

头用第一，智急密果 Practicability First, Intelligence Intensive

电脑编程技巧与维护

COMPUTER PROGRAMMING SKILLS & MAINTENANCE

1994 年 8 月



MICROSOFT
WINDOWS NTTM
COMPATIBLE
32-Bit Application



Microsoft
VISUAL C++

Development System for Windows[®] and Windows NT[®]

Microsoft[®]

中国电脑有希望



北京希望电脑公司成立于1985年1月,隶属于中国科学院,是经济体制改革以来中国大陆新兴的高技术企业之一,实行市场为主导的运行机制,并向股份有限公司过渡。

公司总部位于北京中关村,是“北京新技术产业开发试验区”首批认定的“高技术企业”,享有各项优惠政策。

公司现有职工160余人,工程技术人员占68%,高级技术人员占11.5%,总部除管理部门外,设有CAD部、SGI部,系统技术部,电源产品部、软件部和技术资料部等业务部门;公司在上海、南京、广州、成都设有独资子公司,分别负责华东、华南和西南的市场推广工作;在香港设有合资子公司,负责公司的产品开发和国际推广工作。公司各业务部门和子公司在全国设有100多个分销点,长期客户超过1000家,构成一个有效的全国销售网。

北京希望电脑公司的经营规模在八年间稳步发展,自1988年以来,希望公司一直在“北京新技术产业开发试验区”的排名表中列十名之内。人均历年平均产值超过50万,在全国同类企业中也名列前茅,是一个在全国享有知名度的高技术企业。

☆ 汉化微机CAD工作站、SGI工作站。

☆ “宝合”UPS电源。

☆ HOPE-126无线寻呼中心系统。

☆ UC DOS、dBASE IV 2.0中文版。

AUTO CAD 汉字环境等各种软件。

☆ 计算机技术资料。

地址:北京海淀路82号

电话:2567387、2562329、2565884、
2567819、2579873

信箱:北京8721信箱

邮政编码:100080

电挂: 0755 传真: 2561057



北京希望

电脑公司



Microsoft WIN32

SOFTWARE DEVELOPMENT KIT

Tools for Writing 32-Bit Applications for Windows™

希望汉字系统

汉字系统的希望 UC DOS 3.0

为什么我们总是人们追赶的目标？

※ 支持直接写屏,英文制表符自动识别

西文软件无需汉化即可支持汉字,采用独一无二的英文制表符识别技术,充分保持原版西文软件的神奇风采

※ 国内唯一真正实现零内存的汉字系统

386 以上微机,只要有一定的扩充内存,系统在启动时就可自动将所有程序和数据放入扩充内存,不占用任何 DOS 基本内存,不受 DOS 版本限制;对 286 或没有扩充内存的微机,可利用独家提供的“(虚拟内存管理器 VMS)”为应用程序提供最大的基本内存

※ 经香港金山公司授权,系统配备 WPS 进行文字处理

UCDOS 3.0 提供的 VPS 可同时使用 26 种高精度矢量字库进行模拟显示、打印输出,并在网络环境中首次真正实现共享打印,模拟显示和打印速度可提高 2-3 倍

※ 真正实现网络共享

网络版 UCDOS 3.0 无工作站(包括无盘工作站)数目限制;彻底解决网络中远程终端间的通讯问题;显示字库可存放于服务器上,为各站点保留更多的低端内存

※ 完善的汉字输入方法

系统提供全拼、简拼、双拼、五笔、普通和电报码等多种汉字输入方法,各种输入方法均带有大量词组;独创“记忆词组”,成功地解决了局部词组和专业性词汇输入困难的问题;支持屏幕取词功能,各输入法共享自定义词组

※ 强大的打印功能

国内唯一将点阵字库和矢量字库(26 种)有机结合的汉字系统,保证了低点阵汉字的质量;支持针打、喷打和激打,在任何软件中均可直接打印 2048×2048 点阵以内的任意点阵汉字;独特的打印字库还原技术,还原速度可与硬件媲美,打印速度超过硬字库

※ 特殊显示功能

可在屏幕上显示任何点阵的汉字,大小仅受屏幕尺寸限制;支持点、线、圆、椭圆、扇形、矩形及图形填充等多种作图功能;利用控制字符可实现对简谱文件的后台演奏;提供图像动态保存,所有特显功能均可用于各种图形模式并可在 FoxPro、Borland 系列等直接写屏型软件中使用

※ 彻底支持 DOS5.0、DOS6.0 和 DRDOS6.0

※ 彻底支持鼠标功能

※ 本系统以纯软件方式提供,是便携机用户的最佳选择

敬告用户:

1. UCDOS 及 PTDOS 老用户可凭用户卡进行优惠升级,单用户版 300 元/套,网络版 600 元/套
2. UCDOS 3.0 统一零售价单用户版 880 元/套,网络版 2000 元/套
3. 欢迎广大用户向我公司及各地经销单位索取 UCDOS 3.0 简版,15 元/套

北京希望电脑公司软件部

地址:北京 海淀路 82 号

信箱:北京 8721 信箱

邮码:100080

电话:2579873、8422024、

8422025

传真:2561057

上海希望电脑公司

地址:(200052)上海新华路 416 号

电话(传真):(021)2521656、

2569929

成都希望电脑公司

地址:(610015)成都市新南路
四维村街 6 号(磨子桥附近)

电话(传真):(028)5589787、

5556333

广州希望电脑技术公司

地址:(510620)广州天河体

育西路育蕾三街二号

电话:(020)7505151、7505152、

7505153

传真:(020)7500275

南京希望电脑技术公司

地址:(210005)南京市中山南
路 105 号

电话:(025)4410794、4410549

传真:(025)4410548



即使您对计算机语言一窍不通,您立刻会成为编程大师;即使您已经是一名高级程序员,您立刻会成千倍地提高编程效率

《雅奇 MIS》新一代微机管理信息系统自动生成器

思想變成程序

夢想終成現實

《雅奇 MIS》微机自动编程系统,能广泛地应用于各行各业对信息管理系统自主开发的需要。即使是不懂计算机语言的普通管理人员,利用《雅奇 MIS》编程,也立刻会成为编程大师;而对于高级编程人员,利用《雅奇 MIS》辅助编程,会成千倍地提高编程效率。《雅奇 MIS》自动生成的,一体化的网络或单用户的——财务、人事、档案、购销、生产、计划、商贸等,各行各业的各种图文并茂的信息管理系统,可以满足所有领域的信息管理需要。

《雅奇 MIS》自动编程系统,尤如“母鸡下蛋”,可生成无数套不依赖于生成器,而独立运行的管理信息系统。

《雅奇 MIS》关键技术及七大特点

一、设计过程,直观可视:

程序设计时,完全以人机对话的方式进行。开发过程中的菜单设计、屏幕格式设计、复杂报表设计等,直观可视,极为简单,是完全的所见即所得。

二、开发速度,取决于打字速度:

程序设计时,完全不与计算机语言打交道,任何人只要会用计算机打中文,按动几个简单的功能键,就能利用《雅奇 MIS》开发信息管理系统,而且开发效率是手工编程的千倍以上,一般应用系统可以达到“立等可取”的地步。自动生成的 PRG 源程序,与手工编程结果完全一样,对高级用户的特殊需求,还可再行编辑。

三、界面友好,菜单华丽:

《雅奇 MIS》的用户界面十分友好,生成器及生成的应用程序的菜单,以及各种屏幕界面,都是工作在图形方式下,使用户界面极其优美华丽,速度极快,即使是高水平的手工编程也难以实现。

四、图文并管,功能超凡:

可自动生成彩色图象与文字“图文同屏”、“完全窗口式”显示的图文信息管理系统,即使是现有的,专门用于图文显示的软件,也无法与之相比。

五、报表设计,复杂多样:

“专用报表”、“汇总报表”、“任意报表”,无论如何复杂,只要屏幕能画出的,就能设计生成并打印报表。并且支持八个库数据的同报表打印。

六、“积木化”技术,堪称一绝:

《雅奇 MIS》以“积木化”的模块技术,高度精练出近 30 个功能模块,几乎使所有的,信息管理系统开发所要求的,各种各样的复杂需求包容其中。

七、“智能化”技术,独树一帜:

用《雅奇 MIS》开发的应用系统,生成时有自动记忆功能,后期维护工作十分简单。当需求变化时,只要将运行系统重新挂接到生成器,进行局部的或全部维护修改即可,几乎没有后期维护的负担。

《雅奇 MIS》已在近万家用户中使用,针对不同领域、不同行业的各种复杂信息管理需求,都能以极简单轻松的操作顺利实现,使用效果极佳,用户普遍反映,《雅奇 MIS》是一套真正能“干活”的自动编程工具。

专家预测《雅奇 MIS》将完全取代传统的手工编程方式,而成为信息管理领域软件开发的普及性高科技产品。

全国各地代理经销商名单(同时办理当地代理赠送业务)

单位名称	电话
安徽合肥安兴高科技开发公司	2649222
北京高林技贸公司	2543815
北京三江新技术产业公司	3816193
北京微宏电脑软件研究所	2560964
福建厦门四通公司	5088362
广东广州黄花岗产业区物科实业公司	3832081
广东广州瑞达信息技术服务部	3330898-3506
广东广州市海达科技有限公司	5515979-235
广东广州中兴现代办公设备公司	5515870
广东深圳爱华电子有限公司	3208546
广东深圳意百达科技有限公司	3333336
广东珠海海韵高科技研究所	8892736
广西南宁市数据设备公司	569162
海南省海口市安泰光电子实业有限公司	8662946
河北沧州创新电脑系统服务部	222871
河南开封师专电子技术中心	554795
河南洛阳市亚普技贸公司	426932
河南新乡神通计算机公司	220286
河南郑州艺高电脑新思维广告有限公司	7445877
黑龙江佳木斯信息市场	247931
黑龙江齐齐哈尔惠通计算机公司	425911

单位名称	电话
湖北省武汉市皇龙软件销售中心	7804039
江苏淮阴通经技术发展有限公司	341264
江苏淮阴市清浦电子技术公司	335053
江苏南京电子信息产业集团公司	3342145
江西南昌亚特南达电子电脑有限公司	211926
辽宁本溪市通用计算机发展公司	246277
辽宁朝阳市比塔区银丰科技公司	219720
辽宁锦州华侨电脑科技发展有限公司	236693
内蒙古包头市理想电脑公司	556655-188
山东省泰安市云海科技实业公司	335040
山西大同计算机有限公司	242411
山西太原钢联新技术产业公司	6042720
陕西省汉中市电脑软件公司	219324
陕西西安市新航电子技术公司	4253439
上海海贝船舶技术研究所	4860037
上海两英家文经营部	2583576
上海中兴电脑经营服务公司	4395300-2402
上海市劳动局信息中心	3281394
四川攀枝花市金谷科技贸易发展公司	334187

单位名称	电话
天津津海技术产业公司	3353419
天津山川电子有限公司	7383824-2188
新疆乌鲁木齐电子信息技术开发公司	263078
云南昆明工学院科技开发部	5146647
浙江杭州方欣公司软件天地	8085512
浙江杭州华东电子技术经营部	5160198
浙江杭州星邦信息处理电脑有限公司	5177185
浙江金华联想计算机通信工程公司	327475
浙江省杭州计算技术研究所	5151941

特别注意
免费赠送

无与伦比的超凡功能
令世人瞩目!!!

网络版/单用户版,统一单价:1280元(V3.0) 980元(V2.8)

特 别 注 意

全国各地(包括港、澳、台地区)经销单位,各大院校及科研机构,可以凭介绍信办理免费赠送《雅奇 MIS》手续,每单位限额一套,所赠全套软件均为商品化正版,不得转售。(工本费 280 元)

办理赠送或购买手续,可以通过信函或传真办理,信函地址及收件人:北京中关村海淀路 29 号百骏电子大院二楼(海洋公司旁)张晓飞收。

汇款请寄:

开户行:建设银行海淀支行中关村办事处
帐号:26304493
户名:大连雅奇电脑公司北京科贸公司
邮寄请另加特快专递费 20 元/套。

《雅奇 MIS》全套商品化正版包装包括:

1. 系统盘一套共 5 张
2. 举例生成的应用系统两套
3. 教学录像带一套
4. 用户手册一套
5. 操作手册一套
6. 信誉卡一张
7. 精美包装盒一只

为使全国各地经销单位能尽快就地就近获得全套《雅奇 MIS》赠送软件,望各地具备条件的经销单位,速与北京公司联系,申请在当地代办《雅奇 MIS》赠送业务。

地址:北京中关村海淀路 29 号百骏电子大院二楼(海洋公司旁),电话:2642156 邮编:100080 传真:2642157 联系人:张晓飞

顾问 郭平欣 刘洪昆

名誉社长 周明陶

社长 毕研元

副社长 袁保佑

总编 秦人华

执行总编 秦长久

副总编 王路敬 孙毅

编辑 开元 程钧 易言

艾蒙 田艾 肖贻

苏古 阮薇 温达

管逸群 业成

公关部主任 吕莉 伊梦

出版部主任 祁连顺

发行部主任 孙茹萍

编辑出版 《电脑编程技巧与维护》杂志社

地址 北京市劳动人民文化宫内

邮编 100006

编辑部电话 5123823

公关部电话 5232502

照排 《电脑编程技巧与维护》杂志社

电脑排版部

印刷 北京市地质矿产局印刷厂

订阅处 全国各地邮电局

国内总发行 山东省潍坊市邮电局

邮发代号 24-106

刊号 CN11-3411/TP

每期定价 3.80 元

出版日期 1994 年 8 月 8 日

敬告读者

1995 年报刊征订工作即将开始,本刊
提请您切勿错过当地邮局订阅时间。

本刊订阅代号:24-106

另外,本刊每期所刊实用程序磁盘
(含源程序及可执行程序)订价 15 元/片,
欢迎读者向本社订购,地址:北京市劳动
人民文化宫内《电脑编程技巧与维护》杂
志社发行部,邮编:100006,并注明购买
期号及份数。

实用第一,智慧密集

电脑编程技巧与维护 月刊

1994 年第 2 期

(总第 2 期)

软平台

- 新一代软平台——Microsoft 的 Chicago (5)
- 在 MS DOS5.0、6.0 中使用 2.13H 的虚拟盘 (7)
- 为 PWIN 3.1 快速制作首尾码输入法 (8)
- 加快键盘的响应速度 (9)
- 编写 TSR 程序的一些新技术 (10)

图形图象处理

- BMP 图象文件的快速显示 (13)
- VGA 图形的灰度打印 (16)
- 界面随机填充技术探讨 (18)
- PUT 文件格式的分析与程序调用 (19)

汉字处理

- 从 SPT 到 BMP——谈高点阵汉字的显示 (21)
- 如何在文本模式下显示汉字 (24)
- 西文状态下汉字的无级放大和平滑显示 (25)
- 西文方式下 24 点阵彩色立体汉字放大显示 (27)

编程语言

- Visual C++ 实现多平台应用开发 (29)
- 访问扩展内存的 C++ 子程序 (30)
- C++ 语言环境下的 Windows 编程技术 (32)
- 将 TXT 文本文件转换为可执行阅读文件的方法 (35)
- FoxBASE+ 调用 C 语言的新方法 (36)
- 可编程、超长输入函数的实现 (39)
- 用 MFC 读写数据库 (40)
- DevCast 尽展 Visual Basic 的功能 (41)
- 步长型循环的三值选择技巧 (43)

计算机安全

- COBOL 程序中输出信息的加密方法 (44)
- 实现程序反跟踪的一些技巧 (45)

征稿启事

《电脑编程技巧与维护》是专门为从事电脑编程和系统应用与维护人员创办的专业性和实用性很强的技术杂志,它的主要栏目有:

软平台,该栏目以我国应用面广、发展潜力大的操作系统为基础,介绍这些系统的新功能、新特性,以及利用这些功能和特性进行开发、应用的技术和技巧。**图形图象处理**,该栏目主要介绍图形图象软件的编制技术、技巧和编程的具体方法。它包括软件界面的开发、鼠标器的驱动、各种图形图象格式的介绍及应用等。**汉字处理**,该栏目主要介绍汉字处理的编程实用技术,它包括:汉字的直接写屏、各种字库的组织结构,及汉字的放大、平滑、旋转等多方面的内容。**编程语言**,该栏目主要介绍编程语言的编程技术、技巧及它们所提供的函数及灵活应用技术,它的文章将以实用程序说明应用的方法、技术与技巧。**数据库**,该栏目主要介绍数据库系统的新功能、新用法,以及利用这些数据库系统进行开发工作的经验、编程技术、技巧和方法。**计算机安全**,该栏目主要介绍加密技术、反跟踪技术、反病毒技术及具体病毒的检测和清除技术等。**网络与通讯**,该栏目主要介绍网络的应用技术,包括网络操作系统和在网络环境下应用开发的技术、技巧。**软件维护**,该栏目主要介绍软件功能的用户扩充或扩展及软件故障的具体现象和原因,并给出合理的恢复方法。**实用软件**,该栏目主要介绍如系统维护工具、压缩工具、调试工具的应用技术与技巧。**硬件维护**,该栏目介绍计算机硬件系统、部件的维护与维修经验。

在以上这些方面有经验所得的同仁请不吝赐稿。稿件一旦录用即寄样刊及稿酬。本刊每年都将举办优秀撰稿人评选活动,评选出的优胜者,本刊将免费赠送下一年的《电脑编程技巧与维护》杂志。来稿请寄(100006)北京市劳动人民文化宫内《电脑编程技巧与维护》编辑部。稿件请用稿纸书写整齐或打印出来,所附程序均请打印清楚。作者姓名、单位及通信地址、邮编请用整齐的字迹写在醒目的位置。来稿必须将有关的程序完整地附上。本刊鼓励软盘投稿(同时也请附上一份打印稿),磁盘投寄的稿件一旦录用,稿费我们将加倍考虑磁盘的成本费和邮费。稿件不得一稿两投,本刊在收到稿件后三个月内发录用通知,在此之后没收到录用通知者,稿件可另行处理。

Practice First, Intelligence Intensive

Computer Program Skills & Maintenance

新颖的子目录加密机制	(47)
磁盘损坏时文本文件的一种恢复方法	(49)
CPU“逆行”程序加密法	(50)
Xenix 系统下 tar 盘文件的恢复	(52)

数据库

运用 FoxBASE+ 实现 Windows 的平滑技术	(53)
DOS 下 FoxPro 2.5 通用多窗口处理程序设计	(54)
FoxPro 2.5 的程序自动运行技术	(58)
FoxPro 系统菜单的汉化方法	(60)
使用 FoxPro 的 Browse 命令进行多项选择	(63)

软件维护

优化内存管理、合理使用金山 CCDOS6.0F	(64)
浅谈如何提高 DOS 系统的使用效率	(65)
Xenix 系统引导时常见故障及其维护	(67)

硬件维护

硬盘的使用及维护	(68)
主机电源维修一例	(70)
微机常见热故障的排除	(70)
286 兼容机常见故障的分析及排除	(71)

网络与通讯

采用 OLE 的网络分布式目标通信	(72)
-------------------------	------

电脑博士信箱

DOS 与 Windows	(74)
---------------------	------

电脑反病毒信息公告

本刊全部内容的版权,归本社所有,未经同意,不得转载。

专访 *Microsoft* 高级技术专家

□ 本刊编辑部

问:请问 Windows 将如何发展?

随着 Windows 的发展, DOS 将不成为贵公司主要支持的系统,那么 DOS 的发展方向是什么?

答:我们现在开发一种新版本的 Windows 产品,其代号为“Chicago”这种新产品采用“双剪贴(double clipping)”系统,使 Win32 的应用更完善、可靠,可以跨越更多的平台,此外,我们还要加强在 Windows NT 方面的发展。DOS 作为一种基本的支持系统,其作用在短期内不会被代替,我们将继续推出更高版本的 DOS。另外,我们的一个重要发展方向就是使我们的产品本地化,以便我们的产品能更好地满足世界各地的需要。

问:请问你们在产品汉化方面有什么策略,汉化的过程需要多长时间?

答:我们实行产品汉化的一个重要策略就是与中国大陆的有关公司进行充分合作,吸收大陆本土上的有关技术人员与我们合作,这种方式,对我们的汉化工作十分有利。各种产品的汉化时间是不一样的,这要看汉化的对象、情况及汉化工作的组织等。

问:请问你们在同中国第三方进行合作方面有什么打算?除了现有的合作伙伴,是否还要寻求其它的合作伙伴?

答:是的,现在我们除了与联想公司有很多合作外,还与长城公司建立了合作关系,我们在产品代理、技术服务及培训方面正在同越来越多的中国大陆的公司进行合作。

问:据了解,一些中国用户经常遇到 Microsoft 提供的技术支持不全面或不及时的问题,请问你们将如何解决这个问题?

答:今年是我们公司在中国设立机构的第二年,时间不长,存在一些问题难免的。不过我们正在逐步改进,随着时间的推移,我们将在中国大陆成立更多的技术服务中心,逐步形成技术支持网络,加强对中国软件开发人员的培训,定期举行各种培训班或技术讲座。现在我们已经与中国许多公司签订了授权培训中心的协议,并将举行 Microsoft 的资格考试,为中国的软件技术人员提供更多的便利条件。

问:Microsoft 每次在开发新产品时,都要引入一些新概念、新技术,请问你们是否能将这些新技术、新概念汇编起来,

Microsoft 作为世界著名的软件公司,其产品 MS-DOS、Windows 等在我国已有了广泛的应用,Microsoft 的产品开发策略是我们广大用户所关心的问题之一,许多读者来信、来电问及此事。有鉴于此,本刊编辑部借 Microsoft 高级技术专家来国内访问之际,就中国用户对 Microsoft 公司产品比较关注的一些问题,采访了他们,现整理发表,供用户参考。

系统地介绍给中国用户?

答:我们很注重技术资料的本地化工作。我们产品的资料往往被译成世界多种不同的文字,以满足世界各地不同用户的需要,如在远东地区,我们的翻译工作包括汉语、日语、朝鲜语等。在我们销往中国的产品中,一般都包含有系统的中文资料或英文资料。不过有些新技术、新概念我们不能介绍给国外用户,因为它们具有保密性,也受知识产权保护的限制。但对于我们能够提供的资料,我们将尽力系统地提供给中国用户。

问:请问 FoxPro2.5 的下一个版本是什么?有什么不同?

答:FoxPro2.5 的下一个版本是 FoxPro2.6,它主要是要求运行的环境要高一些,运行速度快一些,支持更多平台的应用开发。

问:最近 IBM 公司与中国电子部签订了一份谅解备忘录,中国将从 IBM 引进更多的技术。中国是一个日益发展的大市场,许多外国公司,如 IBM、Apple、Intel 等纷纷扩大在中国的业务联系,基于这种情况,请问 Microsoft 公司有什么打算,如在技术支持、价格方面有什么策略?

答:各个公司的情况不一样,其发展方向、主要产品的开发不尽相同,因此在中国扩大业务的侧重点也不尽相同,当然竞争肯定是存在的。我们将尽力做好我们自己优势产品的开发与升级,在中国逐步扩大对用户的技术服务网络,满足用户对技术支持的要求。

问:Microsoft 公司的一些产品价格很高,中国用户往往承受不了,对此,贵公司是否有所考虑?

答:我们产品价格的确定因素是多方面的,它主要基于产品的成本、所要对其进行的技术支持等。此外,产品的销售都要保证不仅是国内的,还有国外的经销商的利润,如果没有利润的话,也就不存在买卖活动了。

问:Novell 公司已经推出 Novell DOS 7,请问 Microsoft 是否也有这方面的开发?如果有,两者有什么不同?

答:我们公司对 DOS 7 的开发工作正在进行之中。两者的主要区别在于其内部结构、速度以及用户使用命令的方式等方面。

新一代软平台—— Microsoft 的 Chicago

□ 吴新瞻 译

Microsoft Windows 操作系统的下一个升级产品将是 Chicago, 预计在今年晚些时候出台。Microsoft 深信, Chicago 的出现将使微机更容易使用。随之而来的是对以 Windows 为基础的应用程序的需求增加, 开发基于 Windows 的应用程序的市场扩大。

作为一个 Windows 应用程序的开发人员, 既希望自己编制的应用程序能在现有的 Windows 平台上发挥最大的功能, 也希望能在其他的、未来的 Windows、Chicago 平台上运行。为达到这个目的, Chicago 部门经理 David Cole 向应用程序开发人员提出的要求有五点:

头两点是, 在开发 Windows 应用程序时要使用 Win32 和 OLE 2.0。实际上, Win32 和 OLE 2.0 是在一切 Windows 平台上编写大型应用程序的基本构件。

第三点是开发人员必须遵循“用户界面设计指南 (User Interface Design Guide)”, 这是 Microsoft 正在为未来的 Windows 操作系统开发的界面。

第四点是应用程序应该支持即插即用的功能, 使用户可以在接插一种新的设备后, 打开 PC 机能立即使用, 即系统能自动地感知设备的接插与卸除。

第五点是应用程序应该通过提供登记信息、长文件名、UNC (Universal Naming Convention) 路径名支持以及自己的文件阅读程序, 达到与 Chicago 外壳很好地结合。

按照这五点要求开发出来的应用程序既可以在 Windows NT 和 Windows 3.1 操作系统 (带 Win32 应用编程接口) 上运行, 也可以在 Windows NT 的下一个主要版本 Cairo 上运行。

一、Chicago 的五点要求

1. Win32 的优点和作用

Win32 应用编程接口 Win32 API 可

以使你开发出一个应用程序, 它既可以在 Chicago、Windows NT 和 Windows 3.x 等多种 Windows 平台上顺利地运行, 又能利用支撑平台的特点。其主要原因是 Win32 API 具有一个向上兼容的 Win32 函数、消息和结构集, 它们不论是在 Chicago、Windows NT、Windows 3.x (带 Win32), 还是在未来的 Cairo 平台上都可兼容地使用。

当然, 在不同的 Windows 平台上实现 Win32 API 的程序并不完全一样。例如, 对所有的 Windows 平台, Win32 API 提供的环境都具有 32 位虚拟内存管理 (VirtualAlloc) 和内存映射文件这样一些强大的功能。但是 Chicago 和 Windows NT 可以让应用编程人员使用一些补充的功能, 如线索 (thread) 和长文件名, 而 Windows 3.x (带 Win32) 并不提供这些功能。另一方面, Windows NT 具有, 但 Chicago 和 Win32 并不具有的附加功能是安全功能和 Unicode API。

因为在 Win32 和 Chicago 中剔除了未支持的函数, 用户的应用程序可利用 GetVersion 函数检测到支撑平台是哪一种, 从而选择、利用不同的功能。例如, 当在 Chicago、Windows NT 和 Cairo 上运行时, 应用程序可利用长文件名和线索, 而在带 Win32 的 Windows 3.x 上运行时, 应不用线索, 且采用 8.3 命名规定。

与 16 位的 Windows 应用程序不一样, 在 Chicago 和 Windows NT 上运行的 Win32 应用程序优先按多任务执行, 而且可以在各个保护地址空间内运行多个执行线索。Win32 使用户可以真正并发地执行多个任务。例如, 你可以同时进行编译、做复杂的电子表格计算、打印文档, 以及运行游戏软件。

Win32 API 可以移植至非 Intel 平台, 如 MIPS R4000/R4400、Digital Alpha AXP 及 PowerPC。这样, 只需重

编译, 就能使应用程序在多种硬件配置下运行。

在 Win32 的 SDK 中还包含了一种工具软件 PortTool, 利用它可以将原来的 16 位应用程序移植为 Win32 应用程序。PortTool 在对源代码扫描时, 会指出哪些代码需要改动。

在 MDL (Microsoft Development Library) 库中包含 PortTool 的修订版, 不仅可帮助用户进行应用程序的移植, 还可以向用户说明如何在应用程序中利用支撑平台的功能, 而且仍然在其他平台上可以运行。

所有由 Win32 和 Windows NT 共同支持的函数也得到 Chicago 的支持。因此, 如果你的应用程序要在 Win32 和 Windows NT 上都能运行的话, 只需要在 Chicago 上进行测试就行了。在 MDL 库 5 月版中还包含一张“编写大 Win32 应用程序”表, 它列举了各种平台提供的功能, 可以帮助用户在应用程序中适当地利用这些功能。

2. 对象连接与嵌入

对象连接与嵌入 OLE 2.0 为终端用户提供了一致的方法, 建立复合的文档, 实现应用程序之间的可编程能力, 以及应用程序与桌面区域之间的拖放 (drag and drop) 功能。它还可以使用户通过 Chicago 的外壳观察丰富的文档信息, 例如图标、反应动作及应用程序中特有的其他信息。

Chicago 是 Microsoft 为个人电脑建立的第一个实现文档中心模型 (Document-Centric Model) 的操作系统产品。应用程序要想在 Chicago 环境下更好地得到配合, 应该采用 OLE 2.0 技术。

3. 用户界面设计指南

Windows 操作系统和 Windows 应用程序有一个很大的长处, 这就是它们的一致性。只要你学会了使用 Windows

及至少一个 Windows 应用程序,遇到别的 Windows 应用程序也就不难掌握了。

有几种工具可以帮助你编写 Windows 应用程序时达到一致性,其中之一就是“用户界面设计指南”。它说明了应该如何设计软件,才能够在 Windows 操作系统上运行。其目的是促进 Windows 应用程序内部及相互之间在外观上和功能上保持某种一致性。

Windows 在接口方面的加强体现了从基本的图形用户接口设计进化到面向对象的用户接口设计的发展方向。这种接口不是以应用为中心,而是侧重于以数据为中心。这种设计是沿着 OLE 方向的继续发展。为此,开发人员和设计人员有必要重新考虑一下他们的软件接口,究竟以什么作为基本对象,它们的性质和操作又是什么。

只要通过标准的 Windows API 调用,就可以自动地得到许多新的可视特点。例如,不必去修改你的应用程序,就可以在应用实现中实现一个标题条,条中字符是向左对齐的。同样,可以自动地在应用实现中实现一些 Windows 的控制功能。但是,如果用户需要在应用实现中实现一些特有的控制功能,那就有必要了解设计指南,使用新的公用控制与对话功能。

Chicago 将会引进几种新的 Win32 API,提供对公用控制与对话的支持。这些新的公用控制与对话功能将会放入 Chicago 或 Cairo 发行时提供的动态连接库(DLL)中。因此,即使是当前的 Win32 和 Windows NT 版本,到那时也能获得这些新的公用控制与对话功能。

4. 即插即用使 PC 更好使用

应用程序应该提供一种新型的即插即用消息。Microsoft 正在与 PC 机系统与部件厂商密切合作,旨在开发一种新的即插即用硬件设计标准,它可以在 PC 机上进行自动地与动态地设备配置。即插即用将成为 Microsoft 操作系统产品的一种标准的特点,而 Chicago 是实现即插即用的第一个产品。

即插即用表面上看来是硬件方面的工作。它的确提供了一种关键的功能,使得应用程序可以对系统中的改变智能地作出响应。通过这种途径,应用程序总的说来与 Chicago 将很好地集成在一起。当硬件配置出现动态变化时,应用程序将接收到消息,例如插入了一个传真机的调制解调器,或插入网络适配卡等。无需用户界面的介入,应用程序就可自动

地利用这些新的硬件能力。

这种设计使得 PC 机对新、老用户来说都更加直观。举例来说,如果计算机从网络上卸下,当用户打开网络上的文件时,应用程序就会发出警告。另一种情况是:在应用程序中使用的图标或位图可以根据显示的分辨率而有不同的设置。当应用程序得到 WM_DISPLAYCHANGED 消息时,就可以相应地改变图标和位图的设置。

5. Chicago 外壳支持

Chicago 外壳提供了单一的工具来使用全部系统资源(文档、程序、打印机、实用程序等等),这使得用户使用系统资源更加简便。Chicago 联合全部各不相同的管理程序,如程序管理器、文件管理器、打印管理器、控制面板、Windows 设置程序等,使得全部资源可以组织在灵活的文件夹层次结构中。

文件阅读程序(file viewer)也便利了用户查看文件内容,不论建立文件的应用程序安装与否都可以实现。Chicago 对几种数据类型建立了标准文件阅读程序,而用户还可以为自己的数据类型建立文件阅读程序,或建立包含更多特色的文件阅读程序。将来的开发库版本会提供有关文件阅读程序的更多信息。

Chicago 支持长文件名和 UNC 路径名(\\servername\share)使得文件更容易浏览和定位。应用程序应该使用 UNC 路径名,而不是使用驱动器字母,这样,在下次会话期开始时无需用户重新连接到同一个驱动器字母,就可以访问以网络共享方式打开的文件。

Windows NT 和 Win32 API 也支持长文件名和 UNC 路径名。如使用公用对话(common dialogs),可自动获得长文件名和 UNC 路径名支持。用户只需使缓冲区有足够的空间存储 255 个字符的文件名及另外 260 个字符的 UNC 路径名。

以上五方面是 Chicago 应用程序具有的重要特点,在设计、开发 Chicago 应用程序时应充分注意。这样的应用程序在 Windows NT 和 Windows 3.1 带 Win32)中同样能够运行。

二、Chicago 应用开发的关键

无论是将 16 位基于 Windows 的应用程序转换到使用 Win32 OLE 应用程序,还是利用几个工具产生新的使用 Win32 OLE 的应用程序,你都能很快地开展。利用“用户界面设计指南”你

可以通过现有的公用对话,以及对新的控制的理解来开发、利用新的 Chicago 用户界面的应用程序。你可以用“制作 Win32 应用程序”表来帮助明白三种平台提供的功能,当你要在所有目标平台上测试你的应用程序时,可以肯定一个在 Windows NT 和 Win32 上运行的应用程序也同样能在 Chicago 上运行。因为在不同的平台间有很多差异,所以在所有三种平台上检测是十分重要的。其中关键的区别如下:

1. 对于所有基于 Win32 的应用程序,Chicago 和 Windows NT 是优先的多任务环境,提供单独的受保护的地址空间,而带有 Win32 的 Windows 3.1 是一个非优先的多任务环境,所有的应用程序共用相同的地址空间。

2. 采用 Win32 的 Windows 3.1 具有一个同步的输入队列,Chicago 和 Windows NT 则是一个非同步的输入队列。

3. Win32 在 Windows 3.1 的上层运行,这样它也具有对 GDI(图象设备接口)空间的 64K 的限制。Chicago 和 Windows NT 以超过 32 位堆(heap)来分配空间。这样,能达到最大的可用内存。

4. 带有 Win3.2 的 Windows 3.1 和 Chicago 具有 16 位通用坐标系统,对于文本和图象,X 和 Y 限制在 32 个范围之内。Window NT 采用 32 位通用坐标系统,范围可达 2GB,如果在文本和图象的功能中用全 32 位值,在执行需要的操作之前,Win32 和 Chicago 将截断坐标值的高 16 位。

三、制作 Chicago 应用的十项注意事项

1. 使用 Win32 API。
2. 实现 OLE2.0 的拖放功能。
3. 在 OLE2.0 混合文件中,填充综合信息流。
4. 注册 OLEClassID。
5. 按“用户界面开发指南”的规则做。
6. 使用公用的控制和对话。
7. 支持长文件名和 UNC 路径名。
8. 监测即插即用事件。
9. 在系统注册处注册大的、小的图标,缺省的动词等。
10. 在所有的 Windows 的平台上检测应用软件。

(本文英文资料由 Microsoft 提供)

在 MS-DOS 5.0、6.0 中使用 2.13H 的虚拟盘

□ 曹国钧

MS-DOS 5.0 以下版本中能正确地使用 2.13H 汉字系统的虚拟盘,但在 MS-DOS 5.0/6.0/6.2 的环境中却无法使用 2.13H 的虚拟盘,究其原因,是 2.13H 的虚拟盘的确定问题。

一、2.13H 虚拟盘的安装

在 2.13H 的虚拟盘安装文件 FILE3.COM 中,其虚拟盘的起始地址的默认值为 1024KB,在文件 FILE3.COM 的相对于 100H 位移 027A 处反映为 10H(1024/64=16),在 MS-DOS 5.0 以下版本该值是正确的。但从 MS-DOS 5.0 起,DOS 提供了扩充内存管理程序 HIMEM.SYS 和扩展内存管理程序 EMM386.EXE,它们对微机 640KB 以上的内存进行了管理,即:

(1)安装了 HIMEM.SYS 后,用户可以使用微机的 1M(1024KB)以上的扩充内存,在该区域可设置虚拟盘(RAMDRIVER.SYS)和高速缓冲区(SMARTDRV.SYS)等,并在 1024KB-1088KB 区域保留了 64KB 的扩充内存,通过在 CONFIG.SYS 中设置 DOS=HIGH 可将 MS-DOS 系统的一部分装入该区域,从而节约了 64KB 的常规内存空间。

(2)安装了 EMM386.EXE 后,用户就能使用 640KB-1024KB 的闲置空间,但该程序的设置需占用 512KB 的扩充内存。其中:参数 RAM XXXX 可将 XXXXKB 的扩充空间(XMS)模拟成 XXXXKB 的扩展空间(EMS),这要占用 640KB-1024KB 中的 64KB 的空间作为扩展空间的映射页;

参数 NOEMS 可释放 640KB-1024KB 中的 64KB 空间;

若用户使用的显示适配器为彩色显示适配器(CGA/EGA/VGA),则用参数:

I=B000-B7FF 释放单显区域 32KB 的空间;

FRAME=E000(或 I=E000-EFFF)可释放 64KB 的空间。

通过在 CONFIG.SYS 中设置 DOS=UMB 或在 AUTOEXEC.BAT 中用 LH 命令,可将应用程序或设备驱动程序装入高端内存(UMB)中,从而节约了常规空间。

由上面可看到,2.13H 的虚拟盘的起始位置由以下四个部分计算得到:

(1)A=1024KB,此为微机的扩充空间的起始位置,由 HIMEM.SYS 设置;

(2)B=64KB,由 HIMEM.SYS 设置的 DOS 系统的保留空间 1024KB-1088KB,用 DOS=HIGH 能使用该部分空间;

(3)C=512KB,此为由 EMM386.EXE 所占用的扩充空间;

(4)D=XXXXKB,此为在虚拟盘设置之前的其他使用扩充内存的设备驱动程序,如设置高速缓冲区 SMARTDRV.SYS 等所占用的扩充空间之和。

因此,2.13H 虚拟盘的起始地址应为 A+B+C+D,除以 64KB(一段为 64KB)后再转化成 16 进制即为文件 FILE3.COM 的 027A 处的数值。

二、具体的实现方法

笔者在 AST 486(4M 内存、VGA、MS-DOS 6.0)微机上一个典型配置例子。

CONFIG.SYS 的内容如下:

(A)DEVICE=d:\pwin\HIMEM.

SYS

(B)DOS=HIGH

(C)DEVICE=D:\PWIN\EMM386.EXE

FRAME=E000 I=B000-B7FF

NOEMS

dos=umb

devicehigh=c:\213\ansi.sys

DEVICEHIGH=C:\DOS\SETVER.EXE

devicehigh=c:\dos\dblspace.sys
/MOVE

(D)DEVICEHIGH=C:\dos\SMARTDRV.sys 1024 1024

DEVICEHIGH=C:\DOS\RAMDRIVE.SYS 384/E

shell=c:\command.com /P/E:256

numlock=off

FILES=20

buffers=5

LASTDRIVE=E

STACKS=0,0

上面配置可留给用户约 190KB 的 UMB,能实现 2.13H、SPDOS5.0-6.0F 等汉字系统零内存的设想,其中标以 A,B,C,D 就是计算虚拟盘起始位置所需要的数值,其中:A=1024KB,B=64KB,C=512KB,D=1024KB,则 FILE3.COM 中的 027A 处的数值应改为:

$(A+B+C+D)/64KB$

$= (1024+64+512+1024)/64$

$= 41(29H)$

因此,若将设置虚拟盘的设备驱动程序放到 EMM386.EXE 之前,则 A=1024KB,B=64KB,C,D=0,经计算其起始位置为 $(A+B)/64=17(11H)$,这仅是上面的一个特例,用户按该方式设置虚拟盘,可将文件 FILE3.COM 中的 027A 处的 10 改成 11(该情况已在一些杂志或报刊上讨论过)。

通过以上对 2.13H 的虚拟盘起始位置的分析,根据此原理,用户能更加合理地使用扩展内存(XMS)。

为 PWIN 3.1 制作首尾码输入法

□ 曹 钧

Windows 3.1 中文版已提供了国标、全拼和双拼双音输入法,却未提供其他输入法,如五笔字型、首尾码和自然码等,本文以首尾码为例,说明如何为 Windows 3.1 中文版(PWIN)挂接常见的输入法。

实际上,在 PWIN 中提供了一个通用编码输入法的接口 WINMB.IME,用户只要给出 Windows 规定的通用编码文件,就能经过通用编码编译器 CON-VMB.EXE for Windows 将该编码文件转化成 PWIN 可辨识的输入法文件。因此,获取编码文件是至关重要的。在 PWIN 中,可以提供至多 20 个用户定义的扩充输入法,一般情况下,这已足够了。通用编码文件的格式为:

说明部分 (DESCRIPTION)

[Description]

Name=输入法名称

MaxCodes=输入法最多的编码个数

UsedCodes=用户能使用的编码,如 a~z 等

WildChar=通配符,一般为 z,?,* 等

Sort=0 为在编译时编码文件不重新排序,1 为重新排序

编码部分 (TEXT)

[Text]

<用户编码>

其中<用户编码>格式为:汉字或词组+编码,编码应小写,如:

国务院 gwy

由上面定义格式可看出,编码部分中的(用户编码)是用户特别关心的,若能获取到该部分,则该编码的输入法就能挂接到 PWIN 中,至于编码的处理与查询等均由 PWIN 的通用编码程序完成,用户不必关心。

本文给出的首尾码的编码文件是从现在流行的 2.13H 汉字系统的主文件

CCCC.COM 中直接获取的。在文件 CCCC.COM 的相对于 100H 的偏移 2BD6H~69C0H 为 6768 个汉字的首尾码和拼音码部分,每个汉字占用 4 个字节,其存放格式为:

第一个字:XXXXXX XXXXX XXXXX

首尾码 2 首尾码 1

第二个字:XXXXXX XXXXX XXXXX

拼音码 3 拼音码 2 拼音码 1

其中:X 为拼音的高频字的标志,X=0 或 1。

由上面的格式可看出,首尾码和拼音码各占 2 个字节,每个码占用 5 位,与 64 (大写)或 96 (小写)相加后就是该码的 ASCII 码值,如:汉字“啊”的编码为 146H,6F61H,其格式说明为:

146H	6F61H
000000 01010 01010 0 11011 11011 00001	
10(J) 6(F)	27(D) 27(D) 1(A)

因此,汉字“啊”的首尾码为 FJ,拼音码为 A[[,该汉字不是高频字(X=0)。

在 2.13H 汉字系统中,首尾码的实际输入方法为:

首尾码 1+ 首尾码 2+ 拼音码 1

如汉字“啊”的实际首尾码为 FJA,这样输入的首尾码基本上无重码,加快了输入汉字的速度。

因此,在形成首尾码的编码文件时,获取到一个汉字的首尾码后应再加上该码的第一个拼音码。

程序 SWM.CPP 完成上述方法,在 Borland C++ 下将该程序编译成可执行文件 SWM.EXE。执行程序 SWM 后,在 PWIN 所在的子目录 D:\PWIN 中形成 SWM.TXT 的首尾码的编码文件。在 PWIN 中运行通用编码编译器 CON-

VMB.EXE,将 SWM.TXT 编译成 PWIN 可辨识的首尾码输入文件 SWM.IME。在 PWIN 的主组中选择控制面板图标后,进入控制面板配置,然后选择输入法的图标,安装通用编码输入法,再选用首尾码,则将首尾码安装到 PWIN 中,以后用户只需用 Ctrl+Shift 或鼠标选择该输入法。

下面对 SWM.CPP 说明两点:

1. Windows 的编码文件中的编码应使用小写字母,在程序中是用 96 与 5 位编码相加而成的;

2. 程序 SWM.CPP 中使用了一个转移语句 GOTO LL,是为了跳过汉字库的第 55 区中最后 5 个未定义的汉字,它们的内码分别为 D7FAH~D7FEH。

以上是 PWIN 3.1 添加扩充输入法的方法。从理论上讲,为一通用方法,用户仅需从其他成熟的汉字系统中获取到某种输入法的编码文件,就能快速地将该输入法直接挂接到 PWIN 中。例如:

1. 五笔字型输入法 (WBX) 的编码文件可从联想汉字系统中用该系统提供的编码还原方法 (LX - DOIT/DECODE) 得到;

2. 电报码输入法 (TELE) 的编码文件可从 Super CCDOS 5.0 或 6.0F 的 TELE.COM 文件中(从该文件的绝对位移 08H 开始)得到;等等,用户不妨试一试。

程序 SWM.CPP 的清单如下:

```
#include <stdio.h>
#include <fcntl.h>
#include <io.h>
#include <process.h>
#include <ctype.h>

int main (void)
{FILE *fp, *fp2;
 int handle,handle2,curpos,
 hc=0x0a0d;
 system ("cls");
```

加快键盘的响应速度

□ 顾 强

某些优秀软件中,当操作者始终压住光标键向一个方向移动光标时,屏幕上光标的移动速度会明显地加快。这就涉及到提高键盘的响应速度的问题。

我们知道,计算机有许多端口,每一个端口都有一个固定的端号。当我们要访问端口时必须用其编号,即端口号。AT或兼容机的编号为0x60,是专门用来调整键盘速度的。我们只要向这个端口发一个特定的值,便会改变键盘的响应速度。如果机器的档次在AT以上则每向0x60端口写一个值后则应得到一个回应值0xfa。以下是用C语言编写的程序,这个程序在Turbo C 2.0集成环境下编译通过,并在Compaq 3/25S机上验证。运行时应给一个参数0-15,0为慢速,15为快速。

