算机实用技术系列丛书(三)

最新版

电脑病毒彻底研究

Stone

大榔頭

Disk Killer

快樂星期天

13號星期五

VAGTH: Virus All GOTO Hell

施威銘研究室 著

希望

学苑出版社

计算机实用技术系列丛书(三)

电脑病毒彻底研究

施威铭研究室　著
卢　俊　王小明　改编
李　兵　燕卫华　审校

学苑出版社

(京)新登字 151 号

内 容 提 要

放眼目前充斥市面的解毒程序，从最初的(C) Brian 到现在的“解五毒”、“解十毒”的程序，真是五花八门，而所能找到的电脑病毒书籍也像是病毒的科幻小说，要不然就是解毒软件的采购指南。结果，到头来还是不明白电脑病毒究竟是什么。我们认为，与其不断购买杀毒软件，不如彻底了解病毒，再设法根除病毒。本书详细剖析了三种病毒的源代码，讨论了病毒的寄生、繁殖和传染的原理，以及病毒的其它欺骗伎俩，并列举了与病毒有关的各种系统数据。

需要本书者，请与北京海淀 8721 信箱书刊部联系，邮政编码 100080，电话 2562329。

版 权 声 明

计算机实用技术系列丛书(三)

电脑病毒彻底研究

编　　著：[illegible]研究室
改　　编：卢　俊　王小明
审　　校：[illegible]　燕卫华
责任编辑：陆卫民
出版发行：学苑出版社　　邮政编码：100036
社　　址：北京市海淀区万寿路西街 11 号
印　　刷：[illegible]和印刷厂
开　　本：[illegible]092　1/16
印　　张：13.875　　字　　数：318 千字
定　　数：1 ～ 5000 册
版　　次：1994 年 10 月北京第 1 版第 1 次
ISBN 7－5077－0777－6/TP・9
本册定价：19.00 元

序

不知曾几何时，商人不谈股票，玩电脑的人不谈病毒，就像已经落伍了。传播媒介当然也不会自甘寂寞，俨然一夜之间各种各样的电脑病毒都会从我们身边冒出来一样。然而，不知您是否坐下来冷静地想过没有："电脑病毒"到底是个什么东西呢？

在这种"电脑病毒热"中，除了一群不停地制造病毒、修改病毒的狂热份子以及一些专门调制仙丹"解毒济世"的专家之外，似乎还有更多热衷于花钱买解药的 PC 用户。

不过，这种现象对我们来说并不难理解。事实上，美国电脑病毒工业协会主席就曾受到公开批评：只是"试"着借"病毒恐惧"捞一票罢了。

然而我们认为，与其生活在病毒恐惧中，不停地掏腰包购买最新的解毒程序，不如彻底地了解病毒，培养自我诊断、治疗和预防病毒的能力，而不再依赖于解毒软件，这才是上上策。

最后，我们真诚地希望读者都能够通过阅读与实践来学习我们的经验，提高　　电脑用户的知识水平，而不要永远停留在头痛医头、脚痛医脚的江湖郎中时代。

施威铭研究室

1990. 04. 10

序(增订版)

本书自初版到增订,其间有一年多的时间。事实上,在这一年多的时间里,这本书的增订工作一直在持续进行。我们不仅继续收集、解剖、研究和分析了许多新的电脑病毒,同时也不断地从读者的来信与电话中吸收了许多宝贵的经验与意见,因此在增订版中我们对本书的内容作了以下几项重要更改:

(1) 加入了一年来电脑病毒的演变史。我们详细地剖析了较具代表性,同时也是近年来被认为最有害的几种病毒:"Stone"、"Stone Ⅱ (3月6日)"和"Invader(大椰头)",并分析了其它一些新病毒的技术原理。

(2) 除了已公布的"十三号星期五"与"Disk Killer"之外,对其它病毒都未列出完整的源程序代码,而仅列出原理的重要部分,以便解说。

在此增订版付印之前,回顾这一年来国内发生的病毒事件,不由得令我们感触甚多。自本书初版起,我们就提出了"彻底了解病毒才是对付病毒的上策"的主张。我们认为,那些专门用病毒害人的家伙,就是利用了一般人对病毒的"无知",换句话说,病毒专门"残害"对它认识不清、了解不够的人。即使是专业电脑软件技术人员,如果不下一点功夫,也很可能成为病毒肆虐下的受害者。

