计算机实用程序与使用技巧

上册

茅 晋 李 湍
徐天祎 李健伟 编

清华大学出版社

计算机实用程序与使用技巧

上册

茅晋　李湍
徐天祎　李健伟　编

清华大学出版社

(京)新登字 158 号

内 容 简 介

《计算机实用程序与使用技巧》分上、下两册。主要内容包括操作系统、C 语言、数据库、工具软件使用技巧、病毒防治技术、加密与解密、汉字处理、图形与图象、显示与打印、实用程序、软件评测和网络等十二章，上册为前六章。书中所有程序是众多计算机编程人员和用户经验与智慧的结晶，具有很强的实用性。

本书可供广大计算机用户和程序员参阅。

图书在版编目(CIP)数据

计算机实用程序与使用技巧　上册/茅晋等编. —北京：清华大学出版社，1996

ISBN 7-302-01959-2

Ⅰ. 计…　Ⅱ. 茅…　Ⅲ. 程序系统-基本知识　Ⅳ. TP31

中国版本图书馆 CIP 数据核字(96)第 18734 号

出版者：清华大学出版社(北京清华大学校内，邮编 100084)

印刷者：北京市清华园胶印厂

发行者：新华书店总店北京科技发行所

开　本：787×1092 1/16　印张：31.75　字数：760 千字

版　次：1996 年 10 月第 1 版　1996 年 10 月第 1 次印刷

书　号：ISBN 7-302-01959-2/TP・903

印　数：0001—5000

定　价：36.00 元

目　录

第1章　操作系统

第2章 C 语 言

第3章 数 据 库

第 4 章 工具软件使用技巧

第 5 章 病毒防治技术

第 6 章 加密与解密

第1章 操作系统

1.1 MS DOS 重入分析

MS DOS 是一个单任务的磁盘操作系统，随着应用的不断深入，越来越多的要求需要该系统能够支持多任务。事实上，若干新软件如 Windows 已经支持多任务。尤其是越来越多的人掌握 TSR 程序的设计方法，更需要了解 DOS 的重入问题。在若干公开的文献中，大部分都指出 DOS 是不可重入的，但详细的原因都不甚清楚，本文试图通过对 INT 21H 的总控部分的分析，指出可重入的情况和不可重入的情况。

1.1.1 INT 21H 的总控程序结构

INT 21H 是 MS DOS 的核心部分，要想了解 DOS 的重入问题，只要熟悉 INT 21H 的总控程序结构即可。下面以 MS DOS 3.3 为基础来讨论 INT 21H 的总控程序结构。该程序的框图如图 1.1.1 所示。

根据上述框图分述如下：

(1)进入 INT 21H 的一开始，首先判断 AH 值是否是合法的值，若不合法，退出。

(2)判断 AH 中是否是重入码 51H，62H，50H，33H，64H，如果是则进入相应程序，并执行，然后直接返回。

(3)如果不是上述两种情况，进入真正的 INT 21H，首先是寄存器压栈保护，然后是初始参数设置(1)。它的任务是保存 DS 和 BX 的值到内存单元中，之后将 CS 赋给 DS。

(4)置 DOS 忙标志，DOS 忙标志的信息可由 INT 21H 的 34H 取到。

(5)保存原 PSP 地址(但不恢复)和原用户栈指针；保存当前进程的用户栈指针并设置到 PSP 的+2EH～+31H 单元中。

(6)首先将 SS：SP 指针设置到系统栈的辅助栈(STACK1)上。

(7)设置相应的初始化参数，然后，根据 AH 中的值，计算相应功能调用子程序的地址偏移量。

(8)根据 AH 中的功能号，设置相应的系统堆栈，详细情况见图 1.1.1 的右边部分。

(9)取相应子程序的地址并取执行(CALL)。

(10)DOS 忙标志被置为闲。

(11)恢复用户栈指针和原用户栈指针。

(12)恢复寄存器值。

(13)中断返回，退出 INT 21H 程序。

从以上分析可以看出 INT 21H 的总控程序功能包括四部分：第一是重入功能码的判断并执行；第二是初始化参数设置和恢复；第三是根据 AH 中的功能号，计算程序偏移量并调用相应子程序；第四是根据 AH 中的功能号，设置相应的系统堆栈。由此可见，该总控程序与一般应用系统的总控程序的不同是，应用系统的总控程序只有第二和第三部分。

