。若在当前子目录输入 FF 命令，则将搜寻所有的子目录，直到找到指定的文件为止。

利用批处理技术编写的查找文件程序，一般不如市场上流行的 PC 工具软件包中的查找文件程序效率高。从 DOS 3.0 开始，增加了设置文件只读或 Archive 属性命令 ATTRIB，它只能对当前子目录中的文件工作。从 DOS 3.3 开始，在当前目录里利用参数/S，ATTRIB 命令可对任何目录内的文件设置 Read-only 或 Archive 档案属性。命令：

```
C>ATTRIB file1. txt/S
```

将在所有目录内查找文件 file1. txt 且将文件属性显示在屏幕上。利用 ATTRIB 命令的这些功能，可建立查找文件的批处理程序 FASTFIND. BAT。

批处理程序 FASTFIND. BAT

```
@ECHO OFF
REM FASTFIND. BAT
IF (%1)==( ) GOTO NOFILE          如果不输入准备查找的文件名，就转到标号 NOFILE。
GOTO FILE
: NOFILE
ECHO No File Entered On Command Line
ECHO Syntax is C>FASTFIND file
ECHO Where file is the file you

ECHO wish to find
GOTO END
: FILE
ATTRIB \%1 /S | MORE
```

程序 FASTFIND. BAT 最后一行中，%1 前面的“\”指示 ATTRIB 从根目录开始查找，参数/S 使 ATTRIB 命令可以在所有子目录内查找，命令 MORE 用于逐屏显示 ATTRIB 列出的文件。该程序必须在 DOS 3.3 或更高的 DOS 版本下运行。与 FILEFIND. BAT相比，该文件速度快，且支持通配符。

程序 FASTFIND. BAT 可查找指定的文件及所在目录。日积月累，硬盘上的批处理程序愈来愈多，要记住每个程序的作用与功能是困难的。利用批处理菜单技术，建立用来显示文件名及其功能的菜单，是解决这一问题的方法之一。建立菜单的方法，前面已有介绍，也可以建立一个批处理程序，帮助记忆每个批处理程序的功能。程序 HELP. BAT 只能对应显示文件名和文件的功能。

批处理程序 HELP. BAT

```
ECHO OFF
REM HELP. BAT
CLS
ECHO DISCARD. BAT Will move all unwanted files
ECHO to a directory for holding
ECHO SYNTAX : DISCARD FILE1 FILE2
ECHO MAINTAIN. BAT Will erase temporary files,sort
ECHO files,and run your file
ECHO defragmentation program
ECHO SYNTAX : MAINTAIN
ECHO WARNING : Take two hours to run
ECHO PRINT. BAT Print ASCII files automatically
ECHO SYNTAX : PRINT FILE1 FILE2
ECHO WARNING : Make sure printer is on
```

HELP. BAT 的运行结果为：

```
DISCARD. BAT    Will move all unwanted files
                to a directory for holding
                SYNTAX : DISCARD FILE1 FILE2
MAINTAIN. BAT   Will erase temporary files,
                sort files, and run your
                file defragmentation program
                SYNTAX : MAINTAIN
                WARNING : Takes two hours to run
PRINT. BAT      Will print ASCII files automatically
                SYNTAX : PRINT FILE1 FILE2 FILE3
                WARNING : Make sure printer is on
                or computer will lock up

C : \>
```

批处理程序 HELP1. BAT 中使用了 IF 语句和形式参数，要求显示程序的功能时，必须在命令行上输入文件名。由于 IF 语句的使用，输入文件名的大小写必须与硬盘上文件名的大小写相同。

批处理程序 HELP1. BAT

```
ECHO OFF
REM HELP1. BAT
CLS
IF (%1)==( ) GOTO NOTHING
IF NOT %1==UTILITY GOTO NOUTIL
```

```
ECHO DISCARD. BAT       will move all unwanted files
ECHO                    to a directory for holding
ECHO                    SYNTAX : DISCARD FILE1 FILE2...
ECHO MAINTAIN. BAT      Will erase temporary files ,sort
ECHO                    files ,and run your
ECHO                    file defragmentation program
ECHO                    SYNTAX : MAINTAIN
ECHO                    WARNING : Takes two hours to run
GOTO END
: NOUTIL
IF NOT %1==PRINTING GOTO NOPRINT
ECHO     PRINT. BAT     Will print ASCII
ECHO                    files automatically
ECHO                    SYNTAX : PRINT FILE1 FILE2 FILE3
ECHO                    WARNING : Make sure printer is on
ECHO                    or computer will lock up
GOTO END
: NOPRINT
IF NOT %1==BACKUP GOTO NOBACK
ECHO BACKUP. BAT Will backup all (or some)
ECHO                subdirectories on your hard disk
ECHO                SYNTAX : BACKUP
ECHO                         to backup entire hard disk
ECHO                SYNTAX : BACKUP directory
ECHO                         to backup one subdirectory
GOTO END
: NOBACK
IF NOT %1==MISC GOTO WRONG
ECHO     GAME. BAT Will bring up the game menu
ECHO               SYNTAX : GAME
GOTO END
: WRONG
ECHO INVALID SYNTAX
GOTO NOTHING
: NOTHING
ECHO SYNTAX IS HELP1 CATEGORY
ECHO ENTER CATEGORY IN ALL UPPER CASE
ECHO Valid categories are :
ECHO Utilities,Backup,Printing,Misc
: END
```

10.5 定期自动运行批处理程序

对于具有系统时钟的微机，可以在预定日期自动运行批处理程序，来完成周期性工作，例如每周都需要进行的硬盘备份工作。下面是实现在每星期三自动进行的硬盘增量备份，每星期五进行硬盘全备份功能的批处理程序自动执行的例子。通过这个例子，读者可以了解编写这类批处理程序的方法，进而利用批处理程序定期执行其他应用程序。

要想在每次开机或重新启时，自动执行批处理程序，首先应在 AUTOEXEC.BAT 末尾加上两行：

```
ECHO | MORE | DATE > STOREDATE.BAT
STOREDATE
```

上面第一行中的 ECHO | MORE 为 DATE 提供了回车键，这是回答 DATE 命令所必须的。DATE>STORE STOREDATE.BAT，利用 DOS 管道操作符“>”自动建立了文件 STOREDATE.BAT，并把下列内容自动输入到文件 STOREDATE.BAT 里：

```
Current date is Mon 10-14-1991
Enter new date (mm-dd-yyyy)
```

第一行中的 Current 调用事先建立的文件 Current.bat，并把四个实际参数传递给 Current.bat，AUTOEXEC.BAT 程序中的最后一行运行程序 STOREDATE.BAT，批处理程序 STOREDATE.BAT 又调用了事先建立的文件 Current.bat 并传递五个实际参数，它们是：

%0	Current
%1	date
%2	is
%3	Mon（每次运行时，随周几不同而改变）
%4	10-14-1991（每次运行时，随日期不同而改变）

批处理程序 STOREDATE.BAT 的第一行：

```
Current date is Mon 10-14-1991
```

即为调用批处理程序 Current.bat 的命令行。由 Current.BAT 具体完成每周三执行增量备份，每周五执行全硬盘备份的工作。程序 Current.BAT 清单如下：

批处理程序 Current.BAT

```
ECHO OFF
REM CURRENT.BAT
IF (%1)==( ) GOTO END          如果第一个实际参数没有输入则退出批处理程序
IF (%2)==( ) GOTO END          如果第二个实际参数没有输入则退出批处理程序
IF (%3)==( ) GOTO END          如果第三个实际参数没有输入则退出批处理程序
IF (%4)==( ) GOTO END          如果第四个实际参数没有输入则退出批处理程序
```

```
REM JUMP TO TESTING SECTION      注释行
GOTO TESTS                       转到标号 TESTS
:BACKUP                          执行硬盘备份命令
BACKUP C:\A: /S
GOTO ENDBACK                     完成后,转到标号 ENDBACK
:PARTBACK                        执行硬盘增量备份命令,完成后转到标号 ENDPART
BACKUP C:\A: /S/M
GOTO ENDPART
REM OTHER OPERATIONS HERE        这只是定期执行批处理程序的一个例子,用户可根据自己
                                 的要求,设计其它部分加在这里
REM JUMP TO THEM USING TESTS
:TESTS                           标号行
IF %3==Fri GOTO BACKUP           如果是星期五,则转到标号 BACKUP 执行硬盘备分
:ENDBACK                         标号行
IF %3==Wed GOTO PRATBACK         如果是星期三,则转到标号 PART BACK,执行增量备份
:ENDPART                         标号行
REM OTHER TESTS HERE             可以在这里加入其他的命令行,以完成其他的任务
:END                             结束批处理程序的标号行
```

在 Current.BAT 中可增加其他一些命令,并在满足某些条件时执行,即可完成用户所要求定期执行的任务。在环境中事先设置日期为环境变量:

```
SET DAY=%3
SET DATE=%4
```

批处理程序就可直接调用:

```
%DAY%
%DATE%
```

在调用时,应把环境变量放置在语句中。

批处理程序 Current.BAT 的缺陷是在每周三或周五每重新启动一次机器,都会产生一个增量备份或者完整备份,有时由于种种原因,需要在一个工作日内重新启动系统,就会产生多余的备份。