"Stone Ⅱ"病毒事件就是一个很明显的例证。据我们了解,被这个病毒破坏硬盘的,除了一般的电脑用户之外,还不乏一些使用低级或高级语言的软件公司,而这些人原来所持的态度是"只要用几个扫毒软件扫一扫,若有病毒就找解药来解一解就行了"。他们宁愿花钱购买解毒软件,也不愿意多费点精力去了解电脑病毒究竟是什么。所以,这种要命的观点最终使他们沦为病毒肆虐下的"牺牲者"。其实,我们认为,只要用心研读一下《电脑病毒彻底研究》(本书),Stone Ⅱ就算不上什么难对付的病毒。不幸的是,有些受害者购买了本书却没有阅读,而有些则在病毒发作之后才购买本书,实在让人替他们感到惋惜。

一位友人曾提醒笔者,编写这类书的作者会与病毒制造者展开"马拉松"之战。笔者个人认为,正是由于一般人不了解病毒,那些制造病毒的人才会有优越感。当众人都被病毒搞得晕头转向、束手无策的时候,他们才会获得恶作剧的快感。因此,我们的信念是,必须有人将电脑病毒的奥秘彻底地揭露出来,当人人都对电脑病毒不再感到神秘的时候,病毒这种东西的末日也就为期不远了。值得欣慰的是,"十三号星期五"、"Disk Killer"及其变种确实像我们以前所预测的一样逐渐销声匿迹了。这当然不全是《电脑病毒彻底研究》这本书的功劳,但它终究会形成一股与病毒抗衡的力量,而且这种力量会愈来愈强,于是,病毒终有黔驴技穷的一天,"电脑病毒"终究将只是历史上的一个名词罢了。

施威铭研究室
郑慧彦
1991.04.11

目　　录

第一篇　基　础　篇

第二篇　剖　析　篇

第三篇 参 考 篇

第一篇　基　础　篇

第0章　技术要领

0.1　识别计算机病毒

计算机病毒其实是一些隐藏在计算机存储介质(如软盘和硬盘等)中不断自我复制的程序。由于这些程序具有繁殖、传染、寄居等特性,与"病毒(virus)"这类微生物有近似之处,所以被称为"计算机病毒"。概括地说,计算机病毒大多具有以下几种特性:

(1) 繁殖性:会复制其自身,否则便会绝种。
(2) 传染性:会通过软盘、磁带的携带或者网络及通信连线而传播。
(3) 寄居性:寄居在其它程序文件软盘里,并在启动时运行。

不要小看这三种特性,因为许多病毒设计者就是因此而精巧地设计了病毒的算法。试想,如果一个病毒在繁殖前就发作,那么根本就传播不开了。如果寄居性太差,就会被马上发现并清除掉。如果传染性不强,则很可能会像恐龙一样由于大环境的变迁而绝种,C Brain 的灭绝就是一个例子。

0.1.1　病毒的两种基本类型

PC 机上的病毒具有下列两种基本类型:

(1) 启动型:也可称为"开机型",就是冒充计算机的开机程序,并从开机磁盘里启动的病毒。也有人称这种病毒类型为"系统型"。
(2) 程序型:隐藏在程序文件里,并利用程序文件的运行而启动的病毒。

同样,病毒的入侵,也可依其性质而分成入侵到 BOOT 扇区和入侵到程序文件两种。

当然,也有人提出不同的分类方法,但是病毒程序一定要运行才能进行"繁殖",所以,所有计算机病毒都可依其启动方式而归属于上述两种类型。其实,深入研究计算机病毒后便会了解,病毒的启动方式对其寄居、繁殖等特性都有很大的影响。

0.1.2　如何对抗病毒

从初次出现到现在,绝大多数计算机病毒都具有破坏性,而且好事之徒仍在不断地修改病毒和创造新病毒,所以,对抗计算机病毒的根本方法在于 PC 机用户必须彻底了解病毒的原理和编制技巧,这样才能对症下药,不致于永远处于被动挨打的地位。