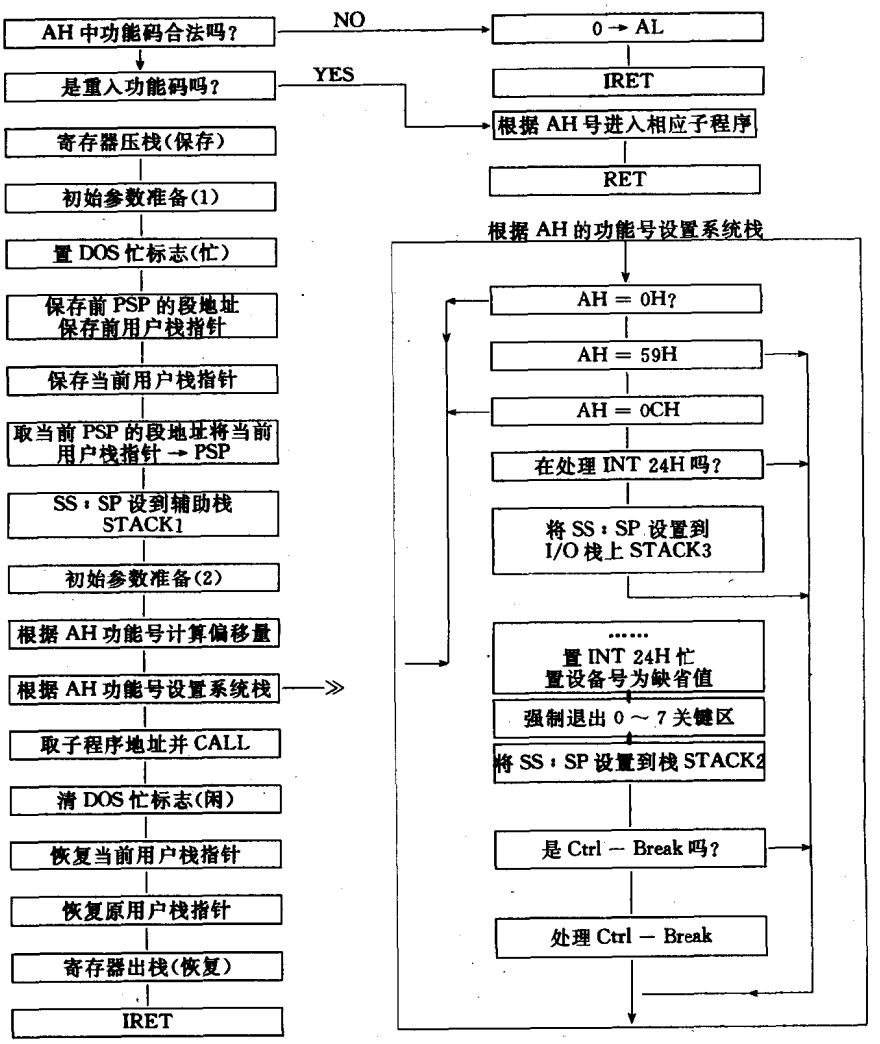

图 1.1.1　INT 21H 的总控程序结构

1.1.2　堆栈切换分析

MS DOS 重入问题的关键是分析 INT 21H 的总控程序，而分析总控程序的核心是了解总控程序对堆栈的使用方法问题。在 INT 21H 中碰到的堆栈有 5 个，图 1.1.2 是 INT 21H 的堆栈及切换状态情况图。