源程序清单如下:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>

int count,status,speed_level;
char speed_number[13]; /* 命令行速度代码 */

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

```
if(argc>1)
{
strcpy(speed_number,argv[1];
count = 0;
speed_level = atoi(speed_number);
speed_level = 15 - speed_level;
outportb(0X60,0XF3);
do {
status = inportb(0X60);
count++;
} while ((count < 1000) && (status != 0XFA));
if (count < 1000)
outportb(0X60,speed_level);
else
printf("\nThe computer can't use the program! \n");
}
else
{
printf("\nUsage: speed_key [speed_level]\n");
printf("speed_level:");
printf("The speed of the keyboard,
it is between (0-15)\n");
exit(1);
}
exit(0);
}
```

```

=====

printf("\n");
printf("MKCODE 1.0V. Wrtten By
CGJ. \N");
printf("Processing,Please wait ..... \n");
if ((fp = fopen("c:\\213\\cccc.com","r+b"))==NULL)
{
printf("In C:\\213,there exists file
cccc.com\n");
exit(-1);
}
if ((fp2 = fopen("d:\\pwin\\swm.txt","w+b"))==NULL)
{
printf("Can't create swm.txt!!! \n");
exit(-2);
}
handle2=fileno(fp2);
char str1[]="Description";
write(handle2,str1,13);
write(handle2,&hc,2);
char str2[]="Name=首尾码";
write(handle2,str2,11);
write(handle2,&hc,2);
char str3[]="MaxCodes=4";

write(handle2,&str3,10);
write(handle2,&hc,2);
char str4[]="UsedCodes='abcdefghi-
jklmnopqrstuvwxyz'";
write(handle2,&str4,38);
write(handle2,&hc,2);
char str5[]="WildChar=?";
write(handle2,&str5,10);
write(handle2,&hc,2);
char str6[]="Sort=1";
write(handle2,&str6,6);
write(handle2,&hc,2);
char str7[]="Text";
write(handle2,&str7,6);
write(handle2,&hc,2);
unsigned char *buffer;
unsigned char swm1,swm2,pym1;
fseek(fp,0x2ad6,SEEK_SET);
handle=fileno(fp);
for (int i=0xb0;i<0xf8;i++)
{ for (int j=0xa1;j<0xff;j++)
{ read(handle,buffer,4);
swm1=( *buffer & 0x1f) | 96;
//64:转化成大写字符

//96:转化成小写字母
swm2=(( *buffer>>5) & 0x07) |
* (buffer+1) <<3 | 96;
pym1= * (buffer+2) & 0x1f | 96;
if (pym1=='{' || pym1==',' || ')')
pym1=' ';
if (j<0xff && j>0xf9 && i==0xd7)
goto LL;
write(handle2,&i,1);
write(handle2,&j,1);
write(handle2,&swm1,1);
write(handle2,&swm2,1);
write(handle2,&pym1,1);
write(handle2,&hc,2);
LL:
}
}
write(handle2,&hc,2);
fclose(fp);
fclose(fp2);
return 0;
}
=====
```

编写 TSR 程序的一些新技术

□ 张 鹰

用未公开的 DOS 技术来编写内存驻留程序,对编程者来讲,并不陌生。因为编写此类程序需要用到 DOS 的终止并驻留功能(INT 21H 功能 31H)或者老的 TSR 中断 INT 27H,所以通常称之为 TSR。一般来讲,编写 TSR 程序要做到以下几点:挂接所需的中断向量;监视 DOS 忙和 DOS 关键性错误标志;保存及恢复 DOS 的 PSP、DTA、扩展错误信息;挂接空闲中断 INT 28H;监视 BIOS INT 13H 调用以决定是否进行 TSR 操作。

这一切都是由于 MS-DOS 的不可重入性。也就是说,当 INT 21H 正在进行某种操作时,其他的 INT 21H 请求将无法插入进来。因为当 MS-DOS 通过 INT 21H 被调用时,它将被切换到以下三个内部堆栈中的一个:I/O 堆栈、磁盘堆栈和辅助堆栈。由于存在着这种堆栈切换操作,如果前台程序正在 INT 21H 内部执行,出现 TSR 调用 MS-DOS,则 MS-DOS 会把 TSR 的数据调入相应的堆栈中,原前台程序的数据会被覆盖掉。

那么,如果保留这一部分数据,待 TSR 结束再恢复这一部分数据就可以解决这一问题。利用 DOS 未公开的 INT 21H 功能 5D06H,就可以访问 DOS 的 SDA(可交换数据区)。

具体格式如下:

输入:AX 5D06H

返回:

CF 置位表示出错,AX 出错码。

CF 清除表示成功

DS:SI 指向 SDA

CX 处在 DOS 内部时必须交换的数据的字节数

DX 任何时候都必须交换的数据的字节数

SDA 是一块包含当前 MS-DOS 内

部数据的空间,包括当前的 PSP 段值以及前面提及的三个堆栈,本质上就是 DOS 的数据段。

使用 SDA 的主要优点是几乎可以在任何时候激活 TSR,这对于轮盘式的多任务切换是极有益的,因为它可以使任务切换的响应速度非常之快。有趣的是,Microsoft Window 3.0 下的多任务管理程序就是利用这一原理来完成的。

这一方法的缺点是需要申请空间来保存这一部分数据,在 DOS 3.1~DOS 3.3 下大小为 73CH 字节,在 DOS 4.X 下为 78CH 字节,以后的 DOS 版本可能会扩大这一区域,编程时可留出更大一部分空间以适应兼容需要(实际上用其他方法编程需要不少指令,因此并没有占用这么大的空间)。如果程序确实占了太大的空间而使其他程序无法运行,可以用下面讨论的 TSR 退出驻留的方法来释放内存。

要使 TSR 退出驻留,就要释放 TSR 所占用的内存,恢复所挂接的中断向量,并恢复 TSR 所改变的一切,关闭 TSR 所打开而未关闭的文件。但当有其他的 TSR 驻留并挂接了相同的向量,退出驻留最好采用先进后出法依次退出,否则,后进的 TSR 的功能将无法实现。

同时使用多个 TSR 程序,一个常见的问题是缺少管理程序的标准。早在 1986 年,一些 TSR 开发者组织了一个团体以建立这样的标准。许多著名的 TSR 作者参加了这个团体,他们后来建立了一个标准:TesSeRact 标准。

TesSeRact 标准确定了一组链入 DOS 多路中断(INT 2FH)的功能,因为 DOS 使用 INT 2FH 同自己的 TSR (诸如 ASSIGN、PRINT 和 SHARE)进行通信。所以 TesSeRact 的开发者认为用相同的接口为自己开发的 TSR 程序

服务是可行的,其功能 C0H——FFH 已载入文档供用户使用。为了保持一致,所有功能的 00H 子功能(AL 为子功能号)被正式保留,用于读取安装状态。返回时,AL 为 00H 表明该功能未安装且可以安装;01H 表明未安装且不可安装;FFH 表明已安装。根据这一标准,可以避免重复安装同一驻留程序。在示范程序中,我们使用 C800H 来检查 TSR 的安装与否,C801H 来决定 TSR 的退出。

释放 TSR 所占用的内存,可以通过查询 MCB(内存控制块)来实现。内存的每一块以一个 MCB 开始,每个 MCB 长一节(16 个字节),而且起始于能被 16 整除的地址,内存块本身的大小也是节的整数倍。这种节边界对齐使得能够用 16 位段地址来确定一个内存块而不必用完整的 20 位地址。

MCB 内部结构如下:

偏移	大小	描述
00H	字节	块类型,5AH 表示最后一块,4DH 表示其他块,其他值出错。
01H	字	PSP 段值,表明所控制内存块的所有者,0000H 表示自由块
03H	字	以节计的内存块的大小
05H	3 字节	未用
DOS 2.X~3.X		
08H	8 字节	未用
DOS 4.X 以上		
08H	8 字节	如为 PSP 内存块,则为 ASCII 字符的程序名,否则为无用串

第一个 MCB 的段值可以用未公开的 DOS 调用 52H 来获得,在返回值的 ES:[BX-2]处。我们搜索 MCB 链,当搜

索到块类型为 5AH 时,结束;搜索到 4DH 时继续;其他值出错。当块类型为 4DH 时,比较其 PSP 段是否与 TSR 的相同。若相同,则说明找到,恢复需恢复的数据,然后将内存块改为自由块(注意,DOS 中可能有好几个内存块有相同的 PSP 值,不应找到一个就结束,应搜索整个 MCB 链)。不要通过文件名来搜索,因为用户可能会改变文件名。

下面我们通过一个实例进行说明:

源程序如下:

```

    . 286
    code segment byte public
    assume cs:code,ss:code,ds:code,es:code
begin:
    jmp start
    newstack db 50h dup (0)
    new _sp dw ?           ;自己的堆栈区
    old _ss dw ?
    old _sp dw ?           ;旧 SS,SP 值保存处
    disk_busy db 0         ;磁盘忙标志
    dac1 db 768 dup (0)
    dac2 db 768 dup (0)    ;保存 DAC 寄存器数值
    dosswap db 2048 dup (0) ;DOS SDA 保存处
    oldint9 dd 9
    oldint13 dd 9
    psp dw ?
newint13:
    mov byte ptr cs,disk_busy,1; 新 INT 13H 服务程序
    pushf
    call cs,oldint13
    mov byte ptr cs,disk_busy,0
    iret
newint9:
    cmp byte ptr cs,disk_busy,1;新 INT9H 服务程序
    jnz no_busy             ;无磁盘操作,转
    pushf
    call cs,oldint9
    iret
no_busy:
    mov cs,old_ss,ss
    mov cs,old_sp,sp
    mov sp,seg new_sp
    mov ss,sp
    mov sp,offset new_sp    ;建立新堆栈
    pusha
    push ds
    push es
    pushf
    call cs,oldint9
    xor ax,ax
    mov es,ax
    mov al,es:[417h]
    mov ah,al
    and al,00000011b
    cmp al,00000011b
    je set_vga              ;按相应键,转关闭 VGA
    and ah,00001100b
    cmp ah,00001100b
    jz rest                 ;转打开 VGA
niret:
    pop es
    pop ds
    popa
    mov ss,cs:old_ss
    mov sp,cs:old_sp
    iret
set_vga:
    mov ax,5d06h

```

程序 PROTVGA.ASM 用汇编语言写成,通过汇编和链接形成 .EXE 文件。键入 PROTVGA 则驻留内存,键入 PROTVGA ? 则卸下 TSR,其他命令行参数无效。程序的主要功能是按下左 Shift+右 Shift 键关闭 VGA 显示器,按下左 Alt+左 Ctrl 时恢复屏幕。程序主要通过修改 VGA 的 DAC 寄存器来进行,读者可以阅读相关的著作。

最后说明一点,TSR 退出驻留时要

注意关闭其打开的文件,否则可能引起系统打开文件而不能操作的错误。一般好的 TSR 程序编制时应考虑这一问题,所以卸下 TSR 时不用再考虑。若 TSR 在装入时用到 DOS 重定向,例如 C>PROTVGA,则可能会留下一个多余的“孤儿”句柄。本文所附程序 PROTVGA.ASM,在 DOS 3.3 下采用 MASM 版本 5.1 编译通过。

```

    int 21h                ;取 SDA
    jc error1
    add cx,dx
    push cx
    push si
    push ds
    push cs
    pop es
    lea di,dosswap
movel:
    mov al,ds:[si]
    mov es:[di],al
    inc si
    inc di
    loop movel
    mov ah,50h
    mov bx,cs:psp
    int 21h                ;置当前 PSP
    mov ah,1ah
    mov bx,cs:psp
    mov ds,bx
    mov dx,80h
    int 21h                ;置当前 DTA
    push cs
    pop ds
    mov ax,1017h
    mov cx,256
    xor bx,bx
    lea dx,dac1
    int 10h                ;保存 DAC 数据
    mov ax,1012h
    mov cx,256
    xor bx,bx
    mov dx,offset dac2
    int 10h                ;关闭 VGA
    pop ds
    pop si
    pop cx
    lea di,dosswap
restorel:
    mov al,es:[di]
    mov ds:[si],al
    inc si
    inc di
    loop restorel         ;恢复 SDA
errorl:
    jmp niret
rest:
    mov ax,5d06h
    int 21h
    jc error2
    add cx,dx
    push cx
    push si
    push ds
    push cs
    pop es
    lea di,dosswap
move2:
    mov al,ds:[si]

```

```

mov es,[di],al
inc di
inc si
loop move2
mov ah,50h
mov bx,cs:psp
int 21h
mov ah,1ah
mov bx,cs:psp
mov ds,bx
mov dx,80h
int 21h
push cs
pop ds
mov ax,1012h
mov cx,256
xor bx,bx
mov dx,offset dac1
int 10h          ;重新打开 VGA
pop ds
pop si
pop cx
lea di,dosswap
restore2:
mov al,es:[di]
mov ds:[si],al
inc di
inc si
loop restore2
error2:
jmp niret
newint2f:
cmp ax,0c800h    ;新的 INT 2FH 服务程序
jz j1
cmp ax,0c801h
jz j2
jmp dword ptr cs:[oldint2f]
j1:
mov al,0ffh
iret
j2:
mov ah,52h
int 21h
mov dx,0
mov ax,es:[bx-2]  ;取第一块 MCB 段值
recmp:
mov es,ax
xor si,si
mov al,es:[si]
cmp al,'Z'        ;是最后块?
jz is_end
cmp al,'M'        ;是有效块?
jnz is_error
mov ax,es:[si+1]
cmp ax,cs:psp     ;PSP 值相同?
jz is_found
recmpl:
mov ax,es:[si+3]
mov bx,es
add ax,bx
inc ax            ;修改 PSP 段值
jmp short recmp
is_end:
cmp cx,0          ;是否有相应块找到?
jz no_found
mov ax,2
jmp short end_dein ;有,带相应参数返回
no_found:
mov ax,1
jmp short end_dein ;无,带相应参数返回
is_error:
mov ax,1
jmp short end_dein
is_found:

```

```

cmp cx,1          ;数据是否已经恢复?
jz changed
mov ax,2509h
lea di,oldint9
mov ds,cs:[di+2]
mov dx,cs:[di]
int 21h
mov ax,252fh
lea di,oldint2f
mov ds,cs:[di+2]
mov dx,cs:[di]
int 21h
mov ax,2513h
lea di,oldint13
mov ds,cs:[di+2]
mov dx,cs:[di]
int 21h          ;恢复所修改的中断向量
changed:
mov ax,0
mov es:[si+1],ax ;改为自由块
mov cx,1          ;置相应标志
jmp recmpl
end_dein:
iret
oldint2f dd 0
tsr_end db '$' ;驻留程序结束
start:
mov ax,0c800h
int 2fh
cmp al,0ffh      ;是否已安装?
jnz install
jmp function
install:
push es
pop word ptr cs:psp ;保存 PSP
push cs
pop es
push cs
pop ds
mov ax,3509h
int 21h
mov si,offset oldint9
mov ds:[si],bx
mov ds:[si+2],es
mov ax,2509h
mov dx,offset newint9
int 21h
mov ax,352fh
int 21h
mov si,offset oldint2f
mov ds:[si],bx
mov ds:[si+2],es
lea dx,newint2f
mov ax,252fh
int 21h
mov ax,3513h
int 21h
lea si,oldint13
mov ds:[si],bx
mov ds:[si+2],es
lea dx,newint13
mov ax,2513h
int 21h          ;保存并修改中断向量
lea dx,msg1
mov ah,9
int 21h
mov dx,offset tsr_end
add dx,110h
mov cl,4
shr dx,cl
mov ax,3100h
int 21h          ;驻留并结束

```

BMP 图象文件的快速显示

□ 顾世强

Windows 在我国已越来越深入人心。随着机器档次的提高,Windows 的普及,BMP 图象文件也就用得越来越多了。使用 BMP 图象文件不但要在 Windows 环境下,同时也要求能在 DOS 中使用,这样 Windows 丰富的图形资源可为 DOS 所用。这无疑对我国目前广大 DOS 用户来说是大有益处的事情。以前我们曾见到过一些软件使用 BMP (或其他格式) 格式的文件进行图象处理,但是图象的处理速度很慢。本文分析了 BMP 文件的格式、VGA 显示器的特点,提出了显示 BMP 图象及特殊效果处理的方法,并给出了源程序。尽管是以

BMP 图形文件为背景,但是这种方法适用于对所有图象的处理。

一、BMP 图象文件格式

BMP 的全称是 Bit Map 即位图,它主要用于 Windows 环境中,其文件结构可分为三大部分(见下表):第一部分是位图文件头 (BITMAPFILEHEADER) 结构;第二部分是位图信息结构 (BITMAPINFO);第三部分是位图阵列结构,即实际的图象数据。存取、变化 BMP 图象文件的关键在于读取或设置文件头和位图信息结构的有关信息,如

宽度、高度、起始位置、存储格式、调色板等。

二、用 EGA 多图形页快速显示 BMP

EGA 提供了多个图形页。在显示 BMP 文件时,完全将 BMP 文件送后台图形页显示。当转换完成后再将后台 BMP 图形页变成前台图形页,这样即可达到 BMP 文件的快速显示。有关函数为:SetActivePage(int Pagenum);SetVisualPage(int Pagenum),将这两个函数加入到程序的合适位置即可达到目的。

BMP 图象文件格式

地址	长度	C 语言结构	内 容	说明	
0	2B	int bfType	文件类型("BM")	BMP 位图文件头 BITMAPFILEHEADER (共 14 个字节)	
2	4B	long bfSize	文件大小		
6	4B	long dfReserved	保留		
0A	4B	long bffbits	图象数据起始位置		
0E	4B	long biSize	位图信息头长度	BMP 文件信息头 BITMAPINFO HEADER (40 字节)	BMP 信息 BITMAPINFO
12	4B	long biWidth	BMP 宽度		
16	4B	long biHeight	BMP 高度		
1A	2B	int biPlance	目标设备(1)		
1C	2B	int biBitcount	每个像素所占位数		
1E	4B	long biComprcssion	压缩类型		
22	4B	long biSizeimage	图象大小		
26	4B	long biXpels	水平分辨率		
2A	4B	long biYpels	垂直分辨率		
2E	4B	long biClrosed	实际颜色数		
32	4B	long biClimportant	重要颜色数		
36	1B	char R	红色相对亮度	彩色表 BITCOLOR	
37	1B	char G	绿色相对亮度		
38	1B	char B	蓝色相对亮度		
39	1B	char reserved	保留		
...		
3E 26 436		长度不定		实际图象数据	

三、用 VGA 的硬件特性 快速显示 BMP

在 VGA 时序发生器中有一个 8 位寄存器,控制它的第 5 位就可以起到打开和关闭屏幕的作用。利用这一特性可以加速对图象的处理。打开和关闭屏幕

的函数见源程序中的 Close_display() 和 Open_display()。将这两个函数加入到适当的地方就可以达到快速显示的目的。

四、源程序

本程序在 TurboC 2.0 的 Large 模

式下调试通过。源程序中还有几个特殊的函数,如:Write_Scre_color(),Read_Scre_color(),Disppear(),appear() 这些函数可以达到对图象进行特殊显示的目的,如:平滑和消隐。同样将这些函数加入到用户程序的适当位置并合理利用的话也可以实现上述功能。

```
#define MAXBLOCK 15
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <alloc.h>
#include <graphics.h>
#include <dir.h>
#include <dos.h>
#include <io.h>
#include <fcntl.h>
#define along(i) ((i+7)/8*8)
long read_color(int);
long rp[16];
unsigned _stklen=0x4000;
int xsize,ysize,q,w;
int far fget_image(int x1,int y1,int x2,int y2,char *fn);
typedef struct {
    int bfType; /* 文件类型("BM") */
    long bfSize; /* 文件大小 */
    long dfReserved; /* 保留 */
    long bfOffbits; /* 图形位置 */
} BITMAPFILEHAND; /* BMP 文件头 */
typedef struct {
    long biSize;
    long biWidth; /* BMP 宽度 */
    long biHeigth; /* BMP 高度 */
    int biPlance; /* 目标设备(1) */
    int biBitcount; /* 每个象素所占位数(1,4,8,24) */

    long biCompression; /* 压缩方式 */
    long biSizeimage; /* 图象大小 */
    long biXpels; /* 水平分辨率 */
    long biYpels; /* 垂直分辨率 */
    long biClrused; /* 实际颜色数 */
    long biClrimportant; /* 重要颜色数 */
} BITMAPINFOHAND; /* BMP 信息头 */
typedef struct {
    char R;
    char G;
    char B;
    char reserve;
} BITCOLOR; /* BMP 信息头 */
int ColorChange;
int ico_color(unsigned char c)
{int color=0;
 if ((c&1)!=0) {color|=4;
  c&=14; }
 if ((c&4)!=0) {color|=1;
  c&=11; }
 color+=c;
 if (ColorChange)
  if (color==7) color=8;
  else if (color==8) color=7;
 return color;
}
/* 绘出 16 色的标准 Windows 位图文件,
返回值: 0:成功
1:文件未找到
2:文件的类型不对
3:文件的颜色数不对
```

```
4:每个象素所占位数不为 4
5:图文件已压缩
6:内存不够 */
int far draw_bmp16(int x,int y,char *fname)
{
    unsigned char c;
    char huge *buf;
    FILE *f;
    int i,j,h,w,k;
    long aa,bb;
    BITMAPFILEHAND hand;
    BITMAPINFOHAND info;
    BITCOLOR color[16];
    if ((f=fopen(fname,"rb"))==NULL) return 1;
    fread(&hand,1,sizeof(hand),f);
    fread(&info,1,sizeof(info),f);
    fread(&color,1,sizeof(color),f);
    if (color[7].R<color[8].R &&
        color[7].G<color[8].G &&
        color[7].B<color[8].B)
        colorChange=1;
    else ColorChange=0;
    if (hand.bfType!=0x4d42) return 2;
    if (info.biBitcount!=4) return 4;
    if (info.biCompression!=0) return 5;
    if (info.biSizeimage==0)
        info.biSizeimage=along(info.biWidth)*info.biHeigth*
        info.biBitcount/8;
    buf=(char far *)farmalloc(info.biSizeimage);
    if (buf==NULL) return 6;
    /* 读入位图数据 */
    fseek(f,hand.bfOffbits,SEEK_SET);
    aa=info.biSizeimage;
    if (aa>0xffff) {
        i=aa/0x8000+1;
        aa=0x8000; }
    else i=1;
    bb=(long)buf;
    for (j=0;j<i;j++){
        fread(buf,1,aa,f);
        buf+=aa;
    }
    fclose(f);
    buf=(char huge *)bb;
    ysize=h=info.biHeigth;
    xsize=w=info.biWidth;
    j=along(w)/2;
    y+=(h-1);
    for (k=0;k<h;k++){
        for (i=0;i<w/2;i++){
            c=* (buf+(long)k*j+i);
            putpixel(x+i*2+1,y-k,ico_color(c&0xf));
            putpixel(x+i*2,y-k,ico_color((c>4)));
        }
    }
    farfree(buf);
    return 0;
}
void Close_display()
{
    while((inp(0x3da)&8)==0);
    while((inp(0x3da)&8)==8);
    outp(0x3c4,1);
```

```

    outp(0x3c5,inp(0x3c5)|0x20);
}
void Open_display()
{
    while((inp(0x3da)&8)==0);
    while((inp(0x3da)&8)==8);
    outp(0x3c4,1);
    outp(0x3c5,inp(0x3c5)&0xdf);
}
void Write_Scre_color(int color,char r,char g,char b)
{
    outp(0x3c8,color);
    outp(0x3c9,r);
    outp(0x3c9,g);
    outp(0x3c9,b);
}
long Read_Scre_color(int color)
{
    long r,g,b;
    outp(0x3c7,color);
    r=(long)inp(0x3c9);
    g=(long)inp(0x3c9);
    b=(long)inp(0x3c9);
    return((b<<16)|(g<<8)|r);
}
Disappear()
{
    unsigned int i,j,m;
    long color,plate[16],r,b,g;
    for (i=1;i<=15;i++)
    {
        plate[i]=read_Scre_color(i);
        rp[i]=plate[i];
    }
    j=63;
    while(j--)
    {
        for(i=1;i<=15;i++)
        {
            color=plate[i];r=color&0x0000ff;
            g=(color&0x00ff00)>>8;b=(color&0xff0000)>>16;
            Write_Scre_color(i,(char)(r),(char)(g),(char)(b));
            if(b<0) b=0;if(b>0) b--;
            if(g<0) g=0;if(g>0) g--;
            if(r<0) r=0;if(r>0) r--;
            color=(b<<16)|(g<<8)|r;plate[i]=color;
            for(m=0;m<0x4ff;m++);
        }
    }
}
appear(long *plate)
{
    unsigned int i,j,m;
    long color,color1,rl,bl,gl,plate[16],r,b,g;
    for(i=1;i<=15;i++) plate[i]=Read_Scre_color(i);
    j=0;
    while(j++<63)
    {
        for(i=1;i<=15;i++)
        {
            color=plate[i];r=color&0x0000ff;
            g=(color&0x00ff00)>>8;b=(color&0xff0000)>>16;
            color1=plate[1];rl=color1&0x0000ff;
            gl=(color1&0x00ff00)>>8;bl=(color1&0xff0000)>>16;
            Write_Scre_color(i,(char)(r),(char)(g),(char)(b));
            if(b<bl) b++;
            if(g<gl) g++;
            if(r<rl) r++;
            color=(b<<16)|(g<<8)|r;plate[i]=color;
            for(m=0;m<0x5ff;m++);
        }
    }
}

```

```

}
int main(int argc,char *argv[])
{
    char buffer[4096];
    /* ---如选 EGA 则 gdriver= EGA,gmode=EGAHI--- */
    int gdriver = DETECT,gmode,errorcode,i;
    char str[30];
    struct palettetype pal;
    gettext(1,1,80,25,buffer);
    printf("Showbmp Version 1.0 Written by DALIAN G. S. Q\n");
    getch();
    if ((argv[1][0]!='/'&& argv[1][1]!='?') || argc==1)
    {
        printf("ShowBMP Version 1.0\n");
        printf("1994,06\n");
        printf("\nUsage: SHOWBMP [BMP files]\n");
        printf("SHOWBMP /? (Read This Help Information)\n");
        printf("\nExample: C:\\>SHOWBMP \\WINDOWS\\CARS.BMP\n");
        return 0;
    }
    else
    {
        /* 建立独立图形运行库,如将此函数去掉则应有 EGAVGA.BGI 文件 */
        registerbgidriver (EGAVGA_driver);
        initgraph(&gdriver, &gmode, "");
        errorcode = graphresult();
        if (errorcode !=grOk) /* an error occurred */
        {
            printf("Graphics error:%s\n", grapherrormsg(errorcode));
            printf("Press any key ...");
            getch();
            exit(1); /* terminate with an error code */
        }
        getpalette(&pal);
        setrgbpalette(pal.colors[7],45,45,45);
        setrgbpalette(pal.colors[9],0,0,255);
        setrgbpalette(pal.colors[1],0,0,43);
        setrgbpalette(pal.colors[8],34,34,34);
        setfillstyle(1,15);
        /* ---bar(0,0,639,479);--- */
        strcpy(str,argv[1]);
        if(strchr(str,'.')==NULL)
            strcat(str,".BMP");
        Close_display();
        i=draw_bmp16(0,0,str);
        Open_display();
        getch();
        for(q=0;q<3;q++)
        {
            Disappear();
            for(w=0;w<=0xffff;w++);appear(rp);
        }
        if (i==0) getch();
        closegraph();
        switch (i)
        {
            case 0 : printf("O. K.");break;
            case 1 : printf("The BMP file not found!"); break;
            case 2 : printf("The file is not BMP file!"); break;
            case 3 : printf("The BMP file's colors is not 16!"); break;
            case 5 : printf("The BMP file is compression!"); break;
            case 6 : printf("Not enough memory!"); break;
        }
    }
    return 0;
}

```

VGA 图形的灰度打印

□ 闫宗孝

VGA 高分辨率彩色图形显示器与早期的 CGA 等显示器有很大的不同,它可以显示丰富的色彩,与其相比,单色打印机所获得的拷贝则显得没有层次感。以下介绍一种能呈现灰度的打印程序,打印效果较好。

VGA 显示存储区分为 4 个位面,显示器上的一点和显示存储区 4 个位面上的点相对应,且 4 个位面占用相同的起始地址 A000:0000。可以通过图形控制寄存器的不同设置,分位面获取显示器上的图形。VGA 图形控制寄存器共有 9 个,通过向 3CEH 端口送索引号,向 3CFH 送数据进行操作。

以下为程序中所需要的设置:

索引号 5 为模式寄存器,读模式设置:5→3CEH;0→3CFH;

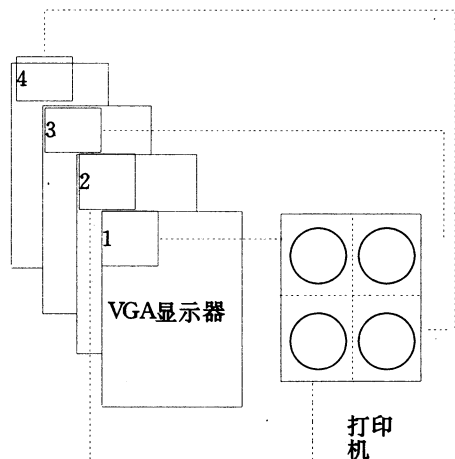
索引号 4 为位面选择读出寄存器,所读位面(0—3)设置:4→3CEH;(0—3)→3CFH;

以下为 1724 打印机的设置:

设置行距:1BH,4AH,12H,0DH,0AH 行距设置为 18,这样图形打印不重叠。

设置图形打印方式(单向):

1BH,47H,05H,00H 打印宽度为 1280。显示器与打印机的对应关系如下图,由于显示器上一点对应打印机上四点,因此可呈现 4 级灰度。



显示器与打印机的对应关系

程序通过修改 INT 5H 中断驻留内存,热键为 Print _ Scrn

运行:C>PS a</; a=0 正常打印;a=1 反向打印。无参数为正常打印。

本程序在 1724(M2024)打印机下编制对于其他打印机需作部分修改。

```
CODE SEGMENT
ORG 100H
ASSUME CS:CODE,DS:CODE,ES:CODE

BEGIN: JMP START
LINEF DB 1BH,4AH,12H,0DH,0AH
VGAGC DB 0DH,0AH,1BH,47H,05H,00H
BUF1 DB 24 DUP(?)
BUF2 DB 24 DUP(?)
FLAG DB 00H
DB 'VGAPS'

NINT5: CLD
PUSH DS
PUSH ES
PUSH SI
PUSH DI
PUSH AX
PUSH BX
PUSH CX
PUSH DX
PUSH CS
POP DS
MOV AX,0A005H
MOV ES,AX
MOV SI,OFFSET LINEF
MOV CX,5
XOR DX,DX

PLINE: XOR AH,AH
LODSB
INT 17H
LOOP PLINE
MOV DX,03CEH
MOV AL,5
OUT DX,AL
MOV DX,03CFH
MOV AL,00H
OUT DX,AL
MOV DX,03CEH
MOV AL,04H
OUT DX,AL
XOR DX,DX
XOR BX,BX
XOR DI,DI
MOV CL,40

LOOPS: PUSH CX
MOV CL,6
MOV SI,OFFSET VGAGC

PRINGC: XOR AH,AH
LODSB
INT 17H
LOOP PRINGC
MOV CL,80

LOOPA: PUSH CX
CALL PLOAD
CALL PPRIN
INC BX
MOV DI,BX
POP CX
LOOP LOOPA
ADD DI,0370H
MOV BX,DI
POP CX
LOOP LOOPS
MOV SI,OFFSET VGAGC
MOV CX,2
XOR DX,DX
```


17

界面随机填充技术探讨

□ 陈捷 陈峻

随着计算机技术在各领域的广泛应用,怎样设计一个新颖的程序界面已成为程序设计者首先应考虑的问题。

在界面设计中,常采用一种由随机的点形成的一幅完整画面的显示方法,这种显示方法使画面显示过程变得更生动、活泼。我们要将一幅画面进行随机显示时,就必须用到随机函数,而 320×200 的画面需要显示 $320 \times 200 = 64000$ 个点,而这么多点随机显示不但造成了时间上的浪费,而且会引起诸如显示时间不一致、画面显示不完整或者占用大量内存等问题。

为了解决上述问题,笔者编写了一

段简单的程序,该程序的功能是在图形方式下显示一幅 400×400 点的画面,画面将随机填充显示黄色的点,直至画面填满为止。

考虑到上述问题,必须将一幅画面的随机填充变为较小画面的随机填充,本程序将画面分为水平、垂直方向等距进行 20 等分,即分为 400 个小块每小块的水平、垂直方向各为 20 个点。程序开始时,动态分配了 400 个字节的填充标志位并初始化为零。紧接着对 400 个点进行随机填充,被填充过的点相应标志位置 1,这样将 400 个点的坐标随机地填入结构数组 array 中。利用结构数组 array 中的 400 个点的坐标,每次写点操作

将随机选择 400 个小块中的一块,再随机填充小块中 400 个点中的一个,由于我们的结构数组 array 中放有 400 个点的随机坐标,可以通过对结构数组 array 的不同顺序操作完成整个画面的显示。对于块坐标用倒序而块中点的坐标用正序选择结构数组 array 中的元素,这样可使显示效果更理想。下述程序中的写点函数体现了上述思想。

本文所附程序在 Borland C++3.1 下调试成功。由于篇幅有限,程序只完成了较简单的显示功能,通过此文的抛砖引玉的作用,希望广大计算机爱好者能共同探讨出更多、更好的方法。以为应用程序界面的开发奠定基础。

程序清单如下:

```
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <graphics.h>
void main(void)
{
    int times=0,i;
    int number;
    int dr,dm,pos;
    struct Array {
        unsigned char x;
        unsigned char y;
    } array[400];
    char *space;
    randomize();
    /* 动态分配 400 个字节 */
    space = (char *)malloc(400 * sizeof(char));
    /* 初始化填充标志位为零 */
    for (i=0;i<400;i++)
        *(space+i)=0;
    while (times<400){
        number = random(400);
```

```
        if (*(space+number)==0) /* 如果该点未填点 */
        {
            array[times].x=number%20; /* 添写数组中 x 的坐标值 */
            array[times++].y=number/20;
            /* 添写数组中 y 的坐标值 */
            *(space+number)=1;
            /* 填充标志位置 1,表示该点已填充 */
        }
    }
    free(space); /* 释放内存 */
    dr=DETECT;
    initgraph(&dr,&dm,"");
    for (pos = 0;pos<400;pos++)
        for (i=0;i<400;i++)
            putpixel(array[(399-i+pos)%400].x*20+array[i].x,
                array[(399-i+pos)%400].y*20+array[i].y,
                YELLOW);
    getch();
    closegraph();
}
```

PUT 文件格式的分析与程序调用

□ 闫 孝

PUT 文件是一种可以被程序调用的图形文件格式,它的格式与 Borland 公司的 Turbo C 中 `getimage()`、`putimage()`; Turbo Pascal 中 `Getimage()`、`Putimage()` 得到的位图格式基本相同,所不同的是,PUT 文件中位图数据从 0—3 位面存放;而 Turbo 系列语言图形数据写屏则相反,以 3—0 位面操作。PUT 文件可以经转换后被程序调用,也可以直接被 Turbo C 和 TurboPascal 程序所调用(有色彩偏差)。同时有些作图工具软件如 A—Four 公司的 IM—AGE72 及最新出版的 CCED5.0 等也支持 PUT 格式文件。本文通过对最为常用的 VGA/EGA 的 PUT 文件格式分析,使用户可以很方便地在程序中调用、制作精美的封面和动画。

PUT 文件由[文件头]+[位图数据]组成,[文件头]包括位图的宽度和高度,它的长度为 4 个字节,前两个字节放位图的宽,后两个字节放位图的高,[位图数据]中逐行存放位图数据,而每行的位图数据则逐位面存放(从 0—3 位面)。

如:某 PUT 文件的文件头为 `x1x2 x3x4 y1y2 y3y4(x1,x2,x3,x4,y1,y2,y3,y4 的长度为 4 位)`,其位图的宽为 $x4 * 16 * 16 + x1 * 16 + x2$;单位为点。将位图以字节存取,则宽度为 $x4 * 2 * 16 + x1 * 2 + \text{inbit}$;单位为字节,则:

如果 $x2=0$ 则 `inbit` 为 0;

如果 $x2 \leq 8$ 则 `inbit` 为 1;

如果 $8 < x2 < 16$ 则 `inbit` 为 2。其位图的高为 $y4 * 16 * 16 + y1 * 16 + y2$;单位为线。

程序一、程序二为演示程序,程序一、二分别用 Turbo C2.0 和 MASM5.0 编制,并在配有 VGA 的 AST386 机器上

调试通过。

程序一源程序如下:

/* 这是一个演示程序。其中 mark.put 文件是台湾智冠公司出品的游戏软件[麻将学园]中的封面画面文件,通过程序的调用在屏幕上显示。并将此图写入文件 mark1.put 中,此文件可以被其它程序调用。也可用某些识别 PUT 文件的作图工具软件,制作软件调用。(如绘图软件 IMAGE72 系列和 CCED 制表软件等)

```

* /
#include <graphics.h>
#include <alloc.h>
#include <stdio.h>
void adjust(signed ** pbuf)
/* 转换子程序,消除色彩偏差 */
{
FILE * putff;
unsigned char * buf1, * lbuf;
signed i, j, k, adr, fno;
int hei, wed;
buf1 = * pbuf;
wed = div(* buf1, 8);
if((( * buf1 - wed * 8) < 8) & (( * buf1 - wed * 8) > 0))
wed = wd + 1;
hei = * (buf1 + 2);
buf1 = buf1 + 4;
lbuf = malloc(wed * 4);
adr = 0;
for(k=1; k<=hei; k++)
{for(i=3; i>=0; i--)
for(j=0; j<wed; j++)
* (lbuf + j + i * wed) = * (buf1 + j + (3-i) * wed + adr);
adr = k * 4 * wed;
for(j=0; j<4 * wed; j++)
* (buf1 + j + adr - 4 * wed) = * (lbuf + j);
}
free(lbuf);
}
main()
{
int graphdriver=DETECT, graphmode;
unsigned * buf;
FILE * putf;
unsigned s, fno;
int width, height;
initgraph(&graphdriver, &graphmode, "");

```

```

putf = fopen("mark.put", "r");
fno = fileno(putf);
s = filelength(fno);
buf = malloc(s);
_read(fno, buf, s);
adjust(&buf);
width = * buf;
* buf = width - 1;
height = * (buf + 1);
* (buf + 1) = height - 1;
putimage(200, 50, buf, OR_PUT);
fclose(putf);
s = imagesize(200, 50, 200 + width, 50 + height);
getimage(200, 50, 200 + width, 50 + height, buf);
putf = fopen("mark1.put", "w");
fno = fileno(putf);
_write(fno, buf, s);
fclose(putf);
free(buf);
getch();
closegraph();
}

```

程序二源程序如下:

;这是一个用汇编语言直接读取 PUT 文件的演示。

;程序中 MARK1.PUT 文件由程序一产生。

```

CODE SEGMENT
ORG 100H
ASSUME CS, CODE, DS, CODE, ES;
CODE
BEGIN: JMP START
LINE      DW 00
COL       DW 00
FILEF     DB 'MARK1.PUT', 0
HANDF     DW 1 DUP(?)
LENGF     DW 1 DUP(?)
DISP      PROC NEAR
PUSH      DS
PUSH      ES
PUSH      SI
PUSH      DI
PUSH      AX
PUSH      BX
PUSH      CX
PUSH      DX
CALL      FILESET
MOV        DX, 3C4H
MOV        AL, 2

```

```

OUT        DX,AL          INC        DX          MOV        AH,3FH
INC        DX          MOV        AL,0          INT        21H
MOV        SI,0          OUT        DX,AL          MOV        AX,[BUFT]
MOV        DI,OFFSET BUFF MOV        AH,3EH          AND        AX,0FFF0H
MOV        CX,LINE          MOV        BX,HANDF          SAR        AX,1
BB1:  PUSH    CX          INT        21H          SAR        AX,1
XOR        AX,AX          ENDN: POP        DX          SAR        AX,1
MOV        AL,01H          POP        CX          MOV        BX,[BUFT]
PUSH        SI          POP        BX          AND        BX,000FH
BEGN:  OUT        DX,AL          POP        AX          CMP        BL,0
PUSH        AX          POP        DI          JE        ADD0
PUSH        DX          POP        SI          CMP        BL,8
MOV        CX,COL          POP        ES          JBE        ADD1
PUSH        SI          POP        DS          ADD2:  ADD        AX,2
T1:   PUSH    CX          RET          JMP        ADD0
PUSH        DS          DISP  ENDP          ADD1:  ADD        AX,1
MOV        AX,0A000H          FILESET PROC NEAR          ADD0:  MOV        COL,AX
MOV        DS,AX          PUSH    AX,          MOV        AX,[BUFT+2]
MOV        AL,ES:[DI]          PUSH    CX          MOV        LINE,AX
MOV        DS:[SI],AL          PUSH    DX          ENDOFI: POP        DX
POP        DS          XOR        AX,AX          POP        CX
INC        SI          XOR        CX,CX          POP        AX
INC        DI          MOV        AH,3DH          RET
POP        CX          MOV    DX,OFFSET FILEF          FILESET  ENDP
LOOP        T1          INT        21H          START:  PUSH    CS
POP        SI          MOV        HANDF,AX          POP        DS
POP        DX          MOV        BX, HANDF          PUSH    CS
POP        AX          MOV        AH,42H          POP        ES
SHL        AL,1          MOV        AL,2          CALL    DISP
AND        AX,000FH          XOR        CX,CX          MOV        AH,00
JNZ        BEGN          MOV        DX,DX          INT        16H
POP        SI          INT        21H          INT        20H
ADD        SI,80          MOV        LENGF,AX          BUFT  DW 2 DUP(?)
POP        CX          MOV        AX,4200H          BUFF  DB 2 DUP(?)
LOOP        BB1          INT        21H          CODE  ENDS
MOV        DX,3CEH          MOV        CX,LENGF          END    BEGIN
MOV        AL,3          MOV        DX,OFFSET BUFT
OUT        DX,AL

```

(上接第12页)

```

msg1 db 07,07,'vgaprot installed new!! Press left <shift>
+Right <shift>',0dh,0ah
db 'to close Vga,Press Left Alt+Crtl to open again.',0dh,
0ah
db "input paramater '?' to deinstall program",0dh,0ah,' $'
function:
xor cx,cx
mov cl,es:[80h]
or cx,cx      ;检查是否有命令行参数?
jnz h_para
jmp displ
h_para:
mov di,81h
rcmpl:
mov al,es:[di]
cmp al,0dh
jz displ
cmp al,' '
jz is_blank
cmp al,'?'      ;检查是否为"?"
jz deinstall
is_blank:
inc di
jmp short rcmpl
deinstall:
mov ax,0c801h
int 2fh      ;卸下 TSR 例程

```

```

add ax,ax
lea di,address
add di,ax
mov dx,cs:[di]
jmp short disp2
disp1:
lea dx,msg2
disp2:
push cs
pop ds
mov ah,9
int 21h      ;显示相应信息
mov ax,4c00h
int 21h
msg2 db 07,07,'Program installed already!',0dh,0ah,' $'
msg3 db 07,07,'Sorry,MCB error! system need reboot!',
0dh,0ah,' $'
msg4 db 07,07,'Sorry,No found install program!',0dh,
0ah,' $'
msg5 db 07,07,'Program deinstalled!',0dh,0ah,07,07,07,'
Warning!! The Int Vector already Recover,ot her TSR maybe dam-
aged!',0dh,0ah,' $'
address dw msg3
dw msg4
dw msg5
code ends
end begin

```


从 SPT 到 BMP

——谈高点阵汉字的显示

□ 肖泰康 顾健农

于有通用的显示字库,一般 16 点阵、24 点阵的汉字显示方法已被广泛地应用。但是在没有合适的矢量汉字库或对矢量字库不熟悉的情况下,如何显示高点阵(如 120 点阵、240 点阵等)的汉字呢?可以想见,在软件封面或显示版权信息的地方出现美观的高点阵汉字将会给软件增色不少。

金山汉字系统提供了多种汉字字体,汉字可大至 480 点阵,并有多种修饰方法,所产生的汉字显示效果极富特色。金山系统的输出可以送入一个文件,即 SPT 格式的图象文件。通过分析 SPT 文件就可以了解 SPT 图象的存储格式,从而提取出有关的汉字信息,这是解决问题的关键。

金山系统的输出图文排版处理后可 JSHZ.C 源程序清单如下:

```
#include <graphics.h>
#include <io.h>
#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <fcntl.h>
#include <alloc.h>
#include <stdlib.h>
#define Highth_Offset 36 /* SPT 文件中图象高度、宽度及图象信息的存储偏移量 */
#define Width_Offset 34
#define Map_Offset 64
#define Down 0x50+0x100
#define Up 0x48+0x100
#define Left 0x4b+0x100
#define Right 0x4d+0x100
#define Enter 0xd
#define BK_Color BLACK /* 新的 SPT 图象的背景和前景色 */
#define HZ_Color YELLOW
#define Getchar(c) if( (c=getch()) == 0 ) c=getch()+0x100
int scr_x=100,scr_y=100;
int rect_x1,rect_y1,rect_x2,rect_y2;
void Cursor(void);
void Get_Rect(void);
void Save_HZ(char *HZ_file);
int Display_Spt(char *spt_file);
```

以存储为不同格式的图象文件,且有压缩与非压缩之分。为简单起见,这里选择非压缩的 SPT 图象文件进行分析。

经过分析我们得知 SPT 文件头占用首部 64 个字节。前 16 字节为 SPT 文件标识。图象像素宽度和高度的存储偏移量为 34、36,各占两个字节。文件头其余字节的值保持不变,可以不管。文件头之后为图象信息。SPT 图象只有两色,所以每字节应包含 8 个点的值,这一点通过下式得到验证:

图象高度×图象宽度/8+文件头长度=文件长度。

至此必要的结构信息已经清楚,利用程序 JSHZ.C 将 SPT 图象在显示器上显示后,选取合适的汉字区域,以 Turbo C 的图象格式存储起来。这样金山系统

所能产生的各种美观的汉字显示效果都能出现在我们用 Turbo C 开发的软件中。

本文所附的程序 HZBMP.C 是用来将前面已经生成的汉字图象叠加到 BMP 图象文件中的。这样在 Windows 的应用程序中也实现了高点阵汉字的显示。用法如下:

1. 用 JSHZ.EXE 获得所需的汉字图象;
 2. 在 Windows 中用 Paintbrush 生成要显示的背景图象,存为 16 色的 BMP 文件。图象宽度要能被 8 整除;
 3. 利用 HZBMP.EXE 将汉字作为前景叠加在 BMP 图象上即可。
- 所附程序用 Turbo C 编写,在普通环境下调试通过。

```
void main(int argc,char *argv[])
{
    int graph_mode=VGAHI,graph_driver=VGA;
    if(argc < 3)
    {
        printf("format error.\n");
        printf("\nUsage:JSHZ filespec1(spt) filespec2(output)\n");
        return;
    }
    initgraph(&graph_driver,&graph_mode,""); /* 初始化图形系统为 VGA16 色高分辨率显示模式 */
    setfillstyle(SOLID_FILL,BK_Color); /* 填充背景 */
    bar(0,0,639,479);
    if( Display_Spt(argv[1]) == 0 )
        Save_HZ(argv[2]);
    closegraph();
}

int Display_Spt(char *spt_filespec)
{
    int spt_handle,spt_width,highth,disp_width;
    int i,j,k;
    unsigned char and = 0x80, *line_buffer;

    if( (spt_handle=_open(spt_filespec,O_RDONLY)) == -1 )
    {
        printf("%s open error!",spt_filespec);
        getch(); return -1;
    }
    lseek(spt_handle,Highth_Offset,SEEK_SET);
    /* 获取 SPT 图象的高度和宽度 */
```

```

_read(spt_handle,&highth,2);
lseek(spt_handle,Width_Offset,SEEK_SET);
_read(spt_handle,&spt_width,2);
if(highth>480) highth=480;
/* 控制显示图象的范围,至多显示一整屏的图象 */
if(spt_width > 640) disp_width = 640;
else disp_width = spt_width;
line_buffer = (char *)malloc(disp_width/8 * sizeof(char));
for(i = 0; i < highth; i++)
{
    lseek(spt_handle,(long)i * (spt_width/8) + Map_Offset,
        SEEK_SET);
    _read(spt_handle,line_buffer,disp_width/8);
/* 读入一行图象信息 */
    for(j=0; j < disp_width/8; j++)
/* 显示图象 */
        for(k=0; k < 8; k++)
            if((line_buffer[j] & (and >> k)) == 0) putpixel(j * 8 + k, i,
                HZ_Color);
    free(line_buffer);
    _close(spt_handle);
    return 0;
}

void Save_HZ(char * HZ_filespec)
{
    char * buffer;
    int hz_handle;
    unsigned long buffer_size;
    if((hz_handle = _creat(HZ_filespec, FA_ARCH)) == -1)
    {
        printf("%s creat error!", HZ_filespec);
        getch();
        return;
    }
    Get_Rect();
    buffer_size = imagesize(rect_x1, rect_y1, rect_x2, rect_y2);
    buffer = (char *)malloc(buffer_size * sizeof(char));
    getimage(rect_x1, rect_y1, rect_x2, rect_y2, buffer);
    lseek(hz_handle, 0, SEEK_SET);
    /* 以 Turbo C 的图象文件格式存储所选取的区域信息 */
    _write(hz_handle, buffer, buffer_size);
    free(buffer);
    _close(hz_handle);
}

void Cursor(void)
{
    line(scr_x - 5, scr_y, scr_x + 5, scr_y);
    line(scr_x, scr_y - 5, scr_x, scr_y + 5);
}

void Get_Rect(void) /* 选取将保存的矩形图象区域 */
{
    int c;
    setwritemode(1);
    Cursor();
    Getchar(c);
    while(1) {
        switch(c) {
            case Up : Cursor(); scr_y -= 5; Cursor(); break;
            case Down : Cursor(); scr_y += 5; Cursor(); break;
            case Left : Cursor(); scr_x -= 5; Cursor(); break;
            case Right : Cursor(); scr_x += 5; Cursor(); break;
            case Enter : rect_x1 = scr_x; /* 选取矩形第一角 */
                        rect_y1 = scr_y;
                        Getchar(c);
                        while(1)
                        {
                            line(rect_x1, rect_y1, rect_x1, scr_y);
                            line(scr_x, rect_y1, scr_x, scr_y);

```

```

                            line(rect_x1, rect_y1, scr_x, rect_y1);
                            line(rect_x1, scr_y, scr_x, scr_y);
                            switch(c)
                            {
                                case Up : Cursor(); scr_y -= 5; Cursor(); break;
                                case Down : Cursor(); scr_y += 5; Cursor();
                                break;
                                case Left : Cursor(); scr_x -= 5; Cursor(); break;
                                case Right : Cursor(); scr_x += 5; Cursor(); break;
                                case Enter : rect_x2 = max(rect_x1, scr_x);
                                            /* 选取矩形第二角 */
                                            rect_y2 = max(rect_y1, scr_y);
                                            rect_x1 = min(rect_x1, scr_x);
                                            rect_y1 = min(rect_y1, scr_y);
                                            Cursor();
                                            return;
                            }
                            line(rect_x1, rect_y1, rect_x1, scr_y);
                            line(scr_x, rect_y1, scr_x, scr_y);
                            line(rect_x1, rect_y1, scr_x, rect_y1);
                            line(rect_x1, scr_y, scr_x, scr_y);
                            Getchar(c);
                        }
                    }
                }
            }
        }
    }
    Getchar(c);
}

```

HZBMP.C 源程序如下:

```

#include <graphics.h>
#include <conio.h>
#include <io.h>
#include <dos.h>
#include <stdio.h>
#include <fcntl.h>
#include <alloc.h>

#define BMP_HeadLength_Offset 10
/* BMP 文件中图象高度、宽度及文件头长度的存储偏移量 */
#define Highth_Offset 22
#define Width_Offset 18
#define Down 0x50 + 0x100
#define Up 0x48 + 0x100
#define Left 0x4b + 0x100
#define Right 0x4d + 0x100
#define Enter 0xd
#define Esc 0x1b
#define Getchar(c) if((c = getch()) == 0) c = getch() + 0x100
int scr_x = 100, scr_y = 100;
int handle_BMP;
int bmp_width, bmp_highth, bmp_head_length;
char * HZ_buffer, * BMP_head_buffer;
int Read_HZ(char * HZ_filespec);
int Read_BMP_head(char * BMP_filespec);
void Display_BMP(void);
void Put_HZ(void);
void Saveimage(char * new_BMP_filespec);
void Cursor(void);
void main(int argc, char * argv[])
{
    int graph_mode = VGAHI, graph_driver = VGA;
    if(argc != 4)
    {
        printf("format error. \n");
        printf("\nUsage: HZBMP filespec1(bmp) filespec2(hzmap) file
            spec3(output)\n");
        return;
    }
    if(Read_HZ(argv[2]) != 0) return;
    if(Read_BMP_head(argv[1]) != 0)

```

```

{
    free(HZ_buffer);
    return;
}

initgraph(&graph_driver,&graph_mode,"");
/* 初始化图形系统为 VGA16 色高分辨率显示模式 */
Display_BMP();
Put_HZ();
Saveimage(argv[3]);
free(HZ_buffer);
free(BMP_head_buffer);
_close(handle_BMP);
closegraph();
}

int Read_HZ(char *HZ_filespec)
{
    int handle_hz;
    long size;

    if((handle_hz = _open(HZ_filespec,O_RDONLY)) == -1)
    {
        printf("%s open error.",HZ_filespec);
        getch();
        return -1;
    }

    size = filelength(handle_hz);
    HZ_buffer = (char *)malloc(size * sizeof(char));
    lseek(handle_hz,0,SEEK_SET);
    _read(handle_hz,HZ_buffer,size);
    /* 读取以 Turbo C 图象文件格式存储的汉字 */
    _close(handle_hz);
    return 0;
}

int Read_BMP_head(char *BMP_filespec)
{
    if((handle_BMP=_open(BMP_filespec,O_RDONLY)) == -1)
    {
        printf("%s open error.",BMP_filespec);
        getch();
        return -1;
    }

    lseek(handle_BMP,Width_Offset,SEEK_SET);
    _read(handle_BMP,&bmp_width,2);
    lseek(handle_BMP,Highth_Offset,SEEK_SET);
    _read(handle_BMP,&bmp_highth,2);
    lseek(handle_BMP,BMP_HeadLength_Offset,SEEK_SET);
    _read(handle_BMP,&bmp_head_length,1);

    BMP_head_buffer = (char *)malloc(bmp_head_length *
        sizeof(char));
    lseek(handle_BMP,0,SEEK_SET);
    _read(handle_BMP,BMP_head_buffer,bmp_head_length);
    /* 读入 BMP 文件头 */
    return 0;
}

void Display_BMP(void)
{
    int i,j;
    unsigned char color;
    lseek(handle_BMP,bmp_head_length,SEEK_SET);
    for(i=bmp_highth-1;i>=0;i--)
        for(j=0;j<bmp_width;j+=2)
        {
            _read(handle_BMP,&color,1);
            /* 以缺省调色板的设置显示 BMP 图象 */
            putpixel(j,i,(color&0xf0)>>4);

```

```

            putpixel(j+1,i,color&0xf0);
        }
    }

void Cursor(void)
{
    line(scr_x-5,scr_y,scr_x+5,scr_y);
    line(scr_x,scr_y-5,scr_x,scr_y+5);
}

void Put_HZ(void)
{
    int c;
    putimage(scr_x,scr_y,HZ_buffer,XOR_PUT);
    /* 显示汉字信息 */
    setwritemode(1);
    Cursor();
    Getchar(c);
    while(1)
    {
        putimage(scr_x,scr_y,HZ_buffer,XOR_PUT);
        /* 清除汉字,恢复背景 */
        switch(c)
        {
            case Right:Cursor(); if((scr_x+=5)>639)
                scr_x=0; Cursor();break;
            /* 调整汉字显示位置 */
            case Left:Cursor(); if((scr_x-=5)<0)
                scr_x=639; Cursor();break;
            case Down:Cursor(); if((scr_y+=5)>479)
                scr_y=0; Cursor();break;
            case Up:Cursor(); if((scr_y-=5)<0)
                scr_y=479; Cursor();break;
            case Enter:
            case Esc:cleardevice(); /* 确认汉字显示位置 */
                putimage(scr_x,scr_y,HZ_buffer,COPY_PUT);
                return;
        }

        putimage(scr_x,scr_y,HZ_buffer,XOR_PUT);
        /* 在新的位置显示汉字 */
        Getchar(c);
    }
}

void Saveimage(char *new_BMP_filespec)
    /* 保存加入汉字信息的 BMP 文件 */
{
    int handle_output,i,j;
    char *buffer_output,*temp;
    unsigned char bmp_color,hz_color;
    if((handle_output = _creat(new_BMP_filespec,FA_
        ARCH)) == -1)
    {
        printf("%s creat error.",new_BMP_filespec);
        getch();
        return;
    }

    lseek(handle_output,0,SEEK_SET);
    _write(handle_output,BMP_head_buffer,bmp_head_
        length); /* 保存原 BMP 文件的文件头 */
    buffer_output = (char *) malloc (bmp_width/2 * sizeof
        (char));
    lseek(handle_BMP,bmp_head_length,SEEK_SET);
    for(i = bmp_highth-1;i>=0;i--)
    {
        temp = buffer_output;
        for(j = 0;j<bmp_width;j+=2)
        {
            _read(handle_BMP,&bmp_color,1);

```

如何在文本模式下显示汉字

□ 张更路

在 西文方式(即文本模式)下显示汉字,主要是基于显示器硬件的发展。英文字母实际上也是由点构成,但MDA和CGA显示器其字符发生器均固化在ROM内,即便现有的全字符汉字系统也无法支持此类显示器,而后的EGA/VGA或更高级的显示器是将字符发生器置于DRAM的页面2,因而非英

文也可以以软件手段,通过字模的置换,达到在文本方式下直接显示的目的。

汉字采用这种显示方法由于避免了图形工作方式及象素操作,无论对CPU还是显示器,负载都大大减轻,因而处理速度快、兼容性强。但要真正完美地实现字符模式下汉字的显示,还要结合动态装入字模、智能西文制表符识别等技术。

本文所附程序用Turbo C2.0编写编译成·EXE文件后;在配有VGA的386兼容机上即可正确运行(程序中printf语句中的字符串是打印机打印出的乱码,实际为扩充ASCII码的192-217),程序所用的汉字库为天汇汉字系统的显示字库。本程序稍加修改,即可实现字符模式下的汉字放大显示。

程序清单如下:

```
#include <dos.h>
#include <stdio.h>
read16(unsigned char * hzs);
unsigned char dot[80][32]; /* 要显示的汉字点阵信息 */
unsigned char far * p;
union REGS r;
void main()
{ /* ----- */
    unsigned myES, myBP;
    char * line = "北京中国科大计算机部张更路";
    read16(line); /* 从字库中读出汉字点阵数据 */
    p = dot[0]; /* 取字模点阵的首地址 */
    myES = _ES; /* 保护 ES:BP */
    myBP = _BP;
    /* ----- */
    _BP = FP - OFF(p); /* 将字模点阵的首地址给 ES:BP */
    _ES = FP - SEG(p);
    _AX = 0x1100; /* 装入字模 */
    _BH = 0x10; /* 给出字符高度 VGA:16 EGA:14 */
    _BL = 1; /* 装入字符集 1 */
    _CX = 0x50; /* 显示字符个数 */
    _DX = 0xc0; /* 起始字符值 */
    geninterrupt(0x10);

    r.x.ax = 0x1103; /* 选择活动字符集 */
    r.h.bl = 1; /* 使用字符集 1 */
    int86(0x10, &r, &r);
    /* ----- */
    _ES = myES; /* 恢复 ES:BP */
    _BP = myBP;
    gotoxy(25, 11); /* 在指定位置显示定义的字符串 */
    printf("懒旅呐魄壬仕掏蝎醒矣裁肿到\n");
```

```
getch();
r.x.ax = 3; /* 恢复缺省设置 */
int86(0x10, &r, &r);
exit(0);
/* ----- */
}
read16(unsigned char * hzs)
{
    FILE * fp;
    int i, n;
    long offset;
    if((fp = fopen("clib16", "rb")) == 0)
        printf("HZ dot file open failed\n");

    for(n = 0; n < 79; n++)
    { if(* hzs >= 176) offset = (long)((* hzs - 161 - 3)
        * 94 + (* (hzs + 1) + 121)) * 32;
      else offset = (long)((* hzs - 161 + 3)
        * 94 + (* (hzs + 1) - 443)) * 32;
      fseek(fp, offset, SEEK_SET);
      for(i = 0; i < 16; i++) {
          dot[n][i] = fgetc(fp);
          dot[n][i + 16] = fgetc(fp);
      }
      hzs += 2;
    }
    fclose(fp);
    return(0);
}
```

```
/* 汉字点阵作为前景对原图象进行覆盖 */
if((hz_color = getpixel(j, i)) != 0)
    bmp_color = (bmp_color & 0x0f) + (hz_color <<
4);
if((hz_color = getpixel(j + 1, i)) != 0)
    bmp_color = (bmp_color & 0x0f) + hz_color;
* temp++ = bmp_color;
```

```
}
_write(handle_output, buffer_output, bmp_width/2);
}
free(buffer_output);
_close(handle_output);
}
```


西文状态下

汉字的无级放大和平滑显示

□李学春 苟列红

在应用软件的开发中,为了提高界面的质量,往往需要对汉字放大显示。虽然,以前有许多文章讨论过这个问题,但都忽视了平滑的方面,使显示出来的汉字粗糙难看;本人在实践过程中,总结出了 16X16 和 24X24 点阵字模地址的计算公式,采用直接读取字模的方法,实现了在西文状态下直接放大、平滑显示汉字的方法。

一、读 16 点阵和 24 点阵字模的方法

不同的中文系统,使用不同的点阵字库,计算字模地址的公式也不尽相同,但读取字模的方法基本是一样的,因为,库文件中每一位表示一个像素点,所以,对于 16 点阵的一个汉字要用 32 个字节表示,而 24 点阵的汉字需要用 72 个字节表示。16 点阵汉字的数据顺序见图 1, 24 点阵的汉字数据顺序见图 2。

第 0 字节	第 1 字节
第 2 字节	第 3 字节
...	...
第 28 字节	第 29 字节
第 30 字节	第 31 字节

图 1 16 点阵汉字的数据顺序

汉字点阵在字库中的位置是和汉字的区位码一一对应的,具体换算方法如下:设汉字的区码和位码分别为 $q1, w1$ 。

源程序清单:

```
program example;
uses hzunit, graph;
Const
  fstr1: array [1..12] of char = '石油大学华东';
  fstr2: array [1..14] of char = '计算数学教研室';
var
  gdev, gmod: integer;
begin
  gdev := DETECT; gmod := 8;
  closegraph;
  {$I-}
  if loResult <> 0 then
    halt(1)
```

我们知道,汉字是用两个字节来表示的,设分别是 $ch1$ 和 $ch2$,则区码和位码分别为:

$$q1 = \text{ord}(ch1) - 160$$

$$w1 = \text{ord}(ch2) - 160$$

16 点阵在字库中的第 0 个字节地址为:

$$\text{address} = (q1 - 1) * 94 * 32 + (w1 - 1) * 32$$

24 点阵字模对 2.13 和长城汉字系统而言,如果 $q1 < 176$,那么该图形符号点阵在 HZK24T 中的第 0 字节的地址为:

$$\text{address} = (q1 - 1) * 94 * 72 + (w1 - 1) * 72$$

对于 $q1 > 175$ 时,汉字分不同的字体分别存放在不同的库文件中,对于每一种字体,在相应库文件中汉字点阵的第 0 个字节地址为:

$$\text{address} = (q1 - 16) * 94 * 72 + (w1 - 1) * 72$$

对于 CCDOS 和 CLIB24 系统而言,在相应库文件中汉字点阵的第 0 字节地址为:

$$\text{address} = (q1 - 1) * 94 + (w1 - 1) + 256$$

对于金山繁体字库而言,在相应库文件中汉字点阵的第 0 个字节地址为:

$$\text{address} = (q1 - 3) * 94 + (w1 - 1) + 188$$

找到了首地址,连续读进一个汉字点阵,经过加工,处理成图形屏幕上指定区域的图形,就把汉字显示出来了。

二、汉字的放大和平滑技术

由于特殊显示效果的需要,很多时

候仅用汉字系统提供的字体是远远满足不了需要的,这就要求在现有字库的基础上采取放大技术,将汉字放大显示。汉字放大即需要将汉字点阵放大,放大的方法有两种,一是将原先显示一个像素的操作,改为在相应位置上画一个填充矩形的操作;二是将显示一个像素的操作改为在相应位置上画一个填充圆形的操作。这样做显示出的汉字特别粗糙,因此需要用磨光技术将一些地方填补成比较光滑的线条。

第 0 字节	第 3 字节	...	第 66 字节	第 69 字节
第 1 字节	第 4 字节	...	第 67 字节	第 70 字节
第 2 字节	第 5 字节	...	第 68 字节	第 71 字节

图 2 24 点阵汉字的数据顺序

磨光技术最简单而有效的方法是,把两个相邻矩形形成的三角形或两个相邻圆形形成的扇形空白区域填充起来,就形成了一个光滑的表面。本文以 2.13 为例,应用 Turbo Pascal 6.0,建立了一个实用的汉字单元,用户可方便地应用它,在西文状态下,显示各种大小的彩色汉字。本文程序在 486DX/33 上调试通过。

```
else
  initgraph(gdev, gmod, '\tp\bgi');
  smoothcc(199, 59, 1.6, 2.0, 1, 'h', fstr1);
  hztext16(330, 400, 2, '李学春');
  smoothcc(200, 110, 1.6, 2.0, 3, 's', fstr2);
  smoothcc(200, 166, 4.0, 4.9, 4, 'f', '研制人,');
  smoothcc(200, 280, 4.0, 4.9, 5, 'k', '李学春,');
  readln;
  closegraph;
end.
unit hzunit;
interface
Uses Graph;
```

```

procedure smoothcc(x,y:integer;chx,chy:real;color:byte;ziti:char;
    st:string);
procedure hztext16(X,Y:Integer;Color:word;S:String);
implementation
const multi:array[0..7] of byte=(1,2,4,8,16,32,64,128);
Type
    kzRec=array[1..24,1..3] of byte;
Var
    fs:file of kzRec;
    mk:array[0..25,0..25] of byte;
    hzklen,address:longint;
    F:file;
    I,J,K,X0,Y0:Integer;
    num:Integer;
    QuCode,WeiCode:longInt;
    Bl:Boolean;
    FileName:string;
Procedure HZ16write (X, Y: integer; QuWeiCode: integer; Color:
word);
Var B:Array [1..32] of Byte;
Begin
    QuCode:=QuWeiCode div 100;
    WeiCode:=QuWeiCode mod 100;
    Address:=(QuCode-1)*94*32+(WeiCode-1)*32;
    FileName:='c:\213\hzk16';
    Assign(F,filename);
    reset(F,1);
    seek(F,Address);
    blockread(F,B,Sizeof(B),Num);
    close(f);
    setcolor(color);
    for i:=1 to 16 do
    for j:=1 to 2 do
    begin
    for k:=7 downto 0 do
    Begin
        x0:=x+j*8-k
        y0:=y+i;
        bl:=(b[(I-1)*2+J] and multi[k])=multi[k];
        if bl then
            line(x0,y0,x0,y0);
        end;
    end;
    end;
end;
procedure hztext16(X,Y:integer;color:word;S:string);
Var
    X1,Y1,Ii,Jj,Qu,Wei,QW:integer;
    Len:word;
Begin
    len:=length(S); ii:=1;
    while ii<=len do
    begin
        while ord(s[ii])<160 do
        begin
            y:=y+8;
            outtextxy(x,y,s[ii]);
            ii:=ii+1; x:=x+8;y:=y-8;
        end;
        if ii<len then begin
            qu:=ord(s[ii])-160;
            wei:=ord(s[ii+1])-160;
            if (qu>0) and (wei>0) then
            begin
                ii:=ii+2;
                qw:=100*qu+wei;
                hz16write(X,Y,qw,color);
                X:=X+16;
            end;
        end;
    end;
end;

```

```

end;
end;
procedure smoothcc(x,y:integer;chx,chy:real;color:byte;ziti:char;
    st:string);
var
    len,ll:integer; ch1,ch2:char;
procedure init_zk(ch:char);
    var st:string;
begin
    st:='c\213\hzk24s';
    case upcase(ch) of
        'S':st:='c\213\hzk24s';
        'F':st:='c\213\hzk24f';
        'K':st:='c\213\hzk24k';
        'H':st:='c\213\hzk24h';
    end;
    assign(fs,st);
    reset(fs);
end;
procedure closefs;
begin
    close(fs);
end;
Function HzToQw(st:string):word;
Var qw:integer;
begin
    qw:=100*(ord(st[1])-160)+ord(st[2])-160-1;
    HzToQw:=qw;
End;
procedure smooth(x,y:integer;chx,chy:real;color:byte;
    ch1,ch2:char);
var
    potx,poty:integer;
    rx,ry:integer;
    plg:array[1..3] of pointtype;
    rhz,kzrec; qw:integer;
    i,j,k,cb:byte;
    offs:longint;
    qu,wei:integer;
begin
    qw:=hztoqw(ch1+ch2);
    qu:=(qw div 100); wei:=qw-100*qu;
    offs:=94*(qu-16)+wei;
    if qu<16 then
    begin
        closefs;
        offs:=94*(qu-1)+wei;
        assign(fs,'c:\213\hzk24t');
        reset(fs);
    end;
    seek(fs,offs);
    read(fs,rhz);
    for i:=0 to 25 do
    for j:=0 to 25 do
        mk[i,j]:=1;
    for i:=1 to 24 do
    for j:=1 to 3 do
    begin
        cb:=rhz[i,j];
        for k:=1 to 8 do
        begin
            if cb<128 then mk[i,(j-1)*8+k]:=0;
            cb:=cb shl 1;
        end;
    end;
    setcolor(color);
    setfillstyle(1,color);
    for potx:=1 to 24 do
    for poty:=1 to 24 do

```

西文方式下 24 点阵彩色 立体汉字放大显示

□赵永飞

编制程序时,经常要求汉字要带立体、醒目、美观、变化多样。本文给出一种西文状态下在屏幕的任意位置放大显示空心立体投影汉字的方法。

点阵的汉字放大原理:横向放大 M 倍,即原来汉字的每点变成了同一行的 M 点,纵向放大 N 倍,即原来显示的每点变成了同一列的 N 点。这样 24 点阵汉字,在经过横向 M 倍,纵向 N 倍放大后,每个汉字则变成了 $24M \times 24N$ 点阵的汉字。

立体空心汉字原理:利用 24×24 点阵字库,把其字模的每点分别向上下左

右延长,同时为其进行立体投影,根据所需要的投影方向、投影的大小把字模向该方向加点,然后再将需要空心的地方的点抹去(或改变颜色),在抹去时要注意不要抹去边上的点。

字模库可采用联想 CCS 汉字系统的 24 点阵字库或希望电脑公司 UC DOS 的 24 点阵字库等。24 点阵字模在汉字库中按列存放,每列 3 个字节,共 24 列,每个汉字占 72 字节,汉字在字模库中的位置用如下公式可以计算出来:(区内码-0xal) $\times 94$ +(位内码-0xal)

上式计算出汉字的记录号后乘以 72

则可得到汉字在字库中字模的第一个字节的位置。然后用 Turbo C 中文件定位函数将文件指针定位于该汉字的第一个字节处,连续读 72 字节则可得到该汉字字模。

本文给出的程序可以直接改编用于屏幕设计。如果觉得读字库太慢,可以考虑将字库装入虚拟盘上。或从一点阵字库中将要显示的字模读出存于另一文件中,然后在使用时直接从所建立的文件中读出,这样就大大提高了显示速度。本程序用 Turbo C2.0 编译,在 AST Premium 386SX/16 上运行通过。

程序清单如下:

```
#include<stdio.h>
#include<graphics.h>
#include<bios.h>
int getbit(unsigned char c,int n);
int puthz(int x,int y,int z,int color,int m,int n,char *p,int k)
/* x,y 表示在(x,y)处显示汉字,z 表示字间隔点数,color 表示汉字
的颜色,m 表示水平放大倍数,n 表示垂直放大倍数,*p 为要显示的
字符串,k 表示阴影的大小 */
FILE *pl;
main()
{
```

```
int Graphdriver,GraphMode;
char *p; clrscr();
Graphdriver=DETECT;
registerbgidriver(EGAVGA_driver);
initgraph(&Graphdriver,&GraphMode,"c:\\tc");
setbkcolor(BLUE);
if((pl=fopen("c:\\st24","rd"))==NULL)
/* st24 为联想汉字系统宋体字库名 */
{ printf("file not open \n");
closegraph();
exit(1); }
```

```
for i:=0 to 1 do
begin
rx:=i*2-1;
for j:=0 to 1 do
begin
ry:=j*2-1;
if (mk[potx+rx,poty+ry]=0)
and (mk[potx,poty+ry]=1)
and (mk[potx+rx,poty]=1)
then
Begin
plg[1].x:=trunc(x+(potx-1)*chx+(chx-1)*i);
plg[1].y:=trunc(y+(poty-1)*chy+(chy-1)*j);
plg[2].x:=trunc(plg[1].x+rx*chx);
plg[2].y:=plg[1].y;
plg[3].x:=plg[1].x;
plg[3].y:=trunc(plg[1].y+ry*chy);
fillpoly(3,plg);
end;
end;
if mk[potx,poty]=1 then
bar(x+trunc((potx-1)*chx),y+trunc((poty-1)*chy)
,x+trunc(potx*chx)-1,trunc(y+poty*chy-1));
```

```
end;
end;
begin
init_zk(ziti);
len:=length(st); ll:=0;
while ll<len do
begin
ll:=ll+1;
if ord(st[ll])>160 then
begin
ch1:=st[ll]; ll:=ll+1;
if ord(st[ll])>160 then
begin
ch2:=st[ll];
smooth(x,y,chx,chy,color,ch1,ch2);
end;
x:=x+trunc(chx*12);
end;
x:=x+trunc(chx*12);
end;
closefs;
end;
end.
```

```

puthz(1,10,6,4,2,2,"西文方式下 24 点阵汉",5);
puthz(131,100,10,4,2,2,"字立体显示",5);
getche();
fclose(pl);
closegraph();
exit(0);
}
int puthz(int x,int y,int z,int color,int m,int n,char *p,int k)
{
    unsigned int i,c1,c2,f=0;
    int il,i2,i3,i4,i5,rec,i6;
    long L; char by[72];
    while((i=*p++)!=0)
    { if (i>0xal)
        { if(f==0)
            { c1=(i-0xal)&0x07f;
              f=1; }
          else
            { c2=(i-0xal)&0x07f;
              f=0;
              rec=c1*94+c2;
              L=rec*72L;
              fseek(pl,L,SEEK_SET);
              fread(by,72,1,pl);
              for(il=0;il<24*m;il=il+1)/ * 水平方向显示 24M 点
                */
              for(i4=0;i4<m;i4++)
              for(i2=0;i2<=2;i2++)/ * 垂直方向显示 24N 点 */

```

```

                for(i3=0;i3<8;i3++)/ * 每个字节的 8 位 */
                if(getbit(by[il/m*3+i2],7-i3))
                for(i5=0;i5<n;i5++)
                { for(i6=0;i6<k;i6++)
                    { putpixel(x+il+i4-1+i6,y+i2*8*n+i3*n+i5-i6,color);
                      putpixel(x+il+i4+1+i6,y+i2*8*n+i3*n+i5-i6,color);
                      if(il!=0)
                      putpixel(x+il+i4+i6,y+i2*8*n+i3*n+i5-1-i6,color);
                      if(il!=24*m-1)
                      putpixel(x+il+i4+i6,y+i2*8*n+i3*n+i5+1-i6,color);
                    }
                }
                for(il=0;il<24*m;il=il+1)
                for (i4=0;i4<m;i4++)
                for(i2=0;i2<=2;i2++)
                for(i3=0;i3<8;i3++)
                if(getbit(by[il/m*3+i2],7-i3))
                { for(i5=0;i5<n;i5++)
                    if((il!=0)&&(il!=24*m-1))
                    putpixel(x+il+i4,y+i2*8*n+i3*n+i5,14+128);
                }
                }
                x=x+24*(m-1)+z;
            }
            return(x);
        }
        int getbit(unsigned char c,int n)
        { return((c>>n)&1); }

```


Visual C++ 实现多平台应用开发

□ 吴新瞻 译



今应用程序开发人员面对着不断增多的平台,包括在 Intel、MIPS 和 Digital 的 Alpha 等处理器上运行的 Windows NT 以及 Chicago 和 Macintosh 操作系统等。本文介绍利用 Visual C++ 系列开发工具,实现在这些平台上开发应用程序的实用技术。

Microsoft 已经宣布了其用一套工具(包括 94 年 Visual C++ 产品)开发 16 位和 32 位应用程序的策略。这个工具系列使开发支持多个平台间对象连接与嵌入(OLE2.0)的 Windows 和 Win32 的应用程序比较容易。实际上,使用 Visual C++ 和 Microsoft 基础类(简称 MFC),开发人员在多个平台上进行开发只需要学会一套工具和一套应用编程接口。

一、Microsoft Visual C++ 系列

Microsoft Visual C++ 系列包括下列产品:

1. Microsoft Visual C++ 1.5

它以 Windows 或 Windows NT 为宿主系统,可建立 16 位的高性能数据库和 OLE2.0 应用程序。这个工具已经可以让你用作升级到 Chicago 的开端。全部用 Visual C++ 1.5 编写的代码完全可以移植到所有其他 Visual C++ 系列成员上去。

2. Microsoft C++ 2.0 for Intel

它以 Windows NT 和 Chicago 为宿主系统和目标系统,对于在 Intel x86 系列处理器上开发的 32 位应用程序提供 OLE 2.0 和 ODBC(开放数据库互连)支持。

3. Microsoft Visual C++ RISC

它以非 Intel Windows NT 平台为宿主和目标系统,当前基于 MIPS R4000 和 Digital 公司的 Alpha AXP 处理器的 Visual C++ 的产品正在开发之中。

4. Microsoft Visual C++ 2.0 对

Macintosh 机的交叉开发版

它是 Visual C++ for Intel 版的增补,以 Apple 公司 Macintosh 机(其处理器为 Motorola 680x0 系列)为目标系统。此产品通过 Windows 代码库来实现 Macintosh 应用程序的开发。

以上的 Visual C++ 产品系列都设计成对多种平台提供相同的工具和语言实现,这样,应用程序开发人员在生成各种应用程序版本时,就能够利用他们原有的代码和编程经验。

二、基础类(MFC)库

Microsoft 的基础类库在 Microsoft C/C++ 7.0 版中首次引入(1991 年)。此后,Microsoft 都注意到使用户能够方便地使用其升级版本。每次 MFC 进行修改之后,用户只需要将源代码重新编译一次,就可得到新版的特色。此外,MFC 还被赋予跨平台的可移植性。这首先在 Windows NT 上使用的 Visual C++ 和 MFC 上实现,它们可以将 16 位的应用程序移植成 32 位的应用程序。

MFC 的下一个版本(3.0 版)也同样将提供用户所希望的可移植性。MFC 3.0 还支持 Win32 和 OLE 2.0,这是 Microsoft 未来操作系统的二个关键成份。使用 MFC 3.0,你就可以编写高性能的 32 位 Windows 应用程序,它们适用于各种平台。

MFC 3.0 是 MFC 2.5 的 32 位版本,最近随 Visual C++ 1.5 推出。由于它提供了新的类,使用户可以很快建立 32 位的 OLE 2.0 应用程序,带有完全的可视编辑及 OLE 自动化能力,这对于使 OLE 2.0 的开发工作简化起了核心的作用。MFC 3.0 还包含了数据库类,这可以使得建立 32 位的具有开放数据库互连性(ODBL)的 C++ 应用程序方便得多。

如果开发人员现在就需要实现 OLE 2.0 或数据库功能,就应从 Visual C++

1.5 和 MFC 1.5 着手进行。这样编出的源程序在 Visual C++ 2.0 系列推出之后,就可以移植到 Windows NT 和 Chicago 上。

Visual C++ 2.0 的各种版本都含有 MFC 3.0。使用 MFC 将使开发出的应用程序有最大的可移植性。

三、一致的跨平台开发环境

除了 MFC 和 Win32 API 之外,还应该理解为什么工具也可以跨平台进行工作。

Visual C++ 2.0 产品系列提供了一个一致的跨越各种开发平台的工具接口。这些工具包括一种新的可视工程管理程序,新的与类/代码浏览程序完全集成的代码编辑程序、新的调试工具,用户定义的工具条,以及一个同开发环境完全集成的源编辑程序。

对于 Visual C++ 2.0 产品系列而言,源编辑程序与代码编辑程序同其余的编程工具一起被集成为单一的环境。当你在 Visual C++ 2.0 中选择一个源程序进行编辑时,相应的源编辑程序便打开,并带有必要的工具选择图象或控制窗口。当你又把注意力返回到代码编辑程序窗口时,源工具窗口便从屏幕上消失,以便留出更多的屏幕空间。

这个环境还包含一个新的图形用户界面,它具有新的可按间距移动的对话框、用户定做的工具条、拖放式的文本、可按间距移动的输出窗口,以及立体式的外观。这些特色在 Microsoft 的 Word for Windows 6.0 以及 Microsoft Excel 5.0 等应用产品中都可以发现。

四、具有一致用户界面的 Wizard

AppWizard 可以帮助 MFC 应用程序开发人员方便地生成所需的各种源代码文件。它包含了各种选择项,可用于实

访问扩展内存的C++子程序

□丁弘 汪国庆

我们编写那些占用大量内存的程序,如图象处理程序以及复杂的三维动画制作程序时,一个注定要解决的问题是如何充分利用计算机内存,因为寥寥640K的基本内存实在难以存储计算过程中产生的如此冗长的图象及计算信息。于是,我们在程序设计时,必须设法充分享用计算机资源所提供的扩展内存(EMS),扩充内存(XMS)等存储空间。然而现今流行的C/C++软件均未提供直接访问EMS、XMS的接口。如Borland C++中的`malloc()`、`free()`、`coreleft()`等内存管理函数仅能访问640K的基本内存。本文向读者介绍一个作者开发的实用扩展内存访问的C++接口。在程序中使用这组函数。就可以象使用正常的基本内存一样,安全自如地使用扩展内存。

EMS内存可以有两兆甚至更多。但一次仅有一小部分可以出现在总线上,

因而表现出来的是EMS的页架。它可以放在任何地方,但通常是放置在视频卡缓冲区的下面,在段0C00H~0E00H。

为了访问EMS内存,应计算出所要取的页,并告诉内存卡来使它出现在页框架中。于是用户可以以页框架的形式读写EMS内存,就如同使用正常的基本内存一样。

当访问不同的页时,你告诉内存卡,将当前页换掉,或将一个页框架中不同的页放进来以使你访问它。

和扩充内存(XMS)相比,扩展内存是通过你在CONFIG.SYS文件中装入的一个扩展内存驱动器,如EMM386.EXE来处理的。这强迫所有想使用EMS的程序和EMS内存硬件通过一个共同的接口来通讯。这带来的好处是,使你不必了解EMS内存处理页的技巧,它允许多个程序同时使用EMS内存,而不必担心一个程序会覆盖另一个程序的数据。

本文所提供的EMS接口是标准的,即不受特定的版本的制约。

函数`isit_EMS()`,检查是否存在EMS驱动器,如果存在则返回1。

在确定EMS驱动器存在之后,才能安全地使用扩展内存中断INT67H。函数`CheckEMS()`检查实际的EMS内存是否存在,如果存在,则返回1。

在本文的程序中,一个EMS页是16384字节长(即16K),这在常数`EMSPagesize`中定义(0x4000)。

函数`EMScoretotal()`,返回EMS内存总页数。

函数`EMScoreleft()`,返回EMS内存所剩的页数。

函数`EMSpageframe()`,返回一个指向高位内存页框架底部的远指针。

函数`EMSsmallloc(n)`,分配扩展内存。EMS内存分配的最小单位是一个16K页,参数`n`指明要分配的页数。该函

现OLE 2.0容器(Container)和OLE 2.0服务器应用。利用专门针对ODBC要求而编写的MFC类,AppWizard也使得开发数据库应用程序比较便利。

ClassWizard可以访问全部C++类库,以及设计和编辑OLE、数据库类。各种Wizard具有一致的用户界面,使得开发跨平台的Win32/OLE应用程序变得简便。

Visual C++ 2.0较之于Visual C++ 1.0有着不少的改进。例如,工程文件的管理功能得到加强,源代码编辑软件也具有新的特色。举例来说,只要连击二次鼠标器,可以把一段文本或其中的某此行、列加亮,然后,移动鼠标器到新的位置,就可以把文本搬到那儿。甚至于可以把文本从一个源文件移到另一个源文件。此外,利用新的分裂窗口功能,可以打开单个源文件的多个视图。由于

源代码编辑软件与代码浏览软件紧密地集成在一起,你可以迅速地从一个特定代码符号(例如函数或类)的定义转到它的各次引用上。

Visual C++对Macintosh机的交叉开发版可以生成远程的680x0 Macintosh机上的目标文件,并且可以用集成的调试软件进行调试。

五、32位的编译程序和增量连接程序

为了保证代码对各种不同的平台具有兼容性,采用共同的语言实现很重要。Visual C++ 2.0的编译程序对于C++模板及例外情况处理的实现,均以C++标准委员会的工作文件为基础。模板是代表C++参数化类型实现的一种语言特点,主要用于一般具有类型安全性的聚合类。例外情况处理的实现使开

发人员针对不同的例外情况分别进行处理。

除了采用共同的语言实现外,Microsoft公司还一直注意寻求加强开发工具性能的途径,以便缩短进行工程开发所需的时间。这包括提供更快的编译程序,更快的浏览程序,以及增量连接程序(Incremental Linker)。

Visual C++ 2.0增量连接程序通过只连接从上一次连接以来已经变化的那部分代码,使得连接时间大大缩短。如果上一次连接后,只有一个源文件发生变化,那么,受它影响也变化的目标代码和调试信息才被重新连接进可执行代码中。这样连接过程要快很多。

Visual C++ 1.5已经可以买到,建议的零售价为599美元。Visual C++ 2.0本年就可以提供。

(本文英文资料由Microsoft提供)

数返回一个反映分配的那部分内存的句柄。

函数 EMSfree(h), 释放扩展内存。参数 h 是一个句柄, 该函数释放句柄 h 所反映的那部分扩展内存, 释放成功, 则返回 1。

函数 EMSmap(h, Ppage, Lpage) 处理物理页和逻辑页的。

```

/* ** EMS. CPP 源代码, 提供访问扩展内存的接口 ** */
#include <stdio. h>
#include <fcntl. h>
#include <io. h>
#include <dos. h>
const int EMSpagesize = 0x4000;
static int AllocatedPages = 0;
static int TotalPages;

static int isit _EMS(void);
static int CheckEMS(void);
static int EMScoreleft(void);
static int EMScoretotal(void);
static void far * EMSpageframe(void);
static unsigned int EMSmalloc(int);
static int EMSfree(unsigned int);
static int EMSmap(unsigned int, int, int);

/* ** 检查是否存在 EMS 内存驱动器 ** */
static int isit _EMS(void)
{
    int fh;
    union REGS rg;
    if(( fh=open("EMMXXXX0", O_RDONLY, &fh)) == -1)
        return 0;
    rg. h. ah = 0x44;
    rg. h. al = 0x00;
    rg. x. bx = fh;
    int86(0x21, &rg, &rg);
    close (fh);
    if(rg. x. cflag) return 0;
    if(rg. x. dx & 0x80) return 1;
    else return 0;
}

/* ** 检查是否存在 EMS 内存 ** */
static int CheckEMS(void)
{
    union REGS rg;
    rg. h. ah = 0x40;
    int86(0x67, &rg, &rg);
    if(rg. h. ah == 0) return 1;
    else return 0;
}

/* ** 返回剩余 EMS 内存页数 ** */
static int EMScoreleft(void)
{
    union REGS rg;
    rg. h. ah = 0x42;
    int86(0x67, &rg, &rg);
    if(rg. x. cflag) return -1;
    if(! AllocatedPages) {
        AllocatedPages = 1;
        TotalPages = rg. x. dx;
    }
    return (rg. x. bx);
}

/* ** 返回总的 EMS 内存页数 ** */
static int EMScoretotal(void)
{
    union REGS rg;

```

映射。h 是扩展内存句柄, Ppage 是物理页号, Lpage 是逻辑页号。映射成功, 则返回 1。

该程序, 在 486DX50 机上用 Borland C++ 2.0 编译运行通过。EMS 驱动器是 EMM386. EXE。

程序清单如下

```

if(AllocatedPages) return (TotalPages);
rg. h. ah = 0x42;
int86(0x67, &rg, &rg);
if(rg. x. cflag) return -1;
AllocatedPages = 1;
TotalPages = rg. x. dx;
return (TotalPages);
}

/* ** 返回指向 EMS 页框架的远指针 ** */
static void far * EMSpageframe(void)
{
    union REGS rg;
    rg. h. ah = 0x41;
    int86(0x67, &rg, &rg);
    if(rg. h. ah != 0)
        return (NULL);
    else return (MK_FP(rg. x. bx, 0));
}

/* ** 分配 n 页的 EMS 内存
    返回所分配内存的句柄 ** */
static unsigned int EMSmalloc(int n)
{
    union REGS rg;
    unsigned int handle;
    if(n>EMScoreleft()) return 0;
    rg. h. ah = 0x43;
    rg. x. bx = n;
    int86(0x67, &rg, &rg);
    if(rg. h. ah) return 0;
    else { handle = rg. x. dx;
        return (handle);
    }
}

/* ** 释放具有 h 句柄的那部分 EMS 内存 ** */
static int EMSfree(unsigned int h)
{
    union REGS rg;
    int i;
    for(i = 0; i<5; i++) {
        rg. h. ah = 0x45;
        rg. x. dx = h;
        int86(0x67, &rg, &rg);
    }
    if (rg. h. ah == 0) return 1;
    else return 0;
}

/* ** 将一逻辑页映射到物理页 ** */
static int EMSmap(unsigned int h, int Ppage, int Lpage)
{
    union REGS rg;
    if(Page<0||Ppage>3) return 0;
    rg. h. ah = 0x44;
    rg. h. al = Ppage;
    rg. x. bx = Lpage;
    rg. x. dx = h;
    int86(0x67, &rg, &rg);
    if (rg. h. ah != 0) return 0;
    else return 1;
}

```

C++ 语言环境下的 Windows 编程技术

□张兴滔

从 Microsoft 公司推出了 Microsoft Windows 3.0 以来,其友好的图形用户界面、强大的内存管理、文件管理、任务管理、多媒体管理等功能越来越为人们所喜爱,同时许许多多的软件开发厂商把应用软件和开发工具软件移植到 Windows 上,或者开发出以 Windows 为软件平台的 Windows 版本的软件,广大的软件开发人员越来越热衷于 Windows 环境下的软件开发。本文将以其著名软件公司 Borland 推出的 C++ V2.0 为基础,结合具体实例,谈一谈 Windows 的编程技术。

Borland C++ V2.0(以下简称 C++)是著名软件公司 Borland International 推出的面向对象的软件开发系统,它的集化程序员工作平台允许对一个程序进行编辑、编译、链接和调试等,它的工程管理文件可以使用户开发大型系统趋于简便,易于管理;它的通过对窗口类进行定义的软接口,实现了对 Windows 的编程,同时还引进了大量 Windows 类的 OOP 编程技术;另一方面,对于熟悉在传统的 DOS 和 UNIX 环境下编程的人来讲,开发 Windows 环境下的应用系统就会面临一个全新的环境以及概念:所熟悉的自顶向下、逐步求精的结构化、程序化程序设计方法已不适用了,而应用程序的完整结构也不存在了,在 C++ 中对 Windows 应用程序的开发和设计,实际上就是 OOP 程序设计方法的充分体现,下面谈谈它的实现过程和 OOP 的一些基本特性、基本概念。

一、Windows 应用程序开发的软环境

笔者建议最好先在 C++ 集成环境程序员平台下编辑 Windows 应用程序的源文件,经编译、链接成功后,存储源文件退出 C++,再在 Windows 环境下运行刚开发的 Windows 应用程序,这样就可以减少许多 C++ 作为 Windows 的

应用程序运行时,开发软件所带来的麻烦,如:当重新编译当前已装入的程序或 DLL 事例(Instance),或应用程序调试时,总可能会因为程序中的致命错误或对一般保护性错误忽略考虑,而导致系统崩溃,应用程序源文件未写盘而丢失等不必要的麻烦。