解决这一问题的方法是在成功地完成了硬盘备份后,建立一个标志文件。再次重新启动机器时,首先进行判断,如果标志文件存在,则说明已完成了硬盘备份,如果标志文件不存在,则进行硬盘备份,然后在适当的时候把标志文件删除。批处理程序 CURRENT1.BAT 在 CURRENT.BAT 的基础上作了改进,当完成了硬盘备份后,自动建立标志文件。

批处理程序 CURRENT1.BAT

```
ECHO OFF
REM CURRENT1.BAT
IF (%1)==( )GOTO END          如果不输入实际参数,则退出批处理程序
```

```
IF (%2)==( )GOTO END
IF (%3)==( )GOTO END
IF (%4)==( )GOTO END
GOTO %3                          这里用形式参数%3作为转向标号,转到的具体位置是实际参
                                 数(周一至周日)的
:Sun                             标号行
REM Sunday commands go here      如果有周日要执行的命令,应放在这里,同时应建立的标志文
                                 件FLAGSUN,用来标志星期日的任务已完成,以避免在周日
                                 重新启动机器时,做不必要的工作
IF EXIST FLAGFRI DEL FLAGFRI     如果存在标志文件,则加以删除
IF EXIST FLAGWED DEL FLAGWED
GOTO END                         退出批处理程序,以避免其它日期命令的执行
:Mon                             执行应在周一执行的命令
REM Monday commands
IF EXIST FLAGFRI DEL FLAGFRI
IF EXIST FLAGWED DEL FLAGWED
GOTO END
:Tue                             执行周二应做的工作
IF EXIST FLAGFRI DEL FLAGFRI
IF EXIST FLAGWED DEL FLAGWED
GOTO END
:Wed                             执行周三应做的工作
REM Wednesday commands
IF NOT EXIST FLAGWED BACKUP C:\A: /S
                                 如果文件FLAGWED不存在,则执行硬盘备份命令。备份
                                 完成后,建立标志文件FLAGWED,这样在星期三重新启
                                 动时,不会进行第二次备份工作。
REM Create flag file             建立标志文件
TYPE Nofile > FLAGWED            这条命令建立了长度为零的标志文件,它不占用磁盘空间。
                                 应注意的是文件Nolile必须事先不存在。
IF EXIST FLAGFRI DEL FLAGFRI     因为不是星期五,故删除FLAGFRI
GOTO END                         退出批处理程序
:Thu                             周四
IF EXIST FLAGFRI DEL FLAGFRI
IF EXIST FLAGWED DEL FLAGWED
GOTO END
:Fri                             周五
REM Firday commands              制作备份并建立标志文件FLAGFRI(与周三同)
IF NOT EXIST FLAGFRI BACKUP C:\A: /S/M
TYPE Nofile > FLAGFRI
```

```
IF EXIST FLAGWED DEL FLAGWED
GOTO END
:Sat                                    周六
REM Saturday commands
IF EXIST FLAGFRI DEL FLAGFRI
IF EXIST FLAGWED DEL FLAGWED
GOTO END
:END                                    结束批处理的程序标号行
```

10.6 间或地执行 DOS 命令

前一节介绍的定期运行批处理程序的方法，可在预定的日期自动运行批处理程序来完成所规定的任务，但是如果在预定的日期没有开机(例如星期三或星期五)，则预定的任务将无法完成(例如上一节中的硬盘备份)。针对这种情况，可设计一种批处理程序，规定在计算机启动次数达到某一数值时，执行某些命令，以完成所规定的任务。

一般情况下，利用文件 AUTOEXEC.BAT，可自动执行 DOS 命令或者批处理程序。在 DOS 提示符下键入批处理程序名也可以运行批处理程序，但是在某些情况下，不是每次启动机器都要执行用户指定的命令，而是启动次数达到指定的数目后，再加以执行。

批处理程序 OCCASIONAL.BAT 说明了这种方法。可把这个文件名放在 AUTOEXEC.BAT 中，即每次开机可自动执行 OCCASIONAL.BAT，如果使用的是 DOS 3.3或更高的版本，可用 CALL 语句调用。否则应使用语句 COMMAND/C OCCASIONAL.BAT 来调用，而且不要忘记文件 OCCASIONAL.BAT 的最后一行应为 EXIT，这样可保证在执行完 OCCASIONAL.BAT 后，能返回 AUTOEXEC.BAT。

批处理程序 OCCASIONAL.BAT

```
@ECHO OFF
FOR %J IN (00 01 02 03 04 05 06 07 08 09) DO IF
 EXIST COUNT.%%J GOTO %%J
                                        利用长度为零的记数文件 count.00,count.01 等来确定上次
                                        运行是第几次运行，并利用 FOR 语句转移到适当的位置
GOTO NOFILE                             如果所有的计数文件都不存在，则转到标号 NOFILE
:00                                     如果文件 count.00 存在则转到这里
REM ALL Work Done Here                  所有命令将在这里执行
BACKUP C:\A:                            执行备份命令
REN COUNT.00 COUNT.01                   重新命名文件
GOTO END                                退出批处理程序
:01                                     如果文件 COUNT.01 存在则转到这里
REN COUNT.01 COUNT.02                   把文件重新命名
GOTO END                                退出批处理程序
:02
```

```
REN COUNT.02 COUNT.03
GOTO END
:03
REN COUNT.03 COUNT.04
GOTO END
:04
REN COUNT.04 COUNT.05
GOTO END
:05
REN COUNT.05 COUNT.06
GOTO END
:06
REN COUNT.06 COUNT.07
GOTO END
:07
REN COUNT.07 COUNT.08
GOTO END
:08
REN COUNT.08 COUNT.09
GOTO END
REN COUNT.09 COUNT.00
GOTO END
:NOFILE                    如果记数文件不存在,则转到这里
REM Creat Counter File     注释行
REM Then Restart process
TYPE NOFILE > COUNT.00     建立长度为零的记数文件
GOTO 00                    转到标号 00,执行 BACKUP 命令
:END                       结束批处理程序运行的标号行
```

在OCCASIONAL.BAT 中有长度为零的计数文件,当 OCCASIONAL.BAT 首次运行时,建立计数文件 COUNT.00,备份完成后,把 COUNT.00 重新命名为 COUNT.01;第二次运行时,只把 COUNT.01 重新命名为 COUNT.02,并不执行备份命令;这样,每启动一次机器,只改变计数文件名,当文件名为 COONT.09 时,把它重新命名为 COUNT.00,并执行一次硬盘备份命令.文件 OCCASIONAL.BAT 的计数文件记录的是计算机的启动次数,而不是当前日期.可以看出,每启动十次即执行一次硬盘备份工作,实现了间或执行命令的目标。批处理程序 OCCASIONAL.BAT 也有缺陷,比如在突然断电,发生硬件故障或者运行应用程序时发生"死机"现象时,就不得不在一天内多次启动机器,当启动次数达到十次时,就会执行备份命令,这种情况下产生的硬盘备份不是用户所需要的。另一问题是如果长时间不关机,比如几周的时间,用户就不能及时得到硬盘备份。

值得一提的是程序 OCCASIONAL.BAT 中,使用了两个技巧:一是第二行中的 IF 语句,把 10 项检验结合在一起,由一个 FOR 语句来完成;另一技巧是使用了长度为零的

计数文件。

文件 COOASIONAL. BAT 执行的只是硬盘备份命令，用户可以编写类似的批处理程序，间或地完成自己的工作，比如检查病毒、删除暂时建立的文件等。

10.7 节省磁盘空间

计算机使用的软盘是由磁道组成的，磁道是一些同心的圆环，由磁盘的中心开始，逐渐向外展开。不同类型软盘的磁道数目可能不同，例如一张双面双密的 5.25 英寸的软盘每面有 40 条磁道，磁道进一步划分为扇区，每个扇区字节数是由磁盘类型决定的，一个或几个扇区组成盘簇。DOS 以盘簇为单位给文件分配磁盘空间。

执行列文件目录的命令 DIR 时所看到的每个文件名后面的字节数只是文件的长度，不是文件实际占用的磁盘空间。每个文件至少占用一个盘簇(长度为零的文件除外)。如果在盘簇大小为 2048 字节的硬盘上执行显示磁盘当前状态命令 CHKDSK：

```
C>CHKDSK
Volume 481009-001 creaed Jan 1,1992 10.04a
  1309440 bytes total disk space
    258048 bytes in 7 hidden files
     20480 bytes 7 directosies
19648512 bytes available on disk

653312 bytes total memory
597184 bytes free
```

然后建立一个非常短小的文件 JUNK. TXT

```
C>COPY CON: JUNK.TXT
VERY SMALL FILE
^Z
1 File(s) copied
```

再执行列目录命令

```
C>DIR

Volume in drive c is 481009-001
Directory of C:\

DOS          <DIR>      1-01-91 10:49p

KS           <DIR>      1-01-91 10:49a

BAT          <DIR>      1-10-91 15:12a
```

```
JUNK        TXT        17      8-26-92          9：10a

4 File(s) 19040464 bytes free
```

可看到，刚刚建立的文件 JUNK. TXT 的长度为 17 个字节，然后再执行一次查盘命令

```
C>CHKDSK
Volume 481009-001 created Jan 1,1992 12：04a
21309440 bytes total disk space
   258048 bytes in 7 hidden files
    20480 bytes in 7 diurectories
   138448 bytes in 287 user files
19646464 bytes available on disk

   653312 bytes total memory
   597184 bytes free
```

