

实用第一, 智慧密集 Practicability First, Intelligence Intensive

第四

电脑编程技巧与维护

COMPUTER PROGRAMMING SKILLS & MAINTENANCE

1994年10月



MICROSOFT
WINDOWS
COMPTON
32-Bit Application



Microsoft
WIN32

SOFTWARE DEVELOPMENT KIT

Tools for Writing 32-Bit Applications for Windows

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,采用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。

ISSN 1006-4052



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

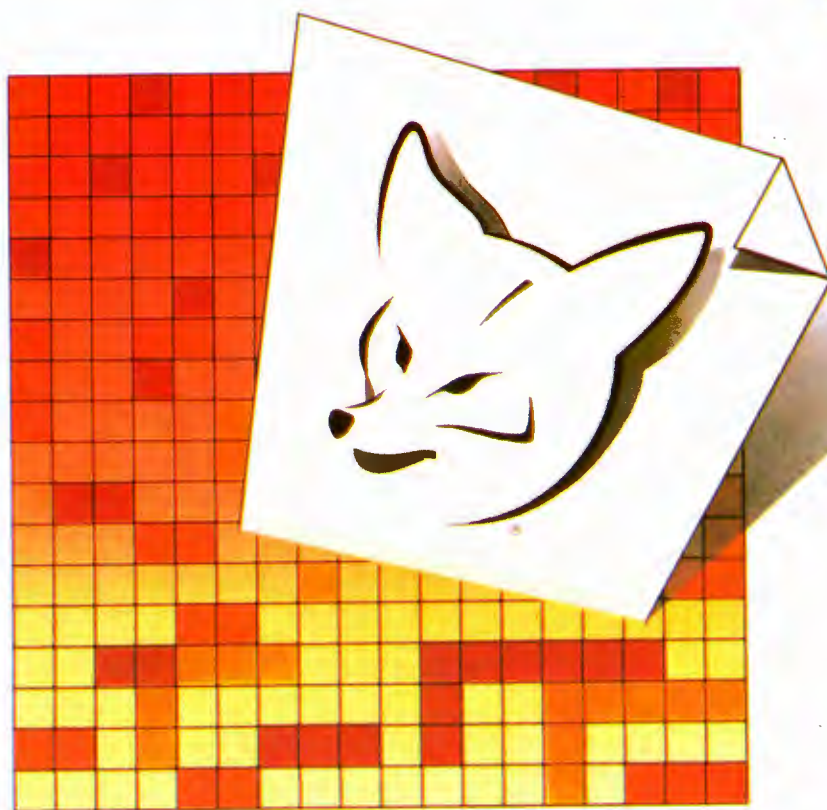
北京希望



电脑公司



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



Microsoft FOXPRO

LIBRARY CONSTRUCTION KIT

For MS-DOS and Windows™

希望汉字系统

汉字系统的希望 UCDOS 3.1

为什么我们总是人们追赶的目标？

1985年，北京希望电脑公司首次推出汉字系统UCDOS 1.0，其优越的性能，得到了广大微机用户的关心、爱护和支持，在此，我们表示衷心的感谢！

随着时间的推移，UCDOS不断地追求汉字系统的最完美境界，终于以最强的功能、最好的兼容性和最惠的价格，遥遥领先于其它汉字系统，成为被追赶的目标。

作为UCDOS家族的最新成员，UCDOS 3.1必将成为汉字系统领域中最耀眼的明珠。

最新奉献

- 首家提供自然码最新5.20版，提供五笔、普通、全拼、简拼、双拼，支持联想输入
- 显示驱动与硬件无关，可支持任意显示设备及任意分辨率和任意种显示颜色下显示汉字、输入汉字等，根据公开接口，各厂商都可以为UCDOS 3.1写显示驱动程序。
如：CGA、EGA、VGA、HGC、TVGA、PVGA等显示设备；支持高速滚屏（硬件）
- 支持大量西文软件直接运行，使用智能的直接写屏和西文制表符识别算法，如实地再现了原版软件的神奇风采
- 支持256色下汉字的正常输入、输出及直接写屏，可在任何扩展图形模式下进行。提供任意尺寸和颜色的矢量汉字显示功能及大量屏幕作图功能
- 全面支持WPS，WPS可在任意高版本DOS和网络环境下运行，并可同时使用UCDOS提供的26种矢量字库，模拟显示和打印速度提高2~3倍
- 支持所有打印机，打印精度最高可达1200dpi，西文制表符打印识别，完美的彩色打印，利用HP打印机特有的数据压缩功能打印，令打印速度提高一至二倍
- 新的Readme，可随意跳转。更完整的手册，为用户及开发人员提供大量方便

UCDOS产品价格如下：

UCDOS 3.1 单用户版	:	980.00 元
UCDOS 3.1 网络版	:	2200.00 元
UCDOS 3.0 单用户版升级	:	100.00 元
UCDOS 3.0 网络版升级	:	200.00 元
UCDOS 3.0 以下单用户升级	:	400.00 元
UCDOS 3.0 以下网络升级	:	800.00 元
注：代理价格面议		

好消息：凡其他汉字系统的正式用户皆可凭用户卡，按半价交叉升级到UCDOS 3.1版！

诚征全国行销伙伴

共同开拓软件市场

希望软件热线：

8422024

8422025

地址：北京海淀路82号希望公司软件部

电话：01-2579873, 2579826, 8422024, 8422025

联系人：周东，张军，夏克

传真：01-8422024, 2561057

收款单位：北京希望电脑公司软件部

开户行：工商银行海淀分理处

帐号：661924-61

邮编：100080



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
海南省海口市安泰电子实业有限公司	8662946
河北沧州市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
江苏淮阴吴通经济技术发展公司	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
上海申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)

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

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。

为使全国各地经销单位能尽快就近获得全套《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 ISSN 1006-4052
 CN24-3411/TP
 广告许可证 京东工商广字 0384
 每期定价 3.80 元 全年定价:45.6 元
 出版日期 1994 年 10 月 8 日

敬告读者

1995 年报刊征订工作已经开始,本刊
 提醒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 4 期

(总第 4 期)

新技术追踪

Microsoft 与 Novell 携手等 13 篇 (4)

趋势与展望

硬磁盘技术的发展趋势及其产品 (8)

软平台

PWIN 3.1 的扩充汉字输入法的制作方法 (12)
 Windows 下 PC 机演奏音乐的一种方法 (18)
 Xenix 系统下文件的压缩和释放 (20)
 利用键盘缓冲区启动不同的操作系统 (23)
 驱动器字母的设置和关闭 (24)

图形图象处理

常见微机图象格式浅析 (26)
 16 色位图文件的 VGA 显示 (31)
 Windows 多文档界面中状态行的实现 (35)

汉字处理

动态小字库的驻留及使用 (38)

编程语言

使用 ADS 为 AutoCAD 做二次开发 (42)
 三次 Bezier 曲线的一种快速软件还原算法 (45)
 立体三维菜单的 C 语言实现 (50)
 Turbo C/C++ 编程技巧——按键的读取 (51)

计算机安全

一种简便而实用的局域网安全策略 (53)

征稿启示

《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都很强的技术杂志,它的主要栏目有:

软平台,该栏目主要介绍操作系统的新功能、新特性以及利用它们进行开发应用的技术技巧。**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。**汉字处理**,该栏目主要介绍汉字处理的编程实用技术。**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其灵活应用技术。**数据库**,该栏目主要介绍数据库系统的新功能、新用法,以及数据库的编程技术、技巧。**计算机安全**,该栏目主要介绍加密技术、反跟踪技术及反病毒技术。**网络与通讯**,该栏目主要介绍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及软件故障的合理恢复方法。**实用软件**,该栏目主要介绍各种工具软件的应用技术与技巧。**硬件维护**,该栏目主要介绍计算机硬件的维护与维修经验、心得体会。

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。同时,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,邮寄稿费时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

应用 C 语言制作软盘密钥的小技巧..... (55)

数据库

成功实现 FoxPro/Windows 环境下的打印技术..... (57)

提高 FoxPro 多平台应用的速度 (63)

FoxPro 2.6 命令及函数 (65)

应用 FoxBASE+2.10 给管理系统增加计算器功能 (67)

软件维护

在 WPS 中实现自动存盘 AutoSave 功能 (69)

Xenix 下与终端有关的软故障及其排除 (70)

增强动态调试程序 DEBUG 的跟踪能力 (72)

硬件维护

多频显示器的特点及维修(一) (74)

实用软件

实用文件分割程序一议 (77)

电脑博士信箱

DOS 与 Windows (81)

电脑反病毒信息公告

反病毒升级信息表 (84)

编读往来

本刊第一届“热心读者”获奖名单 (86)

本刊全部内容的版权归本社所有,未经同意不得转载。

★ 独特的升级技术,最适合中国国情 ★

★ 欢迎加入 ★

★ 帝霸计算机反病毒服务网 ★

★ 热线咨询电话:(01)8123896 ★

新技术追踪

Microsoft 与 Novell 携手

据悉 Microsoft 和 Novell 这两大软件生产厂商最近结束了长期的敌手关系,携手合作。有分析家认为:为此 Novell 将分设四个商业部门,应用产品部门、UnixWare、NetWare 和客户机软件如未来的用于 Internet 的 Mosaic 客户机。Novell 的董事长 Frankenberg 称,他将不再把 Novell 核心业务以外的产品作为重点,如 Novell DOS 7.0。Frankenberg 称 Novell 和 Microsoft 公司联手的主要目的将是两个公司共同合作开发产品以将 Microsoft 的客户机和 Novell 的网络操作系统平台联合起来。为使合作顺利进行,双方都将付出努力。

Chicago 定名为 Windows95

为了让用户易于理解 Windows 各版本的延续性,Microsoft 公司决定将继 Windows 3.0、3.1 及 3.11 等版本后的原称为 Chicago 的版本正式定名为 Windows95。这种定名方法是根据产品名称加发布年份而产生的,但 Microsoft 也表示此定名方法并不意味着 Windows 将会逐年进行升级。

采用 PCMCIA 的远程存取服务器问世

袖针 Modem 生产厂商 Megahertz 公司推出了一种 8 节点远程读写服务器,该服务器采用 PCMCIA Modem。Megahertz MobileLAN 将为拨号网络用户提供远程节点和远程控制能力,并使得网络管理者更容易管理及升级他们的 Modem。内置 14.4Kbps Modem 的服务器将售价 5691 美元。由于包括了 IPX, MobileLAN 支持 Novell 公司的 NetWare 网络操作系统,NetWare Connect 将不再需要。

此外,MobileLAN 将包括一个 SNMP 和两个 MIBs (Management Information Bases) 以支持用户安装和保证安全性并允许监视 Modem 和服务器的运作。该产品的安全措施并不十分理想,MobileLAN 将只包括口令安全。该公司负责人表示,公司计划增加附加安全性如:拨号返回(dial-back)及打包加密(Packet encryption)。

Microsoft 推出新型键盘

Microsoft 公司最近开始销售其第一个键盘产品: Microsoft Natural Keyboard。这种键盘使用起来很舒服, 并有为 Windows 环境设计的特殊键。这种键盘需要 Microsoft 的操作系统如: Windows 3.1 或更高的版本、DOS、Windows NT 的支持。这种键盘目前售价 99.95 美元。该键盘的数字化小键盘(key Pad)和主键盘是分离的, 键盘的前沿高度可以根据使用者的桌、椅高度来调整。键盘有 104 键。

HP 公司提高 DeskJet 系列喷墨打印机的性能

HP 公司改进了 DeskJet 1200C 系列打印机, 新产品在打印质量及内存管理等方面都得到了增强。1200C 及 1200C/PS 喷墨打印机采用 HP 的 ColorSmart 技术。同时打印机能判定打印盒中的墨水是否用尽。HP DeskJet 1200C 具有 6MB 内存, DeskJet 1200/PS 具有 8MB 内存, 增大的内存容量使得这两种打印机打印复杂的图形成为可能。

该系列打印机的标准分辨率为 300dpi, 在只进行黑白打印时, 通过采用 HP 的分辨率增强技术 (Resolution Enhancement) 可使分辨率增加到 600X300dpi。两种打印机都采用 PCL 5 打印机控制语言, 1200C/PS 采用 PostScript Level 2, 35 PostScript Typeface, 及 Adobe 公司的 Adobe Type Manager 技术, 这两种打印机的销售价分别为 1699 美元和 2079 美元。

Microsoft、IBM、Novell 的新选择

面对操作系统向图形化方向的发展, 磁盘操作系统——DOS 这种文本方式的传统操作系统似乎已不再是操作系统的发展主流。面对这种形式 Microsoft、IBM、Novell 对此作法不尽相同。这三家操作系统生产厂商分别向用户推出了新的 32 位系统: Microsoft 将推出 Chicago; IBM 已推出了 OS/2; Novell 将推出 UNIX。IBM 及 Microsoft 表示他们期

望到 1995 年大多数硬件生产厂商将停止预装 DOS 系统, 新版本的 DOS 系统将只作为现存 DOS 系统的升级产品。

IBM PC-DOS 部门经理称 1993 年 IBM 的 PC-DOS 销售了四百万份拷贝, 并期望 1994 年能销售五百万份拷贝, 但实际市场并不乐观, DOS 的市场正在下降。Novell 公司宣布将不再继续的几个台式产品其中就包括 Novell DOS 7。但 Novell 公司的有些人士仍认为在嵌入式系统 (embedded System)、PDA (Personal Digital Assistants) 中 DOS 仍有机会。PDA 将需要具有广泛应用支持以及各种工具和调试器的操作系统, 如 DOS。而 Microsoft 公司将有三种选择: 不再提供新的 DOS 版本; 修改今天的单任务操作系统 MS-DOS 6.22; 或者推出没有 GUI 界面的 Chicago 版本, 从而为现行的 DOS 系统及 32 位 Windows 应用留下一个多任务 DOS。Microsoft 个人计算机操作系统部门经理 Russ Stockdale 表示在 Chicago 销售前后 Microsoft 公司将作出决定。Stockdale 还称用户需要 DOS 以运行现行的应用, 而 Chicago 会做得更好。

IBM 公司认为从现有的 DOS 操作系统支持的 PC 逐渐转到没有 DOS 的 PC 这一过程将很慢。为此, 称之为 PC-DOS 7.0 或 PC-DOS for Workplace Shell 的操作系统产品今年秋天将进入测试阶段, 它将包括 IBM OS/2 中的 Workplace Shell。IBM 的 LaWall 称: 现在一部分用户正使用 Windows, 一部分用户使用 DOS, 尤其是 DOS 上的应用, 用户在寻找新的 32 位环境时, 仍希望这种局面存在。

Ethernet—ATM 转换设备问世

Agile Networks 公司推出了一种称之为 ATMizer 的产品, 该产品是 Ethernet 和 ATM (Asynchronous Transfer Mode) 传输信息的复合转换设备, 其售价为 36000 美元。ATMizer 将支持 1210Mbps Ethernet 结构转换 (frame—Switching) 端口 (10BASE—T 及 10BASE—FL) 及两个 ATM 155Mbps 单元转换 (Cell—Switching) 端口 (可扩展到 6 个)。由于它是为校园网络而设计的, ATMizer 包括能够学习网络拓扑结构的智能软件。ATMizer 支持 TCP/IP、IPX、DECnet、AppleTalk 及 NetBIOS 协议。

Borland 推出基于 Chicago 的工具

Borland 公司在完成其 dBASE for Windows 之后,又开始着手开发公司第一个基于 Chicago 的软件——一系列决策支持工具。

列入开发计划的是 Borland C++ 新版本,现在称之为 Borland C++ 4.5,并将于今年秋天进入市场,这个工具中加入了一个目标构件框架以使 Borland C++ 的应用程序可以转移到 OLE 2.0 上去。据 Borland 公司的 Michael Hyman 称,此项举措使开发人员能够建立 OLE 2.0 服务器和客户机,使用 OLE2.0 Automation 使应用程序自动运行,并且可以创建 OLE 用户控制或 Container。这个目标构件框架可以支持多种 C++ 类,包括 Borland 自己的目标窗口库(OWL),Microsoft 基础类(MFC)等。虽然这一决策支持工具还未完全设计好,但它会包括一个 ReportSmith 的 Chicago 版本以及查询工具,据 Borland 公司产品部副经理 Ken Gardner 称这一决策支持工具系列将需要 Chicago 的支持。

Borland 推出的这一产品系列将会包括多种附加功能,它可以使用户图形化和图表化客户机/服务器信息,还包括一个三维数据显示工具及地理信息系统的绘图软件。其他的一些构件最迟将于 1996 年春季交付使用。

对于 Borland 公司来说开发基于 Chicago 的产品无疑是一个大赌注。Gardner 称“我们当中没有人确切知道 Chicago 何时才能上市,但是这套决策支持软件将会先于 Chicago 走向市场”。

Netwise 推出 Windows 工具

Netwise 最近推出了一种 Windows 工具,这种工具可以容易地将 PowerBuilder 及其他 PC 客户机开发工具与主机 COBOL 应用连接起来。A/IW(Application/Integrator Workbench)使开发者能快速地创建各种应用,这些应用能将台式任务(desktop task)和 IBM 主机 CICS(Customer Information Control System)传输处理应用集成起来,使用户能从自己的 PC 机上读写主机数据。