要了解任何一种计算机病毒,最好的方法就是仔细研究病毒程序的内容,但是,若从程序的机器码着手,不但极为困难,而且事倍功半。如果我们将病毒的机器码翻译成汇编语言程序代码,则分析起来就容易多了。

将机器码翻译成汇编语言代码也就是所谓的反汇编(unassemble),这项工作除了需要适当的工具外,还要求使用恰当的方法。本章最后一节会提出我们在这方面的经验,以供读者参考。

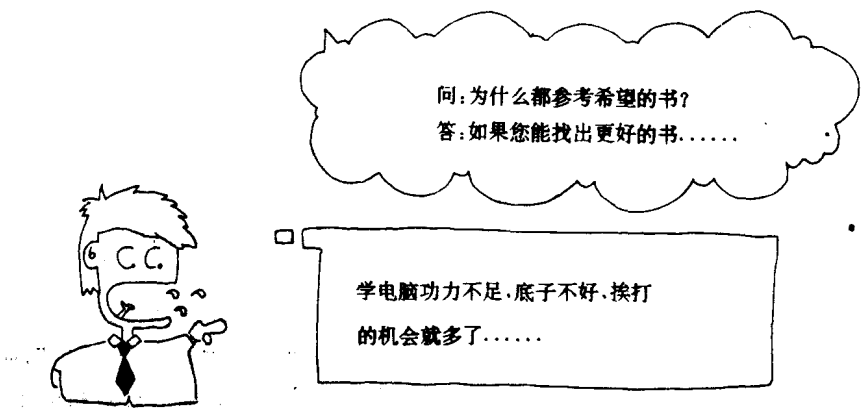
0.2 读者必备的基础知识

本书以指导读者研究计算机病毒、剖析计算机病毒的内部结构、启发读者发掘对抗病毒的方法为目的,并且将重点放在计算机病毒的运作原理及其所运用的特殊技巧上,因而对所涉及的基本知识无法详细地说明,读者必须自己从其它书籍中学习一些基础知识。

以下是我们认为读者所需具备的参考书籍:

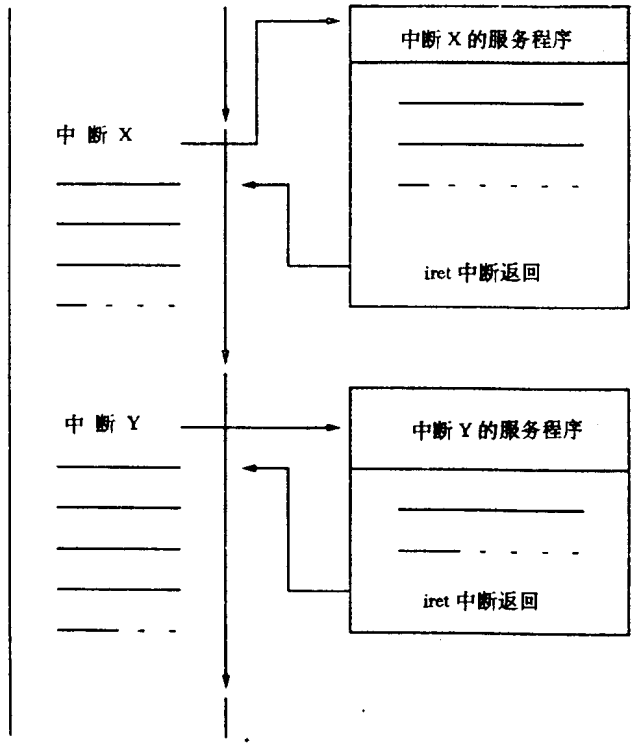
(1) 80x86 汇编语言实务:汇编语言基础书籍;
(2) DOS 技术手册(一):MASM 及反汇编的最佳参考书;
(3) DOS 技术手册(二):本书最重要的数据来源;
(4) DOS 技术手册(三):DOS 功能调用。