可以看到硬盘上的剩余空间为 19646464 字节，比文件 JUNK. TXT 建立前减少了 2048 字节(2K)，也就是说虽然文件 JUNK. TXT 的长度只有 17 个字节，却占用了 2K 字节的磁盘空间。在每簇大小为 2048 字节的硬盘上，每个文件至少占用 2048 字节的磁盘空间。即使文件的长度为 1 个字节，也必须占用 2048 个字节的磁盘空间，如果一个文件的长度为 2049 个字节，则必须占用 2 个盘簇即 4096 个字节。

根据硬盘的种类及所使用的 DOS 版本，硬盘每簇的大小从 1K 至 8K 不等。如果硬盘上存放了很多短小的文件(例如批处理程序)，必然会浪费大量的磁盘空间，而且盘簇的尺寸愈大，浪费的愈多。例如在硬盘上 C：\BAT 子目录中存放了一批批处理程序，使用 Norton 实用程序中的 FILESIZE 来检查这些批处理程序长度之和以及它们实际占用的磁盘空间。

```
C：\BAT>FILESIZE *.BAT
FS-File Size,Advanced Edition,(C) Copr 1987,
Peter Norton
C：\BAT

1. bat              128 bytes
2. bat               31 bytes
3. bat              128 bytes
4. bat              128 bytes
5. bat              128 bytes
6. bat               29 bytes
69. bat             128 bytes
archive. bat        221 bytes
bookcopy. bat       256 bytes
```

```
bookmove. bat          40 bytes
c. bat                 128 bytes
d. bat                 128 bytes
discard. bat           18 bytes
hardcopy. bat          896 bytes
hardxcop. bat          896 bytes
index286. bat          284 bytes
maintain. bat          512 bytes
menu. bat              128 bytes
newpromp. bat          128 bytes
redo-d. bat            384 bytes
tob&a. bat             384 bytes

    5,203 total bytes in 21 files
    43,008 bytes disk space occupied, 88% slack
```

从运行 FILESIZE 的显示结果可以看出，C：\BAT 子目录里所有批处理程序长度的和为 5203 字节，但是实际占有的盘磁空间为 43008 字节。文件的总长度只有实际占有磁盘空间的 12%，即浪费了 88%的磁盘空间。

对于这个问题，目前尚没有完美的解决办法。这里所介绍的两种方法都不完善。一是把所使用的 DOS 版本升级，一般情况，DOS 版本愈高，盘簇的尺寸愈小。

更新 DOS 版本的缺点是繁琐，首先必须把硬盘上的所有文件备份下来，重新格式化硬盘，然后把新的 DOS 版本和已备份的文件装到硬盘上。硬盘上文件的长短是不一样的，有些应用程序很长，多数批处理程序都很短，例如平均每个文件浪费半个盘簇。通过 DOS 版本升级，如果盘簇由 8K 降为 4K，那个每个文件平均浪费的磁盘空间即由 4K 降为 2K。一张 20M 的硬盘，从 DOS2.1 升级为 DOS3.2，盘簇由 8K 降为 4K，至少可释放 4M 字节的磁盘空间。

解决这一问题的第二个办法是把多个短小文件，组成一个长文件。下面的批处理程序 BIGMENU. BAT 是由 6 个批处理程序组成的，6 个程序名分别作为运行 BIGMENU. BAT的实际参数，在命令行上键入。

批处理程序 BIGMENU. BAT

```
ECHO OFF
REM BIGMENU.BAT
REM THIS BATCH FILE REPLACES      组成 BIGMENU. BAT 的 6 个短小的批处理程序。
1.BAT,2.BAT,3.BAT,4.BAT,5.BAT and 6.BAT
IF (%1)==( )GOTO NOCODE
                                  如不输实际参数(文件名),转到标号 NOCODE。
IF %1==1 GOTO ONE                 输入的实际参数(文件名)正确,转到指定的标号,执行批处理程序。

IF %1==2 GOTO TWO
```

```
IF %1==3 GOTO THREE
IF %1==4 GOTO FOUR
IF %1==5 GOTO FIVE
IF %1==6 GOTO SIX
GOTO NOTRIGHT          输入的实际参数(文件名)不正确,转到标号 NO TRIGHT。
:ONE                   执行程序 1.BAT
C:
CD\WORD
WORD %2
GOTO MENU              转到标号 MENU,重新显示菜单。
:TWO                   执行程序 2.BAT
C:
CD\123
123
GOTO MENU
:THREE                 执行程序 3.BAT
C:
CD\DBASE
DBASE %2 %3
GOTO MENU
:FOUR                  执行程序 4.BAT
C:
CD\GAME
XMAN
GOTO MENU
:FIVE                  执行程序 5.BAT
FORMAT A: /V
GOTO MENU
:SIX                   执行程序 6.BAT
BACKUP C:\A: /S
GOTO MENU
:NOCODE                如果没有输入文件名,转到这里。
CLS
ECHO ENTER BIGMENU FOLLOWED
                       告诉用户如何运行 BIGMENU.BAT
BY A MENU CODE TO RUN A PROGRAM
ECHO FOR EXAMPLE BIGMENU 1 TO RUN WORD PROCESSOR
PAUSE                  暂停,给用户阅读信息的时间。
GOTO MENU
:NOTRIGHT              输入了错误的文件名,转到这里。
ECHO YOU ENTERED AN INCORRECT CODE
                       告诉用户应输入正确的文件名
```

```
ECHO ONLY THE CODES 1-6 ARE ALLOWED
ECHO ENTER BIGMENU FOLLOWED
BY A MENU CODE TO RUN
A PROGRAM
ECHO FOR EXAMPLE BIGMENU 1
ECHO TO RUN WORD PROCESSOR
PAUSE
:MENU
REM ENTER COMMANDS HERE TO DISPLAY MENU
                                    假设存在菜单程序 MENU.COM 或者 MENU.EXE,则把调菜单程
                                    序命令写在这里。
REM FOLLOWING COMMAND ASSUMES
A PROGRME CALLED MENU.COM OR MENU.EXE EXISTS
MENU                                显示菜单
```

文件 BIGMENU.BAT 中的最后一行调用菜单显示程序。所显示的菜单至少应包括 7 项选择,前 6 项分别用来执行 6 个批处理程序,最后一项应为 EXIT,即返回 DOS。

这种方法可以节省磁盘空间,存在的主要缺陷是编写,调试时比较困难。由多个短小文件组成的批处理运行时,运行速度也比单独文件慢,但运行结果都是一致的。

如果一个批处理程序,只有一个或几个 DOS 命令,且这几个命令在文件中多次重复出现,则可以用单个字母作为环境变量,把命令存在环境空间里,用环境变量代替文件中的命令,可以缩短整个文件的长度,节省磁盘空间。文件 SAVESPACE.BAT 中,分别用环境变量 E,N 和 O 取代 IF ERRORLEVEL,IF NOT ERRCRLEVEL 和 ECHO,缩短了每行的长度。

批处理程序 SAVESPACE.BAT

```
@ECHO OFF
REM SAVESPACE.BAT
SET E=IF ERRORLEVEL      设置三个环境变量。文件中命令出现的位置,分别用三个环境变量
                         代替。
SET N=IF NOT ERRORLEVEL
SET O==ECHO
%E% 255 %O% 255          如果不调用环境变量,该行为:
                         IF ERRORLEVEL 255 ECHO 255
%E% 254 %N% 255 %O% 254
%E% 2 54 %N% 254 %O% 253
        ... ...
%E% 1 %N% 2 %O% 1
%E% 0 %N% 1 %O% 0
```

SAVESPACE.BAT 的长度为 6477 字节,如果不使用环境变量,它的长度增加几乎

一倍。用这种方法建立批处理程序，同样存在调试困难、运行速度慢等缺点，同时需要较大的环境空间。

10.8 保持 ERRORLEVEL 值

支持 ERRORLEVEL 的 DOS 命令执行后，返回长度为一个字节的 ASCII 码值(0—255)给 ERRORLEVEL，作为错误级别的出口码。DOS 命令正确执行之后，错误级别的出口码为 0，否则为其他的数字。每当 DOS 开始执行命令或者程序时，无论该命令或者程序是否支持 ERRORLEVEL，都先把 ERRORLEVEL 重新设置为零。

在批处理程序文件中，有时需要连续调用几个应用程序，下一个程序的执行又取决于上一个程序执行后返回的 ERRORLEVEL 值，这样必须设法保持上一个程序执行后返回的 ERRORLEVEL 值不变。

CLOSE. BAT 是一个执行结算银行账户的批处理程序，可以实现日结算，月结算和年结算。运行时首先询问要求执行日结算、月结算还是年结算。如果执行日结算，则调用文件 DAILY. COM；如果执行月结算，则必须先执行 DAILY. COM 进行日结算，再调用文件 MONTHLY. COM，进行月结算；如果实行年结算，则必须先进行日结算和月结算，然后调用 YEARLY. COM 进行年结算。由于没有保持 NORTON 实用程序设置的 ERRORLEVEL 值，因此在 CLOSE. BAT 中，DAILY. COM 出现三次，MONTHLY. COM 出现两次。

批处理程序 CLOSE. BAT

```
@ECHO OFF
REM CLOSE. BAT
C:
CD\CLOSING
BE ASK"D)aily,M)onthly or Y)early Closing?"DMY
```

用 NORTOR 中的 ASK 程序询问用执行日结算、月结算还是年结算。根据用户的回答，ASK 程序把 ERRORLEVEL 分别设置为 1,2 或 3。

```
IF ERRORLEVEL 3 GOTO 3
```