传统的编程不能和能利用点设备的程序生成器相提并论,典型的客户机服务器应用仅能将如 Pow-

ersoft 公司的 PowerBuilder 等产品与基于 SQL 的数据连接,但大多数公司的数据仍是非关系数据库。现在的 COBOL 程序能处理其他的数据结构。售价 600 美元的 A/IW 与 Netwise 的 TransAccess 的 A/I 一起运行,支持设计、PowerBuilder、Microsoft Visual Basic、KnowledgeWare、ObjectView、C 语言程序及主机 COBOL 的代码生成。以前 TransAccess A/I 需要 SNA(Systems Network Architecture)与主机通讯,但现在通过在主机及 PC 机上运行 TCP/IP, A/IW 已经不再需要 OS/2 服务器在 PC 机和主机之间作为网关。

IBM 扩展路由器设备

IBM 将扩充其 6611 路由器系列,新增加的三个产品包括七个插槽、四个插槽和两个插槽机型。IBM 将提供两端口的 Ethernet 或 Token Ring LAN 适配器。串行卡支持 RS-232C、V.35、X.21 及 RS-422 接口,IBM 最近仅提供了一个单端口 LAN 适配器卡和两端口的串行卡。IBM 和 Proteon 将合作开发一个远程办公路由器,其 RNP(Remote Network Processor)采用 Motorola MC68360 处理器并将以四种结构提供:其中两种将支持两串口连接及一个单 Ethernet 或 Token Ring 接口;其他两种将支持两串口连接、一个 ISDN 接口及一个 Ethernet 或一个 Token Ring LAN。

RNP 将支持 DLSw(Data Link Switching)及包括 IPX 和 AppleTalk 协议。IBM 的 MNP(Multiprotocol Network Program) 1.3 路由器操作软件升级将增加对 DLSw 和 NetBIOS 的支持。支持 DLSw 将使 IBM 路由器用户能在 TCP/IP 主干网上传输 SNA 信息。现有的 6611 路由器软件和新的软件版本兼容。分析家称:IBM 需要支持 HPR(High-Performance Routing)及 APPN(Advanced Peer-to-Peer Networking)网络节点。

分布式数据库管理系统市场在迅速成长

当前主机数据库工具开发商 Platinum Technology 公司已经同 Tivoli Systems 公司建立了伙伴关

系,采用 Platinum 现有工具开发一个新的 IBM、Oracle、Sybase 及其他数据库生产厂商的分布式数据库管理产品。

Oracle 也将宣布三个支持简易网络管理协议(SNMP)的工具并将随 Oracle 7.1 一起提供。

为了能在 Tivoli 公司的管理环境上运行,Platinum 将在 IBM 的 DB2 数据库中采用它自己的 Platinum Detector 分析工具,而在 UNIX 系统的数据库中采用其子公司 Datura Corp 的基于 UNIX 的 SQL SPY 监测工具以运行于 Tivoli 的 TME 环境中。这些新的工具将能够发送数据库监测信息到 TME 的 Enterprise Console 上去,这是一个从单个工作站上监测单个工作站网络上所有系统状态的跨平台系统管理工具。迄今为止,Platinum 的官员们还没有宣布该产品何时能够提供参考价格。

Oracle 公司许诺将努力使现有的数据库管理工具能够同惠普公司的 OpenView、IBM 的 NetView、DEC 公司的 Polycenter 这样的附合 SNMP 的网络管理工具一起运行。采用了 SNMP 之后,这些工具就可以将 Oracle 的数据库监视信息送到网络上去或传递给其他使用 SNMP 通讯的系统管理工具。据称,Oracle 正在研制 Tivoli 所谓的“事件匹配器(event adapter),它将使得 Oracle 数据库能够对 Tivoli TME 发送和接收信息。Oracle 公司已经承诺它将会支持 Tivoli,但是没有明确何时及怎样兑现。

Platinum 和 Oracle 的支持将会巩固 Tivoli 在分布式系统管理市场中的地位。今年八月,Informix 公司宣布它将在数据库服务器产品中采用 Tivoli 的结构构件,Sybase 也做出类似的举措。

Microsoft 将重点采用 OLE 进行数据库连接

在数据库连接技术领域,尽管 Microsoft 公司并没有大张旗鼓地推行它将采取的举措,但 Microsoft 公司的种种举动表明在开放式数据库连接 ODBC 和 OLE 2.0 之间,他们更偏爱后者。来自 SQL Access Group 的主席 Jon Deutsch 说:“我可以肯定 Microsoft 会继续把 ODBC 做为它的数据库策略存取方法,但是他们现在正在大谈 OLE,如果我是个 ODBC 的用户我一定会关心他们在谈些什么”。

Microsoft 于 1991 年制订了 ODBC 标准,现在 ODBC 已被数据库开发者广泛采用。Microsoft 首先

在它的数据库服务器 SQL Server 95 β 版中采用 OLE 为集成数据库管理工具 Starfighter 和 SQL Server 95 之间提供底层通讯。

Starfighter 中使用一种称为 SQL OLE 的新的 DLL 技术,它可以为 OLE 控制提供存取 SQL Server 的功能。Microsoft 已经开始在 Access 和 Visual Basic 开发工具中提供 OLE 控制。

Microsoft SQL Server 产品部经理 Daniel Basica 谈到:“我们已经采用了所有存取 SQL Server 的接口和方法,并把它们封装成一个 OLE 目标”。虽然 Microsoft 还会把 ODBC 作为 SQL Server 95 的基本数据存取方法,但公司已经在 SQL OLE 中建立了 OLE 数据存取的基础。Basica 说未来的 SQL Server 版本将重点采用 OLE 作为数据存取方法,该版本将在 1996 年早些时候为 Cairo 系统提供。为配合 Cairo 版本的 SQL Server,Microsoft 将同时发布 Database Access Object,这是一个取代 ODBC 的 OLE 接口。

Microsoft 已经在 OLE 中采用了 ISVs,因为这是所有公司未来产品的一项关键技术。在 Microsoft 公司的未来产品中 OLE 2.0 是一个核心构件。Oracle 公司已经开始在其产品中应用了一个基于 OLE 的数据存取层,它被称为 Oracle Objects。

Microsoft 坚持 OLE 将不会取代 ODBC,但在 SQL Server 95 中它会和 ODBC 结合起来。Microsoft 的官员说 OLE 将会提供一个高级数据存取方法,这意味着编程者将不用再写原 ODBC API、SQL Server 的 DBLib 接口 SQL 码这样的低级接口。数据存取的 OLQ 控制(OCX)可以隐藏起那些繁琐的低级接口部分。

Apple 推出 486DX2 附加卡

Apple 为加强对 Windows 的支持推出了一组 50MHz 486DX2 附加卡。Apple 公司暗示当明年 Power Macintosh 配备 PCI 总线时,将销售该产品。同时 Apple 表示计划明年初新一代的 Power Macintosh 支持 PCI 总线。Apple 产品经理 Dave Daetz 承认现有结构无助于现有 486 板使用者,PCI 提供许多可能性可不直接接触处理器。

这是一个关键,许多 Macintosh 的用户同时也 Windows 的用户 Apple 的这一产品可以确保这些用户对 PC 软件的投资,因为这样就不需要购买同类软件的 Macintosh 版。

SKILL

硬磁盘技术的发展趋势及其产品

□远 征

硬

磁盘做为计算机系统的重要外部设备,它的诞生并迅速发展在整个计算机的发展历史过程中起到了举足轻重的作用。从其诞生到现在,短短的二十年中,无论是硬磁盘的市场,还是硬磁盘技术都有了很大的变化。从1973年IBM公司制造出世界上第一台Winchester硬磁盘到今天,硬磁盘技术经历长足的发展:容量从MB发展到GB;读写时间不断地缩短;体积不断地减小,稳定性不断提高。在强烈的市场竞争中,硬磁盘的生产厂商也经历了优胜劣汰的发展演变过程。今天虽然由于新型存储技术的诞生并发展如光盘技术在不断地冲击着硬磁盘产品的市场,但是硬磁盘仍然是目前计算机外部存储设备家族中的重要一员。在我国由于兼容机市场的存在,硬磁盘也成了衡量这些兼容机质量好坏的一个重要的技术指标。因此,了解世界计算机存储设备市场上硬磁盘技术的发展趋势及现状,以及硬磁盘的技术指标对于我国兼容机生产厂商、计算机硬件维修人员和广大的计算机用户都是非常重要的。本文将重点讨论硬磁盘产品的市场现状、硬磁盘技术的发展现状和相关的技术指标,并在文末给出了世界上主要硬磁盘生产厂商的产品及其相关参数表(表1)。

一、硬磁盘技术与市场

1. 硬磁盘市场的回顾与展望

从1973年IBM公司研制出第一台Winchester(温式)磁盘驱动器开始,硬磁盘工业在其经历的二十年发展里程中,1993年应该是最为艰难的一年。尤其是1993年第三

季度著名的硬磁盘生产厂商Conner Peripheral亏损3亿7千2万美元,第四季度Maxtor公司亏损1亿2千2百万美元,亏损数额极为惊人。实际上,1993年除了Seagate技术公司尚能保持盈余之外,几乎所有硬磁盘生产厂商的经营都相当不景气。在这种市场不景气和激烈竞争的双重压力下,各个硬磁盘生产厂商纷纷降低各自的硬磁盘生产产量。但时隔不久,由于1993年各个厂商纷纷并大幅度地减产,使得硬磁盘产品的市场从1993年第四季度开始又处于严重的供不应求的缺货状态,似乎硬磁盘市场又出现了欣欣向荣的景色。在这种“繁荣的市场”的影响下,部分硬磁盘生产厂商于今年年初宣布硬磁盘长价。这种通过各公司之间的价格之战导致各公司之间由于减产而造成的市场“繁荣”,事实上也是下一次价格战的重新开始;这种通过“激烈的竞争导致的杀价与人为降低产量——市场缺货,供不应求——供过于求,激烈的竞争而后再杀价”的恶性循环,一直在硬磁盘产业中周而复始地发生着。这一方面反应了硬磁盘市场的成熟,另一方面也反应出硬磁盘产业似乎已不是商业投资的理想园地。

硬磁盘产业不但是高科技、高风险的产业,更是低利润的产业。硬磁盘工业经过80年代初期厂商的纷纷涌入和因此而形成的产业中的高强度竞争,进入90年代,基本上已经形成了固定的格局:美国的五大磁盘生产厂商——Seagate、Conner、Quantum、Maxtor及Western Digital控制80%以上的世界硬磁盘市场。美国IDC公司的统计表明,1993年全球2.5英寸的硬磁盘销售量为493万9千台,其

中 Seagate 占 28%、Toshiba 占 19%、Conner 占 17%、Quantum 占 12%，其余 24% 的市场由 IBM、Areal、Maxtor、WD 所分享。3.5 英寸的硬磁盘全球销售量为 3934 万 6 千台，其中 Quantum 占 25%、Conner 占 22%、Seagate 占 21%、Maxtor 及 Western Digital 占 12%，其余 8% 则由其它厂商所分享，而在 5.25 英寸的硬磁盘市场方面，全球销售量为 93 万 2 千台，其中 Seagate 占 46%、Micropolis 占 20%、HP 占 12%、Fujitsu 占 11% 其余 11% 的市场则由 DEC、Hitachi 等公司分享。预计 1994 年 2.5 英寸的硬磁盘销售量可望突破 600 万台而达到 626 万 7 千台，3.5 英寸硬磁盘则进一步攀高而上达到 4614 万 2 千台，由于 2GB 以下的市场基本上为 3.5 英寸硬磁盘所取代，5.25 英寸硬磁盘的市场将继续呈现进一步萎缩的局面，1994 年销量可能达到 54 万 7 千台。在激烈的市场竞争中，各个硬磁盘生产厂商将呈现出强者越强、弱者越弱的趋势，根据目前世界硬磁盘市场情况看，有一些市场占有率比较小的硬磁盘生产厂商，可能避免不了退出市场的结局。

2. 硬磁盘技术的发展趋势

硬磁盘是计算机系统的一种重要存储设备，其发展趋势自然会随着计算机系统的发展而发展，并收到计算机系统发展的影响，总体而言，其发展方向不外乎几个方面：更小的尺寸、更大的容量、更高的性能、高可靠性、低功耗、低价格。

目前市场上的 2.5 英寸的硬磁盘最高容量为 500MB，在今年下半年相信 1GB 的 2.5 英寸的硬磁盘将会面市，以应合更快速的 CPU，如 Pentium、PowerPC 和小型化工作站市场的需求。1994 年 3.5 英寸硬磁盘的容量由 130MB 增加到 210MB，同时由于 AT 接口的出现，1GB 以下的硬磁盘市场将被其控制。到今年年底单盘片 500MB 的 3.5 英寸硬磁盘渴望面市。此外，由于 Compaq、AST、DELL 等个人计算机领导厂商，相继加强文件服务器产品的市场。可以预见的是，GB 以上的硬磁盘市场将会快速增长。在硬磁盘高度方面，今年的 1 英寸高、超薄型的硬磁盘将会逐渐在 2GB 以下的硬磁盘市场中成为主流，而半高的硬磁盘也将会渐渐取代原来 5.25 英寸的硬磁盘，除了 2GB 以上的高容量的市场，它已经和 8 英寸的硬磁盘一样逐渐退出了市场，走进了历史的长河，这已经是不可改变的事实了。在 PC 机市场方面，由于绿色的计算机风潮的出现、更佳的能源使用效率及电源管理系统，不但是 2.5 英寸硬磁盘的主要课题，同时也是 3.5 英寸硬磁盘重要发展方向。单从硬磁盘驱动器技术的发展来看，IBM 无

疑是这个行业的领先者。IBM 是第一家将磁阻磁头 (Magnetoresistive Head) 应用在硬磁盘的生产上的厂商，而磁阻磁头也正是今天硬磁盘驱动器容量提高的关键所在，当然 Seagate 和 Dow Corning 合作开发的新一代盘片等，都会逐渐日趋成熟，也将为大容量磁盘的发展奠定基础，并在市场上占有一席之地，甚至得到更大的发展。这些新科技的日趋成熟也意味着硬磁盘的发展又迈向了另一个崭新的时代。

二、影响硬磁盘质量的技术因素

就目前硬磁盘技术的发展而言，影响硬磁盘质量的技术因素包括：未格式化/格式化容量 (Unformatting/Formatting capacity)、磁记录密度、高速缓存 (Cache Memory)、硬磁盘接口 (Interface)、数据传输率 (Data Transfer Rate)、电机转速/等待时间 (RPM/Latency)、查找时间 (Seek Time)、功耗 (Power Consumption)、接触启动停止 (Contact Start Stop 即 CSS)、平均失效时间 (Mean Time Between Failure) 等。这些技术因素中，有些也是比较容易混淆的。

1. 未格式化/格式化容量

一般而言，硬磁盘的规格表上第一个参数就是硬磁盘容量，不管是 IBM 兼容机的用户还是 Apple 机的用户他们所使用的操作系统都以 512 字节为一个扇区。通常磁盘的格式化前后容量的比值大约是 5 : 4，也就是说如果一个硬磁盘的未格式化的容量是 100MB，那么它的格式化容量大约为 80MB 左右。同时，一般情况下，除非特别声明否则硬磁盘生产厂商所谓的 1MB 是指 10^6 (1000000B)，同样 1GB 是 1000MB 而不是 1024MB。

2. 磁记录密度

硬磁盘工业的发展及硬磁盘容量的提高，事实上就是在于磁记录密度的提高。一般而言，它是道密度 TPI 与位密度 BPI 的乘积，通常该值越高表示它所用的磁头与磁盘的等级越好。

3. 高速缓存

高速缓存是指在硬磁盘上为了提高硬磁盘的性能而增加的高速存储器，通常有预读 (Readlook ahead) 或自适应式多段 (Adaptive multisegmented) 两种模式，前者多使用于普及型硬磁盘，而后者多见之于工作站级以上的硬磁盘。

4. 硬磁盘接口

接口是指硬磁盘与系统主机的通讯协议，目前

市场上最常见的磁盘接口有 ATA (IDE) 及 SCSI, 前者适应于新的 CPU 及软件的发展, 各个硬磁盘生产厂商已经开发出 ATA-2 (或 Fast ATA, IDE-2) 的方案以解决 AT BIOS 对 ATA 接口的限制。目前包括 Conner、Western Digital、Seagate、Quantum 都已宣称他们将推出 1GB 的使用 ATA 接口的硬磁盘, 今年推出的 ATA 硬磁盘几乎都能支持 PIO (Programmed IO) Mode 3, 其最大数据传输率为 11.1MB/s, 目前 PIO mode 4、5、6 都在定义中, 其性能表现甚至超过目前的 SCSI 接口。SCSI 接口目前主要用于超过 500MB 的磁盘, 现今市场上最大容量为 Seagate Technology 所推出 9GB 的“Elite 9”。几乎现在的 SCSI 硬磁盘都采用 Fast SCSI II 的规范, 差别在于前者的最大同步数据传输率为 10MB/s, 后者为 20MB/s, 若针对现有的 SCSI 接口硬盘连接上的一个缺点, Conner Peripheral、Seagate Technology 及 Sun Microsystems 共同制定的只有一个连接器 SCA (Single connector Attachment) 的 SCSI 硬磁盘, 其中以 Seagate 最积极推动这种单一连接器的硬磁盘, 该公司宣称将推出 500MB 到 2GB 的单连接器硬磁盘。在性能的改善方面, 新版的 SCSI III 规范已经被提出, 预计以光纤 (Optical Fiber) 为介质, 其数据传输率将可达 125MB/s 以上。