二、Windows 与其应用程序的工作原理及其在 C++ 上的实现

Windows 应用程序是一个事件驱动、信息传递、以 Windows 对象为组织单位的系统,在 C++ 中可看作是一个基于对类实例(对象)组织编程的系统。

1. Windows 与 OOP

在 Windows 环境下进行系统开发设计的主要步骤就是构造一系列图视窗口(即窗口数据),以及实现对这些窗口运行各种操作的函数(即窗口函数)的设计。在 C++ 中 OOP 编程方法的封装性(Encapsulation)正好完成上述任务,比传统程序设计方法实现起来更加优越;在这一系列窗口的构造中,包括主窗口(数据和函数)、子窗口(数据和函数)等等,它们之间的语义(指成员函数和成员数据)可以通过 OOP 的继承性(Inheritance)实现,另外,OOP 中的多态性保证了不同窗口使用同名的窗口函数不会引起混乱。这使得用户开发 Windows 应用程序成为可能。

2. Windows 应用程序的构成及其功能

Windows 应用程序的主控部分非常简单,它主要完成三项工作:

- ① 定义窗口类并且进行初始化登记;
- ② 创建、显示、更新窗口;
- ③ 进行信息处理循环;

可以简单地说,应用程序主控部分的任务就是构造所需的窗口并且提供给 Windows 进行管理,Windows 又为每个

窗口调用其窗口函数,处理输入输出以及其他事件,而这一切都是通过表达不同意思的信息在两者之间传递的结果,因而程序设计主体在于对信息处理过程的设计。

3. 程序主体设计技术

Windows 支持一系列属性,当用户选择了适当的属性,就可构成所需要的窗口,而窗口函数的处理方式则带来一种与传统程序设计方法(如分支、循环、调用等)不同的风格,取而代之的是窗口函数对信息(事例)处理的多路开关式流程。Windows 对每一个窗口的一次信息传递,就会告诉窗口有一个事例发生并以信息方式传过来;窗口函数就可以根据信息代表的不同意思进行处理,这同样适用于主窗口与窗口之间;另一方面,这样的信息传输方式在多个事件构成一个操作的情况下,就会给窗口函数带来破坏,这样就必须引入大量的控制变量来协调各个事件之间的关系,从而带来了窗口函数的复杂结构,由上可见,Windows 应用程序设计的主要工作就是窗口和窗口函数的设计,而关键之处在于窗口函数内部对不同信息进行处理的结构安排上。后面举了一个简单的 Windows 应用程序的设计例子,来说明设计一个完整的应用程序的步骤以及 OOP 程序设计特性在其中的体现,下面首先谈谈窗口函数及其函数处理。

三、窗口及其窗口函数的构造方法

在 Windows 环境下,应用程序的组织实际上就是对窗口及其窗口函数的组织,主要实现构造相应窗口及编制相应窗口函数,在 C++ 中用户可以通过 Windows 类(在 windows.h 中定义)加以实现,下面简单说明一下它们的定义及构成。

1. 主窗口

登记主窗口窗口类的代码如下:

用“宝合”UPS 电源, 再也不怕电网突然掉电



PH-500S



PH-1000

当外电网突然掉电时, 微电脑靠其内部的
 贮能电容通常能维持10MS左右的工作
 时间, 为了避免数据丢失, 磁盘和磁头遭受
 损失, 造成严重后果, 这就需要有一种电
 源系统, 不仅要有不间断供电性能, 而且
 还要有抑制电网噪音, 抗电网干扰, 稳压
 等性能, 在外电网突然掉电时, 以保证微
 电脑系统的正常运行。



PH-1KL



PH-3KL



PH-600

“宝合”UPS 电源获奖一览表

年 度	获 奖 名 称
1989 1991, 1992	北京市新技术产业开发试验区优秀拳头产品
1991	91 年度国家级新产品
1992	被国家科委成果管理办公室推荐为国内优秀产品 北京市科学技术进步三等奖
1993	被北京市技术监督局评为北京市企业标准成果奖 全国第七届运动会指定产品



北京希望电脑公司电源部

地址: 北京海淀路 82 号

邮编: 100080

信箱: 北京 8721 信箱

电话: 2567819, 2561058

传真: 2561057

上海希望电脑公司

地址: (200052) 上海新华路 416 号

电话(传真): (021)2521656, 2569929

南京希望电脑技术公司

地址: (210005) 南京市中山南路 105 号

电话: (025)4410794, 4410549

传真: (025)4410548

成都希望电脑公司

地址: (610015) 成都市新南路四维村街 6 号

电话(传真): (028)5589787, 5556333

广州希望电脑技术公司

地址: (510620) 广州天河体育西路育菁三街二

电话: (020)7505151, 7505152, 7505153

传真: (020)7500275



北京希望电脑公司出版、发行的技术资料，具有内容新、种类多、出版快的特点，并以高效率、高质量、高信誉的服务赢得广大用户的好评。

北京希望电脑公司资料事业部

地址：北京海淀路 82 号

信箱：(100080) 北京 8721 信箱

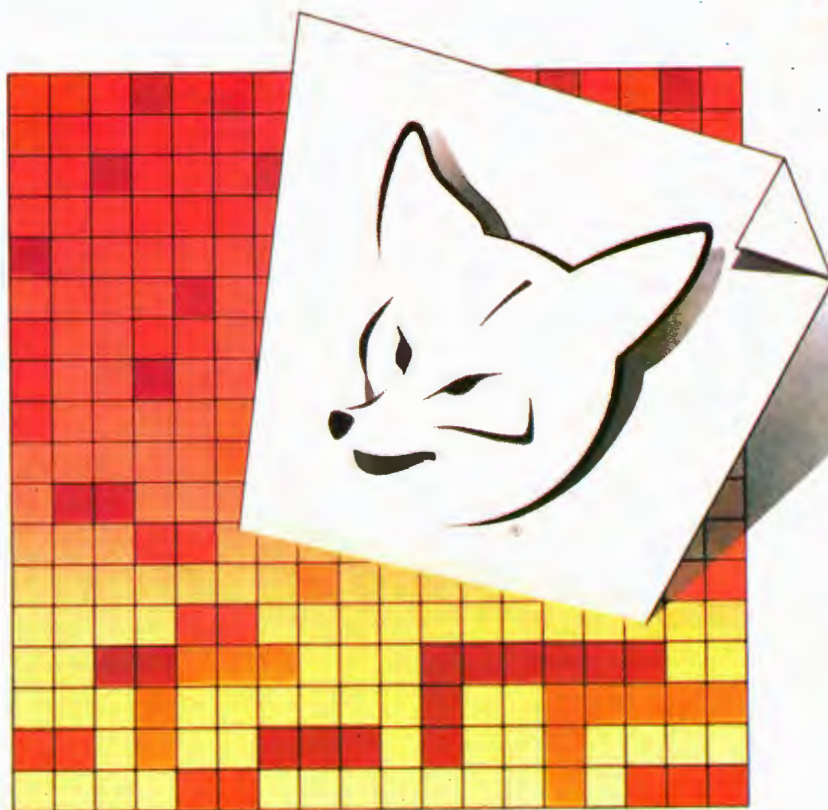
电话：2562329、2541992

传真：2561057





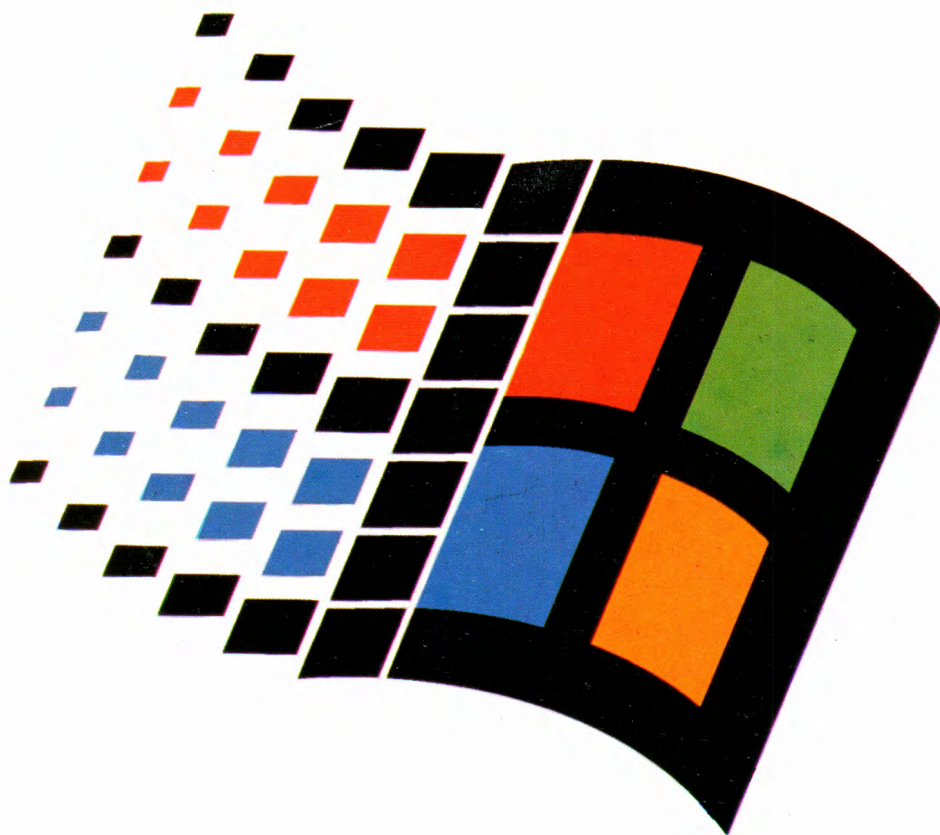
Unlock
the power of
Microsoft FoxPro
for MS-DOS and
Windows.



Microsoft FOXPRO

LIBRARY CONSTRUCTION KIT

For MS-DOS and Windows™



Microsoft
WINDOWS NT™

```
static void Register( void )
{
    WNDCLASS wndclass; // 定义 WNDCLASS 的实例来登记主窗口
    // 下面是对主窗口的属性进行设置
    wndclass.style = CS_HREDRAW | CS_VREDRAW;
    wndclass.lpszClassName = "MyMenu";
    wndclass.cbClsExtra = 0;
    wndclass.cbWndExtra = sizeof( MainWindow * );
    wndclass.hInstance = Main;
    wndclass.hIcon = LoadIcon( Main, "MyIcon" );
    wndclass.hCursor = LoadCursor( NULL, IDC_ARROW );
    wndclass.hbrBackground = GetStockObject( WHITE_BRUSH );
    wndclass.lpszMenuName = MyMenu;
    wndclass.lpszClassName = szClassName;
    if( ! RegisterClass( & wndclass ) ) // 注册登记失败退出
        exit( FALSE );
}
```

Windows 需要知道应用程序中所有不同类型的窗口类信息, 这样你就需要把这些信息填入到一个 WNDCLASS 的结构(在 windows.h 中定义)中并通过 RegisterClass 来注册登记, 你只需为每一个 Windows 类登记一次, 特别地, 同一程序的以前事例已向 Windows 登记了此窗口类, 就可以不再登记了。

Windows 将根据用户在窗口类中指定的窗口信息来建立一个窗口, 在窗口类中不仅定义了窗口的显示特性, 而且还命名了窗口名字并指定了窗口函数, Windows 必须调用它来给窗口传递信息; 因为一个应用程序中的各个事例可以使用主窗口的相同定义(OOP 中的继承性), 因而每一个窗口类只需要定义一次(虽然一个应用程序可以有多个窗口类)。

2. 窗口函数

在上面登记主窗口时, WndProc 为窗口函数, 它主要控制处理下列 Windows 发送的事例: ①窗口创建事例; ②窗口显示事例; ③窗口更新事例; ④窗口撤消事例; ⑤窗口菜单选择事例。

3. 菜单资源的定义及其使用方法

主窗口中有一个带有名为 "MyMenu" 的菜单条菜单, 它是用来建立主窗口时, 在菜单条中建立一个由菜单资源描述文件中描述的信息的菜单, 在主窗口函数中有对菜单选择的控制, 在处理菜单项的选择时, 要遇到一系列对话框, 它们也是资源文件描述定义的。菜单资源文件建立及其描述如下:

```
#include <windows.h>
SIMPLEPAINT MENU // 定义菜单名
BEGIN
    MENUITEM "& Quit!", 100 // 菜单 1-1
    POPUP "& Shape"
    BEGIN
        MENUITEM "& Line" 201 // 菜单 2-1
        MENUITEM "& Ellipse", 202 // 菜单 2-2
        MENUITEM "& Rectangle", 203 // 菜单 2-3
    END
    POPUP "& Pen"
    BEGIN
        POPUP "& Thickness"
        BEGIN
            MENUITEM "& Thin", 301 // 菜单 3-1-1
            MENUITEM "& Regular", 302 // 菜单 3-1-2
            MENUITEM "& Thick", 303 // 菜单 3-1-3
        END
        POPUP "& Color"
```

```
BEGIN
    MENUITEM "& Red", 304 // 菜单 3-2-1
    MENUITEM "& Green", 305 // 菜单 3-2-2
    MENUITEM "& Black", 306 // 菜单 3-2-3
END
END
MyIcon ICON MyIcon.ico // 图标描述
```

4. 图标(ICON)资源的建立和使用方法:

在上面登记主窗口和菜单资源描述中都有一个 MyIcon 量, 它表示该应用程序的图标是由 Windows 的 SDK 建立的。要使用它只需在登记主窗口时, 象以上那样表明, 而且要在菜单所在的资源文件中正确描述。

四、工程文件管理编译和链接 Windows 应用程序

C++ 集成环境程序员平台的优越性, 为 Windows 应用程序的开发提供了极大方便, 但它在结构设计、模型构造等方面又不同于其他程序设计。而且开发一个 Windows 应用程序往往需要多个源文件(头文件、资源文件等等), 这就使得多个相互联系的文件难于管理。C++ 的工程文件正好解决了用户开发 Windows 应用程序时, 多个文件难于管理的麻烦。建立一个 Windows 应用程序的工程文件的方法如下:

(1) 在 C++ 的程序员平台上分别编辑所需的 C++ 源文件(.CPP)、头文件(.H)、模块定义文件(.DEF)以及资源文件(.RC)等几个文件, 并把它们放在同一个目录中。

(2) 打开工程文件管理对话框, 建立一个新的工程文件, 把以上几种文件一起放到这个工程文件中。

(3) 打开工程文件后, 只有选择了正确的编译和链接选择项后, 才能通过 Build All 命令, 对工程文件给出的相关源文件进行编译和链接, 产生正确的应用程序, 它包括许多选择项:

- 在硬件没有配置协处理器时, Windows 应用程序应使用浮点仿真数学库链接;

- 选择编译、链接最终文件生成形式为 .DLL 或 .EXE;

- 选择自动依赖检查(Auto-Dependency), 以便每次所需的多个源文件都能重新编译。

(4) 系统文件、库函数的路径必须加以说明(使编译器和链接系统能找到相应的 Include、lib、classlib 文件)。

(5) windows.h 文件必须放在源文件的开始处。

五、具体例子

下面举个完整的例子, 该程序在 AST 386SX/16 上通过。

源程序清单如下:


```
#include <windows.h> // 头文件的定义
#include <stdlib.h>
#include <string.h>
long FAR PASCAL _export WndProc(HWND hWnd,
WORD iMessage, WORD wParam, LONG lParam);
class Main // 定义 Main 类并进行初始化
{
public:
    static HANDLE hInstance;
    static HANDLE hPrevInstance;
    static int nCmdShow;
    static int MessageLoop( void );
```

```
};
HANDLE Main::hInstance = 0; // 初始化
HANDLE Main::hPrevInstance = 0;
int Main::nCmdShow = 0;
int Main::MessageLoop( void )//与 Windows 之间信息传递函数
{ MSG msg;
  while(GetMessage( & msg, NULL, 0, 0))
  {
    TranslateMessage( & msg );
    DispatchMessage( & msg );
  }
  return msg.wParam; }
class Window //基类 Window 的定义
{
protected:
  HWND hWnd;
public: //定义窗口类的公有函数
  HWND GetHandle(void) { return hWnd; }
  BOOL Show(int nCmdShow) {
    return ShowWindow(hWnd, nCmdShow); }
  void Update(void) { UpdateWindow(hWnd); }
  virtual long WndProc(WORD iMessage, WORD wParam,
    LONG lParam) = 0; };
class MainWindow : public Window // 派生类的定义
{
private:
  static char szClassName[14];
  static void FAR PASCAL LineFunc(int X, int Y,
    LPSTR lpData);
public:
static void Register( void) // 登记主窗口
{
  WNDCLASS wndclass; //定义 WNDCLASS 类的实例
  wndclass.style = CS_HREDRAW | CS_VREDRAW;
  wndclass.lpfWndProc = ::WndProc;
  wndclass.cbClsExtra = 0;
  wndclass.cdWndExtra = sizeof( MainWindow * );
  wndclass.hInstance = Main::hInstance;
  wndclass.hIcon = LoadIcon( Main::hInstance, "MyIcon");
  wndclass.hCursor = LoadCursor( NULL, IDC_ARROW );
  wndclass.hbrBackground = GetStockObject( WHITE_BRUSH);
  wndclass.lpszMenuName = MyMenu;
  wndclass.lpszClassName = szClassName;
  if (! RegisterClass(& wndclass)) { //注册失败则退出
    exit( FALSE); } }
}
MainWindow(void) // MainWindow 的构造函数
{
  // 创建窗口
  hWnd = CreateWindow(szClassName,
    szClassName,
    WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT,
    0,
    CW_USEDEFAULT,
    0,
    NULL,
    NULL,
    Main::hInstance,
    (LPSTR) this);
  if (! hWnd) // 创建窗口失败就退出
    exit( FALSE);
  Show(Main::nCmdShow); //显示窗口
  Update(); //更新窗口
}
long WndProc( WORD iMessage, WORD wParam, LONG
lParam);
void Paint(void);
struct LINEFUNDATA
{
  HDC hDC;
  char FAR * LineMessage;
```

```
int MessageLength;
LINEFUNDATA(char * Message) // 定义构造函数
{
  hDC = 0;
  MessageLength = strlen(Message); //取字符串长度
  LineMessage = new char far[MessageLength+1];
  lstrcpy( LineMessage, Message); };
~LINEFUNDATA(void) {delete LineMessage; }
//定义析构函数
};
char MainWindow::szClassName[] = "Hello, WINDOWS V3.
0!"; //定义串量
void MainWindow::Paint(void)
{
  PAINTSTRUCT ps; RECT rect; FARPROC lpLineFunc;
  LINEFUNDATA LineFuncData( "Welcome !"); //初始化
  lpLineFunc = MakeProcInstance((FARPROC)Mainwindow::
    LineFunc, Main::hInstance);
  BeginPaint(hWnd, & ps);
  GetClientRect(hWnd, (LPRECT) & rect);
  LineFuncData.hDC = ps.hdc;
  SetTextAlign(ps.hdc, TA_BOTTOM);
  LineDDA(0, 0, 0, rect.bottom/2, lpLineFunc,
    (LPSTR) & LineFuncData);
  EndPaint(hWnd, & ps);
  FreeProcInstance(lpLineFunc); }
void FAR PASCAL MainWindow::LineFunc(int X, int Y,
  LPSTR lpData)
{
  LINEFUNDATA FAR * lpLineFuncData =
    (LINEFUNDATA FAR *) lpData;
  TextOut( lpLineFuncData->hDC, X, Y, lpLineFuncData->
    LineMessage, lpLineFuncData->MessageLength);
}
long MainWindow::WndProc(WORD iMessage, WORD wParam,
  LONG lParam)
{
  switch (iMessage) //信息处理过程
  {
    case WM_CREATE:
      break;
    case WM_PAINT:
      Paint(); break;
    case WM_DESTROY:
      PostQuitMessage( 0); break;
    default:
      return DefWindowProc( hWnd, iMessage, wParam,
        lParam);
  }
}
long FAR PASCAL _export WndProc(HWND hWnd,
  WORD iMessage, WORD wParam, LONG
  lParam)
{
  if( iMessage == WM_CREATE)
  {
    LPCREATESTRUCT lpCS;
    lpCS = (LPCREATESTRUCT)lParam;
    return MainWindow::WndProc(iMessage, wParam, lParam);
  }
  else
    return DefWindowProc( hWnd, iMessage, wParam, lParam);
}
#pragma argsused //关闭形参未使用警告
int PASCAL WinMain(HANDLE hInstance, HANDLE
  hPrevInstance, LPSTR lpszCmdLine, int nCmdShow)
{
  Main::hInstance = hInstance;
  Main::hPrevInstance = hPrevInstance;
  Main::nCmdShow = nCmdShow;
```

将 TXT 文本文件转换为 可执行阅读文件的方法

□马 勇

 本文件在 DOS 操作系统中本身无法执行显示,只能借助于 TYPE 等 DOS 命令或某些阅读文件、字处理文件来显示其内容。如果能将文本文件转换成可执行阅读文件,则可以减少用户击键次数并可完善 TYPE 命令的不足,且可避免内存开销过大、系统庞大的问题。其实原理很简单:首先对文本文件进行改造使其符合 C 语言的格式,利 printf() 函数逐行显示 TXT 文本文件中的内容,为完善 TYPE 命令不可分屏显示的不足,可在显示一定行数后加上暂停功能,并借助于 Turbo C 的编译功能将其连接成可执行文件。

为提高手工改写的效率,使人们能方便、有效地进行文本文件的改造,笔者用 C 语言写了一个自动将 TXT 文本文件转换成符合 C 语言格式的小程序。

其命令行接收参数形式的使用为:

C:>TXT - C

得到中文说明

C:>TXT - C AUTOEXEC. TXT AUTOEXEC. C

将 AUTOEXEC. TXT 文件转换成 AUTOEXEC. C 程序

源程序清单如下:

```
# include <stdio.h>
main(int argc, char * argv[])
{
    FILE * fp1, * fp2;
    int ch, ch1, m1 = 0x0a, i = 0;
    char * c1 = " ";
    * c2 = " #include <stdio.h> ",
    * c3 = "main()",
    * c4 = " ",
    * c5 = "printf(",
    * c6 = "\"",
    * c7 = ")",
    * c8 = "\"",
    * c9 = "delay(4000)",
    * c10 = "\\n";
    if(argc < 2) {
        printf("功能:将 TXT 文件自动转换成 Turbo C 程序\n");
        printf("用法:c:>txt - c name1name2\n");
        printf("说明:name1 是 TXT 文件名;\n");
        printf("name2 是 Turbo C 的程序名\n");
    }
```

```
exit(1);}
if ((fp1=fopen(argv[1], "r")) == NULL)
{
    printf("can not open file %s\n", argv[1]);
    exit(1);}
fp2=fopen(argv[2], "w");
fputs(c2, fp2);
fputc(m1, fp2);
fputs(c3, fp2);
fputc(m1, fp2);
fputs(c4, fp2);
fputc(m1, fp2);
fputs(c5, fp2);
fputs(c6, fp2);
while ((ch=fgetc(fp1)) != EOF)
{
    if(ch < 0x0e)
    {
        i++;
        fputs(c10, fp2);
        fputs(c6, fp2);
        fputs(c7, fp2);
        fputs(c1, fp2);
        fputc(m1, fp2);
        if (i == 23) {
            i = 0;
            fputs(c9, fp2);
            fputs(c1, fp2);
            fputc(m1, fp2);
        }
        if ((ch1=fgetc(fp1)) != EOF) {
            fputs(c5, fp2);
            fputs(c6, fp2);
        }
        if (ch > 0x30)
            fputc(ch, fp2);
    }
    else
    {
        fputc(ch, fp2);
        fputs(c8, fp2);
        fputc(0x1a, fp2);
        fclose(fp1);
        fclose(fp2);
        printf("转换成功! \n");
    }
}
```

该程序在 COMPAQ 486 以及长城系列计算机 286~486 上编译通过。

```
if (! Main, :hPrevInstance) {
    Mainwindow, :Register();
    MainWindow MainWnd; //定义 MainWindow 的实例
    return Main, :MessageLoop();
}
```

其图标资源文件描述如下:
MyIcon ICON MyIcon. ico

其模块定义源文件如下:
NAME MyWindow
DESCRIPTION 'My Windows Application Test'
EXETYPE WINDOWS
CODE PRELOAD MOVEABLE
DATA PRELOAD MOVEABLE MULTIPLE
HEAPSIZE 1024
STACKSIZE 5120

FoxBASE+调用C语言的新方法

□傅达峰

众所周知, FoxBASE 数据库语言不是一种完备的编程语言, 它在文件处理、数值计算等方面的功能均有缺陷, 尤其是不能完成涉及系统硬件等方面的底层操作。为此, FoxBASE 提供了与其他语言的接口, 用于完成 FoxBASE 力所难及的功能。

FoxBASE 与其他语言的接口有两个, 其一是用 RUN 命令运行 .EXE 或 .COM 程序, 其二是用“LOAD”和“CALL”命令装入调用 .BIN 程序。第一种方法可以用各种高级语言编程, 但运行时要加载 COMMAND.COM 第二副本, 要占用较多内存且运行时间较长, 而且 FoxBASE 与之参数传递困难; 第二种方法具有占用内存少, 运行快捷而且可以与 FoxBASE 相互传递参数, 但一般要用汇编语言编程, 如涉及文件处理、数学运算等复杂操作, 即使对有经验的程序员来说也是一件麻烦的事。如果有一种方法可以结合两种方法的优点, 那必将极大地方便应用程序的编写。

C 语言是一种具有汇编语言效率与功能的高级语言, 而且还具有极好的扩展性, 可以适用于不同的需要。笔者通过详细研究 Turbo C 的启动过程, 编写了符合 FoxBASE 调用规则的启动代码 COFOX.ASM, 采用此启动代码就可以将 Turbo C 改造成为可编写用于和 FoxBASE 接口的 C 语言的编译系统。

一、Turbo C 语言的启动代码

在 Turbo C 2.0 系统 6 号盘中, 有一个名为 CO.ASM 的文件, 它就是 Turbo C 的启动代码源程序, 用 TASM 可将其编译成 COT.OBJ、COS.OBJ 等各个模式下的启动代码。启动代码的主要功能是初始化程序的运行环境、分析处理命令行参数和系统环境参数并传递给 C 语

言的主函数 main(), 并提供 _exit(), abort() 等退出函数。

Borland 公司提供启动代码源程序 CO.ASM 的原意就是让用户在必要时可以通过修改或改写 CO.ASM 来适应自己的需要, 因此我们可以编制一个遵循 FoxBASE 调用二进制文件 .BIN 规则的 Turbo C 启动代码作为 FoxBASE 与 C 语言的接口。

程序 1 就是笔者编写的实现上述功能的启动代码源程序 COFOX.ASM, 通过以下命令可以编译成所需的启动代码 COFOX.OBJ:

```
C>TASM COFOX, COFOX /
MX
```

与 CO.ASM 不同, COFOX.ASM 提供给 main() 的两个参数 int arg, char far, argp 分别对应于 FoxBASE 传递给 .BIN 程序的 BX 和 DS:BX 值, 在 C 程序中通过对 arg、argp 的操作即可实现 Turbo C 与 FoxBASE 间的参数传递。

二、与 FoxBASE 接口的 C 语言编程

使用 COFOX.OBJ 代替 COT.OBJ 完成 Turbo C 程序的启动过程, 我们就可以用 C 语言方便地编写与 FoxBASE 接口的程序, 但是由于 FoxBASE+ 对二进制程序的特殊要求, 在 Turbo C 编程时要注意以下几点:

(1) 由于 .BIN 程序运行所用的空间不得超过实际程序的大小, 部分标准 C 库函数, 如动态申请释放内存函数就不应被使用。

(2) COFOX.ASM 中定义的栈长度有限, 在 C 语言中应尽量避免使用大项数的局部数组, 也应慎重使用递归算法, 以防止栈溢出。如一定要使用大项数的局部数组或递归算法, 可以相应地增加

COFOX.ASM 中通过 Stacksize 定义的栈长度。

(3) 如果在 C 程序中打开过文件, 则在返回 FoxBASE 之前, 应将所有文件关闭, 然后用 exit() 或 _exit() 退出。

(4) .BIN 程序一旦被 FoxBASE 装入后, 在被释放之前, 其在内存中的位置就固定不变, 因此在 C 程序中只要定义全局变量, 即可在两次调用之间传递参数。例如程序 3 就在两次调用之间传递了密钥 SEED 的值。

(5) 在 C 语言编译时, 应采用 Tiny 模式, 以便编译出的 .EXE 可以被转换成 .BIN 文件。

使用 COFOX.OBJ 代替 COT.OBJ 参加 Turbo C 编译的方法有多种, 其中最简单的是将 COFOX.OBJ 改名为 COT.OBJ 参加编译, 其他方法由于篇幅所限就不一一介绍了, 有兴趣者可以参阅 Turbo C 手册。将 C 程序编译成 .EXE 后, 就可以使用 EXE2BIN 将其转换成 .BIN 文件供 FoxBASE 调用。

三、Turbo C 实现 FoxBASE 内部字符串变量加密、解密的实例

在一些应用如财务、证券系统的口令及用户权限等数据, 一般都需要进行加密, 如使用 FoxBASE 编程其加密强度明显是不够的, 这时我们就可以使用本文提供的方法来实现字符变量的加密、解密, 而其他类型的数据如数值型、日期型也可以转换成字符型后再进行加密。程序 2 就是用 C 语言编制的字符加密、解密程序 BYTELOCK.C, 它的密钥 SEED 是由 1~8 组成的排列数, 只有输入正确的密钥才能得到正确的数据; 程序 3 是使用 BYTELOCK.C 加密、解密字符的实例 SAMPLE.PRG。

程序1 COFOX.ASM 清单如下:

NAME COFOX
PAGE 60,132

```

;-----
; COFOX.ASM -- Start Up Code
; Turbo-C Run Time Library for FoxBASE version 2.1
; Copyright (c) by Darphony Fow,1993,02.01
; All Rights Reserved.
;-----

```

```

stacksize equ 1024 ;STACK SIZE
_TEXT SEGMENT BYTE PUBLIC 'CODE'
ENDS
_DATA SEGMENT PARA PUBLIC 'DATA'
ENDS
BSS SEGMENT WORD PUBLIC 'BSS'
ENDS
_BSEND SEGMENT BYTE PUBLIC 'BSEND'
ENDS
_EMUSEG SEGMENT WORD COMMON 'DATA'
_EMUSEG ENDS
_CRTSEG SEGMENT WORD COMMON 'DATA'
_CRTSEG ENDS
_CVTSEG SEGMENT WORD COMMON 'DATA'
_CVTSEG ENDS
_SCNSEG SEGMENT WORD PUBLIC 'DATA'
_SCNSEG ENDS

```

```

DGROUP GROUP _TEXT, _DATA,
    _EMUSEG, _CRTSEG, _CVTSEG,
    _SCNSEG, _BSS, _BSEND, _STACK
ASSUME CS:_TEXT,DS:DGROUP
extrn _main:near
_TEXT SEGMENT
    org 0H
STARTX PROC FAR
    PUSH DS ;save DS
    PUSH BX ;save BX
    MOV CS:_BX,BX ;save BX to _BX
    MOV AH,30H
    INT 21H
    MOV CS:_version,ax
    ;keep major and minor version number
    MOV AH,0
    INT 1AH ;get current BIOS time in ticks
    MOV WORD PTR CS:_StartTime,dx ;save it for clock() fn
    MOV WORD PTR CS:_StartTime+2,cx
    MOV CS:_SS,SS ;save SS to _SS
    MOV BP,SP
    MOV CS:_SP,BP ;save SP to _SP
    CLI
    PUSH CS ;let SS
    POP SS ;equal to CS
    MOV BP,offset DGROUP:_stackbottom
    MOV SP,BP ;define new SP
    STI
    MOV BX,CS:_BX ;restore BX from _BX
    PUSH DS ;transmit argument -->
    PUSH BX ;'char far *argp'
    PUSH BX ;transmit argument --> 'int arg'
    PUSH CS ;let DS
    POP DS ;equal to CS.
    call _main
_exit LABEL NEAR
_exit LABEL NEAR
    CLI
    MOV SS,CS:_SS ;restore SS from _SS

```

```

MOV BP,CS:_SP
MOV SP,BP ;restore SP from _SP
STI
POP BX ;restore BX
POP DS ;restore DS
RETF ;return to Foxplus
STARTX ENDP

```

```

_abort proc near
    mov cx,lgth_abortMSG
    mov dx,offset DGROUP:abortMSG
    push cs
    pop ds
    mov ah,040h
    mov bx,2
    int 021h
    jmp _exit
_abort endp

```

_TEXT ENDS

_DATA SEGMENT

```

Copyright db 'COFOX V2.1,1993,02.01'
_SP dw 0
_SS dw 0
_BX dw 0
_version label word
_osmajor db 0
_osminor db 0
_errno dw 0
_brklvl dw DGROUP:edata@
_StartTime dw 0,0
abortMSG db
'Abnormal program termination',13,10
lgth_abortMSG equ $-abortMSG
_environ dw 0
_psp dw 0
_DATA ENDS

```

```

_CVTSEG SEGMENT
_RealCvtVector label word
_CVTSEG ENDS

```

```

_SCNSEG SEGMENT
_ScanTodVector label word
_SCNSEG ENDS

```

```

_BSS SEGMENT
bdata@ label byte
_BSS ENDS

```

```

_BSEND SEGMENT
edata@ label byte
_BSEND ENDS

```

```

_STACK SEGMENT
_stacktop db stacksize dup(0)
_stackbottom label byte
_STACK ENDS

```

```

public _errno
public _version
public _osmajor
public _osminor
public _brklvl
public _StartTime
public _RealCvtVector
public _ScanTodVector

```

```
public _exit
public _abort
public __exit
public __environ
public __psp
END STARTX
```

程序 2 BYTELOCK.C 清单如下:

```
/* ----- */
/* BYTELOCK for FoxBASE version 1.00 */
/* Copyright (c) by Darphony Fow,1993,11,17 */
/* All Rights Reserved. */
/* ----- */
```

```
char lock_string(unsigned char far *p);
char unlock_string(unsigned char far *p);
char set_seed(unsigned char far *p);
char lower(char ch);
int farstrlen(char far *p);
```

```
char far *farp;
unsigned char cnt,mask,ch,new;
char seed[8] = {1,2,3,4,5,6,7,8};
```

```
main(int arg,char far *argp)
```

```
{
    char result;
    if (!arg) exit();
    farp = argp;
    switch (lower(*farp))
    {
        case 'u': result = unlock_string(++farp);
                    break;
        case '|': result = lock_string(++farp);
                    break;
        case 's': result = set_seed(++farp);
    }

    *argp = result;
}
```

```
char unlock_string(unsigned char far *p)
```

```
{
    while(*p)
    {
        mask = 0x80;
        new = 0x00;
        for(cnt = 0;cnt<8;cnt++)
        {
            ch = (((*p)&mask)<<cnt)>>(8-seed[cnt]));
            mask = mask>>1;
            new = new|ch;
        }
        *p = new;
        p++;
    }
    return 'o';
}
```

```
char lock_string(unsigned char far *p)
```

```
{
    mask = 0x01;
    while(*p)
    {
        new = 0x00;
        for(cnt = 0;cnt<8;cnt++)
        {
            ch = (((*p)&(mask<<(seed[cnt]-1)))<<(8-seed
                [cnt]))>>cnt;
            new = new|ch;
        }
        *p = new;
        p++;
    }
    return 'o'
}
```

```
char set_seed(unsigned char far *p)
```

```
{
    unsigned char far *tp;
    if (farstrlen(p) != 8) return 'e';
    new = 0x00;
    mask = 0x01;
    tp = p;
    for(cnt = 0;cnt<8;cnt++)
    {
        if((*tp)<'1' || (*tp)>'8') return 'e';
        ch = (*tp) - '1';
        if (new & (mask<<ch)) return 'e';
        new = new | (mask<<ch);
        tp++;
    }
    tp = p;
```

```
for(cnt = 0;cnt<8;cnt++,tp++) seed[cnt] = (*tp) - '0';
return 'o';
}
```

```
char lower(char ch)
```

```
{
    if((ch)>'@' && (ch)<'[') return (ch + 'a' - 'A');
    return (ch);
}
```

```
int farstrlen(char far *p)
```

```
{
    int cnt;
    for(cnt = 0;*p;cnt++,p++);
    return cnt;
}
```

程序 3 SAMPLE.PRg 清单如下:

```
load scrsv
s = 's'
call scrsv with s
s = 'r'
call scrsv with s
if s = "E"
    @ 24,0 say '显示模式改变,无法恢复屏幕'
else
    @ 24,0 say '屏幕恢复成功'
endif
canc
```

可编辑、超长输入函数的实现

□付 辉



者在开发管理系统计算器模块时,涉及到了在固定区域内输入超长表达式的问题。为此笔者编制了输入函数 `char * lget()`,较好地解决这一问题。

函数调用格式为

```
char * lget(char str[],int leftposx,int leftposy,int
length)
```

本函数的特点在于超长输入、可编辑。调用此函数可在指定的屏幕位置(leftposx,leftposy)输入长于length(限定长度)的字符串。并具有仿全屏编辑功能。可随意查看、插入、删除任何位置字符。在本函数的基础上稍加改动即可处理汉字。

应该注意的是在调用此函数前,应说明字符数组str[],数组的大小为您想输入的字符串的大小。

本函数在浪潮286-12、长城386/33c上调试通过。编程环境为Turbo C2.0。

源程序清单如下:

```
#include<mem.h>
#include<stdio.h>
#include<conio.h>
#include "calculat.h"
char * lget(char str[],int leftposx,int leftposy,int length)
{
    int ch, loop=1;
    char * temp; /* 当前显示字符串的起始址 */
    int len=0,i,pos=0,ins=0,xpos;
    /* 输入字符串长度len;字符串中当前字符位置;pos 屏幕上当前
    前字符位置 xpos */
    temp=str;
    gotoxy(leftposx,leftposy);
    while(len<length && loop)
    {
        ch=GETCH();xpos=wherex();
        switch(ch) {
            case LEFT:
                if(pos>0)
                {
                    pos--;
                    if(len<=length || pos<length) /* 当前字符位置或
                    字符串长度小于length */
                        gotoxy(leftposx+pos,leftposy);
                    else { temp--; /* .....和.... 大于length */
                        gotoxy(leftposx,leftposy);
                        for(i=0;i<length && temp+i<=str+len-1;i++)
                            putchar(* (temp+i));
                    }
                }
                break;
            case RIGHT:
                if(pos<len)
                {
                    pos++;
                    if(xpos!=leftposx+length)
                        gotoxy(leftposx+pos,leftposy);
                }
                /* 当前字符光标在可显示区间[leftposx,leftposx+length]内 */
                else if(len>length && temp<str+len-length)
                    break;
            case HOME: /* 从字符串开始显示并将光标置于最右端 */
                temp=str;pos=0;
                gotoxy(leftposx,leftposy);
                for(i=0;i<length && temp+i<=str+len-1;i++)
                    putchar(* (temp+i));
                gotoxy(leftposx,leftposy);
                break;
            case END: /* 显示字符串后length个字符 */
                temp=(len>length)? str+len-length:str;
                pos=len;
                gotoxy(leftposx,leftposy);
                for(i=0;i<length && temp+i<=str+len-1;i++)
                    putchar(* (temp+i));
                break;
            case ENTER:
                loop=0; break;
            case DEL:
                if(pos<=len)
                {
                    if(pos!=len) memmove(str+pos,str+pos+1,len-pos);
                    /* 删除字符串中的字符 */
                    else if(len>length) temp--;
                    /* 删除最后一个字符且字符串长度超过显示长度 */
                    len--;
                    gotoxy(leftposx,leftposy);
                    for(i=0;i<length;i++)
                        putchar(' ');
                    gotoxy(leftposx,leftposy);
                    for(i=0;i<length && temp+i<=str+len-1;i++)
                        putchar(* (temp+i));
                    gotoxy(xpos,leftposy);
                    break;
                }
            case INS:
                ins=!ins; break; /* 退格 */
            case BACK:
                if(pos>0)
                {
                    if(temp>str && xpos==leftposx) temp--;
                    memmove(str+pos-1,str+pos,len-pos);
                    len--;
                    pos--;
                    gotoxy(leftposx,leftposy);
                    for(i=0;i<length && i++) /* 清显示区域 */
                        putchar(' ');
                    gotoxy(leftposx,leftposy);
                    for(i=0;i<length && temp+i<=str+len-1;i++)
                        putchar(* (temp+i));
                    gotoxy(xpos-1,leftposy);
                }
                break; /* 字符输入 */
            default:
                if(ins && pos!=len)
                    memmove(str+pos+1,str+pos,len-pos); /* 插入字符 */
                temp++;
                len++;
                gotoxy(leftposx,leftposy);
                for(i=0;i<length && temp+i<=str+len-1;i++)
                    putchar(* (temp+i));
                gotoxy(xpos,leftposy);
                break;
        }
    }
}
```

```
/* 前当字符光标在显示区间最右端且字符串长度超过显示长度 */
{
    temp++;
    gotoxy(leftposx,leftposy);
    for(i=0;i<length;i++)
        putchar(* (temp+i));
}
}
break;
case HOME: /* 从字符串开始显示并将光标置于最右端 */
    temp=str;pos=0;
    gotoxy(leftposx,leftposy);
    for(i=0;i<length && temp+i<=str+len-1;i++)
        putchar(* (temp+i));
    gotoxy(leftposx,leftposy);
    break;
case END: /* 显示字符串后length个字符 */
    temp=(len>length)? str+len-length:str;
    pos=len;
    gotoxy(leftposx,leftposy);
    for(i=0;i<length && temp+i<=str+len-1;i++)
        putchar(* (temp+i));
    break;
case ENTER:
    loop=0; break;
case DEL:
    if(pos<=len)
    {
        if(pos!=len) memmove(str+pos,str+pos+1,len-pos);
        /* 删除字符串中的字符 */
        else if(len>length) temp--;
        /* 删除最后一个字符且字符串长度超过显示长度 */
        len--;
        gotoxy(leftposx,leftposy);
        for(i=0;i<length;i++)
            putchar(' ');
        gotoxy(leftposx,leftposy);
        for(i=0;i<length && temp+i<=str+len-1;i++)
            putchar(* (temp+i));
        gotoxy(xpos,leftposy);
        break;
    }
case INS:
    ins=!ins; break; /* 退格 */
case BACK:
    if(pos>0)
    {
        if(temp>str && xpos==leftposx) temp--;
        memmove(str+pos-1,str+pos,len-pos);
        len--;
        pos--;
        gotoxy(leftposx,leftposy);
        for(i=0;i<length && i++) /* 清显示区域 */
            putchar(' ');
        gotoxy(leftposx,leftposy);
        for(i=0;i<length && temp+i<=str+len-1;i++)
            putchar(* (temp+i));
        gotoxy(xpos-1,leftposy);
    }
    break; /* 字符输入 */
default:
    if(ins && pos!=len)
        memmove(str+pos+1,str+pos,len-pos); /* 插入字符 */
    temp++;
    len++;
    gotoxy(leftposx,leftposy);
    for(i=0;i<length && temp+i<=str+len-1;i++)
        putchar(* (temp+i));
    gotoxy(xpos,leftposy);
    break;
}
```

用 MFC 读写数据库

□常 久译

开放式数据库连接(ODBC—Open Database Connectivity)因其提供了通用的存储及检索数据的接口而正迅速地发展起来。ODBC 的多面性允许一个应用程序处理及浏览以不同数据库格式存储的数据,如:FoxPro、SQLServer、Microsoft Access、Paradox 等,因此它采用 SQL(结构查询语言——Structured Query Language)关系数据库查询语言的工业标准。最近 Microsoft 推出了 MFC(Microsoft Foundation Class Library)V2.5 简化了应用编程接口(API)并提供了一组使编程更容易的数据库类。MFC 数据库类中的对象、方法(Method)、术语和 Visual Basic 及 Microsoft Access Basic 中的非常类似,如: CDatabase 类和 Visual Basic 中的 Database 对象很相似,它包括一系列同名的成员函数如 ExecuteSQL、CommitTrans 及 Rollback。本文重点讨论如何用 MFC 所提供的类来读写数据库的方法。

定义或查询一个数据源

应用 MFC 数据库类有三个步骤:

- 1) 定义或查询一个数据源;
- 2) 连接到数据源;
- 3) 浏览和处理数据源中的信息。

那么什么是数据源呢?数据源由一组表中数据及必要的读取及查找这一数

据的信息组成。典型而言,一个数据源通过置于 Windows 控制板(Control Panel)中的 ODBC Administration 程序定义。用户规定这个数据表、数据所在位置、数据库名字或目录。最重要的是给数据源定义一个名称。

一旦定义或命名了数据源的名称,一个 MFC 应用仅需规定数据源的名子(DSN)以获得对可能的任一表的读写。DSN 用于 CDatabase 对象和数据源之间的连接。通过 SQLConfigDataSource ODBC API 函数(而不是 ODBC Administration 程序)数据源的名字可被自动地创建,如:

```
SQLConfigDataSource(NULL, ODBC_ADD_DSN, "FoxPro Files (*.dbf)",
"DSN=Test\0Description=Test\0"
"DataDirectory=d:\samples\0"
"FileType=FoxPro 2.5\0SingleUser=False\0");
```

上述程序段创建了一个称之为 Test 使用 FoxPro ODBC 驱动程序的 DSN,数据表在 D:\SAMPLES 目录下。FoxPro 驱动程序用 FoxPro V2.5 格式检索及存储数据。采用 SQLData Sources ODBC API 可使程序发现数据源的名称是否存在,本文后面的程序可使用户检索所有数据源名称的目录。程序 DSN 建立之后,即可读写该数据源的表,该表是怎样

存储的(DBF 文件、MDB 文件)、存储在何处并不重要,因为在查询及处理数据源的数据时,应用程序仅需要知道数据源的名称。

很容易地连接到数据源

MFC 用一个 CDatabase 对象表示数据源。DSN 作为 CDatabase::Open 中的一个参数,CDatabase 对象而后作为 CRecordset 对象的构造者,CRecordset 实现通过 CDatabase 对象定义的数据源的查询。CRecordset 对象也管理 SQL 查询的结果表,以及一些如 MoveFirst、MoveNext、MovePrev、Update、Delete 及 AddNew 的方法,这些方法允许一个应用检索和处理表中的记录。