如果用户回答 Y 实行年结算，转到标号 3。

```
IF ERRORLEVEL 2 GOTO 2
```

如果用户回答 M 实行月结算，转到标号 2。

```
IF ERRORLEVEL 1 GOTO 1
```

如果用户回答 D 实行日结算，转到标号 1。

```
GOTO ERROR
```

如果用户回答的不是 D、M 或 Y，转到标号 ERROR。

```
:1
DAILY/First National Bank/Branch3/Teller4
```

调用程序 DAILY. COM，实行日结算。

```
GOTO END
:2
DAILY/First National Bank/Branch3/Teller4
```

调用程序 DAILY. COM 和 MONTHLY. COM 实行月结算。

```
MONTHLY/First National Bank/Branch3/Manger1
GOTO END
:3
DAILY /Frist National Bank/Branch3/Teller4
```

调用三个程序实行年结算。

```
MONTHLY/First Nationd Bank/Branch3/Manger1
YEARLY/First National Bank/Branch3/Manger2
:ERROR
ECHO Enter D for DAILY CLOSING
```

告诉用户如何正确使用程序 CLOSE.BAT。

```
ECHO Enter M for MONTHLY CLOSING
ECHO Enter Y for YEARLY CLOSING
GOTO END
:END
```

在CLOSE.BAT 中，调用了 NORTON Utilities 中的 ASK 程序。同其它批处理命令一样，批处理程序可以随意调用 ASK 程序，但 ASK 程序必须已在某个子目录下。

如果能够保持 ASK 程序设置的 ERRORLEVEL 值，可以简化 CLOSE.BAT。在 CLOSE.BAT 中，无论执行何种结算，都必须先执行日结算。CLOSE.BAT 中的标号 2 部分执行月结算，由于在标号 1 部分执行 DAILY.COM 时，已把 ERRORLEVEL 归零，标号 2 部分不能用语句：

```
IF ERRORLEVEL 1 MONTHLY
```

来检验 DAILY.COM 是否已执行，因此标号 2 部分只有重复调用 DAILY.COM。在标号 3 部分，需重复执行文件 DAILY.COM 和 MONTHLY.COM，道理是一样的，其结果使 COLSE.BAT 的长度增加，同时延长了运行时间。ERRORLEVEL 值的范围为 0—255，常用的只有少数几个，一般在 0—10 之间。设法把 ERRORLEVEL 值赋给环境变量，把 ERRORLEVEL 值保持在环境里，在运行一个程序或者执行 DOS 命令时，不会由于 ERRORLEVEL 归零而丢失。同时在用 IF 语句检验环境变量时，是作“等于”检验，而不是用 IF 语句检验 ERRORLEVEL 值的“大于或等于”，这样逻辑更为严谨。批处理程序 KEEP.BAT可把 10 以内的 ERRORLEVEL 值赋给环境变量 ERROR。

批处理程序 KEEP.BAT

```
REM KEEP.BAT
FOR %%J IN (0 1 2 3 4 5 6 7 8 9 10) DO IF ERRORLEVEL %%J SET
ERROR=%%J
ECHO ERRORLEVEL IS %ERROR%
```

KEEP.BAT 中的 FOR 语句相当于 11 个 IF 语句，

即：IF ERRORLEVEL 0 SET ERROR=0

```
IF ERRORLEVEL 1 SET ERROR=1
    ... ...
    ... ...
IF ERRORLEVEL 10 SET ERROR=10
```

上述 11 个 IF 语句从小到大检验 ERRORLEVEL 值，在未获得当前 ERRORLEVEL 值之前，IF 语句中的 SET 命令是执行了的，环境变量 ERROR 的值不断更新，当 ERROR 得到了当前的 ERRORLEVEL 值后，其值将保持不变，其后的 IF 语句中的 SET 命令将不执行。

利用 KEEP. BAT，建立 CLOSE2. BAT。

批处理程序 CLOSE2. BAT

```
@ECHO OFF
REM CLOSE2.BAT
C:
CD\CLOSING
BE ASK "D)aily, M)onthly or Y)early Closing? " DMY
CALL KEEP.BAT          调用 KEEP. BAT，把 ASK 设置的
                       ERRORLEVEL 值赋给环境变量 ERROR。
IF NOT %ERROR%==3 IF NOT %ERROR%==2 IF NOT %ERROR%==1 GOTO ER-
ROR
                       如果用户输入的不是 D，M 或 Y，转到
                       标号 ERROR。
  DAILY/First National Bank/Branch3/Teller4
                       实行日结算。由于实行月结算或年
                       结算时，都必须先实行日结算，故不需
                       用 IF 语句检验 ERROR 是否为 1。
  IF NOT %ERROR%== 1 MONTHLY/First National Bank/Branch3/Manager1
                       实行月结算。如果 ERROR 等于 1，只
                       进行日结算；若 ERROR 的值为 2 或 3，都
                       应进行月结算。注意这里的 IF 语句
                       进行的是"等于"而不是"大于或
                       等于"检验。
  IF %ERROR%==3 YEARLY/First NationalBank/Branch3/Manger2
                       实行年结算。
GOTO END
:ERROR
ECHO Enter D for DAILY CLOSING.
ECHO Enter M for MONTHLY CLOSING.
ECHO Enter Y for YEARLY CLOSING.
GOTO END
:END
```

和 CLOSE. BAT 相比，DAILY. COM，MONTHLY. COM 及 YEARLY. COM 在 CLOSE2. BAT 都只出现了一次。CLOSE2. BAT 的运行速度及文件结构都优于 CLOSE. BAT。KEEP. BAT 只涉及了 11 个 ERRORLEVEL 值，如果事先不知道 ERRORLEVEL 值的范围，就可用相类似的办法，把 KEEP. BAT 的应用范围扩大。SAVE-BIG. BAT 是在 KEEP. BAT 基础上建立起来的，可以看出，虽然 SAVE-BIG. BAT 可把 ERRORLEVEL 值赋给环境变量 ERROR，但该文件显得冗长。

批处理程序 SAVE-BIG. BAT

```
@ECHO OFF
REM SAVE-BIG. BAT
SET ERROR=0
FOR %%J IN (1 2 3 4 5 6 7 8 9 10)                    把 ASCII 码值 1—240 分
DO IF ERRORLEVEL %%J SET ERROR=%%J 为每 10 个数一组进 行检验，由于 ERROR 已
                                                     设置为零，所以这 里从 1 开始。

FOR %%J IN (11 12 13 14 15 16 17 18 19 20)
DO IF ERRORLEVEL %%J SET ERROR=%%J
FOR %%J IN (21 22 23 24 25 26 27 28 29 30)
DO IF ERRORLEVEL %%J SET ERROR=%%J
      ... ...
      ... ...
FOR %%J IN (231 232 233 234 235 236 237 238 239 240)
DO IF ERRORLEVEL %%J SET ERROR=%%J
FOR %%J IN (241 242 243 244 245 246 247 248 249 250)
DO IF ERRORLEVEL %%J SET ERROR=%%J
FOR %%J IN (251 252 253 254 255)
DO IF ERRORLEVEL %%J SET ERROR=%%J
ECHO ERRORLEVEL is %ERROR%
```

KEEP. BAT 和 SAVE-BIG. BAT 的目的都是在找出当前 ERRORLEVEL 值后，设置给环境变量 ERROR。

ERRORLEVEL 值的范围为 0 到 255，最小的是一位数零，最大的是三位数 255。一个比 SAVE-BIG. BAT 更简便的方法是分别确定 ERRORLEVEVL 值中三位数中的每个数码。首先确定百位上的数码，它只能是 0、1 或 2 三个数码之中的一个，只需一个循环语句即可实现。

批处理程序 DIGIT-1. BAT

```
@ECHO OFF
REM DIGIT-1. BAT

FOR %%J IN (0 1 2)        此语句把 0—255 分成 0—99，100—199，200—255 三个区域，并把百
                          位上的数码存在环境变量 ERROR 里。
```

```
DO IF ERRORLEVEL %%J00
SET ERROR=%%J
ECHO Left Digit is %ERROR%
```

DIGIT-2.BAT 确定十位上的数码，如果把 ERRORLEVEL 值为二位数，其十位上的数码为 0—9 中的一个。

批处理程序 DIGIT-2.BAT

```
@ECHO OFF
REM DIGIT-2.BAT
FOR %%J IN (0 1 2 3 4 5 6 7 8 9)      用 FOR 语句确定 ERRORLEVEL 值的中间一位数码
DO IF ERRORLEVEL %ERROR%%%J0
SET ERROR=%ERROR%%%J
ECHO Left Two Digits are %ERROR%
```

由批处理程序 DIGIT-3.BAT 调用 DIGIT-1.BAT 和 DIGIT-2.BAT，找出 ERRORLEVEL 值百位和十位的数码后，用同样的方法确定个位数码。

批处理程序 DIGIT-3.BAT

```
@ECHO OFF
REM DIGIT-3.BAT
SET ERROR=
CALL DIGIT-1
CALL DIGIT-2
FOR %%J IN (0 1 2 3 4 5 6 7 8 9)
DO IF ERRORLEVEL %ERROR%%%J
SET ERROR=%ERROR%%%J
ECHO ERRORLEVEL is %ERROR%
```