当进入 INT 21H 程序时，先将原用户栈的指针保存到内存单元中，然后，保存当前用户栈的指针再用系统内部的堆栈。系统堆栈有 3 个，分述如下：

(1)辅助栈(STACK1)，它是供 INT 24H 和 INT 21H 的 59H 功能调用使用的。系统初始将 SS : SP 指向该栈。

(2)I/O 栈(STACK3)，它是专门供 INT 21H 的输入/输出功能调用使用，这些功能调用包括 1H～0CH。

(3)磁盘栈(STACK2)，它是供磁盘操作使用的，这些功能调用包括 AH＝0，AH＞0CH，AH≠59H。

从图 1.1.2 中可以看出，系统在任何时刻 SS：SP 只指向一个系统栈。当系统在一个堆栈上工作时，如果被另一个进程中断，当前堆栈的数据得不到保护，这样系统就崩溃，这就是所谓 DOS 不可重入的原因。

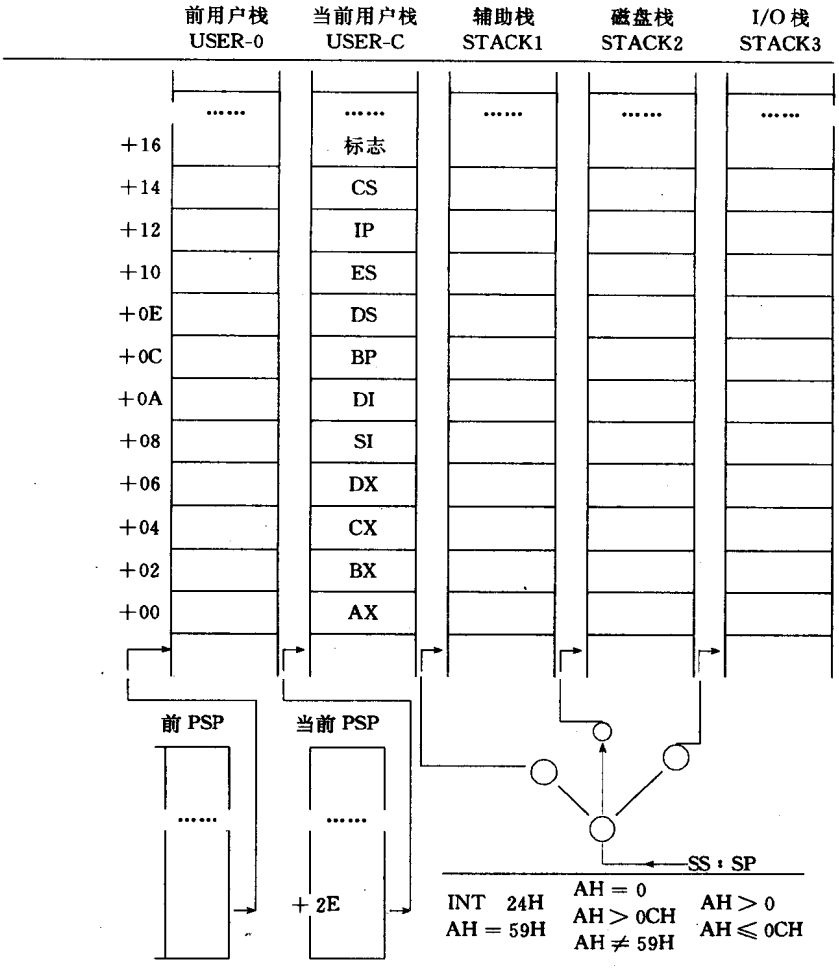

图 1.1.2 INT 21H 的堆栈及切换状态

1.1.3 重入与不可重入分析

重入和不可重入是本文讨论的重点，首先要明白什么叫重入，什么叫不可重入，为清楚起见，定义如下：

不可重入：在执行 INT 21H 的代码时不允许中断它的程序再次调用 INT 21H 的功能，换句话说，在执行 INT 21H 调用时，如又被定时中断 INT 8H 调用 INT 21H，再次进入 INT 21H 的代码(递归)，这种情况被称之为 MS DOS 的不可重入。