注意:本书假定读者具有基本的 DOS 及微机结构的知识,并对汇编语言有一定的了解。本书对十六进制地址中的大小写字母等同对待(A=a,B=b,C=c,D=d,E=e,F=f),并为了简洁起见而在正文中直接使用了一些常用表示法及指令助记符(未作翻译)。



0.2.1 IBM PC 的结构

这方面的知识关系到对自己所使用的机器的了解程度,如 RAM 与 ROM 的特性、BIOS 与 DOS 的关系等,其中与计算机病毒特别相关的是"中断"原理,我们将在下一节特别提出来加以说明。



0.2.2 对操作系统的深入了解

除了 DOS 的基本操作方法之外,“MS－DOS 的系统空间”、“MS－DOS 的用户空间”、“内存管理”、“进程管理”等方面的知识以及对 MS－DOS 磁盘结构的了解,都是读者必备的基础。读者可由“DOS 技术手册(二)”中获得这方面的详细剖析数据,本书在讨论到这方面的相关问题时也会予以概要提示或者补充说明。

0.2.3 阅读汇编语言的能力

在本书中,我们将计算机病毒程序逆向翻译成汇编语言程序来研究,所以读者必须对 8088/80x86 汇编语言相当熟悉(短小精悍的病毒程序大多是以汇编语言写成的)。“汇编语言实务”这本书可以提供这方面的知识。

本书中所有的汇编语言程序都采用 Microsoft Macro Assembler(简称 MASM)格式编写,并按约定将汇编程序的伪指令用大写字母、语言指令用小写字母列出。关于 MASM 方面的知识,读者可参阅“DOS 技术手册(一)”。

0.3 IBM PC 中断

80x86 具有中断(Interrupt,或称 Trap)功能。当中断信号出现时,CPU 会“中断(暂停)”当前的工作而去运行中断所指定的程序。由中断指定运行的程序称为中断服务程序(Interrupt

Service Routine，简称 ISR)。当 ISR 运行完毕后，CPU 会继续运行刚才被中断的程序，所以 ISR 与一般的子程序很类似，然而由于中断具有其它特殊性质，ISR 又与一般的子程序有一些不同。请参阅上页的图。

0.3.1 中断的来源

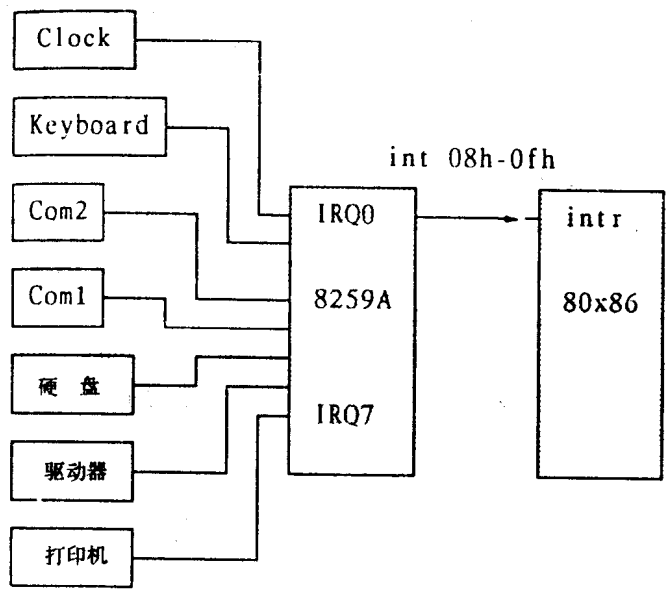
中断的来源有两种：一种来自硬件，一种来自软件。硬件中断可能来自 CPU 内部或外部设备，而软件中断则像一般指令一样以指令的形式存在于程序中，我们可以在 BIOS，BASIC，DOS 系统和 USER 程序以及病毒程序中发现它的踪影。

1. CPU 自发的中断

当 CPU 运行到某些特定点时，会自动执行中断操作，这种中断叫作 CPU 内部中断。例如，当 CPU 执行“除 0”运算时，便会执行第 0 号中断。PC BIOS 设计了一个“除 0”警告服务程序，该服务程序的地址保存在第 0 号中断向量表上。当 CPU 发生“除 0”现象时，该警告程序就会在屏幕上显示“Devide overflow”信息，然后把控制权交给 DOS。