如果 ERRORLEVEL 值在 0 至 199 之间，程序 DIGIT-3.BAT 运行正常。当 ERRORLEVEL 值大于或等于 200 时，可能会导致把查找 ERRORLEVEL 值的范围扩大，即达到 299，而 ERRORLEVEL 的取值范围为 0—255，为了保证查找范围不超过 255，在确定了左边数码为 2 时，必须保证中间一位数码不超过 5。左边一位数码为 2，中间一位数码为 5 时，必须保证右边一位数码小于或等于 5，这样可确保在 0—255 的范围内查找 ERRORLEVEL 值。SAVE-ERR.BAT 是一个完整的保持 ERRORLEVEL 值不超出取值范围的批处理程序。

批处理程序 SAVE-ERR.BAT

```
@ECHO OFF
SET ERROR=
REM SAVE-ERRORLEVEL.BAT
FOR %%J IN (0 1 2) DO IF ERRORLEVEL %%J00 SET ERROR=%%J
IF %ERROR%==2 GOTO 2
FOR %%J IN (0 1 2 3 4 5 6 7 8 9) DO IF ERRORLEVEL %ERROR%%%J0 SET ERROR=
```

```
%ERROR%%%J
FOR %%J IN (0 1 2 3 4 5 6 7 8 9) DO IF ERRORLEVEL %ERROR%%%J SET ERROR=
%ERROR%%%J
GOTO END
:2
FOR %%J IN (0 1 2 3 4 5) DO IF ERRORLEVEL %ERROR%%%J0 SET ERROR=%ER-
ROR%%%J
IF %ERROR%==25 GOTO 25
FOR %%J IN (0 1 2 3 4 5 6 7 8 9) DO IF ERRORLEVEL %ERROR%%%J SET ERROR=
%ERROR%%%J
GOTO END
:25
FOR %%J IN (0 1 2 3 4 5) DO IF ERRORLEVEL %ERROR%%%J SET ERROR=
%ERROR%%%J
:END
```

10.9 实际参数的传递技巧

在批处理程序里可以设置形式参数，执行批处理程序时，在键入文件名的同时，也键入用来替换形式参数的实际参数。批处理程序可以调用 PC 工具软件包中的某些程序，比如 Norton Utilities 中的查找字符段程序 Text Search（TS.EXE），调用时只要键入：

```
TS *.BAT "ECHO OFF"
```

所要查找的字符段 ECHO OFF 必须用引号括起来。上述命令将在当前子目录内的所有批处理程序里查找字符段 ECHO OFF。如果不用引号，TS 程序只查找 ECHO。在批处理程序 TEXFIND.BAT 里调用 TS 程序查找 ECHO OFF，并且把查找的字符段作为实际参数传递给文件里的形式参数：

```
TESTFIND *.BAT"ECHO OFF"
```

实际参数和形式参数的对应关系为：

*.BAT	%1
"ECHO	%2
OFF"	%3

这里因为 DOS 总是把实际参数之间的空格视作分隔等，而不是把空格视作组成实际参数的一个字符。

批处理程序 TEXTFIND.BAT

```
@ECHO OFF
REM TEXTFIND.BAT
TS %1 %2 %3 %4 %5
```

由于没有给形式参数%4 与%5 输入实际参数，DOS 将认为它们不存在。在上面执行文件 TEXTFIND. BAT 的命令行上使用了一般不用作实际参数的引号作为实际参数。为了与通常的作法一致，可以设法由批处理程序把引号加上。

批处理程序 TEXTFND2. BAT

```
@ECHO OFF
REM TEXTFND2. BAT
TS %1 "%2 %3 %4 %5"
```

TEXTFND2. BAT 中，把四个形式参数用引号括起来，执行时，在输入的实际参数小于或等于四个时，它工作得很好。但如果实际参数多于四个，则 TS 程序只查找前四个，文件 TEXTFND3 是解决这个问题的另外一种尝试。

批处理程序 TEXTFND3. BAT

```
@ECHO OFF
REM TEXTFND3. BAT
TS %1 "%2%3%4%5%6%7%8%9"
```

TEXTFND3. BAT 把形式参数%2 至%9 放在引号内，且相互之间无空格，其目的是准备把包括空格在内的几个实际参数传递给它们，如果在屏幕上键入：

```
TEXTFND3 *. BAT REM ECHO IS OFF
```

那么实际参数与形式参数的对应关系为：

*. BAT	%1
空格	%2
REM	%3
空格	%4
ECHO	%5
空格	%6
IS	%7
空格	%8
OFF	%9

文件 TEXFND3. BAT 调用 TS 时的命令实际为 TS *. BAT"REM ECHO IS OFF"，这里忽略了一个重要问题，即空格不能作为实在参数，只能作为实际参数的分隔符，因此这种作法也不能完成这类复杂实际参数的传递。下面的两个批处理程序较好地解决了这种复杂实际参数的传递问题，它们都可以根据实际参数的多少来自动构成调用 TS 程序语句，程序 TEXTFND4. BAT 最多可接受 9 个实际参数。程序 TEXTFND5. BAT 中使用了 SHIFT 命令，因此它所能接受的实际参数没有限制。

批处理程序 TEXTFND4. BAT

```
@ECHO OFF
REM TEXTFND4. BAT
IF (%1)==( ) GOTO ERROR1      如不输入实际参数则转到标号 ERROR1。
```

```
IF (%2)==( ) GOTO ERROR2      该文件至少需输入两个实际参数，
                              如只输入了一个，则转到标号 ERROR2。
IF (%3)==( ) TS %1 %2         执行 TS 程度，然后退出。
IF (%3)==( ) GOTO END
IF (%4)==( ) TS %1 "%2 %3"
IF (%4)==( ) GOTO END
IF (%5)==( ) TS %1 "%2 %3 %4"
IF (%5)==( ) GOTO END

IF (%6)==( ) TS %1 "%2 %3 %4 %5"
IF (%6)==( ) GOTO END
IF (%7)==( ) TS %1 "%2 %3 %4 %5 %6"
IF (%7)==( ) GOTO END
IF (%8)==( ) TS %1 "%2 %3 %4 %5 %6 %7"
IF (%8)==( ) GOTO END
IF (%9)==( ) TS %1 "%2 %3 %4 %5 %6 %7 %8"
IF (%9)==( ) GOTO END
TS %1 "%2 %3 %4 %5 %6 %7 %8 %9"
                              如果输入了 9 个实际参数，文件可执行到这一行。可以看出，
                              由于没有用 SHIFT 命令，最多只能接受 9 个实际参数。
GOTO END
:ERROR1
ECHO No File to Search for Specified
GOTO END
:ERROR2
ECHO NO Text to Search For Specified
GOTO END
:END
```

批处理程序 TEXTFND4.BAT 根据输入实际参数的个数，自动构成调用 TS 程序的命令，其缺点是最多只能寻找 8 个字符段。与 TEXTFD4.BAT 相比；TEXTFND5.BAT 的功能更强。它通过循环，使用了输入的所有实际参数，并在最后用两个环境变量构成了调用 TS 的命令，其中%FILE%是文件名，%TEXT%包括所要查找的字符段。

批处理程序 TEXTFND5.BAT

```
@ECHO OFF
REM TEXTFND5.BAT
REM Allows multiple word searches
IF (%1)==( ) GOTO NOFILE      如果用户没输入第一个实际参数(文 件名)，则转向标号
                              NOFILE。
SET FILE=%1                   由于后面使用了 SHIFT 命令，为了保证文件名不丢失，首先
                              把文件名赋给环境变量。
```

```
SHIFT                                   把所有实际参数左移一个位置。
IF (%1)==( ) GOTO NOTEXT                如果用户只输入了文件名而没有输
                                        入所要寻找的字符段,则转向 NOTEXT 标号行。
SET TEXT="%1                            把第一个引号加在所要寻找的第一
                                        个字符段的左边,并赋给环境变量 TEXT。
SHIFT                                   把所有实际参数左移一个位置。
IF (%1)==( ) GOTO STOP                  如果只输入了两个实际参数(文件名
                                        和一个准备查找的字符段),则转向标号 STOP.
:TOP                                    循环体的开始行。
SET TEXT=%TEXT% %1                      除所要寻找的第一个字符段外,这
                                        一行把所有其余字符段都赋给环境
                                        变量 TEXT,每个字符段之间用空格分开。
SHIFT                                   把所有实际参数左移一个位置。
IF (%1)==( ) GOTO STOP                  如果实际参数已用尽,则转向标号行 STOP。
GOTO TOP                                继续执行循环。
:STOP                                   STOP 标号行。
SET TEXT=%TEXT%"                        加上右边的引号。
GOTO START                              转向标号行 START。
:NOFILE                                 NOFILE 标号行。
ECHO NO files to search entered
                                        告诉用户没输入文件名。
GOTO END                                退出批处理程序。
:NOTEXT                                 如果只输入文件名,则转到这里。
ECHO NO test to search for entered
                                        告诉用户没有输入所要搜寻的字符段。
GOTO END                                退出批处理程序。
:START                                  START 标号。
TS %FILE% %TEXT%                        调用 TS 程序。
END                                     结束标号行。
```

仔细分析 TEXTFND5.BAT,可将其划分为 5 个部分。循环体的开始标号行 TOP 之前为第一部分,这部分处理实际参数少于或等于两个(文件名与第一个字符段)的情况。第二部分从 TOP 开始到 GOTO TOP 为止,是循环体,作用是把其余的字符段赋给环境变量,并用空格把每个字符段分开。从标号 STOP 开始至 GOTO START 共三行为第三部分,这部分把所有准备寻找的字符段右边的引号加上,并转到 START 标号行,去执行字符段查找任务。第 4 部分从 NOFILE 开始,再下数 6 行,为处理错误部分,如果不输入文件名或者要查找的字符段,则转到这部分,显示信息,告诉用户应输入的内容。最后余下的三行为第五部分,调用 Norton 中的 TS 程序,查找字符段。