5. 数据传输率

通常数据传输率有内部 (Internal) 数据传输率及外部 (External) 数据传输率; 前者是与硬磁盘表面记录密度有关, 目前所有的硬磁盘都使用区域位记录法 (Zone Bit Recording, ZBR; 各家的名称有所不同, 但是他们的的方法都一样, 这种方法是 1987 年由 CDC 的子公司 Imprimis Technology “1989 年 10 月为 Seagate 所购并” 所开发出来的), 传统的磁盘内外圈的大小不同。但是它所储存的数据量是相同的, ZBR 就是充分利用外圈的面积来增加储存容量。内部的传输速率越高表示它的磁盘记录的书局越密, 性能也会越高。而外部数据传输率通常是它的接口传输率, 以 ATA (IDE) 而言, PIO Mode 3 为 11.1 MB/s、DMA mode 2 为 13.3MB/s、SCSI I、II 为 5MB/s、Wide SCSI II 为 20MB/s。

6. 电机转速/等待时间

电机转速/等待时间是一般用户时常忽略的一个参数, 由于硬磁盘在存取数据时, 不可能都处于最理想的状态, 也就是说大多数时间, 磁头在两个读或写操作之间, 由于磁头不一定正好在所欲读或写数据的位置, 因此, 磁头就必须等待所欲存取的磁盘表面转动到磁头所在的位置时, 才能进行读或写数

据的操作, 所以主轴电机转速越快, 等待时间越短。电机转速/等待时间是衡量硬磁盘速度的一个重要因素。一般而言, 减少 1ms 的等待时间相当于缩短 2.5 到 3ms 的查找时间。所以基本上主轴电机的转速越快, 表示硬磁盘的性能越高。当然转速越高, 其功耗也就越大, 磁盘散热的问题也就显得越加突出。从目前市场上出售的硬磁盘来讲, 主轴电机转速最快的是 Seagate 的 Barracuda 系列的 7200PRM, 其主轴电机转速几乎为一般硬磁盘的两倍。

7. 查找时间

和电机转速/等待时间参数相比, 查找时间是一般用户在选购硬磁盘时最易比较和最注意的参数, 一般而言, 查找时间有道间、平均及最大三个标准。所谓道间查找, 顾名思义就是磁头从某一磁道转移到其相邻磁道所需的时间。而平均查找时间就是指所有磁头在读或写数据时可能发生移动的平均时间。最大查找时间是指磁头从磁盘最外边的磁道移动到磁盘最里边磁道所需的时间。值得注意的是, 磁头的任何移动, 其时间的计算都包括磁头稳定的时间 (Settling Time)。

8. 功耗

一般而言, 硬磁盘的功耗只有在便携式计算机系统中才受到格外重视, 但是由于绿色个人计算机 (Green PC) 的发展, 一时之间计算机的绿色风潮吹遍了整个计算机产业界, 为迎合这种形式, 今年推出的 ATA 硬磁盘不但添加了电源管理系统, 而且更强调省电的特性。所谓的电源管理系统就是把硬磁盘的工作状态分成正常操作、停滞、启动等模式, 由系统对硬磁盘电源管理系统根据不同的模式执行指令, 从而决定硬磁盘电源的工作状态来达到减少电源损耗的目的。目前电源管理系统, 仅限于 ATA 接口的硬磁盘, SCSI 接口硬磁盘无此功能。

9. 接触、启动、停止

接触 (Contact)、启动 (Start)、停止 (Stop) 也称 CSS 是指在硬磁盘接通电源启动的时候, 主轴电机转动所带动的空气浮力尚不足以使磁头飞离磁盘表面时, 或者在关闭电源之后主轴电机从断电瞬间到完全静止, 这两段时间内磁头是贴在磁盘表面上磨擦运动的, 虽然磁盘盘片表面有一保护层, 但它总是有一定的寿命限制的, 尤其是新的电源管理系统所带来的次数频繁的开、关机, 使得这种磨损次数增加。如果没有较高的 CSS 的话, 可能硬磁盘不是长时间的数据读写用坏的, 而是频繁的开、关机磨坏的。一般而言, 目前配合绿色个人计算机或是便携式计算机系统的硬磁盘都有至少 40000 次的 CSS。

10. 平均失效时间

平均失效时间并不是产品的平均寿命。一般而言,一个硬磁盘的使用年限都只有五年左右,MTBF 可作为产品可靠性的指标。目前在 GB 级的硬磁盘大多已经过 500000 小时的 MTBF,而一般个人计算机的硬磁盘也有 200000 到 350000 小时不等的 MTBF。

三、结 论

硬磁盘技术及市场的发展,是随着计算机技术的发展而发展的,计算机技术的发展对于硬磁盘技术及市场的发展起着举足轻重的作用。硬磁盘技术自身的发展并完善,将使得硬磁盘市场大有改观。

表 1 主要硬盘生产厂商的产品及其相关参数表

硬盘生产厂商	型号	容量 (MB)	接口	查找时间	信息传输率 (AT, MB/S)	信息传输率 (SCSI, MB/S)	高速缓存 (KB)	RPM/平均等待时间 (ms)	磁头数/盘数	磁道/英寸 (TPI)	位数/英寸 (BPI)	启停柱面	MTBF (小时)
Conner	CAF540A/S	541	AT/SCSI-2	12		10	256	4500/6.67	4/2	3253	62500	40K	300000
Conner	CFP303440	343	AT/SCSI	13	7.5	6	64	4011/7.5	4/2	2553	56800	20	250000
Conner	CFA1080A/S	1080	AT/SCSI-2	10.5		10	256	4500/6.67	8/4	3245	65000		300000
Conner	CFP1060S	1080	F SCSI-2	9R/9.5W		10	512	5400/5.56	8/4	3147	65131		500000
Conner	CFS210A	213	AT	14	7.5	N/A	32	3600/8.3	2/1	2774	58566	20K	250000
Conner	CP-30254	251	AT	14	5	N/A	64	4542/6.7	4/2	2450	52270	20K	>250000
IBM/Adstar	DSAA/S3540	548	AT/F SCSI-2	12	8.3	10	96	4500/6.67	4/2	4300	83200		300000
IBM/Adstar	DSAA/S3360	365	AT/F SCSI-2	12	8.3	10	96	4500/6.67	2/1	4300	83200		300000
IBM/Adstar	DFHS-32160	2160	F SCSI-2	8		10/20	512	7200/4.17	8/4	4351	121000		1000000
IBM/Adstar	MS-32600/32	2160	F SCSI-2	8	76-98	10/20	512	5400/5.56	8/4	4351	128900		1000000
IBM/Adstar	0062-S12/SID	1050	AT/F SCSI-2	8.6/10.1		5/10	512	5400/5.56	6/3	4077	86900		800000
IBM/Adstar	WDS-3200	210	SCSI	12	N/A	5	32	4320/6.94	4/2	2222	40000		150000
IBM/Adstar	H3256-A3	256	AT	14	8.3	N/A	96	3600/8.3	3/2	2800	55000		250000
Maxtor	7546A/S	546	AT/SCSI	14	9	5	64	3551/8.45	4/2	2672	50660		>300000
Maxtor	7345A/S	345	AT/SCSI	14	9	5	64	3551/8.45	4/2	2672	50660		>300000
Maxtor	7213A/S	213	AT/SCSI	15	9	10	64	3551/8.45	4/2	1973	42700	20K	>300000
Maxtor	7245A/S	246	AT/SCSI	16	9	10	64	3551/8.45	4/2	2340	42600	40K	>300000
Quantum	LPS540	540	AT/F SCSI-2	12	11(MDA13)	10(6 Async)	128	4500/6.67	4/2	2875	62600		300000
Quantum	Empire 540	540	F SCSI-2	9.5R/11W		10/20	512	5400/5.56	4/2	3014	61300		500000
Quantum	LPS340	340	AT/F SCSI-2	12	6(MDA13)	10(6 Async)	128	3600/8.3	4/2	2670	49273		350000
Quantum	LPS240A/S	245	AT/SCSI	16	5	10	256	4306/7	4/2	1900	38000		>250000
Quantum	LPS270	256	AT/SCSI-2	12	6(MDA13)	10	128	4500/6.7	2/1	2875	62600	40K	300000
Seagate	ST3655A/N	528/545	AT/F SCSI-2	12	5.5/11.1	10	256	4500/6.67	5/3	3000	53000	20K	250000
Seagate	ST5660A/N	545	AT/SCSI-2	12	11.1	10	256	4500/6.67	4/2	3309	68000	40K	300000
Seagate	ST3390A/N	341	AT/F SCSI-2	12	5.5/11.1	10	256	4500/6.67	3/2	3000	53000	20K	250000
Seagate	ST3391A	341	AT	13	5/11.1	N/A	120	3811/7.87	4/2	3081	52500	40K	300000
Seagate	ST32430	2147	F SCSI 2	9R/10.5W		10/20	512	5400/5.56	9/5	4200	80000		800000
Seagate	ST32550	2147	F SCSI 2	8R/9W		10/20	512	7200/4.17	11/6				800000
Seagate	ST3243A	214	AT	16	6	N/A	32	3811/7.87	4/2	2165	38256	20K	250000
Seagate	ST3250A	214	AT	14	5/11.1	N/A	120	3811/7.87	2/1	3081	52500	40K	300000
Seagate	ST3290A	261	AT	16	6	N/A	32	3811/7.87	4/2	2165	38256	20K	250000
Seagate	ST3291A	273	AT	13	5/11.1	N/A	120	3811/7.87	4/2	3081	52500	40K	300000
Seagate	ST9550AG	455	AT	16	11.1	N/A	120	3980/7.54	8/4	3282	59124	50K	300000
WD	WDAC2540	540	AT	6/11	5.75	N/A	128	4500/6.67	4/2	3300	55200		250000
WD	WDAC2340	340	AT	<13	5.75	N/A	120	3322/9	4/2	2481	55200	20K	>300000

* MBF:格式化容量以 MB 为单位

SKILL

PWIN 3.1 的扩充汉字输入法的制作方法

□曹国钧

美

国 Microsoft 公司在 1993 年 10 月推出了它的杰出之作 Windows 3.1 的简体中文版——PWIN 3.1, 该中文版本的出现为 Windows 操作系统在中国的流行提供了强大的基础。PWIN 3.1 中文版已提供了国标、全拼、双拼等三种汉字输入方法, 但未提供其它汉字输入方法, 这给用户在使用 PWIN 3.1 时带来了不便。实际上, 用户不用担心, 在 PWIN 3.1 中提供了通用编码的通用接口, 按照该接口编制编码输入法都能在 PWIN 3.1 中正确使用。PWIN 3.1 提供了通用编码接口文件 WINBM.IME, 用编码编译器 CONVMB.EXE (在 SYSTEM 子目录中) 可以将用户自行定义的用户码 (最多不超过 20 个) 编译成 Windows 可辨识的编码文件 *.IME 形式, 并能直接挂到 Windows 中文环境中。

一、PWIN 3.1 中的通用编码格式

在 PWIN 3.1 中, 用户自定义的编码格式应按如下方法书写, 其中的 && 为注解语句:

```
WBS.TXT:      && 输入法定义的文件名
[Description] && 自定义码的描述
Name=自定义码 && 输入法名称, 将在输入
               && 法菜单中显示
MaxCodes=4    && 修改成编码最大长度
UsedCodes='abcdefghijklmnopqrstuvwxy'
               && 用户使用的实际编码, 如
               && a-Z, ; 等
WildChar=Z     && 用户通配符, 一般为 Z、
               && ?, * 等
Sort=1         && 形成输入法编码时, 从第一
               && 个开始排序;
               && 若 Sort=0, 不排序
```

```
[Text]      && 编码的详细定义, 词组在前, 编码在
               && 后, 即其格式为: 汉字或词组+编码,
               && 编码不能超过 MaxCodes 规定的长度。
```

```
.....
&& 这是 Microsoft 公司的一个练习, 仅作为参考定义
&& 一个很长的词组 abcd
&& 学习 ipnu
.....
```

上例是 PWIN 3.1 主目录\PWIN 中文件 WBS.TXT 的部分内容。

实际上, 用户可根据上例的编码格式定义用户编码, 如下面是一个笔者定义的所谓“用户码”的例子:

```
USET.TXT:
[Description]
Name=用户码
MaxCode=4
UsedCodes='abcdefghijklmnopqrstuvwxy'
WildChar=z
Sort=0
[Text]      && 用户定义编码
邓小平 dxp
中国共产党 zggc
重庆医药设计院 cqyy
```

在 PWIN 3.1 中, 用 CONVMB.EXE 编译程序将 WBS.TXT 和 USER.TXT 分别编译成 WBS.IME 和 USER.IME, 编译后 PWIN 3.1 将自动形成一个扩充编码输入法管理的初始化文件 WINMB.INI (文件内容如下), 用 Ctrl+Shift 可选择自定义码输入法和用户码输入法:

```
[MBInput]
MBInput0=D:\PWIN\SYSTEM\WBS.IME
MBInput1=D:\PWIN\SYSTEM\USER.IME
.....
```

用户也可以在 WINMB.INI 中自行加入外部扩充汉字输入方法。

二、三种编码文件的形成方法

下面笔者提供了汉字系统 CCDOS 中重码较少的三种输入方法:五笔字型输入法 WBX.IME、首尾输入法 SWM.IME 和表型码输入法 BXM.IME 的编码文件的形成方法。

1. 五笔字型 WBX.IME 的制作方法

笔者利用程序 WBXDM.CPP(见程序 1),可以在现有的 SPDOS 汉字系统下自动形成五笔字型单编码文件 WBZX.TXT,并利用 SPDOS 6.0F 的五笔字型内部词组覆盖文件 WBX.OVL(见程序 2: WBXCZ.CPP),产生一万一千多条五笔字型词组文件 WBZX.CZ,将 WBZX.TXT 和 WBZX.CZ 合并(执行命令 COPY WBZX.TXT+WBZX.CZ WBX.TXT)为编码文件 WBX.TXT,并对 WBX.TXT 用 PWIN 3.1 的 CONVMB.EXE 编译后形成了比较通用的 PWIN 上使用的五笔字型输入方法 WBX.IME。其中程序 WBXDM.CPP 和 WBXCZ.CPP 采用 Borland C++或 Turbo C++编译器编译成为 WBXDM.EXE 和 WBXCZ.EXE,在 CCDOS 汉字系统下运行。

2. 首尾码 SWM、IME 的制作方法

笔者给出的首尾码的编码文件是从目前流行的 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 XXXXXX XXXXXX
 首尾码 2 首尾码 1

第二个字: Xxxxxx xxxxxx xxxxxx
拼音码 3 拼音码 2 拼音码 1

其中:X(大写)为拼音的高频字的标志,X=0或1;由上面的格式可以看出,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(□) 27(□) 1(A)

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

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, 从而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(见程序 3)完成上述方法,在 Turbo C 或 Borland C++ 中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 首尾码的编码文件。在 PWIN 中运行通用编码编译器 CONVMB. EXE,将 SWM. TXT 编译成 Windows 可辨识的首尾码输入文件 SWM. IME。在 Windows 的主组中选择控制面板肖像后,进入控制面板配置,然后选择输入法的肖像,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即可。

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 个未定义汉字, 它们的内码分别为 D7FA 至 D7FE。

3. 表型码输入法 BXM.IME 的制用方法

笔者采用 SPDOS 5. X 或 SPDOS 6. 0F 中的外部扩充输入方法表型码文件 BXM. COM 中得到其编码,经分析,编码从文件的偏移 02AA 到 51F9,每一个汉字编码占用 3 个字节,从这 3 个字节中就能获取到表型码的编码,其获取格式如下(其中 bb1 至 bb4 见程序 4 中的含义):

第一字节: $\begin{array}{c} \text{X X X X} \\ \hline \text{bb1} \end{array}$ $\begin{array}{cc} \text{X X} & \text{X X} \\ \hline \text{bb1} & \text{bb2} \end{array}$

第二字节: $\frac{X\ X\ X}{bb2}\ \frac{X}{bb3}\ \frac{X\ X\ X\ X}{bb3}$

第三字节: $\begin{array}{c} \text{X X X} \\ \hline \text{bb4} \end{array}$ $\frac{\text{X X X X}}{\text{bb4}}$

下面就是形成表型码的编码文件 BXM.DAT 的具体方法：

(1) 采用 DEBUG.COM 形成 BXN.COM 文件中格式编码 BX

C:\WPS>DEBUG<BBB.TXT>BX

其中 BBB.TXT 的内容如下:

N BXM.COM

L100

D 2AA 51F9

Q

(2) 整理 BX 文件,使每一个汉字的编码 BX 占用一行,为用户使用的方便,笔者给出了一个自动

整理方法,即在 PE2 中执行宏文件 BXM. PE2(见程序 4),在 PE2 的命令行上键入 BXM. PE2 后,将整理出 BXM 文件,共 6768 行;当然,也可以用 WS、WPS 或 PE2 等软件进行整理,将一些无用的行(如 D 2AA51F9;Q)或列(如地址,最右边的 ASCII 码表示等)删除,并用 ^QA 把“-”符号换成空格,再定义左边界和右边界分别为 1 和 9 即可。

(3) 在 FoxBASE+ 或 FoxPro 中建立如下结构的数据库 BXM. DBF:

字段名	类型	宽度	小数位
BX	C	8	
HZ	C	2	
DM	C	4	

其中 BX 为 BXM 中的编码,HZ 为编码对应的汉字,DM 为实际输入的 ASCII 形式的编码,长度为 4。

(4) 编制 BX 向 DM 转换程序 BXM. PRG(见程序 6),形成了 HZ+DM 的编码文件 BXM. DAT,再将文件头 BXM. TOU(见程序 7)与 BXM. DAT 合并(用 COPY BXM. TOU+BXM. DAT BXM. WIN 命令建立)成 BXM. WIN,则 BXM. WIN 就是可被 PWIN 3.1 的编码编译程序 CONVMB. EXE 编译的编码文件。

三、结束语

上面给出的编码文件的获取方法也可以推广到其他汉字输入方法,如自然码、六笔形、电报码等输入方法,同时,对这些汉字输入方法的编码文件,也能象五笔字型输入一样加入一些词组,以提高输入速度。另外,该方法也能运用于“中文之星”(CStar For Windows)中。

源程序清单如下:

```

/*****
* 程序 1 的名称:WBXDM. CPP      *
* 功能:获取五笔字型单码文件    *
*****/
#include <stdio.h>
#include <conio.h>
#include <process.h>
char key[]="abcdefghijklmnopqrstuvwxyz";
/* 键盘缓冲区的首指针 */
char far *begin=(char far *) (0x40*16+0x1a);
/* 键盘缓冲区的尾指针 */
char far *end=(char far *) (0x40*16+0x1c)
void make __hz(unsigned char *p __);
/* 编码产生的汉字串 */
void clearbuf(); /* 清除系统键盘缓冲区 */
void send __ to __buf(char c,char c1);

```

```

/* 发送一个字符到键盘缓冲区 */
FILE *fp;
/* 清除键盘缓冲区中的内容 */
void clearbuf()
{
    *begin=0x1e; *end=0x1e;
    return;
}
/* 将键码送到系统的键盘缓冲区中 */
void send __ to __buf(char c,char c1)
{
    *((char far *) (0x40*16+*end))=c;
    *((char far *) (0x40*16+*end+1))=c1;
    *end=*end+2;
    if (*end>0x3e) *end=0x1e;
    return;
}
/* 根据五笔字型的编码模拟手工方法获得其汉字 */
void made __hz(unsigned char *p __)
{
    unsigned char buffer[256], *p,c;
    int i;
    for (c='1';c<='9';c++)
    {
        p=p__;
        clearbuf();
        while(*p!=0){
            send __ to __buf(*p,*p);
            p++;
        }
        if (p-p__<4) send __ to __buf(' ',' ');
        send __ to __buf(c,c);
        send __ to __buf(13,13);
        send __ to __buf('@','@');
        i=0;
        while((buffer[i++]=getch())!='@');
        buffer[i]=0;
        if (buffer[0]!='@') break;
        if (buffer[0]==c) break;
        i=0;
        while( buffer[i]>127) i++;
        if (buffer[i]==c)
        { buffer[i]=0;
            fprintf(fp,"%s%s\n",buffer,p__);break;}
        buffer[i]=0;
        fprintf(fp,"%s%s\n",buffer,p__);
    } return;
}
int main(void)
{ unsigned char buff[4];
  int i;
  char i0,i1,i2,i3;
  unsigned char *p;
  printf("\n\n 在 SCDOS 系统中为 PWIN 提取五笔字型单码文件");
  printf("\n 本程序将在磁盘上形成 WBZX. TXT 文件\n");
  printf("\n\n 注意:进入 SPDOS 汉字系统前应调入 WBX 模块,并设置 CapsLock 为小写状态\n");
  printf("按 ALT+F4 进入五笔字型输入状态\n");
  printf("\n 按回车键,程序继续;按 ESC 键,程序终止.....\n");
  if (getch()=='27')exit(1);
  printf("\n\n 程序正在进行,注意:在运行过程中不能按下任意键.....");
}

```

```

if ((fp=fopen("wbzx.txt","w"))==NULL) {
    puts("error!!!");exit(1); }
fprintf(fp,"[Description]\n");
fprintf(fp,"Name=五笔字型\n");
fprintf(fp,"MaxCodes=4\n");
fprintf(fp,"UsedCodes=\\'%s\\'",key);
fprintf(fp,"WildChar=z\n");
fprintf(fp,"Sort=1\n");
fprintf(fp,"[Text]\n");
printf("1.生成一级简码.....");
for (i0=0;key[i0]! =0;i0++){ buff[0]=key[i0];
buff[1]=0;
make __ hz(buff);}
printf("OK!!! \n");
printf("2.生成二级简码.....");
for (i0=0;key[i0]! =0;i0++){
    for (i1=0;key[i1]! =0;i1++){
        buff[0]=key[i0];
        buff[1]=key[i1];
        buff[2]=0;make __ hz(buff);
    }
}
printf("3.生成三级简码.....");
for (i0=0;key[i0]! =0;i0++){
    for (i1=0;key[i1]! =0;i1++){
        for (i2=0;key[i2]! =0;i2++){
            buff[0]=key[i0];
            buff[1]=key[i1];
            buff[2]=key[i2];
            buff[3]=0;make __ hz(buff);}
        printf("OK!!! \n");
        printf("4.生成其他简码.....");
        for (i0=0;key[i0]=0;i0++){
            for (i1=0;key[i1]! =0;i1++){
                for (i2=0;key[i2]! =0;i2++){
                    for (i3=0;key[i3]! =0;i3++){
                        buff[0]=key[i0];
                        buff[1]=key[i1];
                        buff[2]=key[i2];
                        buff[3]=key[i3];
                        buff[4]=0;make __ hz(buff);}
                    Printf("OK!!! \n");
                    printf("\n 程序运行结束!!!");
                    fclose(fp);
                    return 0;
                }
            }
        }
    }
}

```

```

/*****
* 程序 2 的名称:WBXCZ. CPP
* 功能:获取五笔字型外部词组文件
*****/
#include<stdio. h>
#include<stdlib. h>
#include<string. h>
#include<math. h>
int main(void)
{
    int zf,zf1;
    unsigned int n,zm,zm1,zm2,zm3;
    FILE * f1,* f2;
    if ((f1=fopen("C:\\WPS30F\\wbx. owl","r+b"))==NULL)
    { printf("WBX. OVL doesn't Exist!!!%c",07);

```

```

        exit(-1);
    }
    if ((f2=fopen("wbxcz","w"))==NULL)
    { printf("WBXCZ doesn't Create!!!%c",07);
        exit(-2);
    }
    printf("First;Create File WBXCZ,Please Wait....\n");
    fseek(f1,0x32,SEEK __ SET);
    zm=65;putc(zm,f2);
    zf=fgetc(f1);
    zm1=(zf>>2)+64;
    putc(zm1,f2);
    zf1=fgetc(f1);
    zm2=abs((zf&3)<<3)+(zf1>>5)+64;
    putc(zm2,f2);
    zm3=abs((zf1&31))+64;
    putc(zm3,f2);n=0;
    do { n++;
        zf=fgetc(f1);
        if (zf==0xff) {
            zf1=fgetc(f1); if (zf1==0xff) zm++; putc('\n',f2);
            continue;}
        if (zf<0x7f)
        { fputc('\n',f2);putc(zm,f2);
            zm1=(zf>>2)+64;putc(zm1,f2);
            zf1=fgetc(f1);
            zm2=((zf<<14)>>14)*8+(zf1>>5)+64;
            if(zm2<64) zm2+=32;putc(zm2,f2);
            zm3=((zf1<<11)>>11)+64;
            if (zm3<64) zm3+=32;
            putc(zm3,f2);continue;
        } else fputc(zf,f2);
    }while(zm<90);fclose(f1);fclose(f2);
    char * s1,* s2,* s3;
    f1=fopen("wbxcz","r");/* WBX. CZ IS IN USE OF PWIN 3. 1V */
    f2=fopen("wbzx. cz","w");
    printf("Second Step;Create WBZX. CZ,Please Wait.....\n");
    while (fgets(s1,256,f1)!=NULL)
    {
        int i=strlen(s1);s1[i-1]='\0';
        strncpy(s2,s1,4);s2[4]='\0';
        s3=s1+4;strncat(s3,s2,i);
        fputs(s3,f2);fputs("\n",f2);
    } fclose(f1);fclose(f2);return 0;
}

```

```

/*****
* 程序 3 的名称:SWM. CPP
*****/

```

独特的升级技术,最适合中国国情

欢迎加入

帝霸计算机反病毒服务网

热线咨询电话:(01)8123888


```
#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"); printf("\n");
printf("MKCODE 1.0V. Written By CGJ.\n");
printf("Processing. Please wait. ....\n");
if ((fp=fopen("c:\\213\\cccc.com", "r+b"))==NULL)
{ printf("In C:\\213, there doesn't exist 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='abcdefghijklmnopqrstuvwxy'";
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);
write(handle2, &hc, 2);
char str7[]=" [Text]";
write(handle2, &str7, 6); write(handle2, &hc, 2);
unsigned char *buffer, 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:转化成为小写字符
swm2=(( *buffer>5) & 0x07) | * (buffer+1<<3 | 96;
pym1= * (buffer+2) & 0x1f | 96;
if(pym1==' ' | | 1 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;
}
```

```
/* * * * * *
* 程序 4 的名称:BX.M. PE2 *
* * * * *
[df] ;快速演示(DRAW FAST)
set backup 00 ;保留删除的次数为 00
e BX ;编辑文件 BX
```

```
[ca][mb] ;将光标移到数据区,并形成块标记
[line 1272] ;光标移到第 1272 行
[right 10] ;光标向右移动 10 列
[mb] ;定义块
[dm] ;删除块
[line 1] ;光标移到第 1 行
[column 48] ;光标移到第 48 列
[mb] ;定义块
[line 1272] ;光标移到第 1272 行
[column 67] ;光标移到第 67 列
[mb] ;定义块
[dm] ;删除块
[line 1] ;光标移到第 1 行
[column 1] ;光标移到第 1 列
c /- / * ;全局地将“-”替换为空格
[ml] ;定义行块
set margins 1 9 1 ;定义左右边界分别为 1 和 9
[line 1] ;光标移到第 1 行
[dl] ;删除行块
[ml] ;定义行块
[column 1] ;光标移到第 1 列
[rf] ;重新排版
[um] ;消除块标志
[line 6769] ;光标移到第 6769 行(最后一行)
[dl] ;将最后一行的“Q”删除
[cc] ;光标移到命令行
file bxm ;以文件名 BX.M 存盘,并退出 PE2
```

```
/* * * * * *
* 程序 5 的名称:BX.M. PRG *
* (在 FoxBASE+或 FoxPro 中运行) *
* 功能:将 SPDOS 5.X 或 6.0F 中的 BX.M. COM *
* 转化为表型码的输入代码形式 *
* * * * *
set safe off
set talk off
set stat off
set scor off
clea all
* 16 进制数的对应 2 进制数形式
r0="0000"
r1="0001"
r2="0010"
r3="0011"
r4="0100"
r5="0101"
r6="0110"
r7="0111"
r8="1000"
```

★ 帝霸计算机反病毒服务网 ★

★ 专用软件(普及版) ★

★ 特价优惠:20 元/套(含邮费) ★

★ 热线咨询电话:(01)8123896 ★

```

r9="1001"
ra="1010"
rb="1011"
rc="1100"
rd="1101"
re="1110"
rf="1111"
* 2 进制数的 ASCII 代码(即表型码编码)的表示形式
m00000=" "
m00001="a"
m00010="b"
m00011="c"
m00100="d"
m00101="e"
m00110="f"
m00111="g"
m01000="h"
m01001="i"
m01010="j"
m01011="k"
m01100="l"
m01101="m"
m01110="n"
m01111="o"
m10000="p"
m10001="q"
m10001="r"
m10011="s"
m10100="t"
m10101="u"
m10110="v"
m10111="w"
m11000="x"
m11001="y"
m11010="z"
m11011="1"
m11100="2"
m11101="3"
m11110="4"
m11111="5"
sele 1
use bxm
zap
* 将表型码的编码放到字段 BX 中
appe from bxm. sdf
go top
* 形成 6768 个汉字(内码从 B0A1 到 F7FE)
i=176
do while i<=247
  j=1
  do while j<=94
    repl hz with chr(i)+chr(160+j)
    skip
    j=j+1
  enddo
  i=i+1
enddo
* 表型码的代码 DM 形成
@1,1 say "Now Running..."
keyboard chr(13)
go top

```

```

do while. not. eof()
  ff1=subs(bx,1,1)
  ff2=subs(bx,2,1)
  ff3=subs(bx,4,1)
  ff4=subs(bx,5,1)
  ff5=subs(bx,7,1)
  ff6=subs(bx,8,1)
  bb1=subs(r&ff1,2)+subs(r&ff2,1,2)
  bb2=subs(r&ff2,3,2)+subs(r&ff3,1,3)
  bb3=subs(r&ff3,4)+r&ff4
  bb4=subs(r&ff5,4,1)+r&ff6
  repl dm with m&bb1+m&bb2+m&bb3+m&bb4
  * ?? hz
  skip
enddo
* 形成仅有 HZ 和 DM 两个字段
COPY TO bxm__bx
USE BXM__BX
* ^ N 和 ^ W 应在行编辑软件 EDLIN.COM 中直接输入
KEYBOARD "^ N"
KEYBOARD "^ W"
KEYBOARD CHR(13)
MODI STRU
COPY TO BXM DELI
clos data
retu

/* **** */
* 程序 6 的名称: BXM. PE2-
* **** */
[.df]
e bxm. txt
[.ca]
[line 1]
[column 1]
c/"// *
[line 1]
[column 1]
c/,// *
[.cc]
file bxm. dat
:程序 7 文件名: BXM. TOU
[.Description]
Name=表型码
MaxCodes=4 && 编码最大长度为 4
UsedCodes='abcdefghijklmnopqrstuvwxyz12345'
WildChar=?
Sort=1
[.Text]

```

SKILL

★ 帝霸计算机反病毒服务网 ★

★ 专用软件(普及版) ★

★ 特价优惠: 20 元/套(含邮费) ★

★ 热线咨询电话: (01)8123896 ★

Windows 下 PC 机演奏音乐的一种方法

□王 敏

在

DOS 下编程,无论是 BASIC 还是 C、C++ 都提供了演奏音乐的相应的语句或函数,可使计算机利用扬声器发出特殊音乐或奏出悦耳的歌曲,而在 Windows 下编程,情况就不一样了,若要达到同样的目的,就必需利用 SOUND.DRV 这个动态连接库才行。SOUND.DRV 中一共提供了 18 个 DLL 子程序(Windows 3.1)包括:OpenSound、CloseSound、SetVoiceSound、StartSound、SetVoiceQueueSize、SetVoiceNote 等,笔者用 Borland C++3.1 编写了一个音乐程序,介绍怎样利用这些子程序来使计算机奏出音乐。以前笔者曾经介绍过直接将发声频率和持续时间输入来演奏音乐的方法,简称直接法。本文介绍间接用音阶(Octave)和音符(note)输入,简称间接法。

虽然直接法功能不错,但是在谱曲时还是不太方便,因为必须经常查表。GW-Basic 中有一功能更强的指令“PLAY”,它允许我们以更接近音乐符号的方式来设置声音,例如 C、D、E、F、G、A、B 分别代表 Do、Re、Mi、Fa、So、La、Si。同样,我们也可以利用 SOUND.DRV 中的 SetVoiceNote DLL 子程序来做出类似“PLAY”的功能。

本方法与直接法的主要不同点是用接近音乐符号的方式来谱曲,每个音阶都有 12 个音符:C C# D D# E F F# G G# A A# B,其中 # 代表升半音,所以“Do Re Mi Fa So La Si”可用 CDEFGAB 来表示,至于持续时间的长短(即全音符、二分音符、四分音符、八分音符、十六分音符等)则用数字附在该字母的后面,例如 C2 G4 F8 A16 分别代表二

分音符的 Do,四分音符的 So,八分音符的 Fa 和十六分音符的 La。音阶“03”表示第 3 音阶,“04”表示升到第 4 音阶等。

下面介绍一下利用间接法使计算机演奏音乐需用到的几个 DLL 子程序:

OpenSound 打开发声装置,没有参数,返回值为—整数;

CloseSound 关闭发声装置,没有参数,没有返回值;

StartSound 启动发声装置,没有参数,返回值为—正数;

SetVoiceNote 共有四个参数:

第一个参数:nVoice 表示送入发声队列的号码;

第二个参数:nValue 用来送入数字音符(范围:1—84,7 个音阶,每音阶有 12 个音符总共 $7 \times 12 = 84$);

第三个参数:nLength 用来送入每个音符持续时间,例如全音符为 1,二分音符为 2,四分音符为 4,八分音符为 8,十六分音符为 16;

第四个参数:用点来表示持续时间,通常送入 0 即可。

SetVoiceQueueSize 共有二个参数:

第一个参数:nVoice 表示送入发声队列的号码;