CRecordset::Open 用以实现查询初始化(用 CRecordset::Requery 刷新数据),CRecordset::Open 将 SQL 查询作为一个参数接受,如果没有规定 SQL 查询,则调用 Open,Open 调用 CRecordset::GetDefaultSQL 检索包括查询的一个字符串,例如,假设 GetDefaultSQL 函数在 CRecordset 派生类 CMemberset 中,则有

```
CString CMemberset::GetDefaultSQL()
{ return "Select LastName, FirstName
from Members"; } 在默认的情况下,
CMemberset 对象表示 SQL "Select
LastName,FirstName from Members"
```

```
* (str+pos)=(char)ch;
len++; }
else { * (str+pos)=(char)ch;
if (pos==len) len++;
pos++;
if ((xpos==leftposx+lenght || pos==len) &&
len>lenght) temp++;
/* 最右端超常输入时,显示地址加一 */
}
if (len>lenght || ins)
{ gotoxy(leftposx,leftposy);
for (i=0; i<lenght && temp+i<=str+len-1; i++)
putch(* (temp+i));
if (ins && pos<len) gotoxy(xpos,leftposy);
else if (!ins && pos<len) {
xpos<leftposx+lenght ? xpos+1:leftposx+
lenght;
gotoxy(xpos,leftposy); }
```

```
}
else putch(ch);
break;
}
* (str+len)='\0';
return str;
}

int GET(void)
{ int ch1,ch2;
ch1=getch();
if (ch1==0)
{ ch2=getch();
return 256+ch2;
}
else return ch1;
}
```

DevCast 尽展 Visual Basic 的功能

□刘耀东 译

利用软件组成部件的唯一问题是缺乏一种简单明了、前后一致的方法来做到——无对象性，不需要收集对象的基本情况、无公用程序语言。Microsoft 公司的 Microsoft Office 4.0 使这一切为之改观。对象链接和嵌入 (OLE) 2.0 版软件，提供了一个工业领域的对象模型，Office 4.0 则满足了成百上千程序化的、可重新利用的对象需要。Microsoft 公司的 Microsoft Excel 5.0 成为最先随其公用模块化构造语言 Visual Basic 提供而应用于实际工作的实例。

Microsoft Excel 5.0 被用做构造的基础，实际应用中，多数情况下使用 Visual Basic 编程，集成 Microsoft Office 对象快速构造解决方案。

才出现的 DevCast 选择了一个称之为客户信息工作台 (CIW) 的企业信息系

统，来说明利用这些新工具和技术，模块化软件可以被写得更快、更好。

我们以 Microsoft Excel 5.0 为基础，大量采用 Visual Basic 编制程序，集成 Microsoft Office 对象以快速形成处理方案。在这里，还将演示用新的 Visual C++ 1.5 来建立用户自己的 OLE 2.0 对象。

CIW 方案集成了几个常规客户信息、处理任务。客户数据库的管理、通讯、销售和策略报告的制定变得更为容易，而且可以互相集成在一起。用户通过实际应用，可以改进方案的效率、可控制性和通讯。

利用定义良好的函数，用户需求和系统模型的需求图可从中得到益处。这些模型是销售分析、制定预算、报告、通讯、商品清单和项目管理等。

上述解决问题的方案已成为

Microsoft Excel 5.0 的工作规程，每个功能都将被赋予一张用户接口清单及一张描述 Visual Basic 代码位置的模型清单。每个逻辑模型通常依赖于某种“核心”块代码来完成最令人感兴趣的功能。

报告模型对 Microsoft Office 4.0 的程序化控制和多样化的对象范围提供了最为有力的说明。

实现范例

销售分析模型使用了货币转换对象，以便将国际化销售数字转换为本国货币。这一货币转换对象是用做为 OLE 2.0 对象的 Visual C++ 来实现的。在运行一次 CONV.EXE 可执行文件后 (以便注册)，对 Microsoft Excel 5.0 中 Visual Basic 的对象援引是非常简单的：

```
Sub saConvertCurrency ()
    dim currConv as object
```

查寻的结果，在没有作为第二个参数的 SQL 查寻串规定的情况下，CMemberset 对象通过调用 Open 调用 GetDefaultSQL 来获得上述查寻串，并且数据源被查寻，从而返回一个包括 Members 表中所有记录 LastName 及 FirstName 的所有字段。调用函数如 MoveNext 和 MovePrev 允许 CMemberset 记录组 (recordset) 对象在结果集合的整个记录上查寻，结果集合是从 SQL 查寻中返回

检查可能的数据源名称的程序段：

```
BOOL GetDBCDataSourceNames(CStringList
* plist)
{
    ASSERT(PList->IsEmpty());
    HENV hEnv;
    if (SQLAllocEnv(&hEnv)!=SQL_SUCCESS
return FALSE;
    CString strDSN;
    SWORD cbDSN;
    UCHAR szDescription[300];
    SWORD cbDescription;
    RETCODE retCode;
    while(retCode=SQLDataSources(hEnv,
SQL_FETCH_NEXT, (UCHAR FAR *)strDSN,
GetBuffer(30), 30, &cbDSN,
(UCHAR FAR *)&szDescription, 300,
```

的表。

有待解决的问题

现在有两个问题：在应用中记录字段和变量之间怎样传输数据？这些信息怎样显示？MFC 2.5 定义 RFX (记录字段交换—Record FieldXchange) 函数，这些函数将记录字段中的数据传输到用户定义的变量中。事实上，用户可以采用 Class Wizard 映射 (map) 记录组中的各列

1 各字段到数据变量中并让其选择正确的 RFX 函数以调用之。为了浏览数据，MFC 包括称之为 CRecondView 的类，该类很象 Visual Basic 中的 Form。CRecordView 类显示一个对话面板并支持工具棒 (toolbar) 按钮用以跳过一个记录组及显示数据。

MFC 对于 ODBC API 的应用已作了大量的简化。

```
&cbDescription)!=SQL_NO_DATA_FOUND
&&.retCode!=SQL_ERROR
{
    strDSN.ReleaseBuffer();
    PList->AddTail(strDSN);
    {
        if (retCode==SQL_ERROR)
            return FALSE;
        .
        .
        .
        return TRUE;
    }
}
```

(本文英文资料由 Microsoft 提供)


```
set currConv=createobject
("currencyconverter")
worksheet("Sales Data").activate
for each cell in selection
    sellCountry=HEMOCOUNTRY
    buyCountry=
        cell.offset(0,-2).value
    cell.value=currConv.convert
        (cell.offset(0,-1).value
next
```

dim 语句将 CurrConv 说明为一个对象,并且生成了一个对象句柄的存储空间。set 语句则生成 currencyconverter 对象的一个句柄。并且将其分配给 CurrConv。程序然后对选中的当前工作报表的每个单元进行循环扫描,并将其转换为本国货币,在这之后,另一个 set 语句,通过将 currConv 设为 nothing 而释放了句柄。这样就释放了提供给这一货币转换对象请求占用的系统资源。

销售工作数据报表由 SQL Server 数据库得到其原始数据,数据库的连接是通过使用数据菜单中的 GET External 命令来完成的。当利用 Tools 工具菜单中的 Make Add In 命令装入包含在 Microsoft Excel 5.0 中的外部追加命令 XL-QUERY 时,这一 Get External 命令则被插入。外部追加命令 GET External 允许利用连接到 SQL Server 数据库中的开放数据库连接(ODBC)SQL Server 的原始数据信息。一旦原始数据在销售数据工作报表中就位,仅用下面这行代码可足以对数据进行更新:

```
Range(RNG_RAM_SALES_DATA).
ClearFormats
Application.Run"QueryRefresh",Range
(SHEET_SALES_DATA."&"!$A$1")
```

第一条语句中为确定销售数据范围而采用了 ClearFormats (清除格式)方法,以便清除全部格式。下一条语句是运行 QueryRefresh (查询更新),该命令为 XLQUERY add-in 所提供的的一个函数,该函数以保存在工作表上的 ODBC 原始数据为定义参数,该参数也是接收工作表左上角数据的起始单元地址。

报告模型最能说明 Microsoft Office 4.0 的程序控制和对象多样性的强大功能。正如你在下面所能看到的,Visual Basic 用来驱动 Microsoft Excel 5.0 中的 Microsoft Word 软件。ut_setAppobject 函数用途广泛,它可以接受应用名字,并且将对象句柄返回给该应用,以便应用可按照任何其它的 OLE 对象来处理。跟

在该函数后面的程序段,则通过调用该函数以打开 Word 6.0 并且通过 Word Basic 来使用全部的 Word 对象。

```
Function ut_SetAppobject(objOLE2App As
Object,strAppName As String,
    strPath As String,strExtension As
String) As Integer
'Call ut_RunApp to make sure app is running.
If Not ut_RunApp(strPath,strExtension)
Then
'If not ,display message.
MsgBox ut_RepToken(MSGNO_START_
OLE2_APP,strAppName),
vbExclamation,APP_NAME
Else
if so,create the object reference.
Set objOLE2App = Getobject ("", strApp-
Name)
ut_SetAppObject=True
End If
End Function
If ut_SetAppObject(objword,"word. Basic",
OLE2_WORD_PATH,"DOC") Then with
objword
AppRestore For demo purposes.
FileNewDefault
```

通讯使用了新的对话 API(TAPI),为解决问题的程序员设计的经由电话提供程序化控制。下面程序段说明了在自动呼叫情况下 TAPI 的用法。

```
Declare Function ta_tapiRequest-
MakeCall Lib "TAPI.DLL" Alias "
tapiRequestMakeCall"
(ByVal lpszDestAddress As String,ByVal
lpszAppName As Any,ByVal lpszCalledparty
As Any,ByVal lpsz Comment As Any As-
Long
Sub ta_placeCall(strNumber As String)
Dim dwReturn As Long
dwReturn = ta_tapiRequestMakeCall (str-
Number,0,0,0)
If dwReturn<>0 Then
MsgBox ta_GetTAPIError (dwReturn),
vbExclamation,"TAPI Error"
End If
End Sub
```

第一条语句将 ta_tapiRequest MakeCall 函数定义为 Visual Basic 应用函数。该函数可执行 TAPI 提供的 tapiRequestMakeCall 函数。下面的函数只是执行这一函数以完成一独立的过程(在此情形下,DIALER.EXE 是做为 TAPI Software Developer Kit 中的基本 TAPI 服务器提供的,该过程由 TAPI 服务器提供服务的 TAPI 呼叫来完成。

利用 Microsoft Office 对象,已解决了大部分问题。现在我们面临着货币转

换需求,这需要一个客户部件,这一部件可用 Visual C++ 1.5 版来完成。下面的程序段说明了为存取和访问价格数据库而在这一部件中所采用的技术。该程序实现了价格的数值转换,即假定在销售国价格为 m_sellcountry 个单位,在买入国价格则为 m_buycountry 个单位。销售国的价格 m_sellcountry 和买入国的价格 m_buycountry 通过调用 Visual Basic (或任何其它)程序被设置为特征标志,这一程序被执行以响应转换法的一次调用:

```
CConvSet set;
set.m_strFilter="CODE=?";
set.m_FINDCODE=m_sellCountry;
if (!set.open())set.IsEOF() return-1;
amount/=set.m_BUY_USD;
set.m_FINDCODE=m_buyCountry;
if (!set.Requery())set.IsEOF() return
-1;
amount *=set.m_SELL_USD;
return amount;
```

第一条语句将 set 说明为具有 CConvSet 类型的变量,这代表了用 Microsoft Foundation Class Library (MFC) 2.5 数据库所产生的记录集合。下行则产生一由字符串"CODE=?"为标志的参数化的查询。该查询由赋予 m_sellcountry 具有 m_FINDCODE 属性的参数给出。然后执行查询,并且在同一个条件 if 语句内检查零记录。如果以美元为单位得到了一个有效的销售国价格 m_sellcountry 查询,欲求的数值 amount 可由返回的查询队列中的买入国价格 set m_BUY_USD 去除而得到。在程序的第二部分,为了经由 USD 引用获得 m_buyCountry,使用了同样的技术(注意到这次使用了 Request 来代替 Open,因为该查询已经打开)。用该数值去乘以 amount,从而有效地将 m_sellCountry 转换为 m_buyCountry。用 VSD 引用做为计算第三国外汇牌价的公式。

以 OLE 为基础,对部件进行再次利用,在 30 天内就完成了概要至编码设计。利用 WOSA 技术(ODBC, TAPI, Messaging API),在为解决问题而提供的强有力功能时,发出了诸如后端数据库变化、电话服务提供者、邮件系统的主要维护信息。Visual Basic for application 做为一种“粘合剂”,将不同的部件天衣无缝地集成起来,而 Visual C++ 1.5 则使得产生一个完全同类的货币转换对象变得更加容易。

(本英文资料由 Microsoft 提供)

步长型循环的三值选择技巧

□宁正元

循环问题是在程序设计中遇到的最普遍的问题,循环结构是程序设计语言的最重要的控制结构。把需要反复多次执行的任务设计成循环程序,不仅可减少源程序占用的内存空间,还可大大简化程序设计过程。然而,对同一循环问题采用不同的方法所设计出的程序,其运行效率可以有成千上万倍的差别。步长型循环尤为显著,应该引起程序设计者足够的重视。

各种程序设计语言几乎都提供有步长型循环结构。其控制变量的作用有两类,一类仅控制循环次数,另一类除控制循环次数外还在循环体中为循环任务所利用。这后一类循环控制变量的初值、终值和步长值的选择具有很大的技巧性。三值选择适当,可以较大幅度提高程序运行效率。下面,我们以 BASIC 语言的一个典型步长型循环问题的分析设计为例,介绍循环控制变量的三值选择技巧。

问题:把一元人民币兑换为零币,最多可有多少种不同的兑换方法?

最易理解的简单方法是穷举法。让表示五角、二角、一角、五分、二分、一分六种零币的六个循环控制变量 A、B、C、D、E、F 都从 0 开始以一分步长递增到一元(即终值依次为 2、5、10、20、50 和 100),统计出满足等式 $50A + 20B + 10C + 5D + 2E + F = 100$ 的次数而得到兑换方法数。相应的程序如下:

```
5 REM PROGRAM1
10 LET N=0
20 FOR A=0 TO 2
30 FOR B=0 TO 5
40 FOR C=0 TO 10
50 FOR D=0 TO 20
60 FOR E=0 TO 50
70 FOR F=0 TO 100
80 LET S=50*A+20*B+10*C+5*D+2*E+F
90 IF S<>100 THEN 110
100 LET N=N+1
110 NEXT F
120 NEXT E
130 NEXT D
140 NEXT C
150 NEXT B
160 NEXT A
170 PRINT "N=";N
180 END
```

然而,这个程序需执行最内层循环的循环体各语句共 $3 \times 6 \times 11 \times 21 \times 51 \times 101 = 21417858$ 次,在微机 PC-1500 上需约 684 小时才能求得正确结果 $N=4562$ 。

如果在循环控制变量的终值选择上排除掉使 $S \geq 100$ 的情况,最内层循环的循环体只需执行 104560 次,在微机 PC-1500 上约需运行 3 小时 18 分 20 秒,执行时间缩短 207 倍。此时的程序变成:

```
5 REM PROGRAM2
10 LET N=0
20 FOR A=0 TO 2
30 FOR B=0 TO (100-A*50)/20
40 FOR C=0 TO (100-A*50-B*20)/10
50 FOR D=0 TO (100-A*50-B*20-C*10)/5
60 FOR E=0 TO (100-A*50-B*20-C*10-D*5)/2
70 FOR F=0 TO 100-A*50-B*20-C*10-D*5-E*2
80 LET S=50*A+20*B+10*C+5*D+2*E+F
90 IF S<>100 THEN 110
100 LET N=N+1
110 NEXT F
120 NEXT E
130 NEXT D
140 NEXT C
150 NEXT B
160 NEXT A
170 PRINT "N=";N
180 END
```

对现在这个程序,还可通过适当选取 F 的初值和步长值来提高运行效率。由于 A、B、C、E 不论取何值,对于以分为单位的数值而言均为偶数, F 的奇偶性完全取决于 D 的奇偶性,所以 F 的初值可选为 $D - 2 * \text{INT}(D/2)$; 又因为 E 每改变量值 1 需 F 改变量值 2 与之协调,故 F 的步长值又可选为 2。这样,最内层循环的循环体只需执行 53543 次,在微机 PC-1500 上需运行 1 小时 50 分 20 秒,比最初的程序缩短运行时间 372 倍。其程序为:

```
5 REM PROGRAM3
10 LET N=0
20 FOR A=0 TO 2
30 FOR B=0 TO (100-A*50)/20
40 FOR C=0 TO (100-A*50-B*20)/10
50 FOR D=0 TO (100-A*50-B*20-C*10)/5
60 FOR E=0 TO (100-A*50-B*20-C*10-D*5)/2
70 FOR F=D-2*INT(D/2) TO 100-A*50-B*20-C*10-D*5-E*2 STEP 2
80 LET S=50*A+20*B+10*C+5*D+2*E+F
90 IF S<>100 THEN 110
100 LET N=N+1
```

(下转第 46 页)

COBOL 程序中输出信息的加密方法

□张建民

程序编制人员都希望自己所开发的软件具有保密性,不被他人无偿盗用。在程序中加入特定字符,是保护版权的一种较简单的方法。但设定的字符串仍可以通过其他工具软件修改。根据本人用机经验,现介绍三种在 COBOL 程序中加密汉字输出信息的方法。运用此类方法,可使 COBOL 运行程序中的汉字输出信息不易被修改,从而可起到一定的版权保护作用。

我们知道,COBOL 的数据部是用来描述数据结构并确定数据存放顺序的。因此,对于所输出的信息(字符串)进行杂乱排列,或定义数据结构为可计算项,则在编译后的运行程序中很难查找并替换。

方法一,在数据部把要输出的汉字字符串存放位置打乱,使其通过查找并替换运行程序中字符串的目的落空。而在过程部再通过数据传送,将打乱的顺序重新整理输出。如先将“城固县支行”几个字打乱,使得在可执行程序中找不到这一字符串。但通过查看,仍可发现排列杂乱的“城固县支行”几个汉字字符。此种方法简单易行,但保密性不强。

方法二,思路同第一种方法,是将一个汉字内码拆分为两个字符,再打乱存放。例如,“城”字的内码是“B3C7”,将此两个字节位置岔开存放,这样,在运行程序中将找不到“城”字。此种加密,破解度较难。

方法三,是首先将一个汉字的内码转化成十进制数,之后再通过加减一个特定值生成另一个汉字内码的十进制值,并输出该汉字。这样,即使是摸索出了存放结构,但强行改动一个汉字,则会引起此字之后所有汉字的变化。此种加密方法,破解度几乎为零。

以下给出一个完整的 COBOL 程序,该程序用 COBOL Level II 编译通过。

读者亦可以将此思路,应用到其他编程语言中。

```
IDENTIFICATION DIVISION.
PROGRAM - ID. EXAM.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING - STORAGE SECTION.
```

* 第一种加密方法的数据,即将汉字顺序打乱。

```
01 HZ1S.
02 HZ1S3 PIC XX VALUE "县".
02 HZ1S1 PIC XX VALUE "城".
02 HZ1S5 PIC XX VALUE "行".
02 HZ1S4 PIC XX VALUE "支".
02 HZ1S2 PIC XX VALUE "固".
01 HZ1T.
02 HZ1T1 PIC XX.
02 HZ1T2 PIC XX.
02 HZ1T3 PIC XX.
02 HZ1T4 PIC XX.
02 HZ1T5 PIC XX.
```

* 第二种加密方法的数据,将汉字内码拆成两个字节。再岔开位置存放。

```
01 HZ2S.
02 HZ2S1-1 PIC X VALUE X"B3".
02 HZ2S2-1 PIC X VALUE X"B9".
02 HZ2S3-1 PIC X VALUE X"CF".
02 HZ2S4-1 PIC X VALUE X"D6".
02 HZ2S5-1 PIC X VALUE X"D0".
02 HZ2S1-2 PIC X VALUE X"C7".
02 HZ2S2-2 PIC X VALUE X"CC".
02 HZ2S3-2 PIC X VALUE X"D8".
02 HZ2S4-2 PIC X VALUE X"A7".
02 HZ2S5-2 PIC X VALUE X"D0".
01 HZ2T.
```

```
02 HZ2T1-1 PIC X.
02 HZ2T1-2 PIC X.
02 HZ2T2-1 PIC X.
02 HZ2T2-2 PIC X.
02 HZ2T3-1 PIC X.
02 HZ2T3-2 PIC X.
02 HZ2T4-1 PIC X.
02 HZ2T4-2 PIC X.
02 HZ2T5-1 PIC X.
02 HZ2T5-2 PIC X.
```

* 第三种加密方法的数据,将汉字内码转换为十进制值

```
01 HZ3S1 PIC 99999 VALUE 46023
COMP.
* 46023D=B3C7H=城
01 HZ3T.
02 HZ3T1 PIC 99999 COMP.
02 HZ3T2 PIC 99999 COMP.
02 HZ3T3 PIC 99999 COMP.
02 HZ3T4 PIC 99999 COMP.
02 HZ3T5 PIC 99999 COMP.
PROCEDURE DIVISION.
EXAM -1.
```

* 方法一,将打乱的汉字重新编排顺序输出

```
MOVE HZ1S1 TO HZ1T1.
MOVE HZ1S2 TO HZ1T2.
MOVE HZ1S3 TO HZ1T3.
MOVE HZ1S4 TO HZ1T4.
MOVE HZ1S5 TO HZ1T5.
DISPLAY HZ1T.
EXAM -2.
```

* 方法二,将打乱的汉字内码重新编排顺序输出

```
MOVE HZ2S1-1 TO HZ2T1-1.
MOVE HZ2S1-2 TO HZ2T1-2.
MOVE HZ2S2-1 TO HZ2T2-1.
MOVE HZ2S2-2 TO HZ2T2-2.
MOVE HZ2S3-1 TO HZ2T3-1.
MOVE HZ2S3-2 TO HZ2T3-2.
MOVE HZ2S4-1 TO HZ2T4-1.
MOVE HZ2S4-2 TO HZ2T4-2.
MOVE HZ2S5-1 TO HZ2T5-1.
MOVE HZ2S5-2 TO HZ2T5-2.
DISPLAY HZ2T.
EXAM -3.
```

* 方法三,将一个汉字的内码的十进制值加上另一个值,生成下一个汉字的十进制值,并输出该汉字。

```
MOVE HZ3S1 TO HZ3T1.
* HZ3S1=46023; 46023+1541=47564D
=B9CCH=固;
ADD HZ3S1 1541 GIVING HZ3T2.
ADD HZ3S1 7185 GIVING HZ3T3.
ADD HZ3S1 8928 GIVING HZ3T4.
ADD HZ3S1 7433 GIVING HZ3T5.
DISPLAY HZ3T.
STOP RUN.
```

实现程序反跟踪的一些技巧

□ 孙 燕

软件盗版是一个普遍问题,打起官司也费神费力,因此必要的加密手段对于程序员来说是应该掌握的。一般而言,加密分二部分:其一为防拷贝;其二是防跟踪。前者用来防止盗版;后者为了防止解密。两者相辅相成,缺一不可。笔者在多年的加解密实践中,总结出几种防跟踪的方法。这些方法也常见于各种加密软件中。如果在编程中对其灵活运用、合理搭配,能很好地起到软件保护作用。下面以汇编语言为基础加以说明。

一、破坏反汇编工具常用的一些中断地址

系统内存从0:0到0:3FF这1024个字节用来存放系统的中断处理程序入口地址,称为中断向量表。表中每四个字节对应一种中断类型(共有256个中断类型)。四个字节中,前二个字节是中断处理程序所在的段偏移,后二字节是段值。

反汇编工具常用来进行程序跟踪,它的功能能否正常运行取决于中断向量表前9个中断的入口地址是否正确。就拿常用的DEBUG软件来说,每一步的运行离不开INT3的正确入口地址,否则就会死机。另外,如果封住INT9的键盘中断,也一样造成不能运行的后果。程序员可以利用这些特点进行防跟踪设计。

下面几段程序说明具体的做法:

```
CLI
XOR AX,AX
MOV ES,AX
MOV DI,AX
MOV CX,8
REPZ
STOSW
```

上述程序段是将中断INT0~3的入口地址填零。如果修改CX中的值,可使更多的中断失效。

```
XOR AX,AX
MOV SS,AX
CS:
MOV [xx],SP
MOV SP,0024
```

上述程序段是通过将中断向量表作为程序运行的堆栈区使用,造成对中断地址的破坏。首先AX寄存器清零,送入堆栈段寄存器SS,将堆栈指针定为24。这样,随着堆栈操作的执行,中断INT8~0会受到不同程度的破坏。修改SP寄存器的值,会使破坏区域发生变化。程序中的[xx]用来保存原堆栈指针。

利用上述修改堆栈指针的程序,可以修改时钟中断INT8的入口地址,将其指向自编的一段程序。利用这段程序可以探测自己是否被跟踪,这种方法也是经常被采用的。下面举一个例子:

设SS=0,SP=24(即堆栈指向INT8的地址)

```
PUSH CS
MOV AX,xx
PUSH AX
```

上述程序语句将INT8入口地址置为CS:xx。接下来在CS:xx处加入自己的一段检查程序。比如进行时间控制,设定一个时间变量,利用INT8中断约每55ms发生一次,进行记数。正常运行到某段程序,时间是可知的,如果正被跟踪时间变量必然不准,这时可采取相应防治措施。

也可以替换键盘中断INT9的入口地址。一旦引发键盘中断,就进入自编的一段处理程序。在不需要进行键盘输入的地方,如果发生了键盘中断,必然有跟踪者出现,这时可封闭键盘,使跟踪者不能进行下去。封键盘中断有若干方法,下面是其中一种。

```
IN AL,21
```

```
OR AL,02
OUT 21,AL
```

PC I/O 端口21是中断屏蔽寄存器,其中第1位是控制键盘中断的。汇编语句OR AL,02是将第1位置1,禁止该中断发生。在跟踪状态下,键盘不能使用,形同死机。

上面简单介绍了利用修改中断向量表或对中断进行屏蔽的方法防跟踪。在实际编程中,这些方法一般均组合使用,而且互相联系,有的还重复多次。如果孤立地使用,是很容易被有经验的跟踪者避开或破坏掉的。另外要注意的是,在程序开始时,应保存好原来的中断向量表,等到程序结束时恢复原样,以免系统发生故障。

二、利用变码程序隐蔽原代码

另一种常用的防跟踪方法是对程序进行变码,使程序看起来面目全非,再加上多次变码,让跟踪者不知所从或疲不可耐而放弃跟踪。变码方法最常用的是异或,还可再加上其他运算。具体用法一般是将变码程序作为一段子程序调用。下面举例说明其用法:

设要变码的程序起始地址为CS:2E1,终止地址CS:3C9,变码子程序首址是CS:3FC:

```
mov BX,2E1
mov AX,3C9
call 3FC
jmp BX
3FC: push BX
3FD: mov CL,[BX]
inc BX
xor [BX],CL
cmp BX,AX
jl 3FD
pop BX
```

ret

上述变码子程序变码 CS:2E1到 CS:3C9这段程序。变码方法是以字节为单位,顺序往下进行。除首字节不变外,其他字节的值均是前一字节与本字节原值异或的结果,这是比较简单的一种变码。

下面是一个巧妙利用单步中断调用变码程序的例子,不但多次变码,而且加上了防程序改动的措施,变码方法也更复杂一些。

首先将变码程序首址送入 INT3单步中断地址中,这样一发出单步中断指令,便进行变码,而不是正常的单步中断,跟踪者无法运行跟踪程序。

调用程序:

```
XOR DX,DX
MOV CX,X1
y1:INT 3
LOOP y1
PUSH SI
MOV CX,x2
MOV DI,SI
INC DX
```

y2:INT 3

```
LOOP y2
DEC DX
POP SI
NOP
```

变码程序(INT3引发):

```
OR DX,DX
JNZ y3
LODSW
ADD BX,AX
IRET
y3:LODSW
XOR AX,BX
XCHG AH,AL
STOSW
IRET
```

注:在开始一系列变码之前,BX 有一初值。

从上面程序中可看出,不管是调用部分还是实际变码部分,均包含二层功能。第一层是将调用部分以字为单位计算累加和(不计进位),并与上次调用部分的累加和(在 BX 寄存器内)相加,存于 BX 中,标志位 DX=0,x1是调用程序

的字的个数。该层对应于调用部分的程序在语句 LOOP y1以上,对应于变码程序的部分到第一个 IRET。该层作用为保护程序不被改动。

一旦调用部分被修改,累加和必然变化,造成变码错误,导致死机。

第二层相对调用部分是在 LOOP y1以下的语句,相对变码程序是从 y3语句开始到程序完。该层功能主要完成对 x2个字长紧接调用部分(即 NOP 语句以后)后面程序的变码,标志位 DX=1。变码方法是取出一个字与 BX 的值异或,然后该字的两个字节交换,再送回原位。这种变码程序的调用与执行方法,其特点是变码出的新程序马上就被紧接着执行。倘若变码出的程序仍为一调用变码的程序(这很容易作),就形成了多次变码的格局,相信几千次甚至更多次的相似变码过程会使大多数跟踪者知难而退。

关于实现重复循环变码的方法还有多种,变码算法还可更复杂,加上四则运算,左移、右移等方式,使破译更加困难。

(上接第43页)

```
110 NEXT F
120 NEXT E
130 NEXT D
140 NEXT C
150 NEXT B
160 NEXT A
170 PRINT "N=";N
180 END
```

我们还可进一步选择 F 的初值去排除 S<100的情况,即选 F 的初值为 $100-A*50-B*20-C*10-D*5-E*2$ 。此时初值与终值相同,说明了在选定一组 A、B、C、D、E 的值使 $50A+20B+10C+5D+2E \leq 100$ 时,用一分币补足一元的方法有且只有一种。故不需设第六层循环,只需直接在第五层循环体内安排统计计数语句即可。这就少了一层循环,最内层循环的循环体只需执行 4562 次,在微机 PC—1500 上只需运行 2 分 45 秒,比最初的程序缩短运行时间 14924 倍。其程序如下:

```
5 REM PROGRAM4
10 LET N=0
20 FOR A=0 TO 2
30 FOR B=0 TO (100-A*50)/20
40 FOR C=0 TO (100-A*50-B*20)/10
50 FOR D=0 TO (100-A*50-B*20-C*10)/5
60 FOR E=0 TO (100-A*50-B*20-C*10-D*5)/2
70 LET N=N+1
80 NEXT E
90 NEXT D
100 NEXT C
110 NEXT B
120 NEXT A
130 PRINT "N=";N
140 END
```

由于第五层循环每执行一次,都仅给统计变量 N 加 1,所以只要把第五层循环的执行次数值 $INT((100-A*50-B*20-C*10-D*5)/2)+1$ 直接加到统计变量 N 中去,就可省去第五层循环,使程序最终变为四重循环程序。这样仅仅只需执行 343 次最内层循环的循环体,在微机 PC—1500 上不足 40 秒就可得到统计结果,比最初的程序缩短了运行时间 61560 倍。最后程序是:

```
5 REM PROGRAM5
10 LET N=0
20 FOR A=0 TO 2
30 FOR B=0 TO (100-A*50)/20
40 FOR C=0 TO (100-A*50-B*20)/10
50 FOR D=0 TO (100-A*50-B*20-C*10)/5
60 LET N=N+INT((100-A*50-B*20-C*10-D*5)/2)+1
70 NEXT D
80 NEXT C
90 NEXT B
100 NEXT A
110 PRINT "N=";N
120 END
```

由前面同一问题的五个不同程序,我们不难看出循环控制变量的三值选择和循环体的设计,对于循环程序,尤其是多重循环程序的运行效率产生着巨大的影响。同时又可以看出,三值选择与循环体的设计密切相关,要在深入分析理解具体问题的具体特点和要求的基础之上,尽可能做到恰如其分地选择好循环控制变量的初值、终值和步长值;尽可能充分利用循环控制变量的值和循环过程中的某些中间结果值设计好循环体诸语句。在保证正确性的前提下,最大限度地提高循环执行效率。

新颖的子目录加密机制

□金永涛 白焰

对于子目录的加密,已有很多人提出多种方法,如改变子目录的属性字节防止 DIR 查找,改变文件长度字节防止 PCTOOLS 等工具软件查找,还有改变文件起始簇号使子目录内容不能正常使用,尤其是修改簇号法,如果不知道此子目录的原来的正确簇号,就很难把子目录解密,加密效果很好,但这种作法的一个突出的缺点是:每次自己想进入子目录时也要进行重新恢复簇号的烦索的工作,并且一旦忘记原始簇号,将全部丢失子目录中的内容,针对上述问题,本人提出了一种动态的、自动的子目录加密方法。

下面先介绍一个与此功能有关的基本知识,然后再说明具体的实现过程。

一、动态加密原理

DOS 对于子目录的管理提供了四条命令,CHDIR、MKDIR、RMDIR、TREE,用于改变当前目录、建立目录、删除目录和显示目录,对应的 DOS 系统功能调用也提供了四个子功能,即 3BH:置当前目录、39H:建立子目录、3AH:删除子目录、47H:取当前目录。当改变目录时,DOS 的 CHDIR 就调用 DOS 功能调用的 3BH 号功能,此时 DS:DX 处为表示该子目录的 ASCII 字符串。我们可以首先修改要加密子目录的起始簇号、属性和长度域,然后修改 DOSINT 21H 系统调用中断,并驻留内存,监视 3BH 号子功能,根据当前 DS:DX 处的 ASCII 字符串,判断当前子目录是否为指定的子目录,如果是就恢复正确的起始簇号,如果不是,就置簇号为其他值,这样,就完成了子目录的动态加、解密过程。

二、文件目录表的搜索

为了编制通用性的程序,对磁盘上任意存在的子目录进行加密,必须采取搜索磁盘目录表的办法,找到要加密子

目录所在磁盘的具体位置,即逻辑扇区号或物理扇区号,修改目录表的有关字节,并把原起始簇号存入一安全区域如 CMOS 的未用字节单元之中,每一个磁盘尤其是硬盘的目录区起始地址和长度都不相同,可以通过磁盘的 I/O 参数表 BPB 提供的有关参数来计算,BPB 表的具体结构已有很多资料介绍过,此处不再重复,本文后面的程序中利用 DOS 绝对磁盘读 INT 25H 和绝对磁盘写 INT 26H,对 C 盘的根目录进行搜索,寻找子目录 MYSUB,找到后修改相应的字节域。如果采用 BIOS 的磁盘服务功能进行搜索,请修改入口参数为物理扇区号(程序中 BPB 为磁盘 I/O 参数表的入口地址)。

三、起始簇号的保存

方便而又安全地保存子目录的原来起始簇号,是保证该子目录安全的关键,本例中采取了利用 CMOS 未用字节的方法,起始簇号为一个字长,占用两个 CMOS 字节。CMOS 是一个 64 字节的具有掉电保护功能的 RAM 存储器,用来保存有关机器的硬件配置等重要数据,其中有很多未用空闲字节,如 03 和 05 号单元等,可以用来长期保存一些有用数据。对于 CMOS 数据的读写是通过 70H 和 71H 两个端口进行的,具体请见后面程序实例。对于子目录首次加密时,把起始簇号写入 CMOS 中;解密时,再从 CMOS 中读出并写入子目录的相应位置。

四、逻辑扇区到物理扇区地址的转换

由于 DOS 的不可重入性,在修改 INT 21H 之后,新的 INT 21H 模块不能采用 DOS 的绝对磁盘读写功能,即 INT 25H 和 INT 26H,而必须采用 BIOS 的磁盘服务功能 INT13H,在搜索子目录

地址时,如果采取了 DOS 的磁盘功能,得到的是逻辑地址,之后必须转换为物理地址并传送给新的 INT 21H 模块。转换的方法,也可以利用磁盘 I/O 参数表 BPB 来实现,为节省篇幅,此处只给出转换公式,程序省略。

扇区号 = 逻辑扇区号 MOD 每道扇数 + 1

磁头号 = (逻辑扇区号 / 每道扇数) MOD 磁头数

磁道号 = 逻辑扇区号 / 每道扇数 / 磁头数

式中 MOD 为取模即除后取余数。

五、实现过程

以上介绍了加密子目录的有关知识,对于不同的应用场合,可以采取不同的方法,如果编制单独的、只对一个固定的子目录加密的程序,可以省去地址转换过程,利用 NU 等工具软件,可以方便地找出被加密子目录的物理地址,填入程序的相应位置即可。如果要编制通用软件,可以全部由程序自动实现,即把上述各程序有机地结合在一起,首先搜索目录表,定位被加密子目录的地址,修改目录项有关字节,并把起始簇号送入 CMOS 之中,然后把物理地址存入几个内存单元,以便新的 INT 21H 调用,最后修改 INT 21H 使之指向新的模块即可。在新的 INT 21H 模块中,监视 3BH 号子功能,判断 DS:DX 处的字符串,按要求进行加密和解密。

下面给出了一个完整的程序实例,此程序为节省程序段,仅以加密 C 盘根目录中的 MYSUB 子目录为例,并且省略了逻辑地址到物理地址的转换过程,采取了直接把该子目录所在物理地址送入程序中变量的方法。另外本程序加密时填入起始簇号位置的值为 0000,所以在加密状态进入该子目录时显示为根目录内容。

六、例举程序使用方法

例举程序上面已经提到,它本身是一个完整的程序,但是在不同的盘上运行时,需要把相应的子目录的真实物理地址分别填入到 CT SQ CD 变量处,本程序按 .COM 文件格式编写,经汇编、链接并转换为 .COM 文件之后即可运行。首先在 C 盘根目录中建立 MYSUB 子目录,然后运行该程序,运行时首先查找该子目录,并检查是不已经加密,首次运行时程序将修改该目录项的属性、长度和首簇号字节,并把起始簇号存入 CMOS 的 03 和 05 号单元中,如果已经加密则直

接转到驻留程序段修改 INT 21H 中断,驻留并返回 DOS。此时若进入 MYSUB 子目录,将自动解密,退出或进入其他子目录,自动进行加密,并且有信息提示。关机或热启动前请退出该子目录,使之处于加密状态。需要说明的一点是,当处于解密状态时,有时列目录仍为根目录内容,此时只要运行一个不存在的文件,令计算机重读目录表,再列目录时,就可恢复正常的目录内容。当不运行该程序时,用 DIR 或 PCTOOLS 等工具软件均不能发现该目录,即使进入该子目录也是显示根目录中的内容。只要保存好此程序不被他人运行,你的目录始终处于

加密状态,自己使用该子目录时,只要运行一下你的程序即可按正常的操作过程存取你的子目录。

值得注意的是:

1、当你调试程序时一定要设置正确的物理地址,不要使用源程序中的数值,否则将造成严重后果。

2、运用此方法加密子目录后,存在着一个危险,因为修改了起始簇号,被加密子目录空间变为丢失簇,所以 DOS 命令 CHKDSK/F 能够破坏磁盘空间,解决的办法是修改文件分配表 FAT,使之成为坏簇或保留簇即可,方法从略。

加密 MYSUB 子目录程序举例:

```

;MYSUB.ASM(MYSUB.COM)
;1993.12.1
code segment
    assume cs:code,ds:code
    org 100h
start: jmp init
msub1 db '\mysub\
msub2 db 'C:\MYSUB\
;设置读盘缓冲区
fdtbuff db 11 dup(0)
bpb db 501 dup(0)
bpb13 dw 0 ;保存每扇磁道数
bpb15 dw 0 ;保存磁头数
sq db 09h ;以下为物理地址
ct db 02h ;请按实际情况填写
cd db 25h
dat dw 0 ;此目录在扇区中偏移
dat1 dw 0 ;保存逻辑地址
ddd db 0 ;目录表占扇区数
jsz db 0 ;读盘计数
er db '找不到该子目录$'
er1 db '读盘出错$'
bz db 0 ;加密标志
old21 dd 0
msg1 db '已经加密!!!!!!$'
msg2 db '已经解密$'
disp macro line
    mov ah,9
    mov dx,offset line
endm
new21 proc far
push ax
push bx
push cx
push dx
push bp
push si
push di
push es
push ds
cmp ah,3bh
jz jjxx
jmp old
jjxx:
;判断是否进入此子目录
push cs
pop es
mov si,dx
mov di,offset cs:msub1

```

```

mov cx,5
repe cmpsb
jz jml
mov di,offset cs:msub2
mov cx,5
mov si,dx
repe cmpsb
jnz next
jml: cmp bz,0
    jnz jm2
    jmp old
;解密过程
jm2: mov ax,cs
    mov ds,ax
    mov dx,70h
    mov al,5
    out dx,al
    inc dx
    in al,dx
    mov ah,al
    mov dx,70h
    mov al,3
    out dx,al
    inc dx
    in al,dx
    mov si,dat
    mov word ptr [si+26],ax
    mov dl,80h
    mov ch,cd
    mov cl,sq
    mov dh,ct
    mov al,1
    mov bx,offset fdtbuff
    mov ah,03
    int 13h
    mov bz,0
    disp msg2
    jmp old
next:
;加密过程
    cmp bz,1
    jz old
    push cs
    pop ds
    mov si,dat
    mov word ptr [si+26],0
    mov dl,80h
    mov ch,cd
    mov cl,sq

```

```

mov dh,ct
mov al,1
mov bx,offset fdtbuff
mov ah,03
int 13h
mov bz,1
disp msg1
old: pop ds
    pop es
    pop di
    pop si
    pop bp
    pop dx
    pop cx
    pop bx
    pop ax
    jmp cs:old21
new21 endp
;安装过程
init: push cs
    pop ds
    mov al,2 ;读磁盘 BPB 表
    mov cx,1
    mov dx,0
    mov bx,offset fdtbuff
    int 25h
    mov ax,word ptr [bpb+13]
    mov bpb13,ax
    mov ax,word ptr [bpb+15]
    mov bpb15,ax
    ;计算目录表起始地址
    mov al,bpb+5
    cbw
    mul word ptr [bpb+11]
    inc ax
    mov dx,ax
    ;读目录表并查此目录
    push dx
    mov ax,word ptr [bpb+6]
    mov bl,32
    mul bl
    div word ptr bpb
    mov ddd,al
    pop dx
    mov jsz,0
lop1: inc jsz
    mov ah,ddd
    cmp jsz,ah
    ja error

```

磁盘损坏时文件的一种恢复方法

□张存新

磁盘的根目录区和文件分配表受到病毒破坏后,常引起存储于磁盘上的文件不能用DOS命令读写,或读写结果不正确、甚至文件名从磁盘目录区中消失的问题。此时,要有效地恢复文件是困难的,有的用户常采用确定文件位置后重填目录项和文件分配表的方法来恢复文件,但这种方法操作复杂,既繁琐又费时。笔者在实践中摸索出了一种简便方法。该方法利用文本文件的内容可以在屏幕上显示的特点,编制了一个Turbo C实用程序,可方便用于文本文件的恢复,现将编制原理介绍如下:

首先,利用absread()函数读出磁盘扇区的内容,当读出的内容为可打印字符时,就用printf()函数以字符格式显示,当读出的内容为回车符时就换行,当读出的内容为汉字字符时,则显示汉字,当读出的内容为控制字符及其他不可打印字符时则显示一点。每一次显示一个扇区,用户可根据显示的内容来判断,用(y/n)选择存盘,选择后再显示下一扇区的内容。如此,依次显示和存储即可将位于不同扇区的文本文件的内容存储到指

定的文件中。在进行存储时,为避免坏盘扇区的有用内容被存储的文件所覆盖,需将文件存储于另一磁盘上。具体做法是:不在坏盘上运行本程序。例如:要恢复A盘的内容,需在B盘或硬盘上运行本程序。

本文的源程序已在DOS3.31、AST386微机上调试通过。

源程序清单如下:

```
#include "dos.h"
#include "stdio.h"
#include "ctype.h"
main(int argc, char * argv[])
{FILE * fp;
int drive, numsects, sectnum, i, k;
unsigned char buf[512];
char ch;
printf("输入驱动器号(0:A,1:B,2:C):\n");
scanf("%d", &drive);
printf("请输入起始扇区号\n");
scanf("%d", &sectnum);
printf("请输入查看的扇区数\n");
scanf("%d", &numsects);
if(argc < 2)
fp = fopen("filename. %%%", "a");
```

```
/* 不指定文件名,默认为 filename. %%% */
else
fp = fopen(argv[1], "a");
if(fp == NULL)
{printf("不能打开文件\n"); exit(0);}
for(k=0; k<numsects; k++)
{absread(drive, 1, sectnum+k, buf);
for(i=0; i<512; i++)
{if(isprint(buf[i])) printf("%c", buf[i]);
/* 是可打印字符则显示 */
else if(buf[i] == 13) printf("\n");
else if(buf[i] > 160)
{printf("%c%c", buf[i], buf[i+1]);
i++;}
else printf(".");}
printf("\n");
printf("你需要将第%d个扇区的内容写入文件吗?(用y或n选择)\n", k+1);
do{ch = bioskey(0); /* 等待用户选择 */
if(ch == 'y') fwrite(buf, sizeof(char),
512, fp);
}while(ch != 'y' & ch != 'n');
fclose(fp);
}
```

```
mov al, 2
mov cx, 1
mov bx, offset fdtbuff
int 25h
jc error1
mov bx, offset fdtbuff
mov ax, 0
mov di, bx
lop: mov si, offset msub2
add si, 3
mov cx, 5
repe
cmpsb
jz jm
inc ax
cmp ax, 16
jnz lop2
inc dx
jmp lop1
lop2: add bx, 32
mov di, bx
jmp lop
error1:
mov dx, offset er1
```

```
mov ah, 09
int 21h
jmp jss
error: mov dx, offset er
mov ah, 09
int 21h
jss: mov ah, 4ch
int 21h
jm: mov dat1, dx
mov dat, bx
mov si, bx
mov bx, word ptr [si+26]
cmp bx, 0 ; 判断是否已经加密
jz zl
mov dx, 70h; 首次加密保存簇号
mov al, 05h
out dx, al
inc dx
mov al, bh
out dx, al
mov dx, 70h
mov al, 03
out dx, al
inc dx
```

```
mov al, bl
out dx, al
mov word ptr [si+11], 17h
mov word ptr [si+31], 0ffh
mov word ptr [si+26], 0
mov al, 2
mov cx, 1
mov dx, dat1
mov bx, offset fdtbuff
int 26h
disp msg1
zl: mov bz, 1
mov ax, 3521h
int 21h
mov word ptr old21, bx
mov word ptr old21+2, es
mov dx, offset new21
mov ax, 2521h
int 21h
mov dx, offset init
inc dx
int 27h
code ends
end start
```