上面两个批处理程序都把执行 TS 程序所需要的引号加上了,不用用户去加引号。如果要从 *.BAT 文件里查找 REM ECHO IS OFF,这两个批处理程序调用 TS 程序的真实命令为:

TS *.BAT""REM ECHO IS OFF""

所要查找字符段的两边都多了一个引号，这不会影响 TS 程序的执行。

10.10 IF 语句的嵌套

在批处理程序中，把两个或两个以上的 IF 语句写在同一行，即构成了 IF 语句的嵌套：

IF condition1 IF condition2 ... ... DOS command

例如：

IF %1==BAK IF %2==OLD IF %3==TXT DEL *.%1

当 DOS 命令需同时满足多项条件才执行时，把 IF 语句嵌套起来，可加快批处理程序的运行速度，缩短批处理程序的长度。批处理程序 CHECKERR.BAT，对于每个 ERRORLEVEL 值用两个单独的 IF 语句来进行检验。可能存在的 ERRORLEVEL 值有 256 个，因此 CHECKERR.BAT 总共用 512 行来检验 ERRORLEVEL 值。

批处理程序 CHECKERR.BAT

```
ECHO OFF
REM CHECKERR.BAT
IF ERRORLEVEL 255 ECHO 255
IF ERRORLEVEL 255 GOTO END
IF ERRORLEVEL 254 ECHO 254
IF ERRORLEVEL 254 GOTO END
      ... ...
      ... ...
IF ERRORLEVEL 1 ECHO 1
IF ERRORLEVEL 1 GOTO END
IF ERRORLEVEL 0 ECHO 0
IF ERRORLEVEL 0 GOTO END
:END
```

利用嵌套的 IF 语句，对 CHECKERR.BAT 进行了改写，改写后的程序为 CHECKER2.BAT。

批处理程序 CHECKER2.BAT

```
ECHO OFF
REM CHECKER2.BAT
IF ERRORLEVEL 255 ECHO 255
```

如果 ERRORLEVEL 值为 255，则显示 255。ERRORLEVEL 值最大为 255，故只需一个 IF 语句。

```
IF ERRORLEVEL 254 IF NOT ERRORLEVEL 255 ECHO 254
```

所有大于或等于 254 的值都能满足第一个 IF 语句，所

有小于或等于 255 的值都能满足第二个 IF 语句；两者结合起来，只有 254 满足这个 IF 嵌套语句。

```
IF ERRORLEVEL 253 IF NOT ERRORLEVEL 254 ECHO 253
        … …
        … …
IF ERRORLEVEL 1 IF NOT ERRORLEVEL 2 ECHO 1
IF ERRORLEVEL 0 IF NOT ERRORLEVEL 1 ECHO 0
```

CHECKER2. BAT 中，把两个 IF 语句嵌套在同一行上，和 CHECKERR. BAT 相比，IF 的个数没有减少，文件长度减小了将近一半。IF ERRORLEVEL 语句是大于或等于检验，在 CHECKERR. BAT 中，按从大到小的顺序逐个检验 ERRORLEVEL 值，条件一旦满足，应立即退出，否则后继的条件语句无法得到正确的结果。程序 CHECKER2. BAT 中，两个 IF 语句的嵌套使条件成为“等于”而不是“大于或等于”更为严谨，省去了 GOTO END 语句。

IF 语句的嵌套不能减少 IF 的个数，却能缩短文件的长度，使批处理程序的结构更加紧凑。批处理程序 NEST1. BAT 利用 4 个单独的 IF 语句来确定是否输入了正确的 Y 或 N，大写字母和小写字母都是可以接受的，因此用了 4 行 IF 语句。

批处理程序 NEST1. BAT

```
ECHO OFF
REM NEST1.BAT
IF (%1)==( ) GOTO NONE
IF %1==Y GOTO OK
IF %1==y GOTO OK
IF %1==N GOTO OK
IF %1==n GOTO OK
GOTO ERROR
:OK
ECHO CORRECTED VALUE ENTERED
GOTO END
:NONE
ECHO NO VALUE ENTERED
GOTO END
:ERROR
ECHO INVALID VALUE ENTERED
GOTO END
END
```

在 NEST2. BAT 中把 4 个 IF 语句嵌套在同一行上，缩短了程序的长度，加快了执行速度。

批处理程序 NEST2. BAT

```
ECHO OFF
```

```
REM NFST2.BAT
IF (%1)==( ) GOTO NONE
IF NOT %1==Y IF NOT %1==y IF NOT %1==N IF NOT %1==n GOTO ERROR
GOTO OK
:OK
ECHO CORRECT VALUE ENTERED
GOTO END
:NONE
ECHO NO VALUE ENTERED
GOTO END
:ERROR
ECHO INVALID VALUE ENTERED
GOTO END
```

NEST1.BAT 和 NEST2.BAT 这个两个例子说明了 IF 语句嵌套的用法，它们本身并无实用价值。

10.11 FOR 语句的特殊用法

在第 2 章中，曾说过 FOR 语句不能嵌套，例如批处理程序 NESTFOR1.BAT 中，由于存在嵌套的 FOR 语句，执行时将出现出错提示信息。

批处理程序 NESTFOR1.BAT

```
ECHO OFF
REM NESTFOR1.BAT
FOR %%a IN (NESTFOR?.BAT) DO FOR %%b (NEST?.BAT) DO ECHO %%a %%b
```

FOR 嵌套语句，将导致出错提示信息的出现，执行结果如下：

```
C>NESTFOR1
C>ECHO OFF

FOR Cannot be nested
```

在介绍批处理程序的子程序时讲过，由于低于 3.3 的 DOS 版本不支持 CALL 语句，可以使用 COMMAND/C 使子程序执行结束后返回主程序。这里使用 COMMAND/C，可以使 FOR 语句嵌套起来。

批处理程序 NESTFOR2.BAT

```
ECHO OFF
REM NESTFOR2.BAT
FOR %%a IN (NESTFOR?.BAT) DO COMMAND/C
FOR %%b IN (NEST?.BAT) DO ECHO %%a %%b
```

在 NESTFOR2.BAT 中，调用了辅助命令处理程序，使 ECHO 重新打开，运行结果如下：

```
C：\BAT\BAT>ECHO OFF


C：\BAT\BAT>ECHO NESTFOR1.BAT NEST1.BAT
NESTFOR1.BAT NEST1.BAT

C：\BAT\BAT>ECHO NESTFOR1.BAT NEST2.BAT
NESTFOR1.BAT NEST2.BAT

C：\BAT\BAT>ECHO NESTFOR1.BAT NEST3.BAT
NESTFOR1.BAT NEST3.BAT

C：\BAT\BAT>ECHO NESTFOR2.BAT NEST1.BAT
NESTFOR2.BAT NEST1.BAT

C：\BAT\BAT>ECHO NESTFOR2.BAT NEST2.BAT
NESTFOR2.BAT NEST2.BAT

C：\BAT\BAT>ECHO NESTFOR2.BAT NEST3.BAT
NESTFOR2.BAT NEST3.BAT

C：\BAT\BAT>
```

可以看出，DOS 能够准确执行这样的 FOR 嵌套语句。NESTFOR2.BAT 执行后，显示在屏幕上的 5 个批处理程序 NESTFOR1.BAT，NESFOR2.BAT，NEST1.BAT，NEST2.BAT和 NEST3.BAT 都在当前子目录里，根据 5 个文件在屏幕上出现的顺序，可看出嵌套 FOR 语句执行的先后次序。利用 FOR 语句，可以扩大文件通配符的使用范围。多数 DOS 命令，如 DEL，COPY 等支持文件通配符。下面的命令把当前目录中的所有文件复制到 A 盘：

```
C：\BAT\BAT>COPY *.* A：
```

而不必逐个文件进行复制。

少数 DOS 命令不支持文件通配符，比如 TYPE 命令，如果要逐屏显示三个文件，必须使用命令：

```
C>TYPE FILE1 | MORE
C>TYPE FILE2 | MORE
C>TYPE FILE3 | MORE
```

操作比较麻烦，利用 FOR 语句和通配符，可简化操作过程，使三个文件自动逐屏显示。

```
C>FOR %J IN (FILE?) DO TYPE %J | MORE
```

利用 FOR 语句可直接执行一系列 DOS 命令，即把一系列的 DOS 命令作为变量写在括号里，然后逐个执行，程序 3COMMAND. BAT 首先执行 DIR 命令，接着执行 CHKDSK 命令，最后执行 CD\命令。

批处理程序 3COMMAND. BAT

```
@ECHO OFF
REM 3COMMAND. BAT
FOR %%J IN (DIR CHKDSK CD\) DO %%J
```

由于 DOS 以行为单位执行批处理程序，把多个 DOS 命令写在同一行上，由 FOR 语句执行，比单个执行省时。但不能用这种方法执行由两个或两个以上单词构成的命令，因为 DOS 会把单词之间的空格作为分隔符，例如准备执行 CHKDSK 命令，然后执行 NORTON Utilities 用程序的 BE BEEP 命令使扬声器发声，由于 BE BEEP 是由两个单词构成的命令，写成 FOR 语句：

```
FOR %%J IN(CHKDSK BE BEEP) DO %%J
```