第二个参数:nBytes 表示容量,缺省值 192 字节,可容纳 32 个音符,本曲将它定为 2048 以容纳较长的歌曲。

程序用 Borland C++3.1 编写,可运行于 Windows 3.0 或 3.1 下。项目文件:MUSIC2.PRJ 包含 MUSIC2.CPP MUSIC2.DEF,经编译运行以后,只需在窗口 SamplePlayMusicUsingSpeaker 内按一下鼠标左键,就可听到从扬声器中发出的乐曲。

music2.def 程序清单如下:

```
EXETYPE Windows
CODE PRELOAD MOVEABLE DISCARDABLE
DATA PRELOAD MOVEABLE MULTIPLE
HEAPSIZE 4096
STACKSIZE 5120
IMPORTS
    sound. OpenSound
    sound. CloseSound
    sound. StartSound
    sound. SetVoiceSound
    sound. SetVoiceQueueSize
    sound. SetVoiceNote
```

music2.h 文件清单如下:

```
char * YueQu1="G4G8A16G16E4C4D4D8E16D16C2";
char * YueQu2="03A4A8G804D4E8D8C4C8C8C2";
char * YueQu3="C4C8D8E2D8C8D8E803G2
char * YueQu4="A4A804C803A8G8E4G1";
char * YueQu5="C4C8D8G8E8E4A8G4E8G2";
char * YueQu6="E4E8C8D8E8G8F8E1";
char * YueQu7="D4D8E8G4E4D8E16D16C8D8E2";
char * YueQu8="03A4A8G8A8G8A804C8D1";
char * YueQu9="G4G8E8D4C4D8E16D16C8D8E2";
char * YueQua="G4G8E8D8C8D8E8G1";
char * YueQub="E4E8A8G4E4D8E8D8C8C403A4";
char * YueQuc="04D4D8C8E8D8C803A8G1";
char * YueQud="04G4G8A16G16E4C4D4D8E16";
char * YueQue="D16C403A4A4A8G804D4E8D8C4C8C8C2";
const char * frerange="C#D#E#F#G#A#B";
const char * note="0123456789";
char * songstring;
```

music2.cpp 源程序清单如下:

```
#include <stdio.h>
#include<string.h>
#include <owl.h>
#include "music2.h"
#include<math.h>
class TMyApp:public TApplication
{public:TMyApp(LPSTR AName,HINSTANCE hInstance,
HINSTANCE hPrevInstance,LPSTR lpCmdLine,
int nCmdShow)
: TApplication ( AName, hInstance, hPrevInstance, lpCmdLine,
nCmdShow){}
virtual void InitMainWindow(); };
__CLASSDEF (TMyWindow)
class TMyWindow:public TWindow
{public: TMyWindow(PTWindowsObject AParent,LPSTR ATitle)
:TWindow(AParent,ATitle)
{OpenSound();
SetVoiceQueueSize(1,1024);
songstring=YueQu1;
strcat(songstring,Yuequ2);
strcat(songstring,Yuequ3);
```

```
strcat(songstring,Yuequ4);
strcat(songstring,Yuequ5);
strcat(songstring,Yuequ6);
strcat(songstring,Yuequ7);
strcat(songstring,Yuequ8);
strcat(songstring,Yuequ9);
strcat(songstring,Yuequa);
strcat(songstring,Yuequb);
strcat(songstring,Yuequc);
strcat(songstring,Yuequd);
strcat(songstring,Yueque);
};
virtual BOOL CanClose();
virtual void WMLButtonDown(RTMessage Msg)
=[WM__FIRST+WM__LBUTTONDOWN];
};
BOOL TMyWindow::CanClose()
{ CloseSound();return 1;}
void TMyWindow::WMLButtonDown(RTMessage Msg)
{ char singlechar,*single,checkchar;
int count=1,octave=3,playlength=4;
int dots=0;int pitchno,tempnum;
while (singlechar=* (songstring++)) {if((singlechar>=65)
&&. (singlechar<=90))
{ switch(singlechar)
{case 'C':
case 'D':
case 'E':
case 'F':
case 'G':
pitchno=(singlechar-67)*2;break;
case 'A': pitchno=10;break;
case 'B': pitchno=12;break;
};
*single=* (songstring++);
tempnum=atoi(single);
checkchar=*songstring;
if((checkchar>=0x30)&&. (checkchar<=0x39))
{ *single=checkchar;
tempnum=tempnum*10+atoi(single);
songstring++;}
};
if(singlechar==79)
{octave=tempnum;}
else{playlength=tempnum;
pitchno=(pitchno+(octave*12))-1
}; SetVoiceNote(1,pitchno,playlength,dots);
StartSound();
};
}
void TMyApp::InitMainWindow()
{ MainWindow=new TMyWindow(NULL,Name);
}
int PASCAL WinMain(HINSTANCE hInstance,
HINSTANCE hPrevInstance,
LPSTR lpCmdLine,int nCmdShow)
{ TMyApp MyApp ("Sample Play Music Using Speaker",hInstance,
hPrevInstance,lpCmdLine,nCmdShow);
MyApp.Run();
return MyApp.Status;
}
```

Xenix 系统下文件的压缩和释放

□高中伟

在

DOS 操作系统下,已有众多的功能强大的压缩/释放软件供用户选择进行文件的压缩和释放,诸如 LARJ、LHA、PKZIP/PKUNZIP、PKARC/PKXARC 等等。而在日益普及的多用户 Xenix 操作系统下,如何对文件进行压缩和释放呢? Xenix 系统本身提供有两组对文件进行压缩和释放的命令:pack/unpack 和 compress/uncompress (其中 compress/uncompress 是鲜为人知的),虽然它们的功能不象 DOS 系统下的压缩软件那么强大,但它因按字节使用了霍夫曼编码(Huffman 最小冗余法)而能实现高效的文件压缩和释放。下面笔者简单介绍这些命令的使用方法、对各类型文件的操作效果,并介绍其在文件加、解密中的应用。

一、命令格式

1. pack/unpack/pcat

pack

格式: \$ pack [-] [-f] name...

功能:压缩指定的文件,只要成功,各文件名将被改为 name.z

若选用'-' :将对各字节的调用次数设定内部标志;

若选用'-f' :将强行压缩文件,这可适用于整个目录的文件都被压缩(尽管其中的有些文件并不能被压缩)的情况。

unpack

格式: \$ unpack name...

功能:释放 pack 所压缩的文件

pcat

格式: \$ pcat name...

功能:在标准输出上解除压缩后的输出,但仍保留压缩文件。

2. compress/uncompress

compress

格式: \$ compress [-cfu] [-b bits]

name...

功能:压缩指定的文件,只要成功,各文件都将被改为 name.Z

若选用' -v' :显示每个文件被压缩的百分比

若选用' -f' :强行压缩文件

若选用' -b bits' :设置压缩后文件的编码位数的上限,bits 必须在 9 到 16 之间(确省值为 16),否则达不到压缩效果

uncompress

格式: \$ uncompress name...

"compress -d name..." 也可实现文件的释放,效果同 uncompress)

功能:释放 compress 所压缩的文件

zcat

格式: \$ zcat name...

3. 说明

(1)以上的文件压缩和释放操作,都不改变文件的模式、访问权限、存取时间、修改时间等。

(2)对被压缩文件的要求:(a)文件名的长度(包括后缀)不能超过 12 位;(b)文件不能有连结;(c)文件不能是已压缩的,(d)文件不能是目录文件、特殊设备文件,否则不予压缩。

(3)在文件名的表示中,可引用'?'、'*'等通配符。

(4)compress 只能压缩大于 20B 的文件;pack 只能压缩大于 2K 的文件。

(5)由 compress 压缩的文件只能由 uncompress (或者 compress -d)来释放;由 pack 压缩的文件只能由 unpack 来释放。

(6)compress 压缩后的文件名为 name.Z,而 pack 压缩后的文件名为 name.z,即后缀有区别。

二、不同类型文件的压缩、释放效果比较

硬件环境为 HP 386 机,软件环境为汉化 XENIX SYSTEM V 2.3.2 版。

文件 名	文件类型 说明	文件大小	compress/uncompress				pack/unpack			
			压缩后大小	压缩率%	压缩时间	释放时间	压缩后大小	压缩率%	压缩时间	释放时间
xenix	系统核心	437509	248087	43.29	26	15	320319	26.8	23	26
vi	XENIX 编辑器	144652	92103	36.32	6	4	115975	19.8	4	7
libio.a	XENIX 库文件	258535	169867	34.29	17	11	210401	18.6	8	18
libsys.a	XENIX 库文件	127296	80239	36.96	4	4	102538	19.4	6	6
FXA.DAT	COBOL 数据文件	164850	20063	87.82	3	2	35016	78.7	5	3
FXB.DAT	COBOL 数据文件	513216	63699	88.14	18	9	131000	74.5	25	13
cujizan.prg	FoxBASE+ 源程序	107896	36577	66.09	3	3	71978	33.3	3	5
hyq.dbf	FoxBASE+ 数据库	81440	24284	70.18	3	1	35615	56.3	2	3
icb.dat	Informix 数据库	71680	20385	71.56	10	5	31825	55.6	5	10

其中:压缩率指每个文件被压缩的百分比,即压缩率=(文件减少量/原文件大小)×100%;压缩和释放时间均指相对值。

由上面表中可以看出:用 compress 对文件进行压缩比用 pack 对文件进行压缩的效率;文件压缩和释放命令对不同类型文件的操作效果不一样,但是对数据文件的压缩效果却特别好,这种特性很适合用来进行业务系统大量数据的备份;用 compress/uncompress 或 pack/unpack 对文件的压缩和释放时间并不存在哪一方占有绝对优势。总之,一般宜选用 compress/uncompress 进行文件的压缩和释放。

三、文件压缩/释放命令的应用举例

1. 压缩银行会计系统中的总帐 COBOL 数据文件 FXA.DAT 数据文件存放目录为 /usr/icbacct/data

```
$ compress -v /usr/icbacct/data/FXA.DAT
FXA.DAT:Compression:87.82% -- replace with FXA.DAT.Z
$ pack /usr/icbacct/data/FXA.DAT
```

pack:FXA.DAT:78.7% Compression

2. 释放被压缩的 COBOL 数据文件 FXA.DAT

```
$ uncompress FXA.DAT
或者 $ uncompress FXA.DAT.Z
或者 $ compress -d FXA.DAT
或者 $ compress -d FXA.DAT.z
$ unpack FXA.DAT
或者 $ unpack FXA.DAT.Z
```

3. 进行文件的加、解密

我们在工作过程中,可以将需要加密的文件(不管是可执行文件还是文本文件)通过 compress 命令进行压缩,然后‘将压缩后增加的文件名后缀(.Z)去掉,即可起到加密文件的目的,其他人很难猜得出(即使是超级用户),用 uncompress 命令也无法解密,因为需解密的文件名后缀必须为.Z,当需解密时,再将文件名恢复为压缩的文件名形式,然后再运行释放命令。下面简单介绍加、解密银行会计系统中的分户帐文件 FXB.DAT 的过程:

加密过程:

```
$ compress FXB.DAT \; $ mv FXB.DAT.Z
FXB.DAT
```


解密过程:

```
$ mv FXB.DAT FXB.DAT.Z\; $ uncompress
FXB.DAT.Z
```

4. compress/uncompress 在银行业务系统数据备份软件开发中的运用

在银行业务系统的使用过程中,数据的备份是极其重要的,特别是会计、储蓄两个系统的数据备份,这些系统的备份数据量大,而且需要天天备份,保存时间又较长,软盘和磁带的需要量相当惊人。例如一般业务量行处的会计对公系统的数据备份每天需要三片 1.2M 的 5 寸软盘,以备份数据保存半年计算(这是银行制度要求的),约需要 540 片软盘,显然,它不仅给银行带来资金上的困难,而且因为量多势必引起管理上的混乱。巧妙地利用文件压缩和释放命令 compress/uncompress,可以很好的解决该问题。文件压缩和释放命令 compress/uncompress 对不同类型文件的操作效果不一样,但是对数据文件的压缩效果却特别的好,压缩率在 70% 以上,我们正是利用了它的这一特点。

下面给出两个 Shell 程序 backup.sh 和 load.sh,说明压缩和释放命令 compress/uncompress 在会计对公业务系统的数据备份和恢复中的具体运用,其中 backup.sh 为数据备份程序,load.sh 为数据恢复程序,/usr/icbacct/data 是会计系统的数据目录,

为了不使会计系统的运行受到影响,笔者借助于/tmp 目录来进行数据的压缩和恢复,大家不难从程序中看出。利用该程序进行的对公数据备份,每天只需一片 1.2M 的 5 寸盘即可,以备份数据保存半年计算,只需 180 片软盘,大大少于原来的 540 片软盘。

程序 backup.sh 内容如下:

```
clear
rm/tmp/* >/dev/null 2>&1
copy/usr/icbacct/data/tmp
compress/tmp/* >/dev/null 2>&1
echo "\n\n\t\t\t 请把高密盘插入 0 号驱动器....."
echo "\n\n\t\t\t 按<RETURN>键开始备份!"
read a
echo "\n\n\t\t\t 请稍候....."
tar c2v /tmp/* >/dev/null 2>&1
echo "\n\n\n\t\t\t 备份完毕! 请按任意键返回.....\r"
read a
```

程序 load.sh 的内容如下:

```
clear
rm/tmp/* >/dev/null 2>&1
echo "\n\n\t\t\t 请把高密备份盘插入 0 号驱动器....."
echo "\n\n\t\t\t 按<RETURN>键开始恢复备份数据!"
read a
echo "\n\n\t\t\t 请稍候....."
tar x2v >/dev/null 2>&1
uncompress /tmp/* >/dev/null 2>&1
copy /tmp/usr/icbacct/data
echo "\n\n\n\t\t\t 数据恢复完毕! 请按任意键返回....."
read a
```

SKILL

(上接第 69 页)

```
new08:  jmp short a      ;新的 08H 中断开始处
old     equ this dword   ;原 08H 中断的中断向量
old08   dw 0
oldseg  dw 0
number  dw 1092          ;减法计数器 NUMBER
a:      push ds
        push bx
        push si
        push ax
        push cs
        pop  ds
        dec  word ptr number
        jnz  exit
        mov  word ptr number,1092
        mov  ax,40h
        mov  ds,ax        ;DS=40H
        mov  bx,1ah       ;存放键盘缓冲区的首指针
        MOV  WORD PTR [BX],001EH;键盘缓冲区的开始处
        ;(*)可根据应用软件的存盘热键作修改
        MOV  WORD PTR [BX+4],250BH ;送 Ctrl+K
        MOV  WORD PTR [BX+6],1F13H ;送 Ctrl+S
        MOV  WORD PTR [BX+2],22H   ;修改键盘缓冲区的尾
```

指针

```
exit:   pop ax
        pop si
        pop bx
```

```
pop ds
jmp cs:old ;转原 INT 08H
start:  push cs
        pop ds
        mov  ax,3508h ;保存原 INT08H
        int 21h
        mov  old08,bx
        mov  oldseg,es
        mov  ax,2508h
        lea  dx,new08 ;设置新的 INT 08H
        int 21h
        lea  dx,start
        int 27h
```

```
code ends
end begin
```

SKILL

最少的投资
最佳的方案
最多的组合

挂接式
打印机共享器

北京向宇计算机公司

地址:北京复兴路甲 20 号 27 分号 312,314

邮编:100036 电话:8263441,2063366 呼 1511

利用键盘缓冲区启动不同的操作系统

□李彦超

在

工作中有时需要微机启动时进入不同的操作系统,如DOS系统及各种汉字系统。有些刊物介绍过一些实现方法,如在高级语言中通过设置不同文件实现(这些文件用后还得删除),还有用汇编语言设置错误代码来实现等。笔者在工作中探索出一种利用键盘缓冲区设置不同操作系统的方法,该方法使用灵活方便,可以根据需要进入所需系统并直接运行程序。

众所周知,从键盘键入的字符存在一缓冲区中,该缓冲区位于区存0040H段,从1EH到3CH共32个字节,键入的ASCII码占用内存低端的一个字节,间隔地存入缓冲区单元中,分别放在单元1EH,20H,22H,24H,26H,28H,2AH,2CH,2EH,30H,32H,34H,36H,38H,3AH,3CH中,其结构是一个队列,可存放16个字符,队列的首指针位于该段1AH单元中,尾指针位于该段1CH单元中。键入的字符指针又回到1EH(即循环队列)。如果尾指针和首指针相等,则表示缓冲区已满,而如果首指针等于尾指针,则表明缓冲区为空。

掌握了缓冲区的结构和有关存取方式,我们可以在程序中根据需要改变缓冲区的内容,在缓冲区中插入所需命令,就相当于在键盘上打入了这些命令,如果这些命令是启动某种操作系统的命令,就可以用程序控制进入不同的操作系统。

其基本工作步骤如下:

1. 清除缓冲区: `pokeb(0x40, 0x1a, peekb(0x40, 0x1c))`;置缓冲区队列首指针等于尾指针。

2. 禁止键盘中断,键盘中断的控制由转接口33决定,当转接口的第7位和第1位都为1时,禁止所有的键盘中断,当第7位为1而第1位为0时,允许键盘中断。所以禁止所有键盘中断可以执行语句 `outportb(33, 0x82)`。允许键盘中断可执行语句 `outportb(33, 0x80)`。

3. 设置缓冲区首指针指向缓冲区首址 `0x1e`: `pokeb(0x40, 0x1a, 0x1e)`。

4. 设置一变量 `v`(类型为 `unsigned`),用来存放 `0x40` 段中插入字符的偏移量,初始时内容为 `0x1e`,把要执行的命令放入由偏移量 `v` 指向的单元中,一个字符占用一个单元,然后 `v` 加2,即 `pokeb(0x40, v, 命令字符)`; `v+2=2`;如果 `v>0x3c`,说明缓冲区已满, `v` 不能再增大,即 `if(v>0x3c) v=0x3c`。因此,实际上缓冲区内最多可键入15个字符(输出从首指针到尾指针上一位置之间的字符)。如果命令较长,缓冲区容纳不下时,可编成批命令,然后把这个批命令放入缓冲区即可。注意最后要加上回车符(`0x0d`)。