2. 外部硬件中断

外部硬件中断是指 CPU 外部设备(如驱动器、打印机及键盘等)对 CPU 所要求的中断。当外部硬件发出中断请求时，CPU 会由当时 I(Interrupt)标志的值来决定是否执行中断服务。只有当 I 为 1 时 CPU 才会执行中断程序。IBM PC 以一个 8259 中断控制器(IC)来处理同时发生多个硬件中断的问题。8529 会根据各硬件的优先级(Priority)来处理同时到达的中断请求，并选择最优先者向 CPU 发出中断请求。8259 共有 8 条中断输入线(IRQ n)可供外部硬件使用，PC 使用了其中的 7 条，只有第 2 条未使用。



IRQ 编号	int 编号	外部设备
IRQ0	int 08h	CLOCK
IRQ1	int 09h	键盘
IRQ2	int 0ah	无
IRQ3	int 0bh	COM1
IRQ4	int 0ch	COM2
IRQ5	int 0dh	硬盘
IRQ6	int 0eh	软盘
IRQ7	int 0fh	打印机

当 8259 对 80x86 发出 int 信号(由 intr 接收)时,如果此时 I 标志为 1,则 80x86 便接受 8259 的中断请求,此时 80x86 会对 8259 发出一个信号,要求 8259 把中断的编号传送过来。所以,当程序不希望被中断时,便可将 I 标志设置为 0(利用 cli 指令),以便将 CPU 以外的中断都"抑制"住,直到可以接收中断时再用 sti 指令使 CPU 进入可接收中断的状态。PC 把 8259 的 IRQ0～IRQ7 分别对应到 int 08h～int 0fh,所以由上图可以看出,clock 中断为 int 08h,键盘中断为 int 09h,其它依此类推。

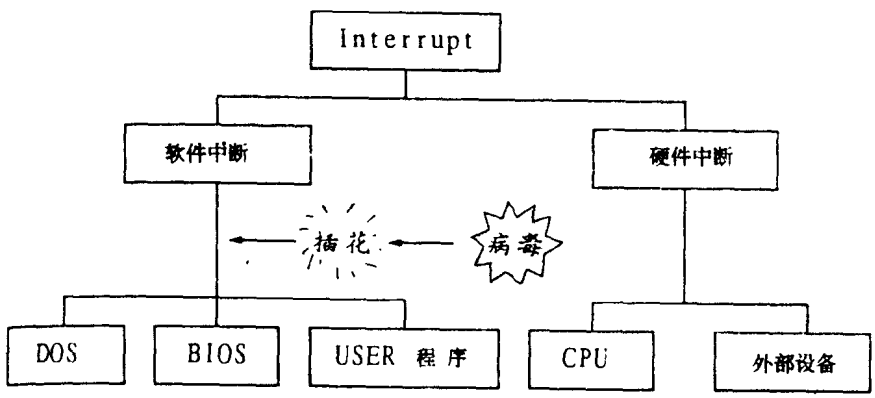
下一章将要论及的 int 13h 就是驱动器 I/O 中断,而 int 13h 是 BIOS 服务中断,它基本上与子程序没有区别,即根本不会通过 8259 请求 CPU 产生中断,而是属于下面要讨论的软件中断之一。

3. 软件中断

至于软件中断,则由 BIOS,DOS,BASIC 系统以及用户程序以下列方式向 CPU 发出请求:

int 中断号码

这是我们最熟悉的方式。

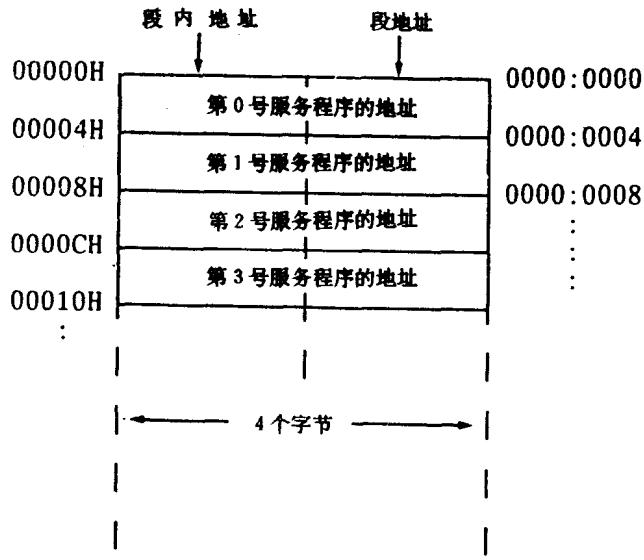


0.3.2 中断的优先权(priority)

所谓优先权,也就是优先执行的权力。权力较高的中断可以比中断权力较低的中断优先执行。简言之,硬件中断的优先权都比软件中断高,所以当 BIOS,DOS 或者用户运行程序时,CPU 与外部设备所发出的中断请求都会被 CPU 优先接受。但是,外部设备的中断也受到 I 标志的限制,这是非常重要的概念,请读者务必牢记。

0.3.3 中断的执行与中断向量表

仅仅指定编号,CPU 是无法运行中断服务程序的。必须给定地址,CPU 才能运行该地址处的服务程序。因此,80x86 的设计者将 00000～003FF 内的区域设置为中断向量(地址)表的存放区,这个区域内保存着中断服务程序的入口地址,每个地址占 4 个字节。所以,第 0 号服务程序的地址在 00000～00003 处;第 1 号服务程序的地址在 00004～00007 处;第 FF 号服务程序的地址在 030FC～003FF 处。



第 n 号服务程序的地址 = 4n

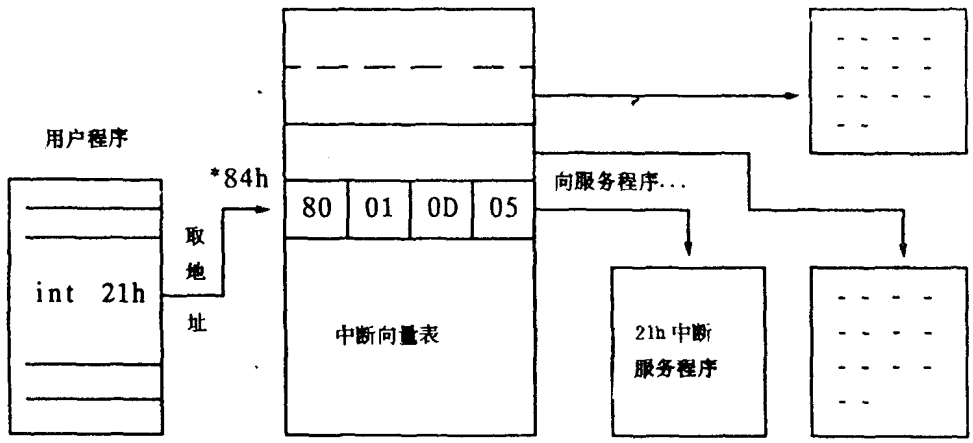
这样,第 n 号中断服务程序的地址保存在中断向量表中第 $4n$ 个字节上。当 CPU 执行到第 n 号中断指令时,便从中断向量表的第 4n 个地址处读取服务程序的地址(共 4 个字节),并运行服务程序。

```
A>debug
-a
0ACB:0100 int 21
0ACB:0102
-t  ← 跟踪看看 int 21h 往哪儿执行

AX=0000  BX=0000  CX=0000  DX=0000  SP=FFE8  BP=0000  SI=0000  D
```

```
DS=0ACB  ES=0ACB  SS=0ACB  CS=070D  IP=0180  NV UP DI PL NZ N

070D:0180  80FC4B       CMP     AH,4B  ← 到 070D:0180 处执行了

—d0:80 110  ← 看看 int 21h 的中断向量,真的是 070D:0180

0000:0080  07 0B EB 00 80 01 0D 07 — 42 02 DA 07 70 02 DA 07
                          ↑
                  21h * 4 = 84h 地址
```