DOS 会把它们当作三个命令：

```
CHKDSK
BE
BEEP
```

必然导致出错提示信息的出现。用这种方法执行 DOS 命令的另外一个缺点是 DOS 不能正确执行带有参数的命令。如果要执行 CHKDSK/F，某些 DOS 版本会把“/”当作分隔符，把 CHKDSK/F 看作 CHKDSK 和 F 两个命令。一般情况下，应避免用 FOR 语句来执行一系列的 DOS 命令。

10.12 建立与 DOS 内部命令同名的批处理程序

在第 2 章中曾说，在建立批处理程序时不可用 DOS 内部命令作为文件名，这是一般情况都应遵循的原则。假如用 DOS 的删除文件命令 ERASE 作为批处理程序名，建立了文件 ERASE. BAT：

批处理程序 ERASE. BAT

```
@ECHO OFF
ECHO Worked!
```

在屏幕上键入：

```
C>ERASE

Required Parameter missing
```

这里 DOS 没有把 ERASE 当作批处理程序，仍然看作内部命令，由于没有输入执行 ERASE 命令必需的参数，所以出现了出错提示信息。

当用户在屏幕上输入不带路径的命令时，DOS 首先检查该命令是否为内部命令，如果是则立即执行。从 DOS3.0 开始，允许在屏幕上键入命令时在命令的前面加上路径名，DOS 遇到这种类型的命令时，则当作外部命令，到路径所规定的目录去寻找。如果 A 盘上有一名为 INFO.EXE 的文件，路径语句中没有包括 A 驱动器，要想执行该文件不必把当前驱动器“C>”改为“A>”，直接键入下列命令即可：

```
C>A：INFO
```

同样，执行批处理程序 ERASE.BAT，如果在前面加上它所在的路径，DOS 则认为它是外部命令

```
C>C：ERASE
Worked!

C>
```

Worked! 的出现，说明程序 ERASE.BAT 执行成功。用 DOS 内部命令作为批处理程序名的实用价值不大，这里只举一个实际应用的例子。

为了防止由于误操作把硬盘格式化，可以把 FORMAT.COM 重新命名为 RENAME.COM。如果 RENAME.COM 所在的子目录为 C：\BAT\BAT，执行磁盘格式化操作时，必须同时键入路径和 RENAME 命令，DOS 将把 RENAME 视作外部命令。如果在 DOS 提示符下直接键入不带路径的 RENAME，DOS 仍把该命令当作文件重新命名的内部命令，这样在某种程度上可防止意外格式化造成磁盘文件的丢失。

10.13 CTTY 的隐含功能

CTTY 是用来改变标准输入/输出设备的命令，多数用户很少使用。在批处理程序里进一步开发 CTTY 的功能，能够把批处理程序的执行过程隐含起来，建立计算机使用记录，也可以有选择地显示文件执行过程中出现的信息。如果一个批处理程序的起始行为 @ECHO OFF，结束行为 COPY *.* A：>NUL，可隐含批处理程序执行过程中产生的大部分信息，但是不能隐含出错提示信息。如果一批处理程序执行过程中产生了出错提示信息，尽管 ECHO 处于 OFF 状态，出错提示信息仍然会出现在屏幕上。程序 NOT-DEL.BAT 执行 DEL 命令时，由于所要删除的文件不存在，尽管使用了 @ECHO OFF 及 >NUL，出错提示信息仍然出现屏幕上。

批处理程序 NOT-DEL.BAT

```
@ECHO OFF
REM NOT-DEL.BAT
REM FILE C：\QQQ DOES NOT EXIST
DEL C：\QQQ > NUL
```

执行后屏幕显示为：

```
C：\BAT\BAT>Not－Del
File not found

C：\BAT\BAT>
```

类似于“File not found”的出错提示信息，不能用@ECHO OFF 或者把信息输送到 NUL 设备的方法，来禁止它们在屏幕上的显示。每当系统启动时，DOS 自动地把键盘当作标准输入设备，把屏幕作为标准输出设备，如果需要，可以用 DOS 内部命令 CTTY 来改变标准输入/输出设备，例如：

```
CTTY COM1
```

DOS 将把第一个异步通信端口作为标准输入输出设备。如果把 NUL(不存在的设备)作为输入输出设备，DOS 将不可能从键盘获得输入，也不能在屏幕上显示任何信息，键盘和屏幕都失去了以往的功能。

利用 CTTY 命令构成的批处理程序 NOT-DEL1. BAT 将不显示 CTTY NUL 与 CTTY CON 这两行之间应出现的任何信息。

批处理程序 NOT-DEL1. BAT

```
@ECHO OFF
REM NOT-DEL1. BAT
REM FILE C：\QQQ DOES NOT EXIST
CTTY NUL
DEL C：\QQQ
CTTY CON
```

在 NOT-DEL1. BAT 中，执行完 CTTY NUL 后，键盘和屏幕不再是标准输入输出设备，此时除了 Ctrl-Break 能中断批处理程序的运行以外，其余的键都失去了功能，企图从键盘键入 CTTY CON 命令来恢复键盘功能也是不可能的，唯一的办法是重新启动。

执行 CTTY NUL 下一行应出现的出错提示信息也不会显示在屏幕上，因屏幕已不是标准输出设备。最后一个语句 CTTY CON 的功能是把标准输入输出设备恢复为键盘和屏幕，这条语句是必需的，否则需重新启动机器。

利用 CTTY 命令可以隐含信息的特点，建立计算机使用记录，能够把计算机启动的时间、启动次数、使用时间长短等资料记录在一个文件里，实际上是建立了使用档案。由于信息是隐含的，因此有助于计算机的管理。LOGBOOT. BAT 是记录计算机启动次数与时间的批处理程序。

批处理程序 LOGBOOT. BAT

```
@ECHO OFF
REM Normally，this would be Part of AUTOEXEC. BAT file
CTTY NUL
ECHO ¦ MORE ¦ TIME >> BOOTLOG. TXT
ECHO ¦ MORE ¦ DATE >> BOOTLOG. TXT
```

CTTY CON

关于 LOGBOOT. BAT，有几点说明如下：

1. 应把这个程序的文件名放在 AUTOEXEC. BAT 里。

2. ECHO ¦ MORE 的作用是为 DATE 和 TIME 命令提供 Enter(回车)键。

3. 利用转向符“>>”，将新的内容添加在原文件之后，而不是重写原文件。

LOGBOOT. BAT 记录计算机两天内的启动次数和启动时间如下：

```
Current time is 8：26：38.12
Enter new time：
Current date is Tue 1-12-1993
Enter new date(mm-dd-yy)：
Current time is 10：31：43.27
Enter new time：
Ccrrent date is Tue 1-12-1993
Enter new date(mm-dd-yy)：
Carrent time i s 9：17：51.49
Enter new time：
Current date is Wed 1-13-1993
Enter new date(mm-dd-yy)：
????????????????
```

可以看出两天内启动了三次，并记录了每次启动的日期和时间。

利用 CTTY 隐含信息的功能，可以建立记录用户使用某应用程序的时间，由于信息是隐含的，启动与结束某程序应用软件的时间记录在一个文件里，计算机管理者可根据文件记录，对用户计时收费。

批处理程序 1-LOG. BAT

```
@ECHO OFF
REM 1-LOG.BAT
CTTY NUL
ECHO Starting Lotus >> C：\LOG\LOTUSLOG. TXT
ECHO ¦ MORE ¦ TIME >> C：\LOG\LOTUSLOG. TXT
ECHO ¦ MORE ¦ DATE >> C：\LOG\LOTUSLOG. TXT
CTTY CON
C：
CD\123
123
CTTY NUL
ECHO Finishing Lotus >> C：\LOG\LOTUSLOG. TXT
ECHO ¦ MORE ¦ TIME >> C：\LOG\LOTUSLOG. TXT
ECHO ¦ MORE ¦ DATE >> C：\LOG\LOTUSLOG. TXT
CTTY CON
```

```
MENU
```

执行程序 1-LOG. BAT 后，LOTUSLOG. TXT 记录了用户使用 Lotus 1-2-3 的起始和结束时间：

```
????????????????
Starting Lotus
Current time is 8 : 06 : 20. 33
Enter new time :
Current date is Tue 1-12-1993
Enter new date(mm-dd-yy) :
Finishing Lotus
Current time is 10 : 16 : 32. 16
Enter new time :
Current date is Tue 1-12-1993
Enter new date(mm-dd-yy) :
```

类似于 1-LOG. BAT 这样的文件，最好和批处理程序菜单结合起来使用。无论 ECHO 处于 OFF 或 ON 状态，CTTY NUL 命令都会把批处理程序执行过程中产生的所有信息隐含起来。利用 ECHO Méssage＞CON，可以在 CTTY NUL 与 CTTY CON 这两条命令行之间有选择地显示信息。

10. 14　实际参数字母的大小写

在某些情况下，实际参数字母的大小写会成为批处理程序能否正确执行的关键。批处理程序 CAPITAL1. BAT，根据输入的实际参数，决定执行那一个程序（DAILY. EXE，MONTHLY. EXE 或 ANNUAL. EXE）。

IF 语句中进行的字符串比较，等号两侧的字符串必须相等，字符的个数、顺序、大小写完全相同，DOS 才能认为相等，否则认为是不同的字符串。在 CAPITAL1. BAT 中，尽管用了 9 行来处理字母的大小写问题，也没有把所有可能的情况全部包括，比如 DAIly。如果不存在大小写问题，则三行就足够了。