CPU“逆行”程序加密法

□张建华

众所周知,CPU 在执行指令时都是按地址从低到高的顺序进行的(跳转指令除外),人们在读程序时也是如此。让 CPU“逆行”应该是一个有趣的设想,特别是在软件的加密当中如能这样,其加密效果将极为理想,在“逆行”当中再配合其他一些加密思想则效果会更佳。

运用单步中断,将使得让 CPU“逆行”的设想成为可能。这里所谓的“逆行”就是让 CPU 从一段指令“倒放”程序(程序中指令从高地址向低地址排放且同一指令的几个字节也按此法则排列)的高地址端一直向低地址方向逐条执行,直到“逆行”结束标志出现为止。我们可以这样设想,CPU 在执行完一条指令之后,产生一个单步中断,利用该中断将一条指令复原,并执行它。执行完此指令后再产生单步中断,通过两次中断的返回地址计算出该指令长度,根据该长度向低地址方向找到下一条指令的起始地址,然后再复原,再执行,一直到标志出现时退出“逆行”进入正常运行状态。

为了实现该设想,你必须:编写自己

的单步中断例行程序;把状态寄存器中的单步标志设置为1(即 FLAG 的第8位置1)。该位一旦被设置,CPU 即在执行完下一条指令之后自动产生单步中断,进入你所写的中断例行服务程序。为了在程序的“逆行”过程中不至于产生错误,该程序段中不能使用循环和回向跳转指令(因每执行一条指令在复原过程中都把刚执行过的那条指令覆盖),且当遇到中断调用指令(INT)时要作些特殊的处理,此时应把它改为顺序执行,同时“逆行”过程中还应关闭硬中断,防止在硬中断中产生“逆行”的副作用。该设想既防止解密者的静态跟踪,同时由于逐条解密执行,即使动态跟踪也将极为困难,又因为所有的解密思想都放于单步中断例行服务程序之中,这使得在改写单步中断的调试程序(如 DEBUG 等)的参与下无法运行原逆向程序段的指令。该“逆行”例程让跟踪者苦恼不堪,而加密者对此却轻而易举。

下面对本文所附“逆行”例程作一些说明。

由于8088系列芯片其指令最长为6个字节,故在逐条“逆行”恢复时一次只恢复六字节指令代码,此时必能产生一条完整的指令。在恢复过程中同时破坏了最近被 CPU 执行过的指令。此例中用 00H 作为“逆行”结束标志,且用循环移位作为指令代码加密的辅助手段。在遇到 CDH(即 INT 指令)代码时必须将其改为顺序执行方式,并在被加密程序段 INT 指令后用11条指令再将其改成“逆行”方式。注意在遇到 CDH 后的恢复过程中,由于要把 CPU 改为顺序执行方式,故此时将把 INT 指令及其后的那11条指令一同恢复并执行。因每次“逆行”程序入口处的那条指令之前不会产生单步中断,故第一条指令应顺序放置并为明文(本例用 NOP)。也可以把本例中 New_int1 例程单独提出并略作修改形成内存驻留程序且先于加密程序运行(只需做一个简单的批处理),此法也同样有效。本例程在 AST 286、MS-DOS 3.3下用 MASM5.0 汇编、连接并转换成 .COM 运行通过。

源程序清单如下:

```
code segment
org 100h
assume cs:code
start: jmp begin
start_addr dw 0 ;设置单步中断后 CPU 执行的起始地址
old_int1 dw 0,0 ;原 INT 1 中断向量
message db "Now,Let's begin..." ,0dh,0ah
db "Press any key to return",0dh,0ah," $"
begin: push cs
pop ds
push cs
pop es
mov ax,3501h
int 21h ;取1号中断向量
mov old_int1,bx
push ax ;压入状态寄存器值
push cs ;压入 CS 值
mov ax,start_addr
push ax ;压入 IP 值
```

```
mov ax,es
mov old_int1[2],ax; 保存1号中断向量
pushf
pop ax
and ax,0feffh
push ax
popf ;清除单步中断位
mov dx,offset new_int1
mov ax,2501h
int 21h ;设置新1号中断向量
lea ax,pro_exit-1
mov start_addr,ax;保存设置单步中断后 IP 地址
pushf
pop ax
or ax,0100h ;设置单步中断位
iret ;转入逆行程序执行
jmp pro_end
```

stand_begin;
;以下为密码程序代码

```
db 00h,90h,0e7h,28h,80h,0b9h
db 82h,24h,80h,0d4h,96h,80h,0d5h,03h
db 0c6h,07h,28h,80h,00h,86h,2ch,4eh
db 0bh,0e6h,60h,99h,90h,0e7h,50h,24h
db 80h,0b9h,82h,80h,90h,96h,80h,0d5h
db 03h,0c6h,07h,28h,80h,00h,86h,2ch
db 4eh,90h,0e6h,84h,5ah,80h,84h,5dh
db 7dh,90h,90h
```

；以下是源程序代码

```
；      nop      ；该指令应为明码
；      cli      ；以下至第一个 IRET 止程序代
码可根据需要
```

```
；      mov dx,offset message ；进行再加密,本例用 ROR 移
；      mov ah,9      ；一位加密
；      int 21h      ；显示信息
；      pushf      ；以下至 IRET 为顺序执行(因在遇 INT 指
令时禁止了单步中断)
```

```
；      pop ax
；      or ax,0100h
；      push ax
；      push cs
；      lea ax,stand_end
；      sub ax,offset go_one
；      add ax,offset stand_begin
；      dec ax
；      push ax ；改 IP 为 go_one 偏移值
；      iret
```

```
；go_one: nop
；      xor ax,ax
；      int 16h
；      pushf
；      pop ax
；      or ax,0100h
；      push ax
；      push cs
；      lea ax,stand_end
；      sub ax,offset go_two
；      dec ax
；      add ax,offset stand_begin
；      push ax
；      iret
```

```
；go_two: nop
；      db 00h ；“逆行”结束标志
```

stand_end:

pro_exit:

```
pro_end:      db 30h dup(20h) ；预留缓冲区用于复原指令
```

```
cld
mov ax,0
mov es,ax
push cs
pop ds
mov di,4
mov si,offset old_int1
lodsw
stosw
lodsw
stosw
sti
mov ax,4c00h
int 21h
```

new_int1: ；以下为 INT1 单步中断例行服务程序

```
cli
push bp
mov bp,sp
push ax
push bx
push cx
push dx
```

```
push si
push di
push es
push ds
push cs
pop ds
mov ax,[bp+4] ；取逆行程序段址
mov es,ax
mov ax,[bp+2] ；取下一条指令偏移地址
sub ax,start_addr
sub start_addr,ax
mov ax,start_addr ；计算下条指令返回地址
mov [bp+2],ax ；送堆栈
push es
pop ds
mov si,ax
mov di,ax
mov al,[si] ；取下条指令首代码
dec si
cld
or al,al
jz int1_end ；“逆行”结束标志出现否?
mov cx,5
rol al,1 ；解码
cmp al,0cdh
jnz int1_begin ；INT 指令出现否?
mov bx,[bp+6] ；如出现则恢复 CPU 为顺序执行
and bx,0feffh
mov [bp+6],bx
mov cx,20h
int1_begin: ；进行恢复
stosb
mov al,[si]
rol al,1
dec si
loop int1_begin
jmp int1_exit
int1_end:
mov ax,[bp+6]
and ax,0feffh
mov [bp+6],ax
mov ax,[bp+2]
sub ax,3
mov [bp+2],ax ；IP 指向 jmp near pro_end 指令
int1_exit:
pop ds
pop es
pop di
pop si
pop dx
pop cx
pop bx
pop ax
pop bp
sti
iret
code
ends
end start
```

300元可省一台打印机

清华大学科学馆

邮编:100084
电话:2594866
联系人:魏宝英

SND 系列打印机共享器

Xenix 系统下 tar 盘文件的恢复

□张 荣



我们在 Xenix 操作系统下工作时,常使用 tar 命令把自己重要的数据从硬盘中备份到软盘上,有时数据量很大,往往需要用多张高密盘连续备份。但有时其中某张盘偶尔会出现问题,这时如果不了解 tar 盘的格式,往往会把自己弄得手足无措。笔者在工作中,曾遇到这样一个问题,以前为了保存数据库和程序而 tar 出来的三张高密盘,其中第二张盘被“火炬”病毒感染了,并且有一个大数据库文件跨在第一张和第二张盘上。当用 tar 命令往硬盘上释放时,到第二张盘,系统提示 Directory checksum error 信息后便退到 \$ 提示符下,后经笔者摸索和分析,终于将盘中的数据库文件挽救出来。现将 tar 盘的格式及文件的恢复过程介绍如下:

tar 盘的格式:盘中文件从磁盘的0道0面1扇区开始顺序存放,每个文件的具体内容前有一个扇区的文件信息头,其内容首先是全路径文件名,然后为文件属性、文件大小及目录校验和如图1。紧接着文件头便是文件的具体内容,直到下一个文件信息头出现。如果文件很大跨在两张磁盘上,那么两部分文件都有文件信息头,并在信息头的后部有1 of 2或2 of 2字样,以表明是文件的第一部分和第二部分。

文件的恢复:假设图一中的文件信息头被病毒覆盖了,我们可用 Norton 中的 NDD 或 DOS 中的 DEBUG,把盘中从0道0面2扇区到下一文件信息头前的所有扇区内容写到一个

DOS 文件中,再用 Xenix 的 doscp 命令把此文件拷贝到 Xenix 系统中,就完成了文件的恢复工作。使用 NDD 很容易完成此过程,而使用 DEBUG,则相对繁琐些,而且必须用 INT 13调用。程序如下:

```
C>debug
-a100
2A27:0100 mov ax,0214;读20个扇区(14H)
2A27:0103 mov bx,200 ;读到 ES:200处
2A27:0106 mov cx,02
2A27:0109 mov dx,00 ;从0道0面2扇区开始
2A27:010C int 13 ;绝对磁盘读调用
2A27:010E int 20
2A27:0110
-g=100
-rcx
:2800 ;20个扇区,每个扇区512字节
-nbc az.dbf
-w200
-q
```

这种方法不仅适用于 Xenix 系统,经笔者试验证明也适用于 Unix 系统,有兴趣的读者不妨上机一试。

```
0B59:0200 62 63 61 7A .2E 64 62 66-00 00 00 00 00 00 00 00 00 bcaz.dbf.....
0B59:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0B59:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0B59:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0B59:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0B59:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0B59:0260 00 00 00 00 20 20 20 36-34 34 20 00 20 20 20 33 .... 644.3.
0B59:0270 31 31 20 00 20 20 20 20-36 32 20 00 20 20 20 20 11. 62
0B59:0280 20 20 20 33 30 30 30 20-20 35 34 35 34 36 33 33 3000 5454633.
0B59:0290 32 33 34 20 20 20 35 34-37 37 00 20 00 00 00 00 234 5477. ....
0B59:02A0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
.....
(文件的具体内容)
0B59:0400 03 55 06 0D 01 00 00 00-82 03 22 01 00 00 00 00 .U.....".....
0B59:0410 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0B59:0420 44 45 48 00 00 00 00 00-00 00 00 43 07 00 85 76 DEH.....C...V
0B59:0430 07 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
```

图1 文件信息头

运用 FoxBASE+ 实现 Windows 的平滑技术

□ 赵 军

随着计算机技术的普及,我们在开发管理信息系统时,总是想设计出好的用户界面。如 PCTOOLS、Windows 等软件,不但具有丰富的菜单功能,而且可以移动窗口在任意屏幕位置显示,本人受 Windows 的启发,在 FoxBASE+ 下开发了一套窗口平滑程序。

本程序可实现窗口动态弹出,用 ↑、→、↓、← 键屏幕范围内平滑移动窗口,选择窗口的显示位置,用 END 键结束选择,并进入主菜单,进行选单操作。退出主菜单后,可继续选择程序清单如下:

```
C>type win.prg
* 运用 FoxBASE+ 实现 Windows 平滑技术
* 编程人:赵军
* 编程日期:1993.12.6
set talk off
set escape off
set scor off
set stat off
@0,0 say chr(14)+"r2]"
set color to /0
@0,0 clear to 24,79
c=1
do while c<=15
r1=21-c
r2=55-2*c
r3=24-c/2
r4=67-c/2
m1=20-c
m2=53-2*c
m3=23-c/2
m4=65-c/2
h=r3-r1
w=r4-r2
set color to /1
@r1,r2 clea to r3,r4
set color to /4
@m1,m2 clear to m3,m4
set color to 10
clear
c=c+1
enddo
set color to /1+
@3,2 clear to 23,77
do while .t.
clear typeahead
@ 0,0 say
chr(14)+"D20,420C12B600,360D30,410C14B580,340D40,
400C10B560,320]"
set color to /0
```

窗口位置,用 ESC 键退出系统。本程序的字幕提示,采用的是 2.13H 汉字系统的特显功能。如对本程序简单改动,可加强其功能,并可直接作为其他管理信息系统的显示界面。本程序的运行环境为 2.13H 汉字系统,并在 FoxBASE+2.0、2.1 版本中调试通过。

其中参数如下:r1、r2、r3、r4 为阴影窗口坐标,m1、m2、m3、m4 为前景窗口坐标,clear typeahead 为键盘缓冲清除命令,使键盘缓冲区为空,保证 inkey() 函数的使用。

```
@ r1,r2 clea to h+r1,w+r2
set color to 2/4
@m1,m2 clea to h+m1-2,w+m2
set color to 1/2
@h+m1-2,m2 clea to h+m1-2,w+m2
@0,0 say chr(14)+"[175|6#2*4@P 世界名车]"
set color to 2/4
@m1+2,m2+3 say" 选购汽车指南 V1.0 "
@m1+5,m2+3 say" ADDR:胜利油田胜利采油厂二矿"
@m1+6,m2+3 say" TEL:242069(05461)"
set color to 1/2
@ h+m1-2,m2+1 say" ↑→↓←→选择窗口的位置"
@ h+m1-1,m2+1 say" END →>进入主菜单"
@ h+m1,m2+1 say" ESC →>退出系统"
m=0
do while m=0
m=inkey()
enddo
set color to /1
@m1,m2 clear to h+r1,w+r2
endif
case m=19
if m2>1
r2=r2-2
m2=m2-2
endif
case m=5
if m1>0
r1=r1-1
m1=m1-1
endif
case m=24
if r1+h<24
r1=r1+1
m1=m1+1
endif
case m=6
set proc to win_1.prg
do win1
close proc
```


DOS 下 FoxPro 2.5

通用多窗口处理程序设计

□刘耀东 王建忠

DOS 下应用 FoxPro 2.5 进行管理信息处理系统开发时,通常要进行数据查询系统的设计工作。FoxPro 2.5 系统本身没有提供同时对多个数据库进行查询的命令。本文提出一种可同时对多个数据库进行查询的程序设计方法,利用本方法可大幅度提高程序设计的效率,快速生成用户界面完美、系统功能完备的数据查询系统。

本文提出的通用数据查询程序具有如下特点:

1. 与数据库内容及结构无关。
2. 字段数目不受限制(1 至 255 个)。
3. 数据类型可分为字符、数值、逻辑、日期型数据。
4. 数据宽度不受限制,可为大字段(数据宽度大于显示窗

口区域宽)。对于大字段,程序自动进行分屏显示算法处理。

5. 查询方式多种多样,可左右移动单个字段,上下移动一条记录,左右整屏移动多个字段,上下整屏移动多条记录。

6. 程序自动显示字段名称。

7. 可同时查询最多 224 个数据库。

8. 自动记忆已查询数据库的当前起始记录序号。

9. 自动统计每个数据库的记录总数。

10. 多窗口重叠显示界面完美。

11. 运行速度快。

欲运行本程序,须首先建立一管理库 WINDOW.DBF,该库结构如下:

```
case m=27
exit
endcase
enddo
set color to /0
@ 0,0 clea to 2,79
@ 0,0 say chr(14)+"[175|24#2*4 @B 欢迎再次咨询]"
@ 0,0 say chr(14)+"[190|140#6*2@1 再见]"
@ 0,0 say chr(14)+"rl]"
set color to 7/0

C> type win_1.prg
proce win1
do while .t.
set color to /0
@ rl,r2, clear to h+rl,w+r2
set color to 2/4
@ m1,m2 clea to h+m1, w+m2
set color to /0
@ 0,1 clear to 2,79
@ 0,0 say chr(14)+"[120|20#3*4@B 选购汽车指南 V1.
0]"
set color to 2/3
@ m1+3,m2+5 prom chr(30)+" 返回 窗 口"
@ m1+4,m2+5 prom chr(31)+" 林肯 丰田"
@ m1+5,m2+5 prom chr(31)+" 奔驰 凌志"
@ m1+6,m2+5 prom chr(31)+" 劳斯莱斯 捷豹"
@ m1+7,m2+5 prom chr(31)+" 凯迪拉克 大字"
@ m1+8,m2+5 prom chr(31)+" 尼桑 皇冠"
menu to p1
```

```
do case
case p1=1
set color to /0
@ 0,0 clear to 2,79
retu
case p1=2
set color to /6
@ m1,m2 clea to h+m1,w+m2
set color to /2
@ m1,m2 clea to h+m1-3, w+m2
set color to 1/2
@ m1+2,m2+1 say "林肯: 美国 1945 年,型号:600,总统级"
@ m1+3,m2+1 say " $ 120000
@ m1+5,m2+1 say " 丰田:日本 1962 年,型号:3.1C,豪华级"
@ m1+6,m2+1 say " $ 30000
wait ""
case p1=3
set color to /6
@ m1,m2 clear to h+m1,w+m2
set color t /2
@ m1,m2 clea to h+m1-3,w+m2
set color to 1/2
@ m1+2, m2+1 say" 奔驰:德国 1942 年,型号:600,总统级"
@ m1+3,m2+1 say" $ 89000
@ m1+5,m2+1 say" 凌志:日本 1983 年,型号:2.0C,普及型"
@ m1+6,m2+1 say" $ 20000
wait ""
endca
enddo
```

字段序号	字段名称	字段类型	字段宽度	小数位数
1	COLOR1	Character	20	0
2	COLOR2	Character	20	0
3	XSHS	Numeric	2	0
4	ROW1	Numeric	2	0
5	WID	Numeric	2	0
6	WJ	Character	100	0
7	COL0	Numeric	2	0

本程序在 MS-DOS 6.0 直接写屏汉字系统及 FoxPro 2.5 for DOS 增强版下运行。

源程序清单:

```

* : * * * * *
* : AUTHOR: LIU YAO DONG && WANG JIAN ZONG
* : PROCS & FNCTS: TYCXXT
* :
* : : DISPREC
* : : LEFTMOVE
* : : RIGHTMOVE
* : : DISPWINDOW
* : * * * * *
SET TALK OFF
SET STATUS OFF
SET SCOR OFF
SET SAFE OFF
SET PROC TO TYCX
SET ESCAPE ON
CLOSE DATA
SELE 225
USE WINDOW && 打开数据库管理文件
DNUM=RECCOUNT() && 计算欲查询的数据库个数
&& 定义数据库查询指针变量数组
DIME FNUM1(DNUM),FNUM2(DNUM),MM1(DNUM),
MM2(DNUM),N3(DNUM),WIDTH(DNUM)
DIME XHBL(DNUM),XHBL1(DNUM),KZBL(DNUM),RR1
(DNUM),RR2(DNUM),REC1(DNUM)
DIME XSHS(DNUM),XSHS0(DNUM),S1(DNUM),S2(DNUM),
S3(DNUM),S4(DNUM)
DIME ROW1(DNUM),YWJ(DNUM),CL0(DNUM),NM(DNUM)
PUBLIC KEY2,IP
IP=1
IF RECCOUNT()>224
RETURN
ENDIF
GO TOP
DO WHILE .NOT. EOF()
REC=RECNO()
ALIS="AA"+STR(REC,IIF(REC<10,1,IIF(REC<100,2,
3)))
&& 读入数据库的控制参数变量
CL0(REC)=COL0
XSHS0(REC)=XSHS
WIDTH(REC)=WID
S1(REC)=COLOR1
S2(REC)=COLOR2
ROW1(REC)=ROW1
YWJ(REC)=TRIM(LTRIM(WJ))
IF .NOT. FILE(WJ+'.DBF')
RETURN
ENDIF
BB=STR(REC,IIF(REC<10,1,IIF(REC<100,2,3)))
&& 计算数据库的结构信息
SELE &BB
YWJ0=YWJ(REC)
USE &YWJ0 ALIAS &ALIS
NM(REC)=FCOUNT()+1
GO TOP
&& 定义数据库库名、长度、坐标、数据类型、小数位数数组名标识
FDNAME="FDNAME"+BB
FDLEN="FDLEN"+BB
XRAY="XRAY"+BB
FDTYPE="FDTYPE"+BB

```

```

FDDEC="FDDEC"+BB
&& 定义数据库结构动态数组
=AFIELDS(DBFSTR)
NM0=NM(REC)
DIME &FDNAME(NM0-1),&FDLEN(NM0-1),&XRAY(NM0),
&FDTYPE(NM0-1),&FDDEC(NM0-1)
STORE 0 TO &XRAY(REC)
&XRAY(1)=CL0(REC)+INT(NM(REC)/10)+6
I=1
GO 1
DO WHILE I<=NM0-1
&FDNAME(I)=DBFSTR(I,1)
&FDLEN(I)=DBFSTR(I,3)
IF &FDLEN(I)<LEN(LTRIM(TRIM(FIELD(I))))
&FDLEN(I)=LEN(LTRIM(TRIM(FIELD(I))))
ENDIF
&FDLEN(I)=IIF(MOD(&FDLEN(I),2)#0,&FDLEN(I)+1,
&FDLEN(I))
IF DBFSTR(I,2)='L'.AND.&FDLEN(I)<4
&FDLEN(I)=4
ENDIF
IF DBFSTR(I,2)='D'.AND.&FDLEN(I)<8
&FDLEN(I)=8
ENDIF
&FDTYPE(I)=DBFSTR(I,2)
&FDDEC(I)=DBFSTR(I,4)
&XRAY(I+1)=&FDLEN(I)+&XRAY(I)+2
I=I+1
ENDDO
GO TOP
XSHS0(REC)=IIF(RECCOUNT()<XSHS0(REC),RECCOUNT
(),XSHS0(REC))
XSHS0(REC)=IIF(XSHS0(REC)<24,XSHS0(REC),23)
GO TOP
REC1(REC)=RECCOUNT()
RR2(REC)=XSHS0(REC)
GO TOP
XSHS(REC)=XSHS0(REC)
WIDTH(REC)=IIF(MOD(WIDTH(REC),2)#0,WIDTH(REC)+
13,WIDTH(REC)+12)+CL0(REC)
DIS=XSHS(REC)
SELE 225
SKIP
ENDDO
USE
IP=1
ALIS="AA"+STR(IP,IIF(IP<10,1,IIF(IP<100,2,3)))
SELE &ALIS
STORE 1 TO RR1
SET COLO TO W+/W+
CLEAR
SET COLO TO W+/RB++
@ 24,0 SAY ' →右移←左移↑上移↓下移[PGUp]上屏
[PGDN]下屏 R 右屏 L 左屏 T 换库'+SPACE(18)
DO WHILE .T.
STORE 0 TO KEY,FNUM1,KEY2,KZBL
STORE 2 TO FNUM2
STORE 1 TO N3,MM1,XHBL,XHBL1
BM=STR(IP,IIF(IP<10,1,IIF(IP<100,2,3)))
FDNAME="FDNAME"+BM
FDLEN="FDLEN"+BM
XRAY="XRAY"+BM
FDTYPE="FDTYPE"+BM
FDDEC="FDDEC"+BM
SELE &BM
DO TYCXXT WITH S1(IP),S2(IP),ROW1(IP),XSHS(IP),
XSHS0(IP),WIDTH(IP),,
RR1(IP),RR2(IP),REC1(IP),NM(IP),FNUM1(IP),FNUM2
(IP),,
N3(IP),MM1(IP),CL0(IP),XHBL(IP),IP
IF KEY2=27

```

```

SET PROC TO
SET SYSMENU OFF
CLEA WINDOWS
RETURN
ENDIF
ENDDO
RETURN
*! *****
*! PROCEDURE: TYCXXT
*! CALLED BY: TYCX. PRG
*! CALLS: DISPWINDOW (PROCEDURE IN TYCX. PRG)
*! : LEFTMOVE (PROCEDURE IN TYCX. PRG)
*! : DISPREC (PROCEDURE IN TYCX. PRG)
*! : RIGHTMOVE (PROCEDURE IN TYCX. PRG)
*! *****
PROCEDURE TYCXXT
  PARA SS1,SS2,ROW0,XSHS,XSHS0,WIDTH,R1,R2,REC1,
  NM,FNUM1,FNUM2,N3,M1,COL0,XHBL,IP
  * SS1:窗口区域,显示字段颜色
  * SS2:窗口边框颜色
  * ROW0:欲在窗口内显示的第一个字段的行坐标
  * COL0:窗口上角列坐标
  * XSHS:窗口高
  * XSHS0:实际在窗口内显示的记录行数
  * R1,R2:数据库显示记录的首尾指针
  * REC1:数据库记录个数
  * NM:数据库字段个数
  * FNUM1,FNUM2:数据库字段显示的首尾字段指针
  * N3:数据库字段显示的当前字段指针
  * M1:数据库字段显示的当前字段字符起始指针
  * IP:数据库工作区指针
  * XHBL:大字段分段显示序号指针
  COL1=IIF(MOD(&XRAY(1)+WIDTH+2-COL0,2) # 0,
    &XRAY(1)+WIDTH+2,&XRAY(1)+WIDTH+3)+
    4
  SET COLOR TO N/W
  DO DISPWINDOW WITH ROW0-2,COL0,ROW0+XSHS,
    COL1,IP,SS2,REC1
  SET COLO TO &SS1
  M2=&FDLEN(N3)
  DO LEFTMOVE
  DO WHILE .T.
    IF KZBL=0
      I=1
      IF KEY2 # 18. AND. KEY2 # 3. AND. KEY2 # 24. AND.
        KEY2 # 5
        @ 0,10 CLEA TO XSHS+1, COL1-COL0
      ENDIF
      DO DISPREC
    ENDIF
    GO R1
    KEY2=0
    DO WHILE KEY2 # 3. AND. KEY2 # 2. AND. KEY2 #
      4. AND. KEY2 # 18. AND. KEY2 # 19;
      . AND. KEY2 # 27. AND. KEY2 # 114. AND. KEY2 # 82
      . AND. KEY2 # 76. AND;
      . KEY2 # 108. AND. KEY2 # 5. AND. KEY2 # 24.
      . AND. KEY2 # 84. AND. KEY2 # 116
      CLEA TYPEA
      KEY2=INKEY(0)
    ENDDO
    IF (KEY2=5. AND. R1=1). OR. (KEY2=24. AND. R2=
      REC1). OR. (KEY2=3. AND. R2=REC1). OR. (KEY2
      =18. AND. R1=1)
      KZBL=1
    ELSE
      KZBL=0
    ENDIF
    DO CASE
      CASE KEY2=84. OR. KEY2=116 && 按键为 T.t 键
        RR1(IP)=R1

```

```

        RR2(IP)=R2
        IP=IIF(IP<DNUM,IP+1,1)
        RETURN
      CASE KEY2=24 && 按键为 ↓ 键
        R2=IIF(R2<REC1,R2+1,REC1)
        R1=IIF(R2<REC1,R1+1,REC1-XSHS0+1)
      CASE KEY2=5 && 按键为 ↑ 键
        R1=IIF(R1>1,R1-1,R1)
        R2=IIF(R1>1,R2-1,R2)
      CASE KEY2=18 && 按键为 [PgUP] 键
        IF R1<=XSHS
          R1=1
          R2=XSHS
        ELSE
          IF R2=REC1. AND. XSHS#XSHS0
            DIS=R2-R1
            R2=REC1-DIS+1
            R1=R2-XSHS
          ELSE
            R2=R2-XSHS
            R1=R1-XSHS
          ENDIF
        ENDIF
        XSHS0=XSHS
      CASE KEY2=3 && 按键为 [PgDN] 键
        IF REC1-R2>=XSHS
          XSHS0=XSHS
          R1=R1+XSHS0
          R2=R2+XSHS0
        ELSE
          IF R2<REC1
            R1=R2
            DIS=REC1-R2
            R2=REC1
            XSHS0=DIS+1
            CLEAR
          ENDIF
        ENDIF
      CASE KEY2=82. OR. KEY2=114 && 按键为 R.r 键
        IF FNUM2<NM
          FNUM1=FNUM2
          DO LEFTMOVE
        ENDIF
      CASE KEY2=76. OR. KEY2=108 && 按键为 L.l 键
        FNUM2=IIF(FNUM1>1. AND. &FDLEN(FNUM2)<
          =WIDTH-2,FNUM1,FNUM2)
        DO RIGHTMOVE
        IF FNUM1=1. AND. FNUM2=1
          FNUM1=0
          DO LEFTMOVE
        ENDIF
      CASE KEY2=4 && 按键为 ← 键
        IF FNUM2<NM
          DO LEFTMOVE
        ENDIF
      CASE KEY2=19 && 按键为 → 键
        IF FNUM1>1
          DO RIGHTMOVE
        ENDIF
      CASE KEY2=27 && 按键为 [Esc] 键
        RETURN
      ENDCASE
      GO R1
    ENDDO
  *! *****
  *! PROCEDURE: DISPREC
  *! CALLED BY: TYCXXT (PROCEDURE IN TYCX. PRG)
  *! *****
  PROCEDURE DISPREC
    I=1
    DO WHILE I<=XSHS0. AND. (. NOT. EOF())

```

```

N3=FNUM1
DO WHILE N3<=FNUM2
  T=&FDNAME(N3)
  DO CASE
    CASE TYPE(T)='N'
      T1="STR(&T,&FDLEN(N3),&FDDEC(N3))"
    CASE TYPE(T)='C'
      T1=T
    CASE TYPE(T)='L'
      IF &T
        T2="".T.""
      ELSE
        T2="".F.""
      ENDIF (&T)
      T1="&T2"
    CASE TYPE(T)='D'
      T1="DLOC(&T)"
      M1=1
    ENDCASE
    IF TYPE(T) # 'N'
      IF (XHBL=1. OR. XHBL=2) . AND. FNUM1<>
        FNUM2 . AND. &FDLEN(N3)<=WIDTH-2
        M1=1
        T1="SUBSTR(" + T1 + ",M1,&FDLEN(N3))"
      ELSE
        T1="SUBSTR(" + T1 + ",M1,M2)"
      ENDIF
    ENDIF
    XS='-'
    IF I=1
      @ 0,0 SAY '记录号'
      IF FNUM1 = FNUM2 . AND. &FDLEN(N3) >
        WIDTH-2
        @ 0,&XRAY(N3)-&XRAY(FNUM1)+&XRAY
        (1)+6-COLO SAY T+;
        REPLICATE(XS,IIF(M2>LEN(T),M2-
        LEN(T),0))
      ELSE
        @ 0,&XRAY(N3)-&XRAY(FNUM1)+&XRAY
        (1)+6-COLO SAY T+;
        REPLICATE(XS,IIF(&FDLEN(N3)>LEN
        (T),&FDLEN(N3)-LEN(T),0))
      ENDIF
    ENDIF
    TT=RECNO()
    IF KEY2=18. OR. KEY2=5. OR. KEY2=24. OR. KEY2
    =3. OR. KEY2=0
      @ 1,0 GET TT FUNCT "JZ"
    ENDIF
    T1=&T1
    @ 1,&XRAY(N3)-&XRAY(FNUM1)+&XRAY(1)+6-
    COLO GET T1
    N3=N3+1
  ENDDO
  CLEA GETS
  I=I+1
  SKIP
ENDDO
*! *****
*! PROCEDURE; LEFTMOVE
*! CALLED BY; TYCXXT (PROCEDURE IN TYCX. PRG)
*! *****
PROCEDURE LEFTMOVE
  KZBL=0
  IF XHBL = 1. AND. FNUM2 = FNUM1. AND. FNUM2 <
  FCOUNT()
    FNUM2=FNUM2+1
    M1=1
    M2 = IIF(&FDLEN(FNUM2) > WIDTH-2, WIDTH-2,
    &FDLEN(FNUM2))
  ENDIF

```

```

IF XHBL=1. AND. FNUM1<FCOUNT()
  FNUM1=FNUM1+1
ENDIF
IF M1=&FDLEN(FNUM1)-M2+1. AND. FNUM1=FNUM2.
  AND. XHBL>1. AND. FNUM1<FCOUNT()
  FNUM2=FNUM2+1
  FNUM1=FNUM1+1
  XHBL=1
ENDIF
IF FNUM2 = FCOUNT(). AND. XHBL = 1. AND. &FDLEN
  (FNUM2)<WIDTH-2
  FNUM1=FNUM1-1
  RETURN
ENDIF
IF FNUM2=FCOUNT(). AND. &FDLEN(FNUM2)>WIDTH-
  2
  FNUM1=FCOUNT()
  IF M2=&FDLEN(FNUM2)-M1
    KZBL=1
    RETURN
  ENDIF
ENDIF
DO WHILE .T.
  IF &XRAY(FNUM2)-&XRAY(FNUM1)+&FDLEN
  (FNUM2)<=WIDTH-2
    FNUM2=FNUM2+1
    IF FNUM2>=NM
      FNUM2=NM
      EXIT
    ENDIF
    XHBL=1
  ELSE
    IF &FDLEN(FNUM2)>WIDTH-2. AND. FNUM2=
    FNUM1
      XHBL1=INT(&FDLEN(FNUM2)/(WIDTH-2))+
      1
      M1=(XHBL1-1)*(WIDTH-2)+1
      IF &FDLEN(FNUM2)-XHBL1*(WIDTH-2)>0
        M2=WIDTH-2
        XHBL=XHBL1+1
      ELSE
        M2=&FDLEN(FNUM2)-M1
        XHBL=1
      ENDIF
      FNUM2=FNUM2+1
    ENDIF
    EXIT
  ENDIF
ENDDO
FNUM2=FNUM2-1
*! *****
*! PROCEDURE; RIGHTMOVE
*! CALLED BY; TYCXXT (PROCEDURE IN TYCX. PRG)
*! *****
PROCEDURE RIGHTMOVE
  XHBL=1
  KZBL=0
  M1=1
  IF FNUM1<=1
    FNUM1=1
    IF &FDLEN(FNUM1)<WIDTH-2
      KZBL=1
      RETURN
    ELSE
      FNUM2=1
      M1=1
      M2=&FDLEN(FNUM1)
      KZBL=1
      RETURN
    ENDIF
  ENDIF

```

(下转第 66 页)

FoxPro2.5 的程序自动运行技术

□ 罗 辉

一些特殊的场合,有时需要对某些键实行虚拟输入,如自动填写某编辑域、自动演示一个软件等等,通过简单的击键或根本不需要击键就可以自动连续地模拟手工完成一系列复杂的动作。在 FoxPro2.5 中,为实现这种键虚拟输入提供了两种方式:预置键盘缓冲区、宏键定义。

一、预置键盘缓冲区

预置键盘缓冲区模拟键盘输入的顺序,预先向键盘缓冲区填入一串键码,而且这些键码将一直保留在键盘缓冲区内直到 FoxPro 请求键盘输入为止。它可用 KEYBOARD 命令实现。其使用格式:

```
KEYBOARD <expC>[PLAIN]
```

其中<expC>可以是一个字符串、一个键表,或一个能返回一个字符表达式的用户自定义函数。如果<expC>是一个键表,它必须用大括号和引号括起来,例如:

```
KEYBOARD '{CTRL+LEFTARROW}'
```

键表虽一些字母、数字或特殊意义的键名的组合。请看下面列出的具有特殊意义的键名表:

特殊键与键名表

特殊键	键名
Left arrow	LEFTARROW
Right arrow	RIGHTARROW
Up arrow	UPARROW
Down arrow	HOME
Home	HOME
End	END
PgUp	PGUP
PgDn	PGDN
Del	DEL
Backspace	BACKSPACE
Spacebar	SPACEBAR
Ins	INS
Tab	TAB
Shift Tab	BACKTAB
Enter	ENTER
{	LBRACE
}	RBRACE
F1 to F12	F1,F2...
Ctrl+F1 to Ctrl+F12	Ctrl+F1, Ctrl+F2...
Shift+F1 to Shift+F9	Shift+F1, Shift+F9...
Shift+F11, Shift+F12	Shift+F11...
Alt+F1 to Alt+F12	Alt+F1, Alt+F2...

特殊键	键名
Alt+0 to Alt-9	Alt+0, Alt+1...
Alt+A to Alt+Z	Alt+A, Alt+B...
Alt+PgUp	Alt+PGDN
Alt+PgDn	Alt+PGUP
Ctrl+left arrow	Ctrl+LEFTARROW
Ctrl+right arrow	Ctrl+RIGHTARROW
Ctrl+Home	Ctrl+HOME
Ctrl+End	Ctrl+END
Ctrl+PgUp	Ctrl+PGUP
Ctrl+PgDn	Ctrl+PGDN
Ctrl+A to Ctrl+Z	Ctrl+A, Ctrl+B...
RightMouse	RIGHTMOUSE
LeftMouse	LEFTMOUSE
Mouse	MOUSE
Escape	ESC

键盘缓冲区可以容纳 128 个击键,如果键盘缓冲区满,其后的击键将被忽略。为了避开 ON KEY LABEL 命令或宏键对该命令的影响,可以选择 PLAIN 子句,例如,如果已经有 ON KEY LABEL 命令给"A"键定义了一个例程,且"A"键已经包含在<expC>中,这时通过选用 PLAIN 子句,使击键"A"不作为特殊的热键看待,而作为普通的字符"A"使用。在下面的例子里,将显示一个窗口用于数据录入。如果输入的公司名已存在于数据库 CUSTOMER.DBF 中,其后有关的字段值将自动用相应的数据库内容填充。

```
CLOSE DATABASES
SET TALK OFF
SET SAFETY OFF
STORE SPACE (40) TO mcompany, maddress, mcomments
STORE SPACE (24) TO mcity
STORE SPACE (2) TO mstate
SELECT1
USE customer
SET ORDER TO TAG company
DEFINE WINDOW menter FROM 7,10, TO 19 70 PANEL
ACTIVATE WINDOW menter
@ 1,3 SAY' 公司:'GET mcompany VALID v _ cust (TRIM
(mcompany))
@ 3,3 SAY' 地址:'GET maddress
@ 5,3 SAY' 城市:'GET mcity
@ 7,3 SAY' 国家:'GET mstate
@ 9,3 SAY' 备注:'GET mcomments
READ
DEACTIVATE WINDOW menter
USE
```

```

FUNCTION v_cust
PARAMETER mcomp
SEEK UPPER (mcomp)
IF FOUND() && 如公司名已存在库中,其余编辑域自动用相应
应字段值填充
    KEYBOARD address + city + state
ENDIF
RETURN . T.

```

二、宏键定义

FoxPro 提供建立宏 MACROS 功能,以完成某些重复性的任务,节省操作时间;或自动演示一段程序,宏键是键击的组合,能使许多 FoxPro 下频繁执行的任务自动化。通过记录一系列字符和命令到一单键或其组合,可以建立宏。这样,你只要按一下被定义的键或键组合,便可执行所记录的一系列击键。用户可以存储宏,在以后需要时,将之恢复到内存后,即可按同样的宏键重演整个序列,就如同手工完成一系列的击键动作一样。

记录的一系列击键动作可以使用单个的字母、数字或其他可打印字符表示,也可以使用特殊键,不过特殊键的键名必须用大括号{}括起来。键与键之间也可以有一定的延时,这时,你可以输入{PAUSE <秒数>}表示执行其前面一个击键后延时所标记的秒数再执行下一个键。其中秒数可以带小数,可以精确到毫秒级。

注意所定义的宏键本身不能再出现在自己所记录的一系列击键中。否则将造成递归循环,出现异常。

1. 有关宏键操作的几条命令

在程序中使用已定义的宏键,可通过几条命令来实现。

(1)、PLAY MACRO <macro name> [TIME <expN
>] 执行一个宏键命令

在 FoxPro 下你可以通过按 Shift + F10 组合键将一系列击键定义为一个宏键存起来。PLAY MACRO 命令可激活这一系列击键。在程序中使用这一命令, 对你编制自动演示程序很有用。

被 PLAY MACRO 命令执行的代表一系列击键的宏键用宏键名<macro name>标识。

TIME 子句注明程序执行一系列击键时,响应每个键之后的时间延时。在表达式 $\langle \text{expN} \rangle$ 中标明的延时量在 0~10 秒之间。例如,如果 $\langle \text{expN} \rangle$ 等于 1,则每一个键击间的延时为一秒。

(2).RESTORE MACROS [FROM<FILE> | FROM MEMO<memo field>]从文件备注字段恢复宏键定义到内存

你可以定义和存储许多宏键到内存中。当退出 FoxPro 时,除非你已经用 SAVE MACROS 命令将它存储到一个宏文件或备注字段中,否则它们将丢失。

当 RESTORE MACROS 命令不在文件或备注字段中, 它将从内存中清除所有的宏键定义, 并将它们恢复成默认值。宏键的默认值是从 DEFAULT.FKY 文件中恢复的。如果没有发现 DEFAULT.FKY 文件, 功能键 F2~F9 将恢复成 FoxPro 系统的标准设置。

(3)、SAVE MACROS TO <file name>|TO MEMO <memo field>将内存中已定义的宏键存储到一个文件或备注

字段中

如果注明将保存内存中已定义的宏键到 MACRO 文件, 则 MACRO 文件必须以字母或下划线打头, 而不能以数字开头。它以 .FKY 为后缀名。如果你不以 .FKY 为文件名后缀, 则必须在 RESTORE MACROS FROM 命令中包含文件名的后缀。

如果存储内存中已定义的宏键到由<memo field>标明的当前记录的备注字段中,则该备注字段不必一定是当前工作区。不过,当存储到另一工作区打开的数据库的备注字段时,字段名前必须包含该数据库的别名。

(4)、CLEAR MACROS 从内存中释放所有的宏键定义

该命令从内存中释放所有的宏键定义,包括用 SET FUNCTION 定义的任何功能键。

2. 程序中使用宏键实例

首先通过系统菜单的 SYSTEM 菜单基中的 Macros 项, 或者在命令状态直接按 Shift+F10 组合键, 调出定义宏键的 Macros Key Definition 对话框, 定义一个宏键 ALT+Y。

宏键 :ALT+Y

宏键名: BROWDEMO

保存该宏键定义的文件名: DEMO.FKY

宏键所代表的一系列键击的组合:

```
{DNARROW}{PAUSE 0.10}{DNARROW}{PAUSE 0.10}
```

&& 模拟下移键

[illegible]


```
{ALT+B}{PAUSE 1.00}R{PAUSE 2.0}
&& 模拟 ALT+B&& 组合键
{TAB}{PAUSE 0.10}{TAB} {PAUSE 0.1}
```

&&R 键、TAB 键、&&ENTER 键

[illegible]

FoxPro 系统菜单的汉化方法

□ 罗 军

 FoxPro 数据库管理软件提供了一个非编程用户接口——菜单系统 SYSMENU, 它几乎包含了 FoxPro 绝大部分常用的命令。使用数据库的用户不用写一条命令便能访问 FoxPro 的各种功能。仅通过对菜单、窗口等进行正确的操作, 即可完成建立、打开、修改、维护、建立关系、过滤设置等数据库操作。进而逐步了解和使用 FoxPro 系统, 完成基本的数据库管理任务。

但 FoxPro 是一个西文软件, 系统的菜单、窗口等信息的显示都用英文实现的, 对使用中文的初级用户, 其学习和使用有一定的难度。

不过 FoxPro 的菜单系统提供了灵活的用户编程接口, 为 FoxPro 的菜单系统的汉化提供了条件。

一、系统菜单的编程接口

FoxPro 的菜单系统提供了 System、File、Edit、Database、Record、Program、Window 等七个菜单基, 每个菜单基相对应一个弹出菜单, 选择其中某一菜单项即可完成相应的操作。在 FoxPro 中, 对系统的菜单基、弹出菜单、弹出菜单中供选择的菜单项都赋予了相应的系统名称, 譬如:

系统菜单基的名称是: _MSYSMENU;

上述七个菜单基相应的名称是: _MSM _SYSTM、_MSM _FILE、_MSM _EDIT、_MSM _DATA、_MSM _RECORD、_MSM _PROG、_MSM _WINDO;

各菜单基相应的七个弹出菜单名称是: _MSYSTEM、_MFILE、_MEDIT、_MDATA、_MRECORD、_MPROG、_MWINDOW;

```

=====
{UPARROW}{PAUSE 0.10}{UPARROW}{PAUSE 0.10}
{UPARROW}{PAUSE 0.10}{UPARROW}{PAUSE 0.10}
{UPARROW}{PAUSE 0.10}{UPARROW}{PAUSE 0.10}
{PAUSE 1.00}{TAB}{PAUSE 1.00}{TAB}
{PAUSE 1.00}{TAB}{PAUSE 1.00}{TAB}
{PAUSE 1.00}{TAB}{PAUSE 1.00}{TAB}
{PAUSE 1.00}{TAB}{PAUSE 1.00}{TAB}
{PAUSE 1.00}{TAB}{PAUSE 1.00}{TAB}
{PAUSE 1.00}{TAB}{PAUSE 1.00}{CTRL+W}
&& 模拟 +W 组合键存盘
然后编辑如下的程序:
set talk off
restore macros from demo && 从 DEMO 文件中恢复宏键 ALT
+Y, 宏键名为 BROWDEMO
select 1
use \FOXPRO25\GOODICS\DEMO\DBFS\customer
go record 10
clear typeahead && 键盘缓冲区清空
play macro browdemo && 激活宏键 ALT+Y, 演示在 BROWS
  
```

而各弹出菜单中的菜单项也有相应的系统名称, 如弹出菜单 _MSYSTEM 中的菜单选项及其相应的名称如下:

菜单选项	相应名称
About FoxPro...	_MST _ABOUT
Help...	_MST _HELP
Macros...	_MST _MACRO
1st Separator	_MST _SP100
Filer	_MST _FILER
Calculator	_MST _CALCU
Calendar/Diary	_MST _DIARY
Special Characters	_MST _SPECL
ASCII Chart	_MST _ASCII
Capture	_MST _CAPTR
Puzzle	_MST _PUZZL

其他弹出菜单中的选项及其名称, 可在 FoxPro 系统联机帮助手册 HELP 中的 MENU - System Menu Name 帮助主题中查阅到。

在菜单系统中, 各种菜单选项的提示信息及其相应的系统名称是分别管理的。因此在程序设计中, 我们可以不改变其所完成的功能, 而为上述系统名称授予不同的屏幕提示信息, 或者在其中添加或删除某些选项, 或者更甚, 在应用软件的菜单系统中直接嵌入系统菜单中的一个或几个选择项, 以达到不同的应用效果。

二、快速菜单生成和系统菜单汉化

从上面的讨论可以看到, FoxPro 的系统菜单的汉化是困难的, 我们可以通过给系统再加上一个汉化的系统菜单“外

```

=====
窗口中一系列击键的动作
browse preference browsem normal
use
clear macros
  
```

执行该程序, 你将看到, 程序打开 FoxPro 2.5 系统自带的演示数据库 CUSTOMER 到 BROWSE 窗口, 编辑光标落在第十个记录的第一个编辑字段上; 然后演示向下一个个地移动记录好象你正按下下移光标键不放一样, 这样向下移动 35 个记录; 再激活 BROWSE 菜单基, 选择改变分区大小的菜单项 (Resize Partition), 模拟 TAB 键, 将一个 BROWSE 窗口分为两个 BROWSE 窗口; 再又模拟按回车键 (ENTER) 认可; 然后又自动改为向上移动 35 个记录; 然后向右移动编辑域, 好象你按了右移光标键一样, 移动 12 个字段后, 象按了 CTRL+W 组合键一样, 程序自动存盘, 退出演示。

壳”，取代原来的系统菜单，来达到目的。方法之一是：直接编程，对整个系统菜单各系统名称授予相应的汉字信息，这比较直观，容易为 FoxBASE+ 用户接受，但较原始。方法之二是：利用系统的菜单生成器完成汉化菜单的设计。

FoxPro 的菜单生成器中提供了一个快速菜单设计功能，在 File 菜单基对应的弹出菜单中选择 NEW，并选中无线按钮 MENU，进入菜单生成器；然后在 MENU 菜单中选择 Quick Menu，即可建立一个与 FoxPro 系统菜单一样的快速菜单，Quick Menu 主菜单区的选择项与系统菜单的选择项相同，每项可调出一个子菜单；各子菜单的条目也与相应系统菜单基的弹出菜单中选择项一一对应。将主菜单区及各子菜单区的 Prompt 栏目中各屏幕信息修改为相应的汉字信息。注意：其中的热键定义最好与原来的保持一致，以免使用混乱。然后以 FOXMENU 的文件名存盘，系统自动生成一个屏幕文件 FOXMENU.MNX。同时在 PROGRAM 弹出菜单中选择 Generate 功能，即可生成相应的菜单程序代码，它以 FOXMENU.MPR 的文件名存盘。

以后每次进入 FoxPro 系统时，首先用 DO FOXMENU.MPR 命令运行一次该程序，系统菜单将以汉字方式显示，给你的操作带来方便。当然，如果在使用中间，你用了 SET SYSMENU TO DEFAULT 等命令恢复到西文菜单方式时，需再次运行该程序才行。

为了便于理解，本文提供了相应的程序代码。你也可以直接输入该程序并运行之，以达到相同的效果。

系统菜单的汉化程序设计，在 FoxPro2.0/2.5、中国龙 2.0、启明星 1.0、华达汉字系统 1.0 等环境下都通过，且程序运行效果很好。

需要说明的一点是，一些与当前操作有关的动态附加的菜单基，如 MENU、BROWSE、SCREEN、REPORT 等菜单的汉化略显复杂，本文未予实现。

```
***** <附:菜单程序 FOXMENU.MPR> *****
SET SYSMENU TO
SET SYSMENU AUTOMATIC
***** 定义菜单棒中各菜单基 *****
DEFINE PAD _MSM_SYSTM OF _MSYSMENU PROMPT
"\<S 系统" COLOR SCHEME 3; KEY ALT+S, ""
DEFINE PAD _MSM_FILE OF _MSYSMENU PROMPT
"\<F 文件" COLOR SCHEME 3; KEY ALT+F, ""
DEFINE PAD _MSM_EDIT OF _MSYSMENU PROMPT
"\<E 编辑" COLOR SCHEME 3; KEY ALT+E, ""
DEFINE PAD _MSM_DATA OF _MSYSMENU PROMPT
"\<D 数据" COLOR SCHEME 3; KEY ALT+D, ""
DEFINE PAD _MSM_RECRD OF _MSYSMENU PROMPT
"<R 记录" COLOR SCHEME 3; KEY ALT+R, ""
DEFINE PAD _MSM_PROG OF _MSYSMENU PROMPT
"\<P 程序" COLOR SCHEME 3; KEY ALT+P, ""
DEFINE PAD _MSM_WINDO OF _MSYSMENU PROMPT
"\<W 窗口" COLOR SCHEME 3; KEY ALT+W, ""
ON PAD _MSM_SYSTM OF _MSYSMENU ACTIVATE POPUP _msystem
ON PAD _MSM_FILE OF _MSYSMENU ACTIVATE POPUP _mfile
ON PAD _MSM_EDIT OF _MSYSMENU ACTIVATE POPUP _medit
ON PAD _MSM_DATA OF _MSYSMENU ACTIVATE POPUP _mdata
ON PAD _MSM_RECRD OF _MSYSMENU ACTIVATE POPUP _mrecord
ON PAD _MSM_PROG OF _MSYSMENU ACTIVATE POPUP
```

```
_mprog
ON PAD _MSM_WINDO OF _MSYSMENU ACTIVATE POPUP _mwindow
***** 定义 SYSTEM 弹出菜单 *****
DEFINE POPUP _msystem MARGIN RELATIVE SHADOW COLOR SCHEME 4
DEFINE BAR _MST_ABOUT OF _msystem PROMPT
"\<A 关于 FoxPro..."
DEFINE BAR _MST_HELP OF _msystem PROMPT
"\<H 帮助..." KEY F1, "F1"
DEFINE BAR _MST_MACRO OF _msystem PROMPT
"\<M 宏定义..."
DEFINE BAR _MST_SP100 OF _msystem PROMPT
"\-"
DEFINE BAR _MST_FILER OF _msystem PROMPT
"\<F 文件夹"
DEFINE BAR _MST_CALCUL OF _msystem PROMPT
"\<C 计算器"
DEFINE BAR _MST_DIARY OF _msystem PROMPT
"\<D 日历/日记"
DEFINE BAR _MST_SPECL OF _msystem PROMPT
"\<S 特殊字符集"
DEFINE BAR _MST_ASCII OF _msystem PROMPT
"ASC\<II 码表"
DEFINE BAR _MST_CAPTR OF _msystem PROMPT
"\<P 屏幕捕捉器"
DEFINE BAR _MST_PUZZL OF _msystem PROMPT
"\<Z 谜题"
***** 定义 FILE 弹出菜单 *****
DEFINE POPUP _mfile MARGIN RELATIVE SHADOW COLOR SCHEME 4
DEFINE BAR _MFI_NEW OF _mfile PROMPT
"<N 新文件..."
DEFINE BAR _MFI_OPEN OF _mfile PROMPT
"\<O 打开文件..."
DEFINE BAR _MFI_CLOSE OF _mfile PROMPT
"\<C 关闭文件"
DEFINE BAR _MFI_CLALL OF _mfile PROMPT
"\<A 关闭所有文件"
DEFINE BAR _MFI_SP100 OF _mfile PROMPT
"\-"
DEFINE BAR _MFI_SAVE OF _mfile PROMPT
"\<S 存储"
DEFINE BAR _MFI_SAVAS OF _mfile PROMPT
"\<V 存储为..."
DEFINE BAR _MFI_REVRT OF _mfile PROMPT
"\<R 恢复"
DEFINE BAR _MFI_SP200 OF _mfile PROMPT
"\-"
DEFINE BAR _MFI_SETUP OF _mfile PROMPT
"\<I 打印机设置..."
DEFINE BAR _MFI_PRINT OF _mfile PROMPT
"\<P 打印..."
DEFINE BAR _MFI_SP300 OF _mfile PROMPT
"\-"
DEFINE BAR _MFI_QUIT OF _mfile PROMPT
"<Q 退出"
***** 定义 EDIT 弹出菜单 *****
DEFINE POPUP _medit MARGIN RELATIVE SHADOW COLOR SCHEME 4
DEFINE BAR _MED_UNDO OF _medit PROMPT
"\<U 不做" KEY CTRL+Z, "^Z"
DEFINE BAR _MED_REDO OF _medit PROMPT
"\<R 重做" KEY CTRL+R, "^R"
DEFINE BAR _MED_SP100 OF _medit PROMPT
"\-"
DEFINE BAR _MED_CUT OF _medit PROMPT
"\<T 剪裁" KEY CTRL+X, "^X"
DEFINE BAR _MED_COPY OF _medit PROMPT
"\<C 复制" KEY CTRL+C, "^C"
DEFINE BAR _MED_PASTE OF _medit PROMPT
```

```

"\<P 粘贴"KEY CTRL+V,"^ V"
DEFINE BAR _MED_CLEAR OF _medit PROMPT
"清除"
DEFINE BAR _MED_SP200 OF _medit PROMPT
"\-"
DEFINE BAR _MED_SLCTA OF _medit PROMPT
"\<A 选择全部"KEY CTRL+A,"^ A"
DEFINE BAR _MED_SP300 OF _medit PROMPT
"\-"
DEFINE BAR _MED_GOTO OF _medit PROMPT
"\<L 跳到某行..."
DEFINE BAR _MED_FIND OF _medit PROMPT
"\<F 查找..."KEY CTRL+F,"^ F"
DEFINE BAR _MED_FINDA OF _medit PROMPT
"\<G 再查找"KEY CTRL+G,"^ G"
DEFINE BAR _MED_REPL OF _medit PROMPT
"\<E 替换并再查找" KEY CTRL+E,"^ E"
DEFINE BAR _MED_REPLA OF _medit PROMPT
"全部替换"
DEFINE BAR _MED_SP400 OF _medit PROMPT
"\-"
DEFINE BAR _MED_PREF OF _medit PROMPT
"\<N 参数设置..."
***** 定义 DATABASE 弹出菜单 *****
DEFINE POPUP _mdata MARGIN RELATIVE SHADOW COLOR
SCHEME 4
DEFINE BAR _MDA_SETUP OF _mdata PROMPT
"\<U 装配..."
DEFINE BAR _MDA_BROW OF _mdata PROMPT
"\<B 浏览"
DEFINE BAR _MDA_SP100 OF _mdata PROMPT
"\-"
DEFINE BAR _MDA_APPND OF _mdata PROMPT
"\<A 添加..."
DEFINE BAR _MDA_COPY OF _mdata PROMPT
"\<C 复制..."
DEFINE BAR _MDA_SORT OF _mdata PROMPT
"\<S 排序..."
DEFINE BAR _MDA_TOTAL OF _mdata PROMPT
"\<T 汇总..."
DEFINE BAR _MDA_SP200 OF _mdata PROMPT
"\-"
DEFINE BAR _MDA_AVG OF _mdata PROMPT
"\<V 求平均值..."
DEFINE BAR _MDA_COUNT OF _mdata PROMPT
"\<O 求记录数..."
DEFINE BAR _MDA_SUM OF _mdata PROMPT
"M 求和..."
DEFINE BAR _MDA_CALC OF _mdata PROMPT
"\<E 财政统计..."
DEFINE BAR _MDA_REPRT OF _mdata PROMPT
"\<R 报表..."
DEFINE BAR _MDA_LABEL OF _mdata PROMPT
"\<L 标签..."
DEFINE BAR _MDA_SP300 OF _mdata PROMPT
"\-"
DEFINE BAR _MDA_PACK OF _mdata PROMPT
"\<P 压缩"
DEFINE BAR _MDA_RINDX OF _mdata PROMPT
"\<X 重新索引"
***** 定义 RECORD 弹出菜单 *****
DEFINE POPUP _mrecord MARGIN RELATIVE SHADOW COL-
OR SCHEME 4
DEFINE BAR _MRC_APPND OF _mrecord PROMPT
"\<A 添加"
DEFINE BAR _MRC_CHNGE OF _mrecord PROMPT
"<E 变化"
DEFINE BAR _MRC_SP100 OF _mrecord PROMPT
"\-"
DEFINE BAR _MRC_GOTO OF _mrecord PROMPT
"\<G 跳到某记录..."

```

```

DEFINE BAR _MRC_LOCAT OF _mrecord PROMPT
"\<L 定位..."
DEFINE BAR _MRC_CONT OF _mrecord PROMPT
"\<C 继续" KEY CTRL+K,"^ K"
DEFINE BAR _MRC_SEEK OF _mrecord PROMPT
"\<S 搜索..."
DEFINE BAR _MRC_SP200 OF _mrecord PROMPT
"\-"
DEFINE BAR _MRC_REPL OF _mrecord PROMPT
"\<P 取代..."
DEFINE BAR _MRC_DELET OF _mrecord PROMPT
"\<D 删除..."
DEFINE BAR _MRC_RECAL OF _mrecord PROMPT
"\<R 恢复..."
***** 定义 PROGRAM 弹出菜单 *****
DEFINE POPUP _mprog MARGIN RELATIVE SHADOW COLOR
SCHEME 4
DEFINE BAR _MPR_DO OF _mprog PROMPT
"\<执行..."KEY CTRL+D,"^ D"
DEFINE BAR _MPR_SP100 OF _mprog PROMPT
"\-"
DEFINE BAR _MPR_CANCL OF _mprog PROMPT
"\<终止执行"
DEFINE BAR _MPR_RESUM OF _mprog PROMPT
"\<R 继续执行"KEY CTRL+M,"^ M"
DEFINE BAR _MPR_SP200 OF _mprog PROMPT
"\-"
DEFINE BAR _MPR_COMPL OF _mprog PROMPT
"\<M 编译..."
DEFINE BAR _MPR_GENER OF _mprog PROMPT
"\<N 产生源程序..."
DEFNIN BAR _MPR_DOCUM OF _mprog PROMPT
"进入 Fo\<xDoc"
DEFNIN BAR _MPR_GRAPH OF _mprog PROMPT
"进入 Fox\Graph..."
***** 定义 WINDOW 弹出菜单 *****
DEFINE POPUP _mwindow MARGIN RELATIVE SHADOW
COLOR SCHEME 4
DEFINE BAR _MWI_HIDE OF _mwindow PROMPT
"\<H 隐藏"
DEFINE BAR _MWI_HIDEA OF _mwindow PROMPT
"\<A 全部隐藏"
DEFINE BAR _MWI_SHOWA OF _mwindow PROMPT
"\<H 全部显示"
DEFNIE BAR _MWI_CLEAR OF _mwindow PROMPT
"\<R 删除"
DEFINE BAR _MWI_SP100 OF _mwindow PROMPT
"\-"
DEFINE BAR _MWI_MOVE OF _mwindow PROMPT
"\<M 移动" KEY CTRL+F7,"^ F7"
DEFINE BAR _MWI_SIZE OF _mwindow PROMPT
"\<S 改变大小" KEY CTRL+F8,"^ 8"
DEFINE BAR _MWI_ZOOM OF _mwindow PROMPT
"\<Z 窗口最大"KEY CTRL+F10,"^ 10"
DEFINE BAR _MWI_MIN OF _mwindow PROMPT
"\<O 窗口最小" KEY CTRL+9,"^ 9"
DEFINE BAR _MWI_ROTAT FO _mwindow PROMPT
"\<C 重绕" KEY CTRL+F1,"^ F1"
DEFINE BAR _MWI_COLOR OF _mwindow PROMPT
"\<L 颜色编辑..."
DEFINE BAR _MWI_SP200 OF _mwindow PROMPT
"\-"
DEFINE BAR _MWI_CMD OF _mwindow PROMPT
"进入命令窗口" KEY CTRL+F2,"^ F2"
DEFINE BAR _MWI_DEBUG OF _mwindow PROMPT
"\<D 调试"
DEFINE BAR _MWI_TRACE OF _mwindow PROMPT
"\<T 跟踪"
DEFINE BAR _MWI_VIEW OF _mwindow PROMPT
"\<V 视图"
RETURN

```

浅谈如何提高 DOS 系统的使用效率

□朱 猛

DOS 是适用于 IBM-PC 机及其兼容机的磁盘操作系统,自 1981 年 DOS1.0 版问世以来,经过不断的改进和增强,目前已发展到 DOS6.0 版,它已成为微机操作系统的标准,受到绝大多数用户的喜爱,在微机用户心中的地位已根深蒂固。由于 DOS 是单用户操作系统,因此它不提供任何用户权限检查机制、系统高度透明、资源完全开放,能使用机器的人就能操纵该机上的所有资源、使用硬盘上所有文件。这种设计对 PC 机是完全合理的,但目前在国内许多单位、这种 PC 机往往不够安全,硬盘文件杂乱、系统配置不合理、资源浪费及使用效率低等。本文针对这些问题,根据笔者多年来使用 DOS 系统的经验,介绍一些实用的使用技巧和管理经验,供用户参考。

一、合理分区使用硬盘

在安装操作系统之前,应根据实际工作需要,用外部命令 FDISK 将硬盘分区。将操作系统和公用软件如 FoxBASE、WPS 和 CCED 等存放在公用区,该区的内容要设法保证安全、稳定,不要存放用户文件和临时文件。为防止有些文件被无意删除,可用 DOS 的 ATTRIB 或 PCTOOLS 的 A 设置只读属性,比如将 COMMAND.COM、AUTOEXEC.BAT、CONFIG.SYS 等文件设置只读属性。其他分区作为用户专用区,用来存放专用软件和数据。在同一个分区中应建立不同类别、不同层次的子目录,将文件分类存放,并定期对文件进行整理,如删除无用文件和临时文件等,以便查找、使用和维护,这样才能避免硬盘上各类文件混在一起、杂乱无序、临时文件过多,同一文件多处存放的情况。同时,建议对硬盘经常使用 CHKDSK 命

令,这对硬盘的维护管理是极为有利的。因为在硬盘(或软盘)上删除文件时,由于断电及驱动器磁头误动作等原因,使被删文件在文件分配表 FAT 中的簇项不能释放,这些未释放的簇 DOS 无法再使用,成为丢失的簇。这些丢失的簇越多,硬盘的利用率就越低。此时必须使用 CHKDSK/F 命令(方法为:在 convert lost chains to files(y/n)? 提示后键入 n)将丢失的簇释放,以便这些簇能再利用。

二、优化系统配置文件

微机启动时,首先要运行一个引导程序,接着 DOS 扫描驱动器的根目录,读取名为 CONFIG.SYS 的 DOS 文件,根据此文件来确定系统的配置。CONFIG.SYS 文件是一个标准的 ASCII 文件,用户应根据实际需要进行合理的配置和优化。最值得注意的命令有:

1. BUFFERS(缓冲区)

BUFFERS 项允许用户定义 DOS 在读写磁盘时存放数据所用的磁盘缓冲区数目。许多用户常有“缓冲区越多越好”的想法,其实如果把缓冲区设得太大,实际看到的是系统性能下降了。缓冲区太大或太小都会影响系统工作效率,应根据内存的容量、最常用的应用类型确定最佳缓冲区数量。

2. FILES(文件)

FILES 项指定了 DOS 一次能打开的最大文件数目,其取值的范围为 8-255。应根据用户正常工作的需要配置最佳值。

3. DEVICE(设备)

DOS 系统提供了用户能安装的两个设备驱动程序,即 ANSI.SYS 和 VDISK.SYS。ANSI.SYS 支持增强的键盘和视频显示,提供屏幕处理特性。VDISK.SYS 可以在内存建立 RAM 磁

盘即虚拟盘,虚拟磁盘的数量、空间、扇区大小及文件数目可根据内存情况任意设定,它具有速度快、效率高和灵活方便的特点。

对 640K 以上内存建立虚拟盘的方法是在 CONFIG.SYS 文件中加上:(参数以 DOS3.0 为例)

```
DEVICE = VDISK.SYS [BBB]
[SSS][DDD][E]
```

其中:

[BBB]: 虚拟盘尺寸,缺省值为 64K。

[SSS]: 扇区尺寸,可允许尺寸为 128、256 或 512。

[DDD]: 虚拟磁盘可以包括的目录数(文件数)。其范围是 2-512,缺省值为 64。

[E]: 通知 VDISK.SYS 使用扩展内存的参数。

下面是一个实际的 CONFIG.SYS 的例子:

```
buffers=30
files=30
device=ansi.sys
device=vdisk.sys 360 512 112
```

三、精心设计批处理文件

DOS 提供了一个功能很强的批文件命令,由在 DOS 下可以运行的键盘命令组成。批命令本身还包括 8 个子命令,这些子命令具有一定的编程能力,使得批处理文件具有很强的处理能力。但由于许多参考书上仅对批处理文件作一般性介绍,而对如何利用批处理文件开发一些巧妙、实用的 DOS 例程很少涉及。下面给出几种批处理文件的设计方法和简单例程。

1. 自动批处理文件

DOS 的批处理文件 AUTOEXEC.BAT 是在系统启动时自动执行的文件,

它直接影响系统的使用效果,应精心设计,现结合实例加以说明。

```
@ echo off
ver
date
time
path c:\; c:\dos; c:\213; c:\foxbase
append c:\; c:\dos; c:\213; c:\foxbase
prompt $p$g$e[37; 44m$e[0;66; "
dir/p"; 13p$e[0;67; "chkdsk/v"; 13p
@ echo on
```

其中,ver、date 及 time 分别显示 DOS 版本、日期及时间。path 命令建立了搜索目录顺序表,当 DOS 调用的文件(仅限于.COM、.EXE 或.BAT 文件)在当前目录中找不到时,可按搜索表的目录顺序依次查找,而 append 命令则设置了数据文件的查找路径。因此,有了 path 和 append 两条命令以后,就可以做到在任意子目录下使用其他目录所有文件。prompt 命令可以设置系统提示符、屏幕色彩和功能键:

prompt \$p\$g 使系统提示符为当前目录名和“>”符号。

prompt \$e[37; 44m 将屏幕设置成前景白色,背景为蓝色,改善了屏幕的视觉效果,使人赏心悦目。

prompt \$e[0; 66; "dir/p"; 13p 将 F8 定义为 dir/p 和回车。

prompt \$e[0; 67; "chkdsk/v"; 13p 将 F9 定义为 chkdsk/v 和回车。

因此,建议在 AUTOEXEC.BAT 中使用 path、append 和 prompt 命令,这样可以大大减少击键次数,简化操作。

2. 普通批处理文件

灵活运用批命令编程和将 DOS 命令序列编入批处理文件,能解决许多实

际问题,收到事半功倍的效果。

下面的批处理文件能自动地把硬盘的一部分当作一个软盘驱动器来使用,使得在单软驱上软盘间拷贝变得十分方便,只需要交换插入源盘和目标盘各一次,缩短了拷贝时间。

```
echo off
echo insert SOURCE diskette in drive A:
pause
md\temp
copy a: . c:\temp
echo insert TARGET diskette in drive A:
pause
copy c:\temp\ . a:
echo y|del c:\temp
rd c:\temp
```

3. 多重批处理文件

在批处理文件中请求执行另一个批处理文件,这种过程就叫多重批处理。实现这一功能会给批处理文件带来更大的灵活性。简单地在批处理文件中调用另一个批处理文件是不能达到此目的的。这是因为被调用的批处理文件执行后,并不能象.COM 文件或.EXE 文件那样回到调用它的.BAT 文件中继续执行后续的命令。利用 COMMAND.COM 文件可以解决这一问题,格式是 COMMAND/C sub, /C 表示进行新的命令处理,sub 是调用子程序名,这样就把一个调用 sub.bat 的问题转化为调用.COM 命令文件的问题,自然,当 sub.bat 执行完后可以正常返回,实现了多重批处理。

四、采取必要的安全措施

对于重要文件或子目录,为了防止被他人查看,可以加密处理。只有掌握密钥者才能够恢复文件或进入子目录,从

而加强了文件的保密性。对硬盘公用区规定只读属性,对专用区设置口令等等,能提高系统的安全稳定性。多数用户都希望自己的文件不被别人轻易拷走,因此,在文件编辑、调试、运行完成后,要删除硬盘上的这些文件及子目录,以便保密。但现在好多工具软件如 PCTOOLS 等都具有恢复删除文件或子目录的功能,必须采取一些措施,使得窃取者无法使用 PCTOOLS 等工具软件对其进行恢复。其实,下面的方法就能实现这一目的:

COPY CON filename (要删除文件的文件名) (CR)

随便输入几个字母后,按 F6 键回车,此时 filename 文件就无法用 PCTOOLS 恢复。

另外,建议使用虚拟盘,将需要编辑、调试和运行的程序放在虚拟盘上,这样不但速度快、效率高,对硬盘的磨损少,而且当你使用完后,只要关机或热启动,所有文件将消失,这不失为一种有效的保护措施。

五、灵活运用 DOS 命令

在 DOS 现有资源的基础上,在实践中和运用过程中,通过发掘 DOS 命令的内在功能,可以探索出一些独到的使用技巧,由此提高 DOS 命令的使用效率和范围。

DOS 是一个极为成功的操作系统,其使用和管理经验之多,远非本文所能概括。本文仅介绍了笔者的一些浅薄看法,旨在起到抛砖引玉的作用。

(上接第 57 页)

```
ENDIF
ENDIF
IF FNUM2>1
FNUM2=FNUM2-1
ENDIF
IF &FDLEN(FNUM2)>WIDTH-2
M2=WIDTH-2
FNUM1=FNUM2
XHBL=2
RETURN
ELSE
M2=&FDLEN(FNUM2)
ENDIF
DO WHILE &XRAY(FNUM2)-&XRAY(FNUM1)+&FDLEN
(FNUM2)<=WIDTH-2
FNUM1=FNUM1-1
IF FNUM1<1
EXIT
ENDIF
```

```
ENDDO
FNUM1=FNUM1+1
RETURN
*! *****
*! PROCEDURE: DISPWINDOW
*! CALLED BY: TYCXXT (PROCEDURE IN TYCX.
PRG)
*! *****
PROCEDURE DISPWINDOW
PARA X1,Y1,X2,Y2,IP,SS,RECNUM.
WNAME="W"+STR(IP,IIF(IP<10,1,2))
IF .NOT. WEXIST(WNAME)
DEFINE WINDOW &WNAME FROM X1,Y1 TO X2,Y2
SHADOW COLOR &SS;
TITLE " 文件名:" +TRIM(LTRIM(DBF())) FOOT " 记
录总数:" +LTRIM(STR(RECNUM,9)) DOU-
BLE
ENDIF
ACTIVATE WINDOW &WNAME
*: EOF: TYCX. PRG
```

Xenix 系统引导时常见软故障及其维护

□石子荣 石学荣

一:某台 386 主机开机后屏幕显示:Errors have been found during the power on seef test in hour computer.

The errors were:

Incorrect configuration data in CMOS memory size in CMOS invalid

从屏幕显示的信息看,CMOS 中配置信息在错误,重新启动机器,按键或其他键(根据说明),进入系统配置状态,用原来抄列的系统正常配置表与当前系统配置信息核对,将不正确的配置信息改正过来即可。

例二:Xenix 系统在安装后重新引导时,若出现“NO OS”后死机,这种情况是由于分区时 Xenix 分区没有分在硬盘柱面的边界上,即安装时起始道号输错,出现警告信息时用户没有理睬,当引导时系统找不到 Xenix 分区,对于这类错误,可以重新分区,将起始道号改正;另外一种情况是由于安装时系统的活动码输错,因而只有重装系统时将活动码改正。

例三:开机后屏幕提示信息:“Boot:0err80”。从屏幕提示的信息分析,可能是系统文件遭到破坏,用自制的引导盘启动系统,在出现“BOOT:”后,输入“fd(52)xenix”正确引导后,再输入命令“/etc/mount/dev/hd0root /mnt”,将硬盘根文件系统安装在引导盘的/mnt下,经检查根目录下的 boot 和 xenix 两文件均存在,为了慎重起见,将根目录的 boot 文件更名为 boot.old,然后将引导盘上的 boot 文件拷贝到硬盘中的根文件系统进行修复,输入命令“/bin/fsck/dev/hd0root”,修复结束后正常关机,重新开机,硬盘引导成功,系统运行正常。

例四:开机后当在屏幕上出现“boot not found”时,一定是 BOOT 文件发生了错误,启动过程中若发生这一问题,一般可能是保存在 CMOS 中的硬盘参数发生了变化按照新的硬盘参数在固定的区域内找不到 BOOT 文件,此时只要重新启动系统,将硬盘参数设置正确,即可排除错误;另外可能是由于 BOOT 文件本身的文件属性或文件丢失的缘故,这种情况可用自制的引导盘将系统启动后,把硬盘的根文件系统挂到软盘上,然后将 BOOT 文件拷到硬盘根目录下即可,对于文件属性不正确的,只要将文件属性用命令“CHOWN”或“CHGRP”或“CHMOD”改正则行。

例五:启动过程中,当在屏幕上显示“xenix not found”错误信息时,必定是硬盘上的 Xenix 文件发生故障,对于这种错误的原因,多是该文件属性不对或该文件所占的磁盘扇区部分被破坏或该文件丢失。我们可参照前面所述的方法,用引导盘启动系统后,把硬盘根文件系统挂在/MNT 目录下,再

将软盘上的系统核心拷贝到硬盘的根目录下,重新启动系统,故障便可排除。

例六:当屏幕上出现:“CAN,T EXEC /ETC/INIT”错误信息时,肯定是/ETC/INIT 文件遭到破坏且不可执行。此时用引导盘启动系统,将硬盘的根文件系统安装到软盘上,再把软盘上的/ETC/INIT 文件拷贝到硬盘上即可。

例七:系统启动时重复执行/ETC/RC 及/ET1CRC.D 下的文件,无注册信息出现。这是由于/ETC/TTYX 文件丢失或损坏造成的。系统启动时执行/ETC/TTYX 文件,为每个打开的终端和虚拟屏幕创建进程号进行读写,它决定系统能打开的终端及数据传输速率和设备文件名等等。解决方法可参照以前的方法,将文件拷贝正确就行。

例八:系统引导时若出现“INIT:/DEV/TTYXX:GETTY KEEPS DYING THERE MAYBE APROBLEM”错误信息时,一定是/ETC/GETTY 文件被破坏,此时可进入系统维护状态,将引导盘内的 GETTY 文件拷贝到硬盘上即可。

例九:系统引导时直接进入单用户状态而不能转多用户,按 Ctrl+D 也不能转。这是由于/ETC/PASSWD 文件丢失的缘故,因为系统在加载时要进行工作方式的选择,此时系统会自动在/ETC 子目录下寻找 PASSWD 文件,若找不到则自动转入单用户状态。处理这种故障时,可将硬盘内的备份文件如 PASSWD.OLD 直接拷贝为 PASSWD,或者将自制的引导盘用命令“/ETC/MOUNT FD 09 6DS15/MNT”挂到硬盘上,再把软盘上的 PASSWD 文件拷贝到硬盘中,也可用 VI 编辑一个 PASSWD 文件。

例十:当屏幕上显示“LOGLN:”而不能进入系统时,肯定是/ETC/LOGIN 文件受到损坏。解决这类故障时,用自制的引导盘启动系统后,将硬盘用命令“/ETC/MOUNT HD0-ROOT/MNT”挂到软盘上,把软盘上的/ETC/LOGIN 文件拷到硬盘内,重新启动系统,故障即可排除。

向宇对等网(X&YNET)

北京向宇计算机公司

地址:北京复兴路甲 20 号 27 分号 312,314

电话:8263441 2063366 呼 1511 邮编:100036

硬盘的使用及维护

□黄 海

在一般单位在配置微机时,常选择 386 及以上机型,这些中、高档微机大多配置有硬盘。有相当一部分使用者对硬盘了解不够,不会正确使用,出现硬盘不能启动、数据丢失等故障,严重者造成硬盘零磁道划伤,给用户的工作、经济造成损失。有必要谈谈硬盘的使用及维护。

现在普遍使用的是采用温彻斯特技术的硬盘(简称温盘)。温盘技术核心是“浮动磁头”技术。顾名思义,它不同于靠磁头与盘面接触读写的方式,如软盘。温盘在没有启动时,磁头被机械机构锁紧在磁盘启停区(启停区是专供磁头起飞着陆的一块没有记录任何信息的区域),启动后,锁紧机构张开(冷启动机器,上电自检完毕后,有轻微的富有弹性的“锵”的一声,即是锁紧机构松开的声音)磁头起飞到有关信息区域上方,此时磁头以约 6 微米的间距悬浮在介质上,所以不会划伤磁盘,读写完毕后步进电机余势未衰,仍达 100 转/分以上,将磁头送回启停区,当电机停止转动时,磁头已悄然停在了启停区。如果磁头在信息区内接触了盘片就可能导致信息的丢失,甚至硬盘的物理损坏,在使用中要注意以下几点:

1. 在检查有关电缆连接无误后,方可上电使用,硬盘工作时(指示灯亮),切忌搬动机器、热启动、按复位键和掉电以防磁头因震动意外碰触盘片和不能停回启停区而划伤盘片。

2. 磁头、磁介质较脆,要避免撞击,如需搬运,应运行 Park.com 程序或 Norton 软件包中的 Disk Monitor 选择 Disk park,确保磁头退回“启停区”。

3. 硬盘可以水平或垂直地放置在机箱中,但切忌将之底座(印刷板一面)向上安放。

4. 避免硬盘在灰尘过重的环境中使用,以免灰尘堵死空气过滤器的通道或进入磁头盘面间隙划伤磁头或磁盘。避免在强电、强磁、高温、潮湿的环境中使用,以免破坏或影响数据。

新购置回一台微机,需要在设置程序 setup(通常兼容机常使用 AMI 的 setup)里对硬盘类型进行设置。在许多情况下硬盘设置不正确是造成硬盘启动失败的重要原因。在高档微机中硬盘接口采用 16 位 IDE 接口,硬盘可以互为使用,有标准的,固定的类型(AMI 是 1-46),用户用“PgDn”或“PgUp”键选择,如果需自定义类型,要求用户自己输入柱面数(Cyls)、扇区数(Sect)、磁头数(Heads)。AST 和 AMI 等型号的 BIOS 能够自动识别硬盘类型,给使用者带来很大便利。接下来的物理格式化(底层格式化)工作已由厂家完成,但要成为一个开工硬盘尚需:

硬盘分区

为使用多个操作系统共享整个硬盘空间用 Fdisk 命令将

硬盘划分成多个分区。DOS3.3×版本能管理 33M 以下的硬盘。对于拥有大容量硬盘的 DOS3.3×的用户,硬盘分区有直接意义。高版本的 DOS 如 DOS5.0 或 DOS6.0 能直接管理高达 2000M 的硬盘,免去分区之苦,但是一些常用的中文软件如 WPS(较低版本)在大硬盘上运行有问题,这是因为它使用了 INT25H、INT26H 中断,若硬盘分区大于 32M 则会在 AX 寄存器中返回错误码。解决办法有两个:一是仍将硬盘分区,每个分区的容量在 33M 以下;二是编写一个 TSR 程序将 INT25H、INT26 中断的参数转化成 DOS 5.0 或 DOS 6.0 处理大硬盘读写时所需的新参数。

对 DOS 分区格式化

使用 FORMAT/S 对所选定的激活分区格式化,以便能从硬盘启动。经过硬盘的初始化、硬盘分区、硬盘格式化这三步工作之后,已经构造好一开工硬盘。

当硬盘使用时间一长,盘上碎片随之增多,碎片是当一个文件被存储到一个硬盘的不同位置即不连续扇区而形成的。碎片影响到系统的速度,使磁头耗废在寻找有关信息区的时间上增多。定期优化硬盘是必需的,一般要做以下几步工作:

1. 删除不需要的文件,然后使用 chkdsk/f 命令。该命令如果发现丢失的分配单元(簇),便提示用户是否将它们恢复成文件,如果确信丢失的分配单元里不含有用信息,键入 NO,chkdsk 删除这些信息,释放被占用的单元;反之键入 Yes,chkdsk 将这些碎片转换为文件 File0001.chk, File0002.chk... 置于根目录之下。

2. 确信已经退出所有程序(如不能在 Dosshell, Windows(中启动)并且确信没有启动任何 TSR 程序。

3. 运行 PCTOOLS 工具包中的 compress 程序。

如果手头没有 compress 程序及其他有此功能的程序,呆以先备份硬盘数据,然后对硬盘格式化消除文件碎片,再用硬盘备份恢复硬盘数据。

如果微机有 XMS 和 EMS,使用 Smartdrv.sys 硬盘高速缓存程序减少从硬盘上读取数据的时间和次数以达到提高速度的目的。

在 config sys 添加:

```
device=c:\DOS\himem.sys
```

```
device=c:\DOS\emm386.EXE R4M
```

```
device=c:\DOS\Smartdrv.sys 1024/A
```

以上命令将 1024K 的扩充内存分配给 Smartdrv,以致高速缓存大小是 1024K。

另外,硬盘的读写速度可以在硬盘低级格式化时调整。提供硬盘低级格式化功能的有:DM 软件, Qaplus 有关选项,调

用 ROM BIOS:

A>Debug

-G = C800:0005

或在设置程序 setup 中,选择 Hard Disk Utility 执行。调整硬盘读写速度是通过低级格式化时正确选择硬盘交错数(interleave)来完成。选择范围是 1—9。一般地,PC 机选 6,386 机选 3,可以适当调整。注意:无论选大了还是选小了都会降低硬盘的读写速度。所以,如果采用了诸多措施仍对硬盘速度不满意,可以在将数据备份后,重新设置参数。

谈起“计算机病毒”大家都有一种谈虎色变的感觉,确实,如若感染上病毒,硬盘数据毁之一旦,辛苦付诸东流。

下面提供一些防护硬盘的具体措施。

1. 如果引导型病毒破坏了硬盘的主引导程序和分区信息表开机启动后,显示:

Drive not ready

insert disk in drive A

strike any key when ready

硬盘丢了,出现这种情况,可以借助 NDD.EXE (Norton Disk Doctor)恢复分区信息。但是最好的办法是防患于未然,确信使用的是未被感染上病毒的命令对硬盘分区和格式化之后,就应立即着手备份主引导记录及分区信息表,程序如下:

c>Debug

-n a:read.com

-a 100

```
XXXX:0100 mov ax,0201
        mov bx,1000
        mov cx,0001
        mov dx,0080
        int 13 ;将硬盘 0 道 0 面 1 扇内容读至内存 XXXX:
                1000 处
        mov cx,0
        lea dx,012a
        mov ah,3c
        int 21
        mov cx,0200
        mov bx,ax
        mov dx,1000
        mov ah,40
        int 21 ;将该内容写入文件 boot.dat 中
        mov ah,3e
        int 21
        ret
        db 'boot.dat'
```

XXXX:XXXX db 0

XXXX:XXXX

-r cx

:XX;取划线处末两位

-w

-n A:write.com

-a100

```
XXXX:0100 mov ah,3d
        lea dx,0128
        int 21
        mov bx,ax
        mov cx,0200
        mov dx,1000
        mov ah,3f
        int 21
        mov ah,3e
        int 21
        mov ah,0301
        mov bx,1000
        mov cx,0001
```

mov dx,0080

int 13

ret

db 'boot.dat'

db 0

XXXX:XXXX

-r cx

:XX;取划线处末两位

-w

-Q

执行 read.com 命令,将在 A 驱生成一个 boot.dat 文件,该文件保存主引导程序及分区表信息,一旦需要,使用 Write.com 命令便又可将该信息写回硬盘。另外一种较简便的方法是使用相关软件,如使用 CPAV 中 Bootsafe,亦可得到一个保存主引导程序和分区表信息的文件,在需要的时候,用 bootsafe/r 将之写回硬盘。

2. 在 AUTOEXEC.BAT 中安装 Norton 的 Disk Monitor 中的 Disk Protect,常驻内存,用户可以选择对硬盘的系统区,某个或某类文件乃至整盘加以保护,从而防止病毒侵袭。如果没有 Norton 的话,可安装 Vsafe 及 vwatch。为节约常规内存可将之装入高端内存:

在 config.sys 中添加:

device=C:\DOS\himem.sys

device=C:\DOS\Emm386.sys noems

dos=high,umb

在 AUTOEXEC.BAT 中添加:

LH C:\CPAV\Vsafe/1+/2+.../8+

LH C:\CPAV\Vwatch

vsafe/1+/2+/3+/4+/5+/6+/7+/8+可使计算机在低级格式化硬盘、程序驻留内存、数据写磁盘,可执行文件被病毒感染,引导扇带毒,有数据欲写入硬盘引导扇或分区表、欲改写软盘引导扇区及欲改变可执行文件时产生相应警告,Vwatch 监测病毒,一旦发现病毒就立即报警。

3. 定期检测、清除病毒、常用软件有 CPAV,SCAN/,Kill 等,CPAV 在清除国产病毒方面稍嫌乏力,并且具有一定“毒副性”使某些应用程序不能启动,如果配合使用多种杀毒软件,就相得益彰了。

4. 一定要养成经常备份数据的良好习惯,一旦出现最坏情况需格式化硬盘时,免受更大损失。

最后建议大家审慎使用硬盘压缩工具。常见的有 MS-DOS6.0 及以上版本的 Doublespace 和 stac 电子公司的 Stack3.1。鉴于目前市场上大容量硬盘价格相对昂贵及软件愈来愈庞大,使用硬盘压缩工具能有效地将硬盘存储空间提高一倍以上,但是,对数据的压缩和解压缩需要额外开销,影响了系统性能,根据美国 PCworld 测试中心数据,特别是对数据库软件 Paradox3.5 的使用中,速度损失达 50%! 其他也有 2%~7% 的速度损失。而且,压缩硬盘上如有 1 个字节数据出错,将会影响 8KB 数据! 因而,对其安全性予以考虑不是多余的。另外在压缩硬盘中使用诸如 2.13H 汉字系统的用户将会发现汉字的显示及打印有问题,因此,建议时对 C 盘不要压缩,在 C 盘装入常用的软件如 DOS,WPS 等,而压缩其余逻辑盘,装入大型软件如游戏,PCTOOLS 8.0. Borland C++3.1 等,这样既能充分体现硬盘压缩工具极高的性价比,又能一旦不测将损失控制在最低限度。

主机电源维修一例

□ 闫辰国

故障现象:一台 AT/286 计算机主机工作中突然停止运行,关机重新加电,主机面板指示灯不亮,风扇不转。

分析与维修:首先该计算机是在没有稳压电源和 UPS 的环境下直接在市电电压上工作的。我们知道主机运行需靠电源供给 $\pm 5V$ 和 $\pm 12V$ 直流电压,但电压没有输出到主机板上,因此可以断定电源部分很可能出现了问题。

打开主机,取下电源,静态观察保险丝完好无损,印刷板上无明显烧烤糊迹象。上电测得 $\pm 5V$ 和 $\pm 12V$ 均无电压输出。AT/286 主机电源属于他激式脉宽调制型开关电源,它具有过流、过压、缺相保护、“启动”等功能。任何一端出现故障,相关的自动保护控制线路将向脉宽调制组件送去故障信号,

脉宽调制组件在故障信号的作用下,使它的未级驱动端送出来的调制脉冲的宽度变为零,迫使直流稳压电源进入电压输出为零的停机状态。针对交流保险丝未熔断、电路板上的元器件没有明显的被烧焦烤糊、滤波电容未出现冒顶爆裂以及电路元器件未见明显的脱焊、虚焊的现象,根据以往的维修经验,判断故障点很可能发生在高频降压变压器的副边绕组部分。因为主机绝大多数组件均采用 $5V$ 电源供电,因而电源 $5V$ 输出端的负载电流最大,随之而来的故障率也较高。测得 $300V$ 直流高压桥式整流线路及未级驱动功率晶体管均正常。测量二极管,结果发现二极管 D19 被击穿短路,更换一个性能相同或相近的二极管后,计算机恢复正常。

微机常见热故障的排除

□ 赫建

微机常见故障中有一类是芯片和元件热性能差引起的,多发于兼容微机,特别是一些质量差的水货,严重时系统将不能正常工作。其实处理也较简单,请看下面几个实例:

1. 故障现象:一台 YORKTAL386/33 微机,联接一大屏幕显示器(美国 MX-210 20" 彩显, 1280×1024 , 256 颜色, 8bit + 2bit 存储器),面内插 AGC 显示卡,运行 2 小时后微机自动重启,且自检不到头又重启。

处理和分析:经开机检查无异常,机内温度较高,显示卡上全是存储器芯片,时值盛夏,分析可能是热性能故障所致,采取关机措施,两小时后再开机。运行 1 小时后重复上述故障。对此,将大屏幕显示器改联到另一台微机运行正常。

分析原因,该批微机质量普遍不好,显示卡上发热元件又多,微机上热性能差的器件工作在热临界线附近,一旦超出便发生故障。

2. 故障现象:上述微机刚刚修完软盘驱动器,验机时发现打印机不好用,运行 AutoCAD 时鼠标器不好用,重新启动,故障依旧。

处理和分析:经查系统开机显示参数中并行口消失,串口正常,开机查外观无明显异常,当天气温偏高。分析多功能卡及相关器件热性能差,导致接口故障,当时临近下班,采取关机处理,第二天早上开机正常。

3. 故障现象:一台 AST Bravo 486/25 微机,运行中无故死机,冷、热启动均无效,键盘不通,除发光管可按亮外,其

余无反应。

处理和分析:开始怀疑键盘故障,经换用正常,当时已连续运行一整天,室内温度偏高,也是夏季。分析该机已使用两年多,使用频率很高,接近故障期,特别是键盘接口有关器件老化,热性能差造成故障。仍采用关机方法,次日正常。