5. 把 `v` 存入缓冲区尾指针中,即 `pokeb(0x40, 0x1c, v)`。

6. 允许键盘中断。

把本文程序编译连接生成 `.EXE` 文件后,放入 `AUTOEXEC.BAT` 文件中即可。该程序在系统启动后,显示一清单,根据选择可进入DOS系统、UCDOS汉字系统和金山汉字系统。稍加扩充还可模拟运行程序。

```
//smdos.c
#include<stdio.h>
#include<dos.h>
```

(下转第52页)

驱动器字母的设置和关闭

□牟达森

我

们知道,驱动器字母在系统启动后即被设置好,其最后一个字母是由 config.sys 中的 LASTDRIVE 语句设定的,除非改变 LASTDRIVE 值后重新启动,通常是无法把它删除的。如果能在命令行下或程序中设置或关闭一个逻辑驱动器,将是十分有用的。笔者通过对有关 DOS 内部数据结构的分析,编写了一个小程序,实现了上述思想。

一、当前目录结构 CDS

系统中每个驱动器都对应一个 CDS, CDS 是从 DOS 3.0 起引入的,其作用是存放驱动器的一些信息。在此以前的 DOS 中,这些信息都存在于 DPB(驱动器参数块)中。所有驱动器的 CDS 组成 CDS 数组存放在系统中。如果设 LASTDRIVE=F,则 CDS 数组就有 5 个元素。CDS 在 4.0 以前长度为 51h 字节,在 4.0 以后(包括 4.0)为 58h 字节。其

偏移量	大小	作用
00h	67 字节	当前目录路径名
43h	字	标志 位 15:网络驱动器和 IFS 位 14:逻辑驱动器和有效驱动器 位 13:JOIN 驱动器 位 12:SUBST 驱动器
45h	双字	指向该驱动器的 DPB
本地驱动器		
49h	字	当前目录起始簇号
48h	双字	?
网络驱动器		
49h	双字	指向重定向器 IFS 记录
40h	字	INT21h 功能 5F03h 的参数
本地和网络驱动器		
4Fh	字	逻辑驱动器根目录在路径中的偏移量
51h	7 字节	用于 DOS4.0 以后

结构及含义如上表。

二、CDS 入口地址的获得

与 CDS 密切相关的还有 DOS 的表之表(list of lists),它存放着系统中众多参数块的入口地址及其他一些内容。CDS 的入口地址就在表之表的偏移 16h 处,而在偏移 21h 处存放着 LASTDRIVE 的值。顺便提一下,改变此表中的 LASTDRIVE 值是不能增加逻辑驱动器的字母数的,因为 CDS 数组的大小在系统启动后即已确定。使用 INT 21h 功能 52h(属于保留功能)可以得到表之表的入口地址,从而便可得到 CDS 在系统中的位置。

三、程序说明

程序通过改变 CDS 中的属性字,使得某一驱动器变为合法或非法。当把一驱动器改成非法后,任何对它的操作都是无效的,就象系统中没有安装此驱动器一样。而恢复之后则一切变为正常。

具体使用方法如下:

```
DRVSET < drive: >/< NET | PHYS |
SUBST | JOIN | OFF >
```

如: C>DRVSET F:/OFF\ 关闭 F 驱动器

C>DRVSET F:/PHYS 恢复 F 驱动器
笔者用它关闭存放数据的逻辑驱动器,在一定程度上起到了保护数据的作用,效果甚佳。也许读者能发现它更大的作用,本文同时也说明了存取 DOS 内部数据的方法,这对编写高水平的程序无疑是很有用的。本文所列程序在 386DX、DOS3.31、5.0、6.0 下调试通过。

源程序清单如下:

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

#define NETWORK (1<<15)
#define PHYSICAL (1<<14)
#define JOIN (1<<13)
#define SUBST (1<<12)
typedef enum { UNKNOWN=-1,FALSE=0,TRUE=1 } OK;
typedef unsigned char BYTE;
typedef unsigned int WORD;
typedef unsigned long DWORD;
typedef BYTE * DPB;
/* 本例没用到 DPB 的具体内容,故没有具体定义 */

#pragma pack(1)
typedef struct {
    BYTE current __path[67];
    WORD flags;
    /* NETWORK,PHYSICAL,JOIN,SUBST */
    DPB far * dpb; /* 指向 DPB 的指针 */
    union {
        struct {
            WORD start __cluster; /* 当前目录的起始簇号 */
            DWORD unknown;
        } LOCAL; /* 本地驱动器 */
        struct {
            DWORD redirifs __record __ptr; /* 指向重向器 IFS 记录 */
            WORD parameter; /* INT 21h 功能 5f03 的参数 */
        } NET; /* 网络驱动器 */
    } u;
    WORD backslash __offset;
    /* 逻辑驱动器根目录在路径中的偏移量 */
    /* DOS 4+ 此处还有 7 个字节 */
} CDS;

void error(char * s)
{ puts(s);
  exit(1); }
/* 取得当前目录指针
运用了 DOS 中尚未公开的功能:INT 21h 的 52h—返回表之表地址。
*/
CDS far * curr __dir (unsigned drive)
{
    static BYTE far * first __CDS=(BYTE far *)0;
    static OK ok=UNKNOWN;
    static unsigned currdir __size;
    static BYTE lastdrv;
    if(ok==UNKNOWN) /* 只初始化一次 */
    { unsigned currCDS __ofs,lastdrv __ofs;
      if((ok=(__osmajor<3)? FALSE:TRUE)==FALSE)
        return(CDS far *)0; /* DOS 3.0 以下无 CDS */
      /* 根据 DOS 版本计算当前目录结构和 LASTDRIVE 在 DOS
      表之表中的偏移 */
      currCDS __ofs=(__osmajor==3 && __osminor==0)?
      0x17:0x16;
      lastdrv __ofs=(__osmajor==3 && __osminor==0)? 0x1b:
      0x21;
```

```
asm push si
asm mov ah,52h
asm int 21h /* ES:BX 指向 DOS 的表之表 */
asm mov si,lastdrv __ofs
asm mov ah,byte ptr es:[bx+si]
lastdrv=__AH; /* lastdrv 中为 LASTDRIVE 的值 */
asm mov si,currCDS __ofs
asm les bx,es:[bx+si]
asm mov word ptr first __CDS+2,es
asm mov word ptr first __CDS,bx
/* first __CDS 指向第一个 CDS */
asm pop si
if (first __CDS==(BYTE far *)-1L) ok=FALSE;
currdir __size=(__osmajor>=4)? 0x58:0x51;
/* 计算当前目录结构的大小 */
}
if (!ok) return (CDS far *)0;
if (drive>=lastdrv) return(CDS far *)0;
return (CDS far *) (first __CDS+(drive * currdir __size));
}

main(int argc,char * argv[])
{
    CDS far * drv;
    unsigned drive;
    if (argc != 2)
        error("Usage:DRVSET <drive:>/<NET|PHYS|SUBST|JOIN
|OFF>");
    else {
        if (argv[1][0]>='A') drive=toupper (argv[1][0])-'A';
        else error(" Usage: DRVSET <drive:>/<NET|PHYS|SUBST|
JOIN|OFF>");
    }
    if (! (drv=curr __dir(drive)))
        error("can't get current directory structure");
   strupr(argv[1]);
    if (argv[1][1]!=':' && argv[1][2]!='/')
        error(" Usage: DRVSET <drive:>/<NET|PHYS|SUBST|
JOIN|OFF>");
    if (strcmp(argv[1],"OFF")) drv->flags=0;
    else { if(strcmp(argv[1],"NET"))
        drv->flags|=NETWORK;
        else { if(strcmp(argv[1],"PHYS"))
            drv->flags|=PHYSICAL;
            else { if(strcmp(argv[1],"JOIN"))
                drv->flags|=JOIN;
                else { if(strcmp(argv[1],"SUBST"))
                    drv->flags|=SUBST;
                    else error("Usage:DRVSET<drive:>/<NET|PHYS|SUBST|
JOIN|OFF>"); }
                }
            }
        }
    printf("%c:",argv[1][0]);
    if(! drv->flags) fputs("INVALID",stdout);
    if (drv->flags & NETWORK) fputs("NET",stdout);
    if (drv->flags & PHYSICAL) fputs("PHYSICAL",stdout);
    if (drv->flags & SUBST) fputs("SUBST",stdout);
    if (drv->flags & JOIN) fputs("JOIN",stdout);
    fputs("driver.",stdout);
    putchar('\n');
    return 0;
}
```

常见微机图象格式浅析

□胡 进

当

今计算机图形图象技术的应用越来越广泛,为了能在不同的场合和条件下更有效地利用和处理图象,许多公司制定了多种规范或标准来对原始图象数据进行压缩和存储,要建立一个处理图象的系统,就必须要了解这些规范,即相关的图象文件格式。为了便于用户使用这些图象格式,本文汇总了几种国内最常见的图象文件格式规范供广大读者参考。

一、BMP 图象文件格式

表 1 BMP 图象文件格式

地 址	长度 (字节)	说 明
00 H	2	位图文件的标记“BM”
02 H	4	以字节为单位的位图文件的大小
06 H	2	保留字(一般置 0)
08 H	2	保留字(一般置 0)
0A H	4	图象数据的起始地址
0E H	4	位图信息头长度。共 40 字节
12 H	4	位图图象宽度的像素值
16 H	4	位图图象高度的像素值
1A H	2	图象目标设备的位平面数值为 1
1C H	2	每个像素的位数(1,4,8,24)
1E H	4	位图图象压缩类型。0—未压缩 1—REL 8 2—REL 4
22 H	4	以字节为单位的压缩图象大小
26 H	4	目标设备水平分辨率
2A H	4	目标设备垂直分辨率
2E H	4	实际使用颜色数
32 H	4	重要颜色数
36 H	1	蓝色相对亮度值
37 H	1	绿色相对亮度值
38 H	1	红色相对亮度值
39 H	1	位图保留字,其值为零

以下为实际位图数据

BMP 格式图象文件是目前最流行的几种图象文件之一,它也是 Microsoft Windows 的操作系统支持的主要文件格式,在这里我们主要介绍一下 BMP 的非压缩存储格式。BMP 文件格式相对来说比较简单,一般由以下三部分组成:

位图文件头(共 14 个字节);

位图信息头结构(共 40 个字节);

位图彩色映象表和图象数据;

BMP 的图象文件格式具体定义如表 1。

BMP 文件头偏移 22H 处(压缩图象大小值)不能简单地通过把图象长、宽和每像素所占字节数相乘求得。由于 BMP 图象数据是按行存放的,每行按双字对齐。所以位图长度是由实际行宽与行数相乘的结果,每个像素所占字节数相乘所得。

文件头 2EH 偏移处是实际使用颜色数,它说明位图实际使用的颜色表中的色彩索引值。当图象每个像素对应 1 bit、4 bit 或 8 bit 时就必须有一个彩色映象。彩色图象的大小一般分为 2、16 或 256 个表项。如果此值非 0,则它包含使用的颜色数目,同时它也是彩色映象的表项号码。如果此值为 0,则彩色映象为全大小。从上表我们可以看出位图数据是紧跟在彩色映象表后面的,一般而言,对于每个像素是 1 bit 的位图,由于每个像素只占一位,因此 1 个字节表示可有 8 个像素点。第一个字节最高位对应最左边的像素,依此类推……。

对于每个像素对应 4 bit 的位图,若图象没有被压缩则每 1 个字节可表示 2 个像素,高 4 位对应最左边的像素,低 4 位对应右边的象

素且每行填充到一个 4 字节边界。

对于每个象素对应 8 bit 的位图；每个字节表示一个象素，每行填充到一个 4 字节边界。

对于每个象素 24 bit 的位图，每个象素为 3 个字节，顺序为红色、绿色和蓝色的值。每行填充到 4 字节的边界。

这里为了便于实际应用，笔者给出了用 C 语言定义的 BMP 图象文件格式，具体定义如下：

```
typedef struct TagBmpFileHeader
```

```
{
    int    BmpType;    LongBmpSize ;
    int    BmpReserve1 ;int BmpReserve2 ;
    long   BmpOffsetBits ;
}BmpFileHeader
```

```
typedef struct    TagBmpInfoHeader
```

```
{
    Long    BInfoSize ;
    long    BWidth ;
    Long    BHeigh ;
    int     BPlane ;
    int     BBitNum ;
    Long    BCompress ;
    Long    BImageSize
    Long    BxPerMeter
    Long    ByPerMeter
    Long    BImportantClr
}BmpInfoHeader
```

```
typedef struct TagRGBQUAD
```

```
{
    int    RGBBlue ;
    int    RGBGreen ;
    int    RGBRed ;
    int    RGBReserved ;
}RGBQUAD
```

二、GIF 图象文件格式

GIF 是“Graphics Interchange Format (图象互换格式)”的缩写，它是 CompuServe 公司制定的图象存储规范。GIF 采用散列法(hash—method)进行图象数据的压缩编码。它用一个最多 256 种颜色的调色板实现对 24bits 彩色图象的支持。图象最大可为 64KX64K 个象素点。

GIF 图象文件分为文件头和文件体两大部分，文件头包括文件标志、彩色表、图象信息描述等，它的结构可随文件包含的实际图象数据而变化，相当灵活。下表 2 给出了 GIF 图象文件的文件头格式。位于 0AH 处有一个全局标志字节，位 7 表明文件有无彩色表，1 则有，全局色表将紧随彩色表的存在标识符，0 则无，背景色索引将无意义。后三位表示每个象素所要求的位数，每个象素所要求的位数是将后三位数加 1 而得的值。之后包含 1 bit 的排序标志。为

0 则全局色表不进行排序；为 1 则全局色表按重要性排序，最重要的颜色在先，次之在后，依此类推。另外还有 3bits 表示全局色表大小用于计算全局色表所含字节数。

表 2 GIF 图象格式文件头

地址	长度(字节)	说明
00 H	6	GIF 文件标记一般为 GIF87a 或 GIF 89 a
06 H	2	图象的水平分辨率(逻辑屏幕宽度)
08 H	2	图象的垂直分辨率(逻辑屏幕高度)
0A H	1	全局标志字节，文件有无彩色表，每象素位数
0B H	1	图象背景颜色
0C H	1	GIF 文件保留字节，值为 0
0D H	不定	全局彩色表起始，共长度与每象素位数有关

全局颜色表由代表红色、绿色、蓝彩色组合的字节序列组成。全局颜色表用于不包含局部颜色表的图象和简单文本的扩展。其长度等于 $3 \times 2 \times (\text{颜色表大小} + 1)$ 字节。全局颜色表之后是局部图象数据。

表 3 GIF 局部表头数据

偏址	长度(字节)	内容
00 H	1	图象分隔符，值为 0x2C
01 H	2	图象左边界的象素列数
03 H	2	图象顶部边界的象素列数
05 H	2	图象以象素计的宽度
07 H	2	图象以象素计的高度
09 H	1	标识字节

这里的标识字节各位段含义如下：

bit 7 局部颜色表标志一位长。值为 1 表示有局部颜色表；为 0 表示无局部颜色表。bit 6 交叉标志一位长，值为 1 表示图象出现交叉，为 0 表示图象无交叉。bit 5 排序标志两位长，指示局部颜色表是否排序 bit 4 3 保留位二位长，bit 2 0 局部颜色表大小三位长。

若有局部颜色表，则其将紧跟在局部图象表头之后。该表也是由红色、绿色、蓝色三种颜色组合的

最少的投资

最佳的方案

最多的组合

挂接式

打印机共享器

北京向宇计算机公司

地址：北京复兴路甲 20 号 27 分号 312,314

邮编：100036 电话：8263441, 2063366 呼 1511

字节序列组成的。此部分之后,为图象编码数据。GIF 文件压缩后的数据是分块存储的,每块的第一字节表示该块中数据的长度。解压缩操作实际上是按块读出每块中的数据。GIF 文件的压缩是基于变长码的,其长度最大可达 2^{12} ,即最大允许 4096 个表项。下面以压缩一个 256 色图象为例来看一下 GIF 文件的编码实现过程。首先初始化表的前 256 项,对应表项号码为 0 到 255,然后程序开始分析读入图象的数据。程序将当前字符串拷贝到当前首标并重复读入字符,浏览预制字符串表,看是否能找到此字符串。如未匹配,则表项值加 1 送到输出的程序串中,并使当前字符串长度增加 1 字节。如匹配,则不必增加表项。当形成 GIF 文件时,要常输出一个特殊的程序码“clear”,告诉译码器重新初始化它的字符串表。只要编码软件判断出这个表已满,便会输出这个程序码以通知解码器放弃这个表,重新开始。这个程序码定义为图象颜色的最大值。依此类推,重复以上过程直到压缩完成为止。

GIF 文件头结构用 C 语言表示如下所示:

```
typedef struct GIFFileHeader
{
    char Version[6];
    unsigned ScrWidth;
    unsigned ScrHeight;
    unsigned GlobalClrSize : 3;
    unsigned SortFlag : 1;
    unsigned ClrResolution : 3;
    unsigned GlobalClrFlag : 1;
    int BkClr;
    int Reserved;
} GIFFileHeader;

typedef struct GIFLocalHeader
{
    int ImageSeparator;
    unsigned ImgLeft;
    unsigned ImgTop;
    unsigned ImgWidth;
    unsigned ImgHeight;
    unsigned LocalClrSize : 3;
    unsigned LocalReserved : 2;
    unsigned LocalSortFlag : 1;
    unsigned LocalCrossFlag : 1;
    unsigned LocalClrFlag : 1;
} GIFLocalHeader;
```

三、PCX 图象文件格式

Zsoft 公司开发的 PCX 是一种广泛流行的标准图象文件格式,它能表示多达 256 色的彩色图象,目前绝大多数扫描仪、图象编辑软件都支持这种格式。

PCX 文件由文件头、位图数据和彩色表三部分组成。PCX 文件头长度固定为 128 字节,但对于 16

色 PCX 和 256 色 PCX 文件,彩色表的位置有所不同,位图数据是采用扫描线行程压缩法存放的。

表 4 PCX 文件头结构

地址	长度 (字节)	内 容
00 H	3	PCX 文件标记段(0A0501),01H 处为版本号
03 H	1	每个位平面的每个像素的位数
04 H	4	PCX 图象的左上角坐标(Xmin,Ymin)
08 H	4	PCX 图象的右下角坐标(Xmax,Ymax)
0C H	2	图形适配器水平分辨率
0E H	2	图形适配器垂直分辨率
10 H	1	红色相对亮度值
11 H	1	绿色相对亮度值
12 H	1	蓝色相对亮度值
...
40 H	1	Zsoft 保留值,值为 0
41 H	1	彩色/灰度位平面数
42 H	2	每一水平行的一个彩色位平面所需要的内存字节数。
44 H	2	调色板方式,1 为彩色或黑白 2 为灰度方式
46 H	2	视屏输出的水平像素数-1
48 H	2	视屏输出的垂直像素数-1
4A H	54	以空格填满文件头

在过去 PCX 图象格式只能支持单色或 16 色彩色图象但是 PC Paintbrush IV 以后版本可以支持 256 色全彩调色板,在上表中的表头偏移 01 H 处是版本标志,它的值对应的版本信息如下:

- 0=PC Paintbrush 2.5 版
- 2=PC Paintbrush 2.8 版,有调色板
- 3=PC Paintbrush 2.8 版,无调色板
- 4=PC Paintbrush for Windows 版
- 5=PC Paintbrush 3.0 以上版

位于偏移 46H、48H 处的数据,只有在 Paintbrush IV 或 Paintbrush IV Plus 版本中才有意义。PCX 文件中包含有调色板信息,其中的彩色表每一项由 3 bits 组成(分别对应红色、绿色、蓝色亮度值),对 16 色 PCX 文件彩色表为 48 bits,位于文件头中部。对 256 色 PCX,其彩色表位于文件后部图象数据后面的 786

最少的投资

最佳的方案

最多的组合

挂接式

打印机共享器

北京向宇计算机公司

地址:北京复兴路甲 20 号 27 分号 312,314

邮编:100036 电话:8263441,2063366 呼 1511

bits 里。同时在文件倒数第 769 字节置扩展调色板信息 0X0C。

PCX 是一种历史悠久的图象存储格式,版本很多,因此为了对 PCX 文件正确解码就需要根据版本号,确定每个象素对应的位数及彩色位平面数。

表 5 PCX 图象数据信息

每象素的位数	位平面数	说明
1	1	单色图象
1	2	4 色图象
1	3	8 色图象
1	4	16 色图象
2	1	4 色,CGA 头调色板
2	4	16 色图象
4	1	16 色,EGA 头调色板
8	1	256 色图象
8	3	16.7 M 种彩色

PCX 图象文件数据在文件头偏移 80H 处开始存放,数据是以 RLL 编码方式进行压缩的,其基本思想是用一个重复计数值来记录相邻重复的字节数,且压缩仅对一条扫描线进行。面对彩色图象每根扫描线按其位平面数分成若干段进行,其算法如下:

设扫描线上某段由连续出现 N 次的字节 B 组成:

(1)if(N=1) {if(B<0xc1 h) 输出 B;
else 输出 (0xclh,B); }

(2)if(N>63) {输出 (0xFFh,B);
N=N-63 ;
goto (1) ; }

(3)if(N>1 && N<63) {输出 (0xc0h+N,B); }

上述算法就是位映象格式到压缩格式的转换算法。由此压缩过程对连续出现一次的字节 B,如果 B>0xc0h,则压缩时在该字节前加上压缩标志 0xcl h。如果 B<=0xcc h,则不加。对连续输出 N(N>1)次的字节 B 则压缩为 0xc0+N,B 这两个字节。

PCX 文件结构的 C 语言描述如下:

```
typedef struct PCXFileHeader
{
    char Sign; char Facture; char Version;
    char Encoding; char BitsPerPixel;
    int Xmin, Ymin; int Xmax, Ymax;
    int HRes; int VRes;
    char Palette [48]; char Reserved;
    char NumOfPlanes;
    int BitsPerLine; int TypeOfPalette;
    int XPixel; int YPixel;
    char Filler[54];
} PCXFileHeader;
```

四、TIFF 图象格式文件

TIFF (Tag Image File Format) 格式是由 Aldus

和 Microsoft 公司联合开发推出的, TIFF 在组织上比较复杂,编程非常灵活,它可以支持任意大小的图象,支持从单色到 24 bits 真彩色及灰度图象,且具有与设备无关性。TIFF 描述非常全面,但它需要大量的编程工作进行解码。

TIFF 文件一般由文件头、参数指针表、参数数据表和图象数据四部分组成。

表 6 TIFF 格式文件头

地址	长度(字节)	说明
00 H	2	字节顺序:MM(Motorola 格式)或 II(Intel 格式)
02 H	2	版本标记号,其值为 42
04 H	4	指向第一个参数块的指针

文件头后面根据指针指向第一个参数指针表及参数数据表。

表 7 TIFF 参数指针表(偏址由本指针表开始)

偏址	长度(字节)	说明
00D	2	参数块指针数目计算,设值为 n
02 D	12	第一参数数据表
14 D	12	第二个参数数据表
...
n * 12 - 10D	12	第 n 个参数数据表
n * 12 + 2D	4	如果有,指向下一个参数指针表,否则为 0

表 8 TIFF 参数数据表结构(偏移由本块开始算起)

偏址	长度(字节)	说明
00 H	2	参数标记码
02 H	2	参数数据类型
04 H	4	参数长度字段
08 H	4	参数数据指针或指向参数数据的指针

在 TIFF 参数数据表结构中,前两个字节为标记码。共用标记码值为 254 到 221,可以在 TIFF 规定中查到。当此值大于等于 32768 时为赋给各个公司的软件私有码。

参数类型码指出 TIFF 支持的数据类型,其代码为:

1=1 字节整型数 BYTE 字节类型

最少的投资

最佳的方案

最多的组合

挂接式

打印机共享器

北京向宇计算机公司

地址:北京复兴路甲 20 号 27 分号 312,314

邮编:100036 电话:8263441,2063366 呼 1511

2=1 字节 ASCII 码 CHAR 单字节字符
 3=2 字节整型数 SHORT 短整数类型
 4=4 字节整型数 LONG 长整数类型
 5=8 字节分数 RATIONAL 有理数类型
 4 字节分子,
 4 字节分母
 参数长度字段指定数据字段中值的数目,而不

是字节数目。实际字节数目可以通过将参数长度乘以数据类型中的字节数求得。最后四个字节通常是指向数据字段开始处的一个指针,也就是文件开始到数据开始之间的字节偏移数。

在 TIFF 文件中经常会遇到一些参数码,为了帮助读者理解这些码的意义,这里给出了一些常用 TIFF 参数码及它们的意义说明。

表 9 常用 TIFF 参数码及其意义

参数码	类 型	名 称	内 容
256(100H)	SHORT	图象宽度	图象以像素为单位的宽度
257(101H)	SHORT	图象长度	图象以扫描行为单位的垂直高度
258(102H)	SHORT	每样点位数	像素深度
259(103H)	SHORT	压缩码	图象数据的压缩代码码: 1. 未压缩 2. CCITT Group 3 改进一准霍夫曼行程长度编码 3. CCITT Group 3 二维编码 4. CCITT Group 4 二维编码 5. LZW 压缩 32773;PackBits 压缩
262(106H)	SHORT	彩色顺序	0 最小值为白,最大值为黑,其余为灰度 1. 最小值为黑,最大为白,其余为灰度 2. RGB 真彩色,最大、小值为饱和度,无调色板 3. 伪彩色调色板,最大、小值由调色板定义
273(111 H)	LONG	象块偏移	图象块的偏移地址
277(115 H)	SHORT	每像素样点数	值为 1:单色、灰度图象 值为 3:24 位真彩色图象
282(11A H)	RATIONAL	水平分辨率	像素/单位的 X 方向分辨率
283(11B H)	RATIONAL	垂直分辨率	像素/单位的 Y 方向分辨率
284(11 C H)	RATIONAL	平面配置	值为 1:一个位平面。 为 2:彩色位平面
291(123 H)	SHORT	灰度响应曲线	数据指向一个灰色校正表。缺省为 2。
296(128 H)	SHORT	分辨率单位	值为 1:无,为 2:英寸,为 3:厘米,缺省值为 2。
301(12D H)	SHORT	彩色响应曲线	指向三个连续存放的表的指针,每个表对应于 R、G、B 的彩色校正表
319(13F H)	LONG	原色	CIE 值的指针
320(140 H)	SHORT	调色板	指向调色板的指针

TIFF 文件头的结构可以用 C 语言描述如下:

```
typedef struct TIFFHeader {
    unsigned    Bytes ;
    unsigned    Version ;
    unsigned    Long    NextPoint ;
} TIFFHeader ;

typedef struct IFDBlock {
    unsigned    tag ;
    unsigned    type ;
    unsigned    long Length ;
    unsigned    long value ;
} IFDBlock ;
```

以上介绍了几种常见的图象格式供广大软件开发人员参考,希望能够对实际工作有所帮助。上面介

绍的图象形式有的相对比较复杂,但是只要仔细研究剖析并实践就会发现,处理各种形式的图象并不是一件特别难的事,相信大家都能掌握。 **SKILL**

最少的投资

最佳的方案

最多的组合

挂接式

打印机共享器

北京向宇计算机公司

地址:北京复兴路甲 20 号 27 分号 312,314

邮编:100036 电话:8263441,2063366 呼 1511

16 色位图文件的 VGA 显示

□ 高铁栓

W

indows Paintbrush 彩色绘图软件,操作简单、方便,深受用户的喜爱,为了在 DOS 下利用该软件的图形,作为软件封面或其他用途,需设计程序来显示该软件的图形,Paintbrush 的存盘文件格式有: PCX, 单色位图、16 色、256 色、24 位真彩色位图等多种。所谓位图文件就是按位(点)格式存储的图象文件。本文仅以 16 色位图文件为例,说明 Windows 位图文件的显示方法。

一、位图文件. BMP 的结构

位图文件由文件头、调色板数据、图象数据三部分组成,文件头在文件首部,共 53 个字节,文件头之后是调色板数据,其长度随颜色数不同而不同,单色位图文件调色板数据共 $4 \times 2 = 8$ 个字节,16 色位图文件共 $4 \times 16 = 64$ 个字节,256 色为 $4 \times 256 = 1024$ 个字节;调色板与图象数据之间隔一个字节 00h, 00h 之后是图象数据,其长度也随颜色数不同而不同。单色位图文件图象数据为 $\text{width} * \text{height} / 8$ 个字节,一个字节代表 8 个像素点,16 色位图文件图象数据为 $\text{width} * \text{height} / 2$ 个字节,一个字节代表两个像素点;256 色位图文件为 $\text{width} * \text{height}$ 个字节,一个字节代表一个像素点。下面针对 16 色位图文件加以说明。

1. 文件结构

16 色位图文件结构如下表:

字节号	意义
00-01	位图文件标志,为 BM
02-05	文件总长度(字节数)

06-07	未定义
08-11	文件首部长(文件头+调色板+1)
12-17	未定义
18-19	图象宽度(像素数)width
20-21	未定义
22-23	图象高度 height
24-27	未定义
28-	每像素点所占位数(为 1,4,8 等) bits __ per __ pix
29-52	未定义

16 色位图文件的 bits __ per __ pix = 4。

2. 调色板数据结构

每组调色板数据由四个字节组成,第五个字节为 0,其他三个字节依次为蓝(B)、绿(G)、红(R)的颜色值,均为高 6 位有效,即每个字节均右移两位成为蓝、绿、红的实际有效颜色值。设置 VGA 调色板的方法有两种:

(1)调用显示中断的 10h 或 12h 功能实现。有些系统没有提供该功能,因此不提倡采用这种方法。

(2)对 VGA 硬件编程。VGA 的数字/模拟信号转换寄存器(DAC)用来控制显示器的颜色,可以通过改变颜色寄存器的红、绿、蓝的百分比实现。DAC 的写地址挑选寄存器,端口地址为 3c8h,用来挑选调色板寄存器号。16 色 640×480 模式的 16 种颜色对应的调色板寄存器号分别为 0、1、2、3、4、5、20、7、56、57、58、59、60、61、62、63,开机后,这 16 个寄存器都写入了默认 RGB 颜色值,往这 16 个调色板寄存器写入 RGB 颜色值,便可改变显示颜色值。改变调色板颜色寄存器的方法是往 DAC 的数据寄存器(端口地址为 3c9h),写入 18 位颜

色值。先写红色,再写绿色,最后写入蓝色。如欲改变色号0的显示颜色,可采用如下办法:

```
MOV DX,3C8H
MOV AL,0 ;色号0对应的调色板寄存器号为0
OUT DX,AL
MOV DX,3C9H
MOV AL,R ;R为存放红色值的变量
OUT DX,AL
MOV AL,G ;G为存放绿色值的变量
OUT DX,AL
MOV AL,B ;B为存放蓝色值的变量
OUT DX,AL
```

或者使用 Turbo C 的 outportb() 函数:

```
outportb(0x3c8,0);
outportb(0x3c9,R);
outportb(0x3c9,G);
outportb(0x3c9,B);
```

采用方法(2),可以把16色位图文件的16组调色板数据装配至调色板寄存器,使得图象显示颜色与位图文件相符。

二、图象数据分析

16色位图文件的图象数据一个字节表示两个像素点,高4位表示一个像素点的颜色,低4位表示一个像素点的颜色,屏幕上一行用 width/2 个字节表示。位图文件图象数据的最后一行是屏幕的第一行,图象数据倒数第二行是屏幕的第二行,依此类推。

三、VGA 的 16 色显示原理

VGA 的 16 色 640×480 显示模式,模式号为 12h,采用如下方法可把 VGA 设置为 640×480 的高分辨率图形模式:

```
MOV AX,0012H
INT 10H
```

采用如下语句可恢复为文本方式:

```
MOV AX,0003H
INT 10H
```

VGA 的 16 色 640×480 模式的显示卡内存是位平面组织方式,有四个位平面,一个像素点的四位颜色数据被依次写入四个位平面,每个位平面一位,屏幕上即显示出相应颜色的点。颜色数据写入显示卡内存的方式有四种:写方式0,写方式1,写方式2,写方式3。通过对 VGA 的图形控制寄存器(GDC,端口地址为 3ceh)的 5 号索引寄存器(GDC5 称为方式寄存器)编程,设置写方式。针对 16 色位图文件图象数据的组织方式,采用写方式2比较好,此种方式下,CPU 传来的数据作为颜色数据(低4位有效),被自动扩展为四个字节,一次写入对应的位平面中。把数据写入方式设置为写方式2的方法如下:

```
MOV DX,3CEH
MOV AL,5
OUT DX,AL ;选择5号索引寄存器(即方式寄存器 GDC
5)
MOV DX,3CFH
MOV AL,2
OUT DX,AL
```

位屏蔽寄存器(GDC 8)用一个字节来控制哪个像素位允许/禁止 CPU 写,每个像素位均有四位,分布在四个位平面中,须往 GDC 8 中写入一个位屏蔽码(或称位掩码 bit __ mask),来决定 8 个像素点中的哪一个像素显示。

要在屏幕上显示出像素点,必须计算出该点在显示内存中的地址偏移量 offset 及字节内位置(用位掩码表示)。计算方法如下:

我们知道(0,0)是屏幕左上角像素的坐标,(x,y)点在显示内存中字节偏移量为:offset=y*80+x/8,知道了偏移量之后,就可以使用地址 a000:offset 来访问该字节。由于一个字节对应 8 个像素点,究竟是哪一个像素显示,还必须计算出位屏蔽码,方法是:shift __ count=x mod 8, bit __ mask=80h shift __ right by shift __ count。

知道了偏移量与位掩码之后,就可以在屏幕上画点。算法为:

(1)设置像素位掩码;