注意,此例中 int 21h 中断向量表内的值(070d:0180)可能与用户所操作的结果不同,这是因为 MS—DOS 会移位,而 int 21h 的中断向量地址也是 MS—DOS 的一部分,因此会因不同的系统设置而移位。

在此例子中,当用 DEBUG 中的 T 命令来跟踪 int 21h 中断过程时,结果是 int 21h 刚开始运行时就先转移到 070D:0180 地址上,而用 D 命令转储中断向量表之后再观察,070D:0180 正是位于 0000:0084～0000:0087 位置上的数据(80 01 0d 07),所以 MS—DOS 的中断过程正是按这样的方式运行的:



注意:int 21h 的中断向量存放在向量表中 21h * 4=84h 的位置上。

对中断向量表的观察,在病毒研究中是非常重要的一个环节。读者现在不妨用 DEBUG 中的 D 命令列出 0000:0000～0000:03FF 中所有中断向量的内容来观察一番(最好印成报表保存,以备后用)。

IBM—PC 将 int 00h～int 1fh 设计成供 BIOS 及 8259 中断控制器使用,int 20h～int 3fh 供 DOS 使用,int 60h～int-67h 供用户使用,其余部分则供 BASIC 及以后使用。有关 BIOS 中各中断服务程序的用法,我们也会针对程序的需要酌情补充说明,并列在本书后面的参考篇里。此外,我们还将在阅读本书时会用到的一些重要信息整理在参考篇中供读者参考和查阅,例如:

(1) EXE 文件的结构图;

(2) PSP 的图示;

(3) COM 文件与 EXE 文件的比较；

(4) DOS int 21h 中各种有用的功能调用。

1. 中断与堆栈(stack)

当 CPU 执行 INT xxh 指令时，会先依次将 flags，CS 和 IP 推入堆栈中，然后才执行中断服务程序。当执行到中断服务程序内的 IRET 指令时，会依次从堆栈中取回 IP，CS 和 flags，再返回到当初 INT 发生时的地址处往下继续运行。当然，若有必要，在中断服务程序中也可以用 INT 指令来运行其它服务程序。

在此务必注意的是，INT 和 IRET 指令所使用的堆栈是调用者的堆栈，而在中断服务程序中，若没有特别地改变 SS 和 SP，则执行 push 和 pop 指令时所使用的堆栈区域仍然是同一个区域。所以，在进入中断服务程序之后，若 push 了几个 WORD 下去，执行 IRET 前就需要 pop 出几个 WORD 出来。若没有将 push 下去的值 pop 出来，或者没有先 push 就 pop(将由 INT 指令 push 下去的值 pop 出来)，就会使 IRET 所 pop 出的 flags，CS 及 IP 与原来的不同，从而不能返回到原来发生 INT 的地址处。但是，病毒程序则常利用这个原理，先把欲前往的地址 push 到堆栈中，再运行 IRET，这就等于执行 jump 的操作，而不是 IRET 了。这种方法常会见到。

2. 中断与远程调用、远程跳转

如果不使用 INT 指令，则可以采用远程调用(跨段区的调用)的方法来执行中断服务程序。当然，执行之前必须先获取中断向量。在程序中，常用远程指针的方式来存放中断向量：

```
int21h_vec LABEL DWORD        ; 声明 2 word 的标号
int21h_ip  DW    ?            ; 这个 data word 放 IP
iht21h_cs  DW    ?            ; 这个 data word 放 CS
```

当要调用 int 21h 中断服务程序时，就可以用 call 远程指针的方式来调用 int21h_vec，例如：

```
call DWORD  PTR cs:int21h_vec
```

或

```
call FAR    PTR cs:int21h_vec
```

但是，我们在前面曾说过，中断服务程序运行结束时会 pop 出 IP，CS 和 flags，而执行 call 指令时只会 push 出 CS 和 IP，所以我们在 call 之前必须先将 flags 的值 push 到堆栈中去。请看下面这个实例：

```
;程序名称:FCALL.ASM
. model small
. code
start:          jmp     get_vec
;
int21h_vec      LABEL   DWORD
int21h_ip       DW      ?
int21h_cs       DW      ?
```