上述几个实例,具有代表性,根据故障情况和笔者多年对其他电器设备的维护经验,认为此类故障有普遍性,值得一提。该类故障多为器件本身质量差,如未经老化筛选等,或者是某些器件年久老化、趋于损坏,总之外特性是热故障。对于此类故障,一般无须特殊处理,若需坚持工作者,应采取降温措施,如使用空调、电风扇等,重要的场合须更换器件,彻底根治。

286 兼容机常见故障的分析及排除

□陈永红 穆大明

故障现象 1:

开机后,有时能进入硬盘,有时不能进入硬盘,即使进入硬盘,也会出现死机。

故障的分析及排除:

根据故障现象,分析认为可能是某一部位接触不良所致。打开主机箱,采用拔插法,将与硬盘有连接的插头重新插一遍。重新开机运行,机器恢复正常,故障排除。

故障现象 2:

当主机与打印机联机时,打印机无反应。

故障的分析及排除:

经检查,打印机是好的。

首先,用观察法进行检查,当检查到固焊在底板上的并行口控制芯片 U29 时,发现其表面裂开了一个小口,用手触摸芯片表面微热,估计故障为芯片受热老化所致。

更换 U29 芯片后,重新开机运行,机器恢复正常。

故障现象 3:

主要开机数分钟后,死机,且屏幕显示杂乱彩色图象。

故障的分析及排除:

从故障现象看,很象是计算机病毒所至,但经检查不是病毒所引起的。

打开机箱,用观察法进行检查,首先看元器件特别是集成电路是否有工作异常现象。结果发现,80286 芯片,由于质量的问题,造成温度过热,从而导致死机。更换 80286 芯片后,再重新开机运行,机器恢复正常,故障排除。

故障现象 4:

主机加电后,屏幕突然抖动一下,显示器光标消失,系统死机。

故障的分析及排除:

从出现的故障现象分析,这种故障很可能与时钟发生器部分有关。

用示波器检测 82284 产生的 CLK、reset 信号波形情况。结果,CLK 信号是正常的,而 reset 信号永远处于低电平,进一步检查 82284 各管脚的输入端,发现 Power Good 信号电路出现异常,这就使得时钟发生器 82284 的复位输入端 reset 不能产生一个低电平有效的负跳变,从而使其复位输出端 reset 也产生不了一个有效的高电平,CPU80286 不能完成复位动作,导致了整个系统死机。

在 82284 芯片的复位输入端 reset 的第 12 脚对地之间加焊一只 50μF 的电解电容。这样就可以通过电解电容自身充电时间来产生开关延时,以便使得从 Power Good 送来的信号有效。重新开机运行之后,机器运行正常,故障排除。

故障现象 5:

当用硬盘启动时,系统经常出现死机。

故障的分析及排除

根据故障现象,怀疑可能是某种病毒侵扰,然而,用一些流行的防病毒软件进行检查,没有发现任何异常现象。

开机按 Ctrl+Alt+Esc 进入 Setup 状态,屏幕显示系统参数,经检查表明各项参数正确无误。由此可得出,系统软环境基本完好,转入硬件测试。

经过各种检查测试,均未发现硬件有问题,那能不能是因为硬盘性能不佳,动作速度慢而引起的故障呢?故试着将面板系统时钟由高速(16MHz)调至常速(8MHz),然后,重新开机运行,机器运行正常,故障排除。

(上接第 63 页)

```
@i,1 say pcimag
endfor
use c:\user\magazine\magazine\magazine
cTable=browpick(dbf(),"Magazine Catalog-All Titles")
if "<>"cTable
clear
list fields Name,Postcode
use
delete file(cTable)
endif
wait
clear
@0,0,24,79 box
for i=1 to 23
```

```
@i,1 say pcimag
endfor
select magazine
cTable=browpick(dbf(),"Magazine Catalog-Selected Titles",
"Name,Postcode","Postcode='100036'")
if "<>"cTable
clear
list fields Name,Postcode
use
delete file(cTable)
endif
close all
set status on
Return
```

采用OLE的网络分布式目标通信

□王世铎 译

Microsoft 的 OLE 2.0 (Object Linking and Embedding) 提供了各种集成应用程序的方法。OLE 定义并实现了这样的机制,即允许应用程序作为软件“目标”(数据和处理这些数据的相关函数)的集合与其他应用程序“连接”。这种连接机制和协议被称之为 COM (Component Object Model——组件目标模型)。

OLE 的现今版本支持用户的台式计算机上集成信息的很多功能。可以想像,用户们能很容易地在不同的计算机上集成目标,就象它们都在本地一样。例如,利用这种功能,旧金山的用户可以把自己计算机上的电子表格单元(Cell)与在纽约的企业数据库相连接。每当数据库更新信息时,用户的电子表格能自动地反映这些变化。

为了支持不同计算机上的目标集成,Microsoft 正在开发 OLE 的新版本,它具有 COM 原有功能的全部特点。分布式目标通信 (Distributed Object Communication) 是 OLE 2.0 开发的下一步工作。提供与目前 OLE 2.0 相同的标准目标间的协作,而且允许在整个计算机网络之间进行这种协作。例如,在基于 Windows 的计算机上的 address-book 程序可以与 Unix 系统上的不同的地址目标连接,并输入它的地址信息,用户不必知晓网络上不同的平台之间进行的这种交互作用。

更为重要的是,OLE 的新版本无须改变现有的应用程序。目前 OLE 2.0 的应用程序可以直接与其他机器上的其他应用程序连接,不需要改变应用程序源代码,同时不需要重新编译应用程序。

应用程序不需要改变,因为只是扩充了基础目标通信机制(这种机制对于应用程序是透明的)。新版 OLE 不修改 OLE 2.0 的应用编程接口(API)和目标接口,因此应用程序可以用同样的方法调用 OLE 2.0 的相同功能。如果支持这些功能调用的服务碰巧在一台不同的计算机上,OLE 基础结构便自动地将这种请求发送到远程服务。

一、分布式 OLE

分布式功能是 OLE COM 的自然扩充。OLE 当前与未来版本之间的最重要区别是用于在目标间传送操作和数据的远程过程调用机制。在 OLE 2.0 中,有用于单机上的内部目标通信的“轻型”远程过程调用机制 (LRPC——“Lightweight”Remote Procedure Call)。LRPC 允许目标通过处理边界来传送信息,这些处理边界是用以保护操作系统内各应用之间互相影响的。也就是说,LRPC 是一种内部目标处理的通信工具,允许在同一计算机上处理操作(如目标)之间进行对话。新工

具使用 Microsoft 的 RPC (而不是,LRPC),通过增加网络上目标通信,而扩充了 OLE 2.0。

Microsoft 的 RPC 为 LRPC 增加了许多功能,其中最突出的是,网络上进行 RPC 调用的能力。RPC 系统允许应用程序调用远程过程,这些过程就象在与调用应用程序相同的地址空间一样。采用 RPC,应用程序可以用调用本地过程相同的方法调用远程过程。由于主叫程序和应答过程之间的数据传送是透明的,有可能在不改变非分布式应用程序使用的编程模式的情况下,创建在多处理或多计算机之间运行的应用程序。

Microsoft 的 RPC 更为绝妙且很重要的一个方面就是提供一套相关工具,能够自动地生成为传输数据而将数据打包的集合代码。这一功能允许程序员方便地定义新的目标接口,保证目标进行单处理以外的通信。Microsoft 的 RPC 与开放软件基金会 OSF (Open Software Foundation) 在其分布计算环境 (DCE——Distributed Computing Environment) 规范中定义的 RPC 标准兼容,因为 Microsoft 的 RPC 基于 DCE 的 RPC,所以支持 Microsoft 与 RPC 的系统能够与许多基于 DCE 的系统交换信息,包括 Open VMS MVS, AS/400 以及 20 多个 UNIX 系统。Microsoft RPC 增加了某些向上与用于定义目标的 DCE 接口定义语言 (IDL) 兼容的扩充,但不改变线路级的协议。

二、目标层

尽管 RPC 提供了许多分布式目标系统需要的关键功能,如网络传送独立性,安全性和签名服务的 APIS,以及不兼容处理器和操作系统之间的数据转换,但 RPC 还不够。OLE 在 RPC 之上的目标层提供了进一步对分布式编程结果以及目标特有的多形功能的提取和简化的方法。

在目标层,OLE 2.0 的 LRPC 结构已经提供目标的本地事件和非本地事件(目标在主叫程序地址空间之外)之间的透明性。建立分布式目标系统最为困难的是得到这种透明性。尽管增加完全的 RPC 机制,但 OLE 的新版本并不改变整个结构。

图 1 显示了客户应用程序连接到运行于不同机器上的一个目标服务器的方法。在客户机上运行 RPC 代理 (Proxy),使客户与目标服务进行通信,就象在同一台机器上一样。代理 (Proxy) 简单地将功能调用打包成标准的 RPC 信息,该信息使用在客户机上运行的网络运输在网络上进行传输。服务器上的应答 RPC 存根 (Stub) 从客户那里接收信息,将其拆包,

并送到目标服务器。

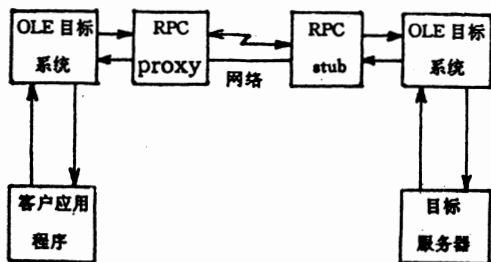


图 1 RPC 应用程序在远程系统上的调用方法

请注意，“客户”和“服务器”这两个词只是分别指使用者和目标服务的提供者。而不是说，客户就是指台式计算机，服务器，就是指大型后端计算机。事实上，在 OLE 中，应用程序之间的多数关系是双向关系，即软件子程序之间同时互为客户端和服务端。

三、分布式目标将重新定义计算

由分布式目标支持的 OLE 允许把一个应用程序分为许多不同的子目标；每个子目标都能单独地在不同的计算机上运行。由于 OLE 提供了网络的透明性，因此整个网络就象一个具有巨大处理能力和功能的大型计算机。例如，数据库应用程序可以按一套子程序来建立：查询，报告，格式生成和事务管理等。这些子程序都能够在具有它所要求的处理能力，I/O 带宽和硬盘容量的计算机上运行。其结果，计算变得更为有效了，因为软件能更加与所需求的准确的硬件能力相匹配。计算也能不断升级，因为一个应用程序或一组应用程序就象杠杆一样使整个网络的资源变为无限的资源。

四、公共目标模型

Microsoft 和 Digital 都认为需要建立不同操作系统上的目标之间的联系，并且开发了一种体系结构，允许 OLE 与 Digital 的 OLE 多任务平台相互操作和 Digital 的多任务平台目标系统，目标代理(Object Broker)之间相互操作。这种被称为公共目标模型(COM—Common Object Model)的体系结构，定义了基于 DCE RPC 的公共协议和一组 OLE 核心功能，对此 Digital 和其他感兴趣的公司在他们的产品范围内将给

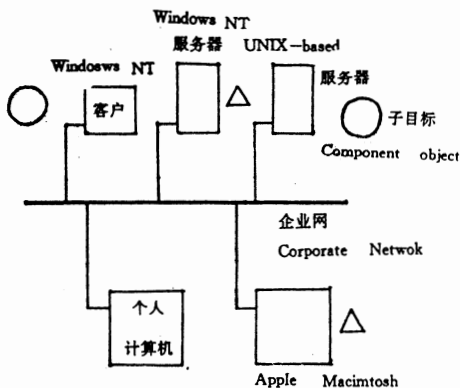


图 2 OLE 和 Object Broker 使企业目标级计算成为可能

予支持。公共目标模型是子目标模型的直接突破，并且具有完全与 OLE 的向上兼容性。如图 2 所示，当 OLE 与目标代理共同工作时，Windows, Windows NT 和 Windows NT 服务器等操作系统可以连接运行在各种平台上的目标，包括 OSF/1, HP-UX, SunOS, IBM, AIX, UNIX 和 Open VMS。

对于用户来说，用户收益是很大的。用公共目标模型，用户可以在企业中的任意一个平台上存取信息，无须了解所连接是哪种应用程序或达到目标所需要的通信机制的方式，采用简单的技术，如 OLE2.0 的牵引和置放(Drag Drop)技术，用户可以操作整个企业的目标，无须知晓这些目标在什么地方或使用什么应用程序来建立这些目标。

Microsoft 和 Digital 约定对 COM 遵循开放处理的原则，以面向广泛的工业要求。系统集成人员，企业应用程序开发人员，独立的软件销售商以及其他感兴趣的第三方人员对设计评审安排在 1994 年的上半年。在进行这些评审之前，将出版供评审用的草案。

五、Windows 的发展途径

今天的 OLE2.0 技术是 Microsoft 操作系统战署方向的主要基础的一步。Microsoft 的 Windows NT(Cairo)和 Windows(Chicago)操作系统的下一个主要版本将依靠 OLE，并与其兼容。虽然从 OLE2.0 发展到 Cairo 和 Chicago，在计算机使用的方法上有很大变化，然而对于现有的应用程序来说，达到这种功能性的途径还是平缓的。未来许多先进技术将会简单地继承现有的应用程序，而不改变这些程序本身。Microsoft 和 Digital 努力将保证 OLE 的分布式目标权能在 UNIX 和 Open VMS 平台上以及台式计算机、Windows NT 服务器上被采用。因此，独立的软件销售商和企业系统集成工程师现在就可以着手于 OLE2.0 进行开发，这些开发结果可以应用于在未来的分布式，多相的，基于目标的计算机系统。

为了显示 OLE 采用 COM 在平台间的操作，Microsoft 和 Digital 演示了示范应用程序，将在 OSF/1 操作系统运行的目标与 Microsoft Windows NT 计算机上的目标连接。示范应用程序支持 Microsoft Excel 和在 OSF/1 服务器上运行的库存查询目标之间的与标准 OLE 2.0 兼容的“热链接”。对于用户是透明，Object Broker 中的 OLE 单元允许 Microsoft Excel 的是电子报表与 OSF/1 计算机上的源数据连接。OLE 和 Object Broker 还有一种新功能(机制)，用来发现 OSF/1 服务器，并且动态地将库存查询数据传送到 Microsoft Excel 电子报表中。用户可以从屏幕上实时地看到更新的库存数据，而不必知道数据不同的来源。

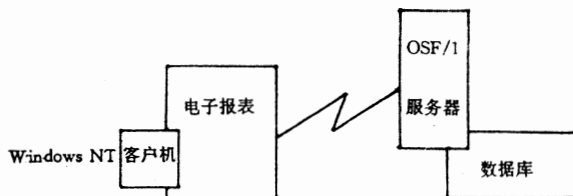


图 3 Windows NT 客户机与 OSF/1 服务器

(本文英文资料由 Microsoft 提供)

DOS 与 Windows

当格式化一张软盘时,MS-DOS 会分配给磁盘一个序列号,请问 DOS 是怎样将一个唯一的序列号分配给每一个磁盘的? 用户能阻止 DOS 分配序列号给磁盘吗? 如果不能,怎样从磁盘上删除序列号而用自己的序列号代替?

! 如果你用的 MS-DOS 版本是 4.0 或 4.0 以上的,在格式化或重新格式化磁盘之前,FORMAT 命令会立即自动地分配给每一张软盘或硬盘一个随机挑选的序列号。

因为被分配的序列号是一个 4 字节(32 位)十六进制数字,而 DOS 有超过 400 万种变化的数字,从这些变化的数字中可以随意选择序列号,从而使两张磁盘不可能被分配给同样的序列号。

但是,你不能使 FORMAT 命令不在磁盘上放置序列号,而且 DOS 也没有一种命令或选择可以让你修改序列号。唯一的改变序列号的方法是利用磁盘编辑工具,如 NORTON 软件或 DOS 的 DEBUG 命令,通过往磁盘上写入一个新的序列号来改变旧的。因为序列号是存放在磁盘的一个重要区域内的,因此绕过操作系统而直接往磁盘内写入内容是很冒险的,有时输入一个简单的字符也许会破坏整张磁盘上的信息。

为避免意外地毁坏磁盘,只有对编辑磁盘数据有相当水平并且有很丰富的这方面的经验的人,才能尝试修改序列号。如果不顾冒险硬要修改序列号,则首先要将序列号置在一个连续的偏移量为 39(27 个 16 进制的)的 4 字节数据区,它在磁盘的引导记录里。软盘的引导记录在 0 面 0 扇区。每个硬盘的引导记录在紧接着数据区前面的扇区中。

当使用 DIR 命令查看硬盘上的“DOS”子目录时,发现在“DOS”子目录里出现了一些杂乱的批处理文件,这些批文件的名字看起来很怪,例如,7241DOS.C.BAT,另外还有一个长度为 120K 的文件,名叫 DOSSHELL.SWP。这些奇怪的文件是从哪儿来的? 它们有什么用? 可以从硬盘上删除这些文件吗?

! 当从 DOS 的图形接口程序 DOSSHELL.EXE 启动一个应用程序或退到 DOS 提示符下时,DOSSHELL 的工作交换器将产生一个临时的批处理文件来容纳通往应用软件的全部路径名称。这个批处理文件就好比是一张公路交通图,指导着工作交换器完成程序交换的日常工作。上面所提到的长度为 120K 的文件,它装着 DOSSHELL.EXE 所需要的一些信息,当从 DOS 提示符下或从 DOSSHELL.EXE 的活动工作表中提及的应用程序中退回来以后,这些信息用于重新启动 DOSSHELL.EXE。

这些批处理文件都是临时性文件,在正常情况下,DOS 在每一个工作交换过程的最后清除掉这些文件。如果 DOS 子目录下有大量这样的批处理文件,可能是因为没有正确使用工作交换器的缘故。

假如启动 DOSSHELL,然后按 Shift+F9 键退出 DOSSHELL 到命令提示符下,就在系统将工作状态置于 DOS 提示符之前,DOSSHELL 产生一个小的批处理文件叫做 XXXXDOSC.BAT,在这里 XXXX 是一个随机的十六进制数,当在 DOS 提示符下完成工作以后,按 EXIT 返回到 DOSSHELL。

DOSSHELL 将清除它放置在 DOS 子目录里的指派作为控制临时文件的那个临时批文件 XXXXDOSC.BAT。

但是,如果没有在操作过程结束之前就返回到 DOSSHELL,这些临时文件就被保留着,存放在硬盘上。当你运行工作交换器的活动工作表中包含的一个应用程序后,不是从应用程序中退出并返回到 DOSSHELL,而是按下了 Ctrl+Alt+Del 重新启动机器时,临时批处理文件将不会被删除,而是被保留下来了。

为了避免积累额外的文件,记住要返回到 DOSSHELL 并通过选择“文件菜单”中的“退出”选项来退出程序,这样会确保所有的临时文件从硬盘上清除掉。

如果电源电压过低,经常导致机器不得不重新启动,可以在自动批处理文件接近结束的地方放置如下的语句来清除 DOSSHELL 留下的后顾之忧。

```
DEL C:\DOS\???? DOSC.BAT
IF EXIST DOSSHELL.SWP
DEL DOSSHELL.SWP
```

还有一种办法,留出一定的内存空间作为所谓的 RAM 盘,并且指派这个盘作为储存这些临时文件的地方。那么当你关闭系统或重新启动后,内存会被清除,所有的储存在内存中的临时文件也就同时被清除了。

最近,笔者将 AST Premium 486/25 微机的 DOS 版本从 5.0 升级到 6.0,但自从安装了 6.0 后,无论哪一次从 D 盘上的应用程序退出来时,都会收到“Invalid COMMAND.COM”的错误信息,为什么 DOS 需要寻找 COMMAND.COM 文件? 到底是什么引起的

这个问题? 当前的 COMSPEC 环境设置是:

COMSPEC=\COMMAND.COM

! COMMAND.COM 文件是非常重要的 DOS 文件, 用来执行 DOS 命令。如果 DOS 在磁盘上找不到 COMMAND.COM 文件, 计算机将不能正常工作。

DOS 在装载 COMMAND.COM 时, 总是将它分为两部分装载到内存中, 较小的部分叫做暂驻部分, 包含了执行 DOS 内部命令(如 COPY、DEL、REN 等等)所需要的代码。较大的部分叫做常驻部分, 当 DOS 运行应用程序时会去寻找它。

因为有些程序非常大, 它们运行时可能会偶然地覆盖住 COMMAND.COM 驻留在内存中的暂驻部分, 如果发生了这种情况, COMMAND.COM 的常驻部分必须能够在磁盘上找到 COMMAND.COM, 以便它可以重新装载损坏或丢失的暂驻代码部分。

COMSPEC 环境变量的作用是告诉 DOS, COMMAND.COM 在哪个盘上。如果 COMSPEC 设置不完全就会发生上述错误。通常情况下, 在 AUTOEXEC.BAT 文件中指定的 COMSPEC 路径应该包括启动盘的盘符。如果省略 COMSPEC 变量设置中的启动盘的盘符, 在 DOS 需要重新安装 COMMAND.COM 时, 会自动寻找当前盘的根目录区。如果当前盘是 D 盘, 而 COMMAND.COM 存在于 C 盘上, 系统就会用 "Invalid COMMAND.COM" 这条错误信息提示用户。

另外, 检查所有盘的所有目录来确保它们没有储存旧版本的 COMMAND.COM 文件也是一个好办法。为保证是用正确的版本工作, 应该在根目录区的 COMMAND.COM 与 DOS 子目录中 COMMAND.COM 比较一下, 这两个文件应该有相同的文件长度和日期、时间标记。

笔者刚刚安装上 DOS 6.0, 用 DBLSPACE.EXE 压缩硬盘上的文件时, 这个程序告诉我它不能继续下去, 因为硬盘中存在着数据错误, 程序提示运行 CHKDSK 并且带上参数 /F 后再运行 DEFRAG 来纠正错误。当我运行 DEFRAG 时, 得到这样的一条信息: "Error reading cluster 62,600; Use a disk

repair program to correct the problem. Then run DEFRAG again." 什么是磁盘修复程序? 我可以带着这个没有解决的问题运行 DBLSPACE 吗?

! 磁盘修复程序可以完成许多低级的硬盘维护功能。例如修复损坏的磁盘数据扇区或文件分配表(FAT), 或重新安装一个被破坏的主引导记录。所有这些易被破坏的数据项对硬盘的操作都是十分重要的。市场上提供了许多种优秀的磁盘修复软件可以选择, 其中有两种最流行的磁盘修复工具: Norton 软件包和 PCTOOLS 的 DISKFIX。

为此, 先使用磁盘修复程序来修补扇区损坏, 而不要带着问题使用 DBLSPACE 程序。DEFRAG 程序向用户报告磁盘损坏的情况。几乎所有使用 DBLSPACE 时出的问题, 都是由坏的磁盘扇区破坏了 DBLSPACE 的压缩卷文件(CVF)引起的。

硬盘可能包含了成百的从来没有使用过的文件, 但是, 用户又不愿意删除不熟悉名字的文件, 因为担心应用程序最终有可能会需要这些文件。你了解一种常驻内存的、可以鉴别应用程序从来也没有使用过的软件吗? 最好这个软件可以利用数据库来跟踪 Windows 3.1 和 DOS 的文件。

! 还不知道有哪一种共享或公用范围内的软件可以完成这种精确的工作, 但是可能一种商业软件包叫做 "磁盘历史学家" (Disk Historian) 可以完成这种工作。这种基于 Windows 3.1 上的软件通过保持文件使用方面的精确数据, 来帮助你恢复被浪费的磁盘空间。这个软件提供了诸如你曾经多少次访问过一个文件, 最后访问的日期和时间, 以及第一次使用文件的日期等信息。

在一台只有 640K RAM 的 286 PC 机上安装 MS-DOS 5 有哪些优点和缺点? DOS 5 的内核比 DOS 3.0 的内核小吗?

! 在 286PC 机上安装 DOS 5, 其好处不仅仅是简单的操作系统的升级, 使用 DOS 5, 可以享受许多应用程序的好处, 诸如 UNDELETE.EXE、UNFORMAT.COM 和 DOSKEY.COM, 这些功能在 DOS 4.0 及以前的版本都是看不到的。但是, 如果计算机只配备有 640K 的内存, 无法使用 DOS 5 具备的

许多内存管理特性。这些特性包括装载 DOS 内核(它的中心代码, 不包括外部程序), 设备驱动程序和内存驻留程序(TSR)到扩展内存。

因为 DOS 5 内核和 COMMAND.COM 的容量比 DOS 3.0 相应的部分大得多, 所以 DOS 5.0 下的应用程序能使用的 640K 内存的空间就相应减少, 从而影响应用程序的运行。

DOS 中的 LOADFIX 命令的作用是什么? 什么时候用这个命令?

! DOS 5 提供的内存管理功能, 可以将 DOS 的有关部分或其他程序从常规内存转换到高位内存, 从而释放了内存的前 64K 的空间, 这个空间在早期的 DOS 版本中为 DOS 保留。许多为 DOS 4.0 及以前的版本设计的 DOS 应用程序都禁止装载到这个区域。一旦这样的程序发现它们将要被安装到操作系统保留的内存空间时, 它们就会结束并返回一个错误信息。

LOADFIX.COM 是 Microsoft 公司提供的实用程序, 以确保那些设计好的不在计算机常规内存的前 64K 里运行的 DOS 应用程序能够装载到 64K 以上的地址。

DOS 没有提供一种直接更改子目录的方法, 那么怎样在 DOS 系统下更改子目录?

! 在 DOS 操作系统中没有提供更改子目录名的命令, 如果要变更子目录名, 只能先建一个新目录, 然后把旧目录中的文件复制到新目录中, 最后再删除旧目录。这样做既浪费时间又不方便, 下面这段用 Turbo C 2.0 编写的小程序能够快速方便地更改子目录名。将它编译连接后放到 DOS 子目录下作为一条外部命令使用, 将给您带来一点方便。

源程序如下:

```
C:\>TYPE CDN.C
#include <stdio.h>
void main(int argc, char *argv[])
{
    if(argc != 3)
    {
        printf("\nUsage : CDN [old _
        directory _name new _directory
        _name]\n");
        exit(1);
    }
    if(rename(argv[1], argv[2]) == 0)
```

```
exit(0);
printf("Invalid parameter\n");
}
```

在装入 DOS 6.0 后,不慎破坏了 DOS 6.0 的一个应用文件 GRAPHICS.COM。想从最原始 DOS 系统盘拷贝好的文件到 C:\DOS 目录下来替换坏的文件是很容易的事。但当将正确的系统盘插入到软驱中,然后敲入 DIR 命令时,看到大部分文件名都在它们的扩展名中包括下划线字符。当拷贝应用文件到 DOS 子目录后,并不能用原来的文件名启动它了,实际上,根本就无法进入这个应用软件。这是为什么呢?

! 从 DOS 5.0 开始,大多数 MS-DOS 应用程序以压缩格式存放在原始系统盘上,必须用 EXPAND 命令将文件从 DOS 系统盘上拷贝到硬盘上。EXPAND 命令将给文件解压缩后,拷贝到指定的目录中。然后将文件改名为 GRAPHICS.COM。

如果你的 DOS 6 系统盘是 1.2M 的,那么插入 2 号盘到 A 驱动器,然后键入如下命令,就可以将一个未被损坏的 GRAPHICS.COM 文件拷贝到 C:\DOS 子目录中:

```
EXPAND. EXE A:\GRAPHICS.
COM C:\DOS\GRAPHICS.COM
```

在购买新的 Super VGA 监视器时,我用 DOSSHELL.EXE 来比较不同型号监视器的质量。试过的所有监视器,DOSSHELL 都提供了 25 行、30 行、34 行、43 行和 60 行屏幕显示。又测试了原有的 EGA 监视器,DOS 5 的 DOSSHELL.EXE 只提供了 25 行和 43 行显示,起初猜想这可能是 EGA 的局限性,但是,当新买的 SVGA 监视器安装好后,DOSSHELL 仍然只能提供 25 和 43 行显示。怎样才能使 DOSSHELL 提供所有的 SVGA 屏幕格式呢?

! 为了解决这个问题,需要更换两个视频限定文件,这两个文件存放在 DOS 子目录下,它们控制着 DOSSHELL 的屏幕。这两个文件是: DOSSHELL.VID 和 DOSSHELL.GRB。替换这两个旧文件的新文件以压缩格式存放在原始 DOS 系统盘上。

如果不再需要旧的 DOSSHELL.

VID 和 DOSSHELL.GRB 文件,可以删除它们。但是明智的方法是将它们保存下来,直到确定替换成功。

建议将文件改名,使用如下命令:

```
REN DOSSHELL.VID DOSSHELL.
@ID
REN DOSSHELL.GRB DOSSHELL.
@RB
```

然后再开始替换,将原始 DOS 系统盘插入 A 驱,敲入如下命令:

```
EXPAND A:\VGA.VI_
C:\DOS\DOSSHELL.VID
EXPAND A:\VGA.VI_
C:\DOS\DOSSHELL.GRB
```

这样就可以将文件解压缩并把它们放置到 C 盘的 DOS 子目录里。重新启动 DOSSHELL.EXE 来确认新文件产生的效果是否与期望的一致,如果是,就可以删除先前改名的 DOSSHELL.@ID 和 DOSSHELL.@RB 文件了。

笔者自己编了几个 BASIC 程序,帮助新用户熟悉我们的网络。但是,这个程序总是要进入 QBASIC 的编辑器,需要用户键入另外的命令来运行程序。能否不用给用户介绍 QBASIC 的环境而直接启动这些程序?

! 可以不在屏幕上显示 QBASIC 的编辑器而直接启动它,只要再在命令行加上一个开关 /RUN 和想要运行的程序的名称。例如,启动 QBASIC 并装载 MYFILE.BAS:

```
QBASIC/RUN MYFILE.BAS
```

怎样在 DOS 命令状态发送一个信息到打印机?可以使用一个批处理文件,或者有更简单的方法吗?

! 可以在 DOS 提示符下发送信息到打印机,利用如下的输入命令:

```
ECHO ^L > LPT1:
```

输入命令时,先敲入 ECHO,然后输入一个空格再按下 Ctrl 键不放,然后按下 L 键 DOS 就会加上如下字符到命令行: ^L。

最后再敲入 >LPT1: 结束命令行输入,然后按回车键。

虽然这个方法在 DOS 命令状态工作得很好,但是你却不能从一个批处理文件中使用它。为了能从批处理文件中发送一个信息到打印机,必须首先建立另一个小的批文件。给这个批文件命名为 FF.BAT。首先键入如下命令,建立一

个批文件:

```
COPY CON FF.BAT
```

然后键入:

```
ECHO
```

跟上一个空格,象以前那样,按下 Ctrl 键不放,然后敲一下 L 键,DOS 会加入如下字符到批文件中: ^L

结束这个批文件,敲入如下字符:

```
>LPT1:
```

然后敲 F6 键,最后敲回车键结束。

这样可以直接从 DOS 命令状态下执行 FF.BAT 文件,也可以从另一个批文件中调用它。要想从另一个文件中调用 FF.BAT,必须安装 DOS 3.2 或更高的 DOS 版本。早期的 DOS 版本不允许从一个批文件中运行另一个批文件。在上面的例子中若利用打印口 LPT2,就用 LPT2 来替换 LPT1。

笔者正在开发一个基于 Windows 的应用程序,需要存储一个文件,采用 GetSaveFileName 例程时发现对话框里的文件列表呈暗灰色,好象它们都是处于非激活状态,请问能让这些文件列表呈黑色或者是其他的颜色吗?

! 这是实际使用的标准,文字变成灰色显示最终用户哪些是已经创建的,鼓励用户支持唯一的文件名,请注意,即使文字已变成灰色,你仍然能选中它,同时也能存入你的文件。

看起来,我们似乎不能产生名为 DISPLAY.EXE 的任何应用程序,请告诉我为什么在 Windows 下将任何 EXE 文件改名为 DISPLAY.EXE 时都会失去其功能?

! 这种问题经常遇到,要弄清楚这个问题必须深入到 Windows 的内部结构,当运行 Windows 3.1 时,系统在启动时加载一个驱动程序名为 DISPLAY.DRV,在 Windows 内部文件表中以 DISPLAY 名称来存储。链接程序用 .EXE 或者 .DLL 文件中的名字存储在 .DEF 文件中的名称,这样,加载一个名为 DISPLAY.EXE 的文件时,就和 Windows 内部的文件表中文件名相重,这样就会使应用程序失去了功能,因为 Windows 拒绝其执行。

但这种现象在 Windows NT 中不会出现,因为在其内部的文件列表中记录了全部的路径,因而不会混淆。

电脑病毒泛滥至今,一直困扰着广大计算机用户,给用户带来的损失颇大,为此应广大读者的要求,本着实用第一、智慧密集的办刊原则及为广大计算机用户服务的宗旨,本刊与帝霸计算机反病毒服务网联合举办了电脑反病毒信息公告栏这一新颖实用的栏目,希望它能够在为广大读者建立一个安全干净的电脑工作环境的过程中尽一点微薄之力。如果能对读者有一些真正的帮助,也就达到了我们苦心运作,拼搏多年的目的。

帝霸(DEBUG)计算机反病毒服务网是中国第一家专业从事计算机反病毒服务的机构。其自主开发的反病毒专用软件是一个具有高度智能化的反病毒应用软件系统,它最大的特点是立足于国内,自适应升级,它的检测部分的升级信息可以通过本刊的独家公告公布给广大读者,使具有专用软件的读者能迅速对专用软件自行升级,来对付

编者的话

新出现的病毒,可以说,只要您拥有了帝霸反病毒专用软件,加上您所拥有的本栏目,就不用再害怕病毒的侵扰了。

本刊以后的每一期计算机反病毒信息公告栏中都将公告国内新出现的病毒的升级信息,读者可以将这些信息参数通过帝霸反病毒专用软件强大的库管理功能自行录入帝霸反病毒专用软件中,这样您就可以自己对付这些病毒了,非常实用,非常方便。

为了使广大读者能更好的利用这个栏目,为了使广大读者能迅速获得帝霸反病毒专用软件,本刊已与帝霸反病毒服务网全面合作免费向读者赠送帝霸反病毒专用软件普及版(只收取成本费和邮寄费,总计20

元)。读者需要的话,可以邮局汇款至(100006)北京市劳动人民文化宫内《电脑编程技巧与维护》杂志社,联系电话:5123823,5232502,联系人:刘伟。

帝霸计算机反病毒服务网专用软件 DBNET

帝霸计算机反病毒软件是帝霸(DEBUG)计算机反病毒服务网自行开发的专用软件,是一套自成系统的,集查找病毒、清除病毒、快速自我升级于一体的、最具实用价值的新一代开放式反病毒软件,它的全新开放式和自升级的反病毒概念,可以让使用者参与实际反病毒的工作,有效地提高使用者自身的反病毒能力。

开放式技术是当今世界计算机技术发展的潮流和方向。帝霸计算机技术研究所的技术人员经过多年对计算机病毒与反病毒技术的研究,顺应时代潮流,将开放式技术溶于反病毒技术之中,形成了一套完整的科学理论,在学术上取得了重大突破,获得了各方面专家的一致赞誉。以此理论为基础投入大量的人力物力完成了它的商品化,形成了今天的帝霸计算机反病毒软件。

该软件采用独创的数据结构方法来描述病毒,从而可以很快地分析新出现的病毒,形成一套升级信息表,用户获得这张表即可自我升级帝霸反病毒软件。

该软件由三大功能模块组成:1.全面、快速的检测功能模块;2.安全彻底的病毒消除功能模块;3.功能强大、使用方便的反病毒库管理模块。

该软件的主要特点有:1.采用C++与汇编语言混合编程,提高了程序的运行效率;2.采用类Windows的精美图形界面,极大地提高了用户使用的方便性;3.勿须汉字系统支持,直接在西文状态下显示汉字图标;4.采用开放式反病毒技术概念,用数据结构的方式描述病毒,从而使用户不需要重新编程即可完成对新病毒的检测及清除的自身升级;5.使用户自身的反病毒能力与最新的反病毒技术和反病毒信息同步。

在该软件的基础上,帝霸计算机技术研究所又进一步组建了帝霸反病毒服务网,希望该软件能更好地为广大计算机用户服务,为广大用户建立和维护一个安全干净地计算机工作环境。

帝霸计算机反病毒专用软件分普及版和增强版两个型号,两者都可以消除目前国内流行的各种病毒。增强版可以通过邮寄给网员的完整的升级信息表来检测并清除以后新出现的病毒,普及版只能通过《电脑编程技巧与维护》杂志每月所刊登的部分升级信息表来升级并检测新病毒。增强版对入网会员免费提供,普及版则对所有人员均免费赠送(收取20元/套的成本费和邮寄费)。普及版可按下述任一地址邮购:

★(100006)北京市劳动人民文化宫内《电脑编程技巧与维护》杂志社

联系电话:(01)5123823,5232502 联系人:刘伟

★(100836)北京海淀定慧西里19号楼帝霸计算机技术研究所

联系电话:(01)8123896,8263441 联系人:管逸群

帝霸计算机反病毒服务网简介

帝霸计算机反病毒服务网(Debug Computer Anti-Virus Service Net 简称 DEBUG NET),是中国第一家在计算机反病毒领域内提供快速反应的专业服务网,得到了国内外有关方面,特别是《电脑编程技巧与维护》杂志的大力支持和协作,它的运营将填补我国计算机安全方面的一项空白,必将为我国计算机正常健康地运作,建立正常干净的计算机工作环境、开发环境作出贡献。

收费标准及服务内容:

一、A类:300元/年(面向单位用户)

服务内容:

1. 每年免费提供一套本网最新版本的专用软件。
2. 每月免费赠送一本《电脑编程技巧与维护》杂志。
3. 每月按时邮寄一份新病毒的完整升级信息表,网员可以对本网专用软件自行升级。
4. 网员发现病毒,本网可在24小时内做出快速反应,全面分析,形成升级信息表,并传送给网员,网员可以将其输入本网专用软件,将此病毒清除(传送可用电话或传真)。此项服务一年中4次以内免费,超过4次按30元/次收费。
5. 向网员通报特定病毒的发作及流行情况,及时提醒网员加强预防。
6. 为网员提供反病毒技术咨询、有偿修复硬盘、恢复数据等服务。

7. 不定期举办反病毒技术研讨会和学习班,学习和研讨反病毒技术的发展,介绍计算机反病毒技术,网员可获优先优惠。

8. 为网员推广计算机技术产品牵线搭桥。

9. 为网员提供项目咨询和相关的技术服务。

10. 为网员提供合法的常用软件,仅收成本费。

二、B类:90元/年(面向个人用户)

服务内容:

1. 每月按时邮寄一份新病毒的完整的升级信息表,用户可以对本网软件自行升级。不定期为网员提供相关的开发经验、技术、技巧。

2. 网员发现新病毒,本网以尽可能快的速度给网员解决问题,并将升级信息表列入邮寄的资料中,回寄给网员,解决网员的问题。

3. 每年本网发表的软件升级版本,均免费赠送一套。

4. 通报新病毒发现地点及泛滥区域,提醒用户预防。

5. 有偿提供修复硬盘,恢复数据等服务。

6. 本网举办各种技术研讨会和学习班时,可获优先。

三、专项服务:价格面议(面向系统用户)

服务内容:

按网员要求提供全方位服务。

入网方法

1. 申请入网者先填写入网申请表,并寄往下述地址:

(100836)北京海淀定慧西里19号
楼帝霸计算机技术研究所

2. 入网费在寄出申请表的同时可通过邮局汇往上述地址。

3. 帝霸计算机反病毒服务网在收到申请表及入网费后即将网员证、有关票据、帝霸反病毒服务网专用软件寄给网员。

4. 网员在收到网员证后即成为正式网员,可按相应服务类别享受服务。

帝霸计算机反病毒服务网入网申请表(复制有效)

联系人	单位名称			年龄
所在部门	职称		职务	
地址	通讯地址			
电话	传真	邮编		
从事工作				
微机型号				
网员类别	<input type="checkbox"/> A类:300元/年 <input type="checkbox"/> B类:90元/年 <input type="checkbox"/> 专项服务			
入网费	汇出人民币(大写) <input type="checkbox"/> 邮局 <input type="checkbox"/> 银行			
	汇出时间	19 年 月 日		
单位公章	年 月 日	经办人		

★帝霸计算机反病毒服务网热线咨询电话:(01)8123896 热线汉字寻呼:(01)8298866 呼1111。

《电脑编程技巧与维护》订阅单(复印有效)

地 址				邮 编	
单 位				收 件 人	
起订日期	年 月 日			共 计	期
订阅份数		款 数		汇款方式	

《电脑编程技巧与维护》订阅单(复印有效)

地 址				邮 编	
单 位				收 件 人	
起订日期	年 月 日			共 计	期
订阅份数		款 数		汇款方式	

诚征全国直销代理

自《电脑编程技巧与维护》月刊征订启事和各报刊杂志的消息及广告刊出之后,不少单位的图书馆、资料室、培训部门、书刊经销商以及广大热心读者来信、来电询问本刊的出版发行情况,并要求作为本刊的直销代理,有些读者、同仁甚至不辞劳苦来本刊编辑部谈直销事宜。这一方面反应出本刊“实用第一、智慧密集”的办刊原则的正确性,另一方面又反应了读者想尽早、方便地得到本刊的愿望,为此,我社决定,在全国诚征直销代理(单位或个人),其原则是:方便读者互惠互利、大家共享、共同发展。

具体办法如下:

① 包销:杂志社给予包销客户以较大优惠。② 代销:杂志社给予较好的优惠,在结清第一次代销杂志的款项后,按代销客户的要求发放下批杂志。为保证包销与代销客户的利益,本杂志社对首次代销的客户以 50% 的优惠价提供 10 份样刊试销。代销和包销的优惠率及合同如下:

包销与代销优惠率:

包销	优惠率	代销	优惠率
10~50 份	20%	10~50 份	18%
50~100 份	25%	50~100 份	22%
200 份以上	30%	200 份以上	25%

包销与代销合同(复印有效)

甲方名称					
通信地址					
邮政编码		电 话		传 真	
甲方代表签字			经办人		
乙方名称	《电脑编程技巧与维护》杂志社				
通信地址	北京市劳动人民文化宫内				
邮政编码	100006	电 话	5232502	传 真	
乙方代表签字	孙茹萍		经办人	刘伟	

经双方协商同意×××××××(甲方)愿作乙方(《电脑编程技巧与维护》杂志社)的包销或代销

者,按×××份的优惠率进行结算。本合同自双方代表签字之日起生效。

订 阅 须 知:

△本刊每月八日出版,每期订价:三·八元,全年订价:四十五元六角。一九九四年出版发行六期。邮局公开发行,邮发代号二四—一〇六。
△请将本卡寄至:(一〇〇〇〇六)北京市劳动人民文化宫内《电脑编程技巧与维护》杂志社发行部。联系电话:(〇一〇)五一二三八二三。
△如从邮局汇款,请按上述地址汇寄;如从银行汇款,请汇至:开户银行:中国农业银行北京分行东四北分理处一四二服。银行帐号:八〇一四〇五四。
帐户:电脑编程技巧与维护编辑部。款到后即按月寄刊。

读者意见征询卡(复印有效)

使本刊越办越好,请读者填写此卡。我们将根据寄回的征询卡,挑选 100 名本刊热心读者,免费赠送本刊 1995 年全年杂志。

读者姓名:_____ 年龄:_____ 职务或职称:_____

工作单位:_____ 电话:_____

通信地址:_____ 邮编:_____

▲您对本刊所设栏目的看法:(好:√,一般:○,不好:×)

- | | | | |
|--------------------------------|-------------------------------|----------------------------------|---------------------------------|
| <input type="checkbox"/> 新技术追踪 | <input type="checkbox"/> 软平台 | <input type="checkbox"/> 图形图象处理 | <input type="checkbox"/> 汉字处理 |
| <input type="checkbox"/> 编程语言 | <input type="checkbox"/> 数据库 | <input type="checkbox"/> 计算机安全 | <input type="checkbox"/> 网络与通讯 |
| <input type="checkbox"/> 软件维护 | <input type="checkbox"/> 实用软件 | <input type="checkbox"/> 硬件维护 | <input type="checkbox"/> 电脑博士信箱 |
| <input type="checkbox"/> 新产品 | <input type="checkbox"/> 明星企业 | <input type="checkbox"/> 反病毒信息公告 | |

▲您认为本期各栏目的质量如何(好:√,一般:○,不好:×)

- | | | | |
|-------------------------------|----------------------------------|----------------------------------|-------------------------------|
| <input type="checkbox"/> 软平台 | <input type="checkbox"/> 图形与图象处理 | <input type="checkbox"/> 汉字处理 | <input type="checkbox"/> 编程语言 |
| <input type="checkbox"/> 数据库 | <input type="checkbox"/> 计算机安全 | <input type="checkbox"/> 网络与通讯 | <input type="checkbox"/> 软件维护 |
| <input type="checkbox"/> 硬件维护 | <input type="checkbox"/> 电脑博士信箱 | <input type="checkbox"/> 反病毒信息公告 | |

▲您对本刊的总体看法:(好:√,一般:○,不好:×)

- | | | | |
|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| <input type="checkbox"/> 文章内容 | <input type="checkbox"/> 栏目设置 | <input type="checkbox"/> 版面安排 | <input type="checkbox"/> 编辑质量 |
| <input type="checkbox"/> 服务水平 | <input type="checkbox"/> 广告质量 | <input type="checkbox"/> 印刷质量 | |

▲您认为本刊是否适合你的需要?

- ☐ 很适合 ☐ 基本适合 ☐ 不适合

▲您最喜欢的本期栏目及文章是哪个(篇)?

▲您还希望本刊增加哪些栏目或哪些方面的内容?

▲您是否还有其他建议?

填卡须知:

请在每个选项前的方框内按要求作标记。需填写的项目,请用工整的字迹填写,写不下可另附纸。填好后沿虚线剪下(或复印),寄往:(100006)北京市劳动人民文化宫内《电脑编程技巧与维护》杂志社。

读者服务卡(复印有效)

需要本刊代为联系本刊中所出现的有关公司的读者,请填此卡。

读者姓名:_____ 职务或职称:_____ 传真:_____

工作单位:_____ 电话:_____

通信地址:_____ 邮编:_____

对本刊____年第____期第____页 ☐ 彩色广告 ☐ 黑白广告 ☐ 正文中出现的
公司的_____产品感兴趣

希望:☐ 寄取公司资料 ☐ 寄取产品目录 ☐ 寄取产品资料
☐ 询问价格 ☐ 建立业务联系 ☐ 其他_____