批处理程序 CAPITAL1. BAT

```
@ECHO OFF
REM CAPITAL1. BAT
IF (%1)==( )          GOTO NOTHING
IF (%1)==(DAILY)      GOTO ONE
IF (%1)==(daily)      GOTO ONE
IF (%1)==(Daily)      GOTO ONE
IF (%1)==(MONTHLY     GOTO TWO
IF (%1)==(monthly)    GOTO TWO
IF (%1)==(Monthly)    GOTO TWO
```

```
IF (%1)==(ANNUAL)GOTO THREE
IF (%1)==(annual)    GOTO THREE
IF (%1)==(Annual)    GOTO THREE
: NOTHING
ECHO This Batch File Requires
ECHO A Parameter. The Valid
ECHO Ways To Start It Are:
ECHO CAPITAL1 daily
ECHO CAPITAL1 monthly
ECHO CAPITAL1 annual
GOTO END
: ONE
DAILY
GOTO END
: TWO
MONTHLY
GOTO END
: THREE
ANNUAL
GOTO END
: END
```

回避字母的大小写问题有多种方法，每种方法都有本身的优缺点。在 CAPITAL2.BAT，利用输入的实际参数作为转向语句 GOTO 的标号，GOTO 语句与 IF 语句不同，GOTO Label 中的 label(标号)要求字母相同，顺序相同，而不考虑大小写。

批处理程序 CAPITAL2. BAT

```
@ECHO OFF
REM CAPITAL2. BAT
IF (%1)==() GOTO NOTHING
GOTO %I
: NOTHING
ECHO This Batch File Requires
ECHO A Parameter. The Valid
ECHO Ways To Start It Are:
ECHO CAPITAL2 daily
ECHO CAPITAL2 monthly
ECHO CAPITAL2 annual
GOTO END
: DAILY
DAILY
GOTO END
: MONTHLY
```

```
MONTHLY
GOTO END
:ANNUAL
ANNUAL
GOTO END
:END
```

在键幕上键入下列命令：

```
C>CAPITAL2 daily
```

上述命令行中的实际参数 daily 将替换文件中的形式参数%1，文件里的标号是：DAILY，GOTO 语句不考虑大小写，因此 GOTO daily 将转到标号 DAILY。

解决实际参数字母大小写问题的另一种方法是利用环境变量。设置环境字符串命令 SET 的格式为：

```
SET[name=[parameter]]
```

添加到环境中后，变量名(name)中的小写字母都被转换成大写字母，例如：

```
SET adc=xyz
```

变量名 abc 将转换成大写字母 ABC。利用 SET 命令的这一特点，把实际参数作为环境变量名，输入实际参数时，不必考虑字母的大小写问题。

批处理程序 CAPITAL3. BAT

```
@ECHO OFF
REM CAPITAL3.BAT
SET DAILY=
SET MONTHLY=
SET ANNUAL=
SET %1=YES
IF %DAILY%==YES GOTO DAILY
IF %MONTHLY%==YES GOTO MONTHLY
IF %ANNUAL%==YES GOTO ANNUAL
GOTO ERROR
:ERROR
ECHO This Batch File Requires
ECHO A Parameter. The Valid
ECHO Ways To Start It Are:
ECHO CAPITAL3 daily
ECHO CAPITAL3 monthly
ECHO CAPITAL3 annual
GOTO END
:DAILY
DAILY
```

```
GOTO END
:MONTHLY
MONTHLY
GOTO END
:ANNUAL
ANNUAL
GOTO END
:END
```

在 CAPITAL3.BAT 中，首先从环境中清除了 DAILY，MONTHLY 和 ANNUAL 三个变量，再把 YES 作为值赋给它们：

```
SET %1=YES
```

上述命令行把输入实际参数中的小写字母都转换成大写字母，确保了后面 IF 语句的成功执行。用这种方法构成的批处理程序，实际参数必须是数字时，会出现意想不到的问题。例如准备执行 Lotus 1-2-3，必须把 123 作为实际参数，语句：

```
SET %1=YES
```

将正常工作，语句：

```
IF %123%==YES
```

DOS 认为%1 是一个形式参数，而不是把 123 作为一个整体当作变量名。在所有的环境变量中，只有 PATH 的值(参数部分)中的小写字母被 DOS 转换成大写字母，例如：

```
PATH=C:\dos;C:\lotus
```

进入环境后则成为：

```
PATH=C:\DOS;C:\LOTUS
```

这为解决实际参数字母的大小写问题提供了又一种方法，编写这类批处理程序，应按下列步骤进行：

(1) 把当前路径暂时存入一个环境变量中。

(2) 利用设置 PATH 命令，把实际参数赋给变量 PATH，PATH 命令将把实际参数中的小写字母转换成大写字母，某些 DOS 版本的 SET PATH=Value 不能把路径中的小写字母转换成大写字母，因此不要使用 SET PATH=Value，使用 PATH=Value 在任何 DOS 版本下都能把 Value 中的小写字母转换成大写字母。

(3) 把存于 PATH 中已转换成大写的实际参数赋给另一个环境变量。

(4) 把 PATH 的值恢复为当前路经。

(5) 从环境中清除步骤 1 中保持当前路径的环境变量。

CAPITAL4.BAT 是用这种方法的，它可以接受由字母或数字组成的实际参数。

批处理程序 CAPITAL4.BAT

```
@ECHO OFF
```

```
SET OLDPATH=%PATH%
PATH=%1
SET VARIABLE=%PATH%
PATH=%OLDPATH%
SET OLDPATH=
IF (%VARIABLE%)==(DAILY)      GOTO DAILY
IF (%VARIABLE%)==(MONTHLY) GOTO MONTHLY
IF (%VARIABLE%)==(ANNUAL)   GOTO ANNUAL
GOTO ERROR
:ERROR
ECHO This Batch File Requires
ECHO A Parameter. The Valid
ECHO Ways To Start It Are:
ECHO CAPITAL4 daily
ECHO CAPITAL4 monthly
ECHO CAPITAL4 annual
GOTO END
:DAILY
DAILY
GOTO END
:MONTHLY
MONTHLY
GOTO END
:ANNUAL
ANNUAL
GOTO END
:END
```

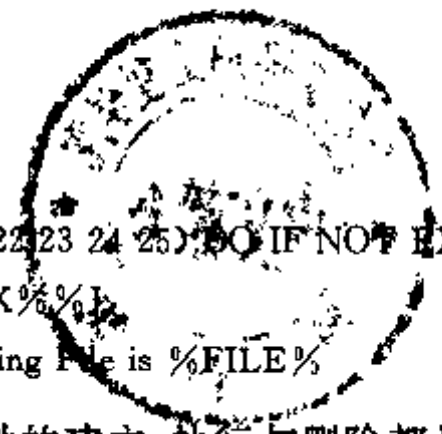
这种方法必须同时在 DOS 环境里保存三个环境变量，需要较大的环境空间。

10.15 删除临时建立的批处理程序

批处理程序执行过程中，有时需要建立临时性工作文件，这些临时性工作文件在建立它们的批处理程序执行过程中，完成使命后即被删除。临时性工作文件不得与已有的批处理程序重名。本书在不同的批处理程序里使用的临时性工作文件名都为 JUNK.BAT。为了防止临时性工作文件与已有的批处理程序重名，可用批处理程序 FINDFREE.BAT 检查当前目录里是否存在与准备使用的临时性工作文件同名的文件。

批处理程序 FINDFREE.BAT

```
@ECHO OFF
REM FINDFREE.BAT
FOR %%J IN (1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
```



```
18 19 20 21 22 23 24 25) DO IF NOT EXIST JUNK%%J SET
FILE=JUNK%%J
ECHO Working File is %FILE%
```

临时性工作文件的建立、执行与删除都是在一个批处理程序执行过程中完成的，有时用户不易觉察到这个过程。在编写含有临时性工作文件的批处理程序时，由于存在删除文件语句，应采取谨慎态度，否则会得到“Batch file missing”的出错提示信息。删除语句 DEL 必须放在临时性工作文件执行结束之后，不能放在临时性工作文件内部，因为不允许企图删除一个正在运行的批处理程序。批处理程序 JUNK1.BAT 与 JUNK2.BAT 都是临时性工作文件，两个文件除了结束符 ^Z 的位置不同外，其余完全相同，JUNK1.BAT 中的最后一行由单独一个文件结束符组成。JUNK2.BAT 中的文件结束符与删除命令在同一行。

批处理程序 JUNK1.BAT

```
@ECHO OFF
REM JUNK1.BAT
DEL JUNK1.BAT
^Z
```

在 DOS 提示符下执行 JUNK1.BAT 的结果为：

```
C>JUNK1
Batch file missing

C>
```

批处理程序 JUNK2.BAT

```
@ECHO OFF
REM JUNK2.BAT
DEL JUNK2.BAT^Z
```

DOS 执行批处理程序时，如果在当前执行行里没有发现文件结束符，则认为文件没有结束，就要继续寻找下一行。JUNK1.BAT 在执行过程中，在没有结束的情况下把本身删除，产生“Batch file missing”的出错提示信息是必然的。JUNK2.BAT 中删除行的最后是文件结束符，DOS 认为是 JUNK2.BAT 执行结束后才删除的，故不会出现出错提示信息。

[General Information]
书名=MS--DOS批处理程序应用与技巧
作者=
页数=171
SS号=0
出版日期=