重入：不可重入的反定义即 INT 21H 的代码可在多个并行进程中，允许一个进程在任何一个时刻中断另一个进程，称之为重入。

通过对 INT 21H 的总控程序和系统堆栈切换分析，我们可以很清楚地了解到 MS DOS 的重入情况分为如下两种情况。

1. 完全可重入的功能调用

从上述分析中可以看到，有一些功能调用是完全可以重入的，它们是：

(1)50H，设置活动进程的 PSP 段的地址；

(2)51H，取当前活动进程的 PSP 段地址；

(3)33H，取/置 Ctrl-Break 状态；

(4)62H，取当前活动进程的 PSP 段地址；

(5)64H，设置未知标志(在 PRINT.COM 中使用过)。

这些功能能够重用的原因是它们的操作比较简单和单一，而且不用系统堆栈。

2. 不同堆栈切换的重入

从上述分析中可以很清楚地了解到，SS：SP 任一时刻指向一个系统栈，如果在磁盘栈上工作，被另一个进程中断到 I/O 栈上工作，这时系统是可以重入的。

不是上述两种情况，一般说来是不可重入的，这是若干公开文献中所指出的。但是，如果是这样，那将大大限制 TSR 的发展，同时也极大地限制了 DOS 的应用范围。DOS 不可重入的最重要的原因是当前进程的堆栈和状态数据得不到保护。事实上 DOS 的真正的重入方式，并不在于它的代码可重入性，而是因为它的数据的可保存性。换句话说，如果当一个进程被切换到另一个进程时，前一个进程的所有状态数据得以保存，当该进程被恢复时，这些数据也被恢复，这就是 DOS 未公开的秘密——DOS 重入方法：DOS 的上下文切换方法。这里简单介绍一下其基本思想。DOS 有一个数据交换区 SDA(Swap Data Area)，它包括系统的三个堆栈和其它若干状态数据。它分为两部分：第一部分叫一直交换区(Swap Always Area)，大约 18H 字节；第二部分叫交换区，大约 73CH 字节，不到 2KB。后者包括前者。该数据区位于临界区资源 PV 操作数据区的下面。取该区的首地址和大小可通过一个未公开的功能调用来实现，它是：

DOS 3.1～3.3　5　　D06H

入口：AX＝5D06H

出口：DS：SI 指向 SDA

　　DX：当 DOS 忙时，一直交换区的大小

　　CX：交换区的大小

DOS 4.X　　5D0BH

入口：AX＝5D0B

出口：DS：SI 中返回一个表，其内容如下：

表 1.1.1

偏　移　量	大　小	描　　述
00H	字	SDA 的数目
SDA 入口：		
02H	双字	SDA 地址
06H	字	数据区长度和类型
		第 15 位＝1　要交换“交换区”
		＝0　要交换“一直交换区”
		第 4～0 位　数据区的字节数
08H	双字	下一个 SDA 的入口
	……	

关于什么时候保存一直交换区，什么时候保存交换区，也可用未公开的 INT 2AH 中断来监视。

（龚正科）

1.2 巧用 DOS 中的 I/O 重定向命令

本文介绍 DOS 手册中一个大多数人未予重视的命令，即 I/O 重定向命令，结合其它 DOS 编程命令给出了 4 个应用实例，即使用 I/O 重定向命令修改程序，调试、修改汇编语言程序，显示内存容量，备份硬盘结构信息。

PC DOS 2.0 以上都支持一个类似于 Unix 系统的 I/O 重定向功能，如果用户能熟悉它并结合 DOS 的其它功能加以灵活运用，将会更方便有效地使用微机，提高应用水平。

DOS 的输入、输出通常是在标准设备（键盘和显示器）上进行的，但也可重新定义为另外的设备。此设备可以是磁盘文件或设备文件，即可以改变输入、输出的方向。DOS 承认的常见设备文件有：CON（控制台，即键盘/显示器）、AUX（异步通信口）、PRN（并行打印机）、NUL（虚拟设备）等等。

1. 输出改向

输出改向可以在命令中键入一个大于号“>”加上输出文件名或设备名。如果该输出文件已经存在，则清除该文件并进行复写。格式是：

>〔d:〕〔path〕filename

使用两个连续的大于号“>>”，可用来添加一个文件。如果该文件已存在，则输出将附加于原文件之后，不会破坏原文件。若文件不存在，则该命令类似于上述命令。

A>dir b:*.exe>b:dirs.txt

A>dir b:*.com>>b:dirs.txt

这两个重定向的列目录命令，可产生一个目录文件 dirs.txt，包括扩展名为.com 和 .exe 的文件目录。进而用命令：

A>type b:dirs.txt>prn

A>copy b:dirs.txt>lptl:

这两个命令达到的效果是一样的，即重定输出设备为并行打印机。

2. 输入改向

类似于输出改向，在命令行中使用小于号“<”及输入文件名，可进行输入改向。格式是：

<〔d:〕〔path〕filename

3. 使用实例

(1)利用重定向功能修改程序

在软件移植过程中经常要对原有软件进行修改，特别是软件的 I/O 接口部分。这类修改的特点是：难度不高，工作量大，需要非常细心，又很有规律。我单位有一个用 dBASE Ⅲ 编写的人事工资管理系统，原运行环境为 PC/XT 机，汉字 CC DOS2.1，现需改在 286 机的高分辨率环境下工作。由于制表符区位的差异，使 30 个报表程序中的制表符都要进行修改。如用 EDLIN 程序处理，虽然可以用字符串替换功能，但由于 EDLIN 只提供命令方式而不支

持程序方式，若每个程序有 20 种制表符需要修改，则共需 30×20=600 次的字符串替换命令。用重定位功能可以实现修改过程的自动化。方法如下：

①用 EDLIN 或 COPY CON:AA.BAT 建立一个名为 AA.BAT 的批命令文件，其中插入了以下形式的命令：

FOR%%FIN(*.PRG)DO EDLIN%%F<XG

其中:XG 是一个任取的文件名，其内容是 EDLIN 程序的命令序列。在本例中，可用 EDLIN 自身建立，如下：

1,R 〈原字串 1〉^Z〈替换字串 1〉

1,R 〈原字串 2〉^Z〈替换字串 2〉

…

…

1,R 〈原字串 N〉^Z〈替换字串 N〉

E

②现在程序修改工作只需在 DOS 提示符下运行 AA.BAT，即可一次修改完所有的程序。用类似的方法可修改各类程序，只要对 XG 文件进行细致的设计，就可获得满意的结果。

(2)用 DEBUG 的 U 命令可以很方便地进行反汇编

如果没有汇编语言源程序，又需要修改源程序，尤其是插入修改，一般是比较麻烦的。但实际上只要有扩展名为 .COM 和 .EXE 的可执行文件，再结合重定向命令就方便多了。如有一个 T.COM 文件，长度为 200H 字节。键入以下命令：

A>DEBUG T.COM>T.ASM

—u 100 3ff

—q

T.ASM 就是 T.COM 的机器码和反汇编结果。你可以用 EDLIN 等编辑软件调用 T.ASM，进行分析和打印，修改时将其中的机器码去掉，借助于编辑软件进行插入、替换等修改。工作完成后，将第一行的 U 命令改为：

f 100 ffff 0

a 100

存盘，然后键入命令：

A>debug<t.asm

在退出 DEBUG.COM 后，用命令方式进入 DEBUG 环境中，再用 D 命令查看文件长度(最后一个非零字节的地址减去 100H)，设置寄存器 BX,CX，再将汇编好的内容以文件名 T.COM 存盘即可。

(3)在 AUTOEXEC.BAT 中显示内存容量，加强对病毒的监视能力

许多病毒常修改内存总量(减少 1～2K)，使病毒程序常驻内存。PC 机的内存总量参数存放在 0:0413H 和 0:0414H 单元中，只要知道微机的实际容量，对比一下就可以判定内存中是否染上病毒。这种方法对于那些操作系统型病毒、感染 COMMAND.COM 的外壳型病毒以及利用修改内存总量隐藏自己的病毒都很有效。一旦发现参数不符可及时采取措施。由于 0:0413H 和 0:0414H 两单元存放的是基本内存参数，所以对一些扩展了内存的 AT，286,386 机同样适用。常见的内存配置和两单元的对应关系如下：

实际内存	0:0413H	0:0414H
640K	80H	02H
512K	00H	02H
384K	80H	01H
256K	00H	01H

在批文件中加入以下命令:DEBUG<DEBUG.DAT,其中DEBUG.DAT是DEBUG程序的命令序列,建立方法同(1),内容为:

```
D 0:413    L    2
Q
```

(4)备份硬盘结构信息

微机用户都知道像DOS系统盘是很重要的系统软件,需要做好备份。实际上对拥有硬盘的用户来说,硬盘的结构信息(包括硬盘引导区BOOT,DOS分区引导区DOSBOOT,文件分配表FAT,文件目录FDT)也是非常重要的系统软件。尤其是现在许多恶性病毒发作时都破坏硬盘的结构信息,从而达到破坏整个硬盘数据的目的。经常备份硬盘的结构信息可以最大限度地减少病毒造成的数据损失,对于引导型的病毒也可以起到消毒的作用。

以DOS 3.3管理的43M硬盘C:为例,假设硬盘结构信息的四部分分别保存到文件BOOT.DOS, DOSBOOT.DOS, FAT.DOS, FDT.DOS中。实现保存的批处理文件BACK.BAT内容如下:

```
@ECHO OFF
IF NOT EXIST *.DOS GOTO A
ATTRIB  -R *.DOS
IF NOT EXIST *.BAK GOTO B
ATTRIB  -R *.BAK
DEL *.BAK
:B
RENAME *.DOS *.BAK
:A
DEBUG<BACK.DAT>NUL
ATTRIB +R *.DOS
ATTRIB +R *.BAK
ECHO BACKUP SUCCESSFUL!
```

实现恢复的批文件REST.BAT内容如下:

```
@ECHO OFF
IF EXIST *.DOS GOTO B
ECHO DOS INFORMATION NOT FOUND!
GOTO END
:B
DEBUG<REST.DAT>NUL
ECHO RESTORE SUCCESSFUL!
:END
```

其中BACK.DAT和REST.DAT为DEBUG.COM的命令序列。它们的内容为:

BACK.DAT

```
A 100
MOV AX,0201
MOV BX,1000
```

REST.DAT

```
A 100
MOV AX,0310
MOV BX,1000
```

```
MOV CX,0001
MOV DX,0080
INT 13
INT 3
G=100
N BOOT.DOS
R CX
200
R BX
0
W 1000
L 100 2 0 1
W 100
N DOSBOOT.DOS
L 100 2 1 55
R CX
AA00
N FAT.DOS
W 100
L 100 2 AB 20
N FDT.DOS
R CX
```

```
MOV CX,0001
4000
W 100
Q
MOV DX,0080
INT 13
INT 3
N BOOT.DOS
L 1000
G=100
N DOSBOOT.DOS
L 100
W 100 2 0 1
N FDT.DOS
L 100
W 100 2 1 55
W 100 2 65 55
L 100
L 100
W 100 2 AB 20
Q
```

注意：使用时需将文件 DEBUG.COM，ATTRIB.COM，BACK.BAT，REST.BAT，BACK.DAT，REST.DAT 都放在同一软盘上。

用户在做好备份后，可用 COMP 命令比较 BOOT.DOS 和 BOOT.BAK，以及 DOSBOOT.DOS 和 DOSBOOT.BAK 是否一致。若原机器无毒，比较结果两文件不一样，则说明有病毒侵入。同时也可结合(3)来判断。发现有毒后，解毒方法很简单：在无毒环境下，将备份好的硬盘结构信息用 REST.BAT 恢复即可。注意：这时恢复要用 *.BAK，因为 BOOT.DOS 和 DOSBOOT.DOS 可能已经包含着病毒的程序。

（缪晓明）

1.3 DOS 内部命令

1.3.1 DOS 内部命令的修改和扩充

DOS 由以下 4 个主要部分组成：引导块，IBMBIO.COM，IBMDOS.COM，COMMAND.COM，其中 COMMAND.COM 负责接受键盘命令，并执行相应的命令子程序；若是像 DIR，COPY，RENAME 等内部命令，就直接由驻留在内存的代码执行；若是外部命令，则调用相应的可执行程序执行。DOS 内部命令是由 COMMAND.COM 初始完成并驻留内存的，其优先级最高，利用 DEBUG.COM 将 COMMAND.COM 调入内存，可找到一个 DOS