```
mov dx,3ceh
mov al,8
out dx,al
mov dx,3cfh
mov al,bit __ mask
out dx,al
```

(2)从与该像素对应的地址读取。

假定 ES 中为 A000H, BX 中存放字节的偏移,则可用 MOV AL,ES:[BX] 语句来完成。

(3)设置 AL 为像素的颜色,使用 MOV ES:[BX],AL 语句可以把颜色写入位平面即可在屏幕上显示出该颜色的像素点。

经过以上分析,可以很容易地写出 16 色位图的显示算法:

- (1)打开文件;
- (2)判断是否为位图文件;若不是,转(11);
- (3)判断是否为 16 色文件,若不是,转(11);
- (4)读入图象宽 width 和高 height;
- (5)读入调色板数据;
- (6)设置 VGA 为 16 色 640x480 模式,模式号为 12h;
- (7)设置 VGA 的 16 个色号对应的调色板;
- (8)读入一行图象数据,在屏幕上显示;

用“宝合”UPS 电源，再不怕电网突然掉电



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蕾三街二号
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

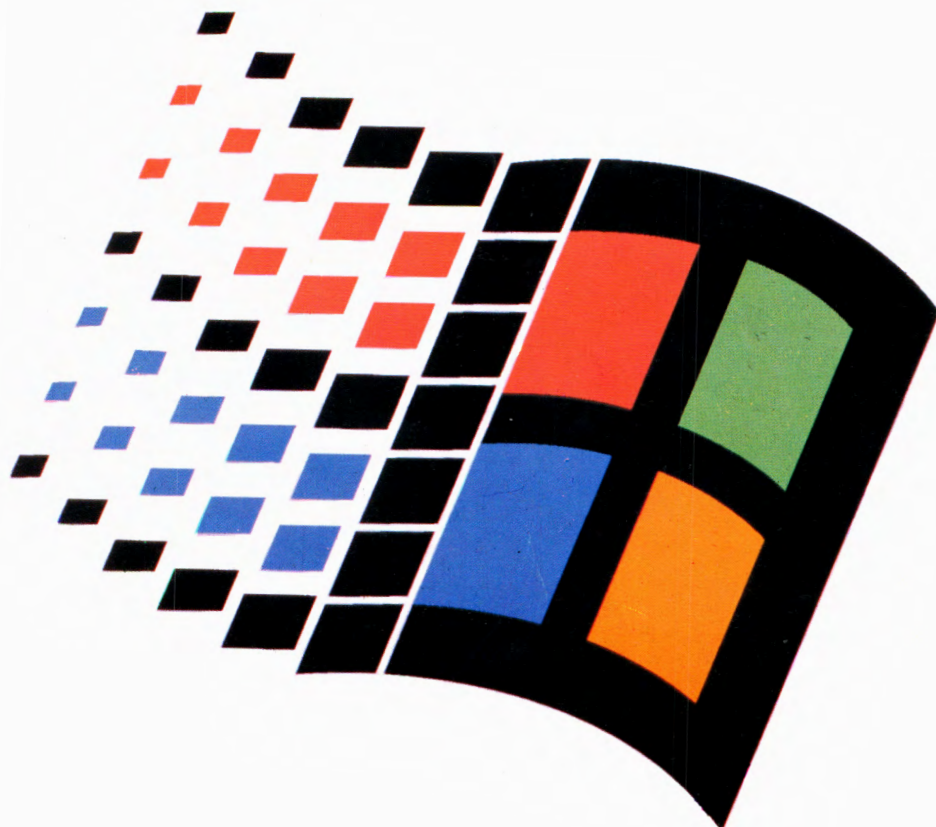
地址：北京海淀路 82 号

信箱：(100080) 北京 8721 信箱

电话：2562329、2541992

传真：2561057





Microsoft® **WINDOWS NT**™



MICROSOFT
WINDOWS NTTM
COMPATIBLE
32 Bit Application

微软公司北京代表处

北京新世纪饭店写字楼五层551室

中国北京首都体育馆南路六号

邮政编码: 100046

电话: (86-1)849-2148~50

传真: (86-1)849-2151



Microsoft VISUAL C++

Development System for WindowsTM and Windows NTTM

(9)若图象没有显示完,返回(8);

(10)恢复 VGA 文件模式;

(11)结束。

该程序在 Borland C++ 2.0 下编译通过,可在有 VGA 卡或兼容 VGA 卡的所有计算机上运行。

源程序清单如下:

```
#include <stdio.h>
#include <conio.h>
#include <mem.h>
#include <alloc.h>
#include <dos.h>
#include <graphics.h>
unsigned int width,height;
unsigned char bits __per __pix,id[3];
void xs16s(char * filename);
unsigned char far * farptr(unsigned char far * p,long l);
void main(int argc,char * argv[])
{
    if(argc<2) { printf("input error");return;}
    xs16s(argv[1]);
    return;
}
#pragma inline
void xs16s(char * filename)
{
    FILE * fp;
    if((fp=fopen(filename,"rb"))==NULL)
    {
        printf("%s open error\n",filename);
        return;
    }
    fseek(fp,0L,0);
    fread(id,1,2,fp);
    if(memcmp(id,"BM",2))
    {
        printf("%s is not bmp file.\n",filename);
        return;
    }
    fseek(fp,28L,0);
    fread(&bits __per __pix,1,1,fp);
    fseek(fp,18L,0);
    fread(&width,1,2,fp);
    fseek(fp,22L,0);
    fread(&height,1,2,fp);
    if(bits __per __pix !=4) return;
    char * p,* pal;
    if((p=(char *)malloc(4*16+1))==NULL)
    return;
    if((pal=(char *)malloc(3*16+1))==NULL)
    return;
    fseek(fp,53L,0);
    if(fread(p,1,65,fp)!=65) return;
    for(int ll=0;ll<16;ll++)
    {
        * (pal+ll*3)=((*(p+ll*4+3))>>2);
        * (pal+ll*3+1)=((*(p+ll*4+2))>>2);
        * (pal+ll*3+2)=((*(p+ll*4+1))>>2);
    }
    asm{
```

```
        mov ax,0012h
        int 10h
    }
    unsigned char red0,green0,blue0;
    red0= * pal++;
    green0= * pal++;
    blue0= * pal++;
    outportb(0x3c8,0);
    outportb(0x3c9,red0);
    outportb(0x3c9,green0);
    outportb(0x3c9,blue0);
    red0= * pal++;
    green0= * pal++;
    blue0= * pal++;
    outportb(0x3c8,1);
    outportb(0x3c9,red0);
    outportb(0x3c9,green0);
    outportb(0x3c9,blue0);
    red0= * pal++;
    green0= * pal++;
    blue0= * pal++;
    outportb(0x3c8,2);
    outportb(0x3c9,red0);
    outportb(0x3c9,green0);
    outportb(0x3c9,blue0);
    red0= * pal++;
    green0= * pal++;
    blue0= * pal++;
    outportb(0x3c8,3);
    outportb(0x3c9,red0);
    outportb(0x3c9,green0);
    outportb(0x3c9,blue0);
    red0= * pal++;
    green0= * pal++;
    blue0= * pal++;
    outportb(0x3c8,4);
    outportb(0x3c9,red0);
    outportb(0x3c9,green0);
    outportb(0x3c9,blue0);
    red0= * pal++;
    green0= * pal++;
    blue0= * pal++;
    outportb(0x3c8,5);
    outportb(0x3c9,red0);
    outportb(0x3c9,green0);
    outportb(0x3c9,blue0);
    red0= * pal++;
    green0= * pal++;
    blue0= * pal++;
    outportb(0x3c8,20);
    outportb(0x3c9,red0);
    outportb(0x3c9,green0);
    outportb(0x3c9,blue0);
    red0= * pal++;
    green0= * pal++;
    blue0= * pal++;
    outportb(0x3c8,7);
    outportb(0x3c9,red0);
    outportb(0x3c9,green0);
    outportb(0x3c9,blue0);
    red0= * pal++;
```

```

green0= * pal++;
blue0= * pal++;
outportb(0x3c8,56);
outportb(0x3c9,red0);
outportb(0x3c9,green0);
outportb(0x3c9,blue0);
red0= * pal++;
green0= * pal++;
blue0= * pal++;
outportb(0x3c8,57);
outportb(0x3c9,red0);
outportb(0x3c9,green0);
outportb(0x3c9,blue0);
;red0= * pal++;
green0= * pal++;
blue0= * pal++;
outportb(0x3c8,58);
outportb(0x3c9,red0);
;outportb(0x3c9,green0);
;outportb(0x3c9,blue0);
red0= * pal++;
green0= * pal++;
blue0= * pal++;
outportb(0x3c8,59);
outportb(0x3c9,red0);
outportb(0x3c9,green0);
outportb(0x3c9,blue0);
red0= * pal++;
green0= * pal++;
blue0= * pal++;
outportb(0x3c8,60);
outportb(0x3c9,red0);
outportb(0x3c9,green0);
outportb(0x3c9,blue0);
red0= * pal++;
green0= * pal++;
blue0= * pal++;
outportb(0x3c8,61);
outportb(0x3c9,red0);
outportb(0x3c9,green0);
outportb(0x3c9,blue0);
red0= * pal++;
green0= * pal++;
blue0= * pal++;
outportb(0x3c8,62);
outportb(0x3c9,red0);
outportb(0x3c9,green0);
outportb(0x3c9,blue0);
red0= * pal++;
green0= * pal++;
blue0= * pal++;
outportb(0x3c8,63);
outportb(0x3c9,red0);
outportb(0x3c9,green0);
outportb(0x3c9,blue0);
free(p);
free(pal);
unsigned char * pp, * p0;
int bytes=width/2;
unsigned char c;
asm{ mov dx,3ceh

```

```

mov al,5h
out dx,al
mov dx,3cfh
mov al,2
out dx,al
}
if((pp=unsigned char *)malloc(width+1))==NULL) return;
p0=pp;
unsigned int offset,x,y=0;
unsigned char shift __count,bit __mask,color;
for(unsigned int i=1;i<height;i++)
{
pp=p0;
fseek(fp,118L+(long(height)-long(i))*bytes,0);
for(int j=0;j<bytes;j++)
{c=fgetc(fp);
*pp++=((c&0xf0)>>4);
*pp++=(c&0xf);
}
pp=p0;
x=0;
for(j=0;j<width;j++)
{
color= * pp++;
offset=y*80+x/8
shift __count=x%8;
bit __mask=((0x80>>shift __count)&0x00ff);
asm{
mov al,8
mov dx,3ceh
out dx,al
mov al,bit __mask
mov dx,3cfh
out dx,al
mov ax,0a000h
mov es,ax
mov bx,offset
mov al,es:[bx]
mov al,color
mov es:[bx],al
}
x++;
}
y++;
}
asm{
mov al,5
mov dx,3ceh
out dx,al
mov al,0
mov dx,3cfh
out dx,al
}
free(p0);
getc();
asm{mov ax,0003h
int 10h
}
fclose(fp);
}

```

Windows 多文档界面中状态行的实现

□赵艳利

自

1992年Windows 3.1操作系统面市以来,其功能强大的图形窗口界面、多任务的操作环境、以及强大的内存管理功能倍受PC机用户的青睐。Microsoft Windows的推出,使得操作计算机的方式和软件开发过程发生了革命性的变化,国外各大软件公司纷纷推出了自己的for Windows的软件产品,Borland公司也因此推出了Borland C++ 3.1 for Windows的软件开发工具。Borland C++ 3.1成功地引入了面向对象(OOP)的程序设计方法,OOP编程大大降低了软件开发的困难程度,同时,Borland专门为开发Windows应用软件提供了Object Windows Library(简称OWL),对象窗口库OWL这种新的应用程序机制的采用,避免了用传统C代码编写Windows程序在其主窗口回调函数中一段庞大的开关语句,提高了程序的可读性,使程序容易维护和调试。OWL对应用程序隐藏了许多底层细节,用户只需很少的代码就能设计出完美的Windows用户界面。MDI多文档界面是Windows操作系统的标准界面,它可以在一个Windows应用程序中显示和操作多个文本。开发Windows的多文档MDI界面是Windows软件开发中最具挑战性的工作之一,利用OWL开发MDI界面较用C代码容易得多。如果用户使用过Borland C++编译程序的TCW或IDE版本,就会发现该软件窗口底部有一个状态行,该状态行随时为用户提供一些反馈信息以确定哪些命令在给定时刻是有效的,哪些热键是激活的,或其他的描述表相关的信息,使得程序界面更加友好,众多的商

业软件都提供了此种手段。众所周知,OWL对用户隐藏了许多细节,这使得用户在MDI界面中使用状态行更加困难。笔者在编程实践中积累了一些经验和技巧,现分步介绍如下:

一、MDI 界面的结构

在每个MDI应用程序中都包含一个称为MDI框架窗口(frame window)的主窗口,在该框架窗口的用户区有一个不可见的窗口即MDI用户窗口(client window),它对动态创建的MDI子窗口(MDI child Window)提供幕后管理。框架窗口通过标准菜单Tile、Cascade、Arrange Icons、Close All来操作MDI子窗口,MDI子窗口可以重叠,最大化,最小化,每个最小化的窗口都有一个图标,MDI子窗口不会出现在框架窗口外,MDI子窗口不能有菜单,它的功能由框架窗口的菜单来控制。

二、状态行的实现

实现状态行有许多不同的方法,一种可以用子窗口来实现,另一种采用绘制方式来实现状态行。众所周知,由于OWL隐藏了某些细节,再加之MDI界面的特点,用子窗口方式在MDI界面中实现状态行,状态行难以实现和管理。基于上述原因,在MDI界面中实现状态行最好采用绘制方式。

本文采用绘制方式实现状态行。状态行在MDI界面中占据框架窗口位置,而用户看不见的用户窗口通常占据框架窗口用户区的整个空间,为了能正确实现状态行,在构造用户窗口时要给状态行留出空间,同时,当作为主

窗口的框架窗口大小改变或移动时,状态行和用户窗口要调整各自的位置占满整个用户区,并且各自不发生干涉,这就要在创建用户窗口时应考虑状态行的影响,改变缺省用户窗大小,为状态行留出空间。由于 OWL 隐藏了创建用户窗口的细节,因此用户必须在主窗口构造函数中调用 SetupWindow() 显示地创建用户窗口。

```
void TMainWindow::SetupWindow()
{
    HMENU FrameMenu;
    RECT R;
    InitClientWindow();
    RemoveClient();
    FrameMenu = GetMenu(HWindow);
    ClientWnd -> ClientAttr -> hWindowMenu = GetSubMenu (
    FrameMenu
    ,ChildMenuPos);
    GetClientRect(HWindow,&R);
    if (ClientWnd->Attr.X==CW__USEDEFAULT)
    {
        ClientWnd->Attr.X=R.left;
        ClientWnd->Attr.Y=R.top;
    }
    if (ClientWnd->Attr.W==CW__USEDEFAULT)
    {
        ClientWnd->Attr.W=R.right-R.left;
        ClientWnd->Attr.H=R.bottom-R.top-StatLineHeight; // 为状态行
        // 留出空间
    }
    ...
}
```

程序通过主窗口 WM__PAINT 绘制状态行,为使状态行在主窗口最大化、最小化、移动或改变大小后,能够始终保持状态行的正确位置及大小,程序要响应 WM__SIZE 来调整状态行,并重画状态行。

```
void TMainWindow::WMSize(TMessage&.Msg)
{
    ...
    MoveWindow(ClientWnd-HWindow,
    XClient,
    YClient,
    ClientWidth,
    ClientHeight-StatLineHeight; //用户窗口高度减去
    //状态行高度
    TRUE);
    InvalidateRect(HWindow,&StatLineRectangle,0);
}
```

三、状态行的访问

状态行是一个系统对象,为使其被其他任意对象访问,最好的方法就是创建一个全局系统管理员来访问状态行。

```
class TSystemManager;
class TMainWindow;
TSystemManager * TheSystem;
class TSystemManager
{
    TMainWindow * mainwindow;
public:
    TSystemManager(TMainWindow *);
    void StatLineMessage(char *);
    char * StatLineMessage();
}
```

系统管理员通过全局指针 TheSystem 被访问。该指针由主窗口类来初始化,当应用程序被终止时, TSystemManager 对象必须被释放。该项工作可在主窗口的 CanClose() 函数中完成。

```
BOOL TMainWindow::CanClose()
{
    delete TheSystem;
    return TRUE;
}
```

源程序清单如下:

```
#include <edit.h>
#include <filewnd.h>
#include <string.h>
#include <strstrea.h>
#include <mdi.h>
#include <owl.h>
LPSTR APPLICATION__NAME="MDI STATLINE DEMO";
#define SIZEMSGW(arg) (LOWORD(arg.LParam))
#define SIZEMSGH(arg) (HIWORD(arg.LParam))
class TSystemManager;
class TMianWindow;
TSystemManager * TheSystem;
class TSystemManager
{
    TMainWindow * mainwindow;
public:
    TSystemManager(TMainWindow *);
    void StatusLineMessage(char *);
    char * StatusLineMessage();
};
class TMainWindow : public TMDIFrame
{
    char message[80];
    RECT StatusLineRectangle;
public:
    TMainWindow(LPSTR,LPSTR);
    virtual void WMSetFocus(TMessage&.) =
    [WM__FIRST+WM__SETFOCUS];
    virtual void WMKillFocus(TMessage&.) =
    [WM__FIRST+WM__KILLFOCUS];
    virtual void MWSize(TMessage&.) =
    [WM__FIRST+WM__SIZE];
    virtual void About(TMessage&.) =
    [CM__FIRST+CM__ABOUR];
    virtual void SetupWindow();
    virtual void Paint(HDC,PAINTSTRUCT&);
    void Message(char *);
    char * Message(){ return message;}
```

```

BOOL CanClose();
};
TMainWindow::TMainWindow(LPSTR Title,LPSTR Menu)
:TMDIFrame(Title,Menu)
{ ChildMenuPos = 0; }
#define STATUS __LINE__ BACKGROUND RGB(192,192,192) //
定义状态行背景颜色
#define STATUS __LINE__ TEXT RGB(128,0,0) //定义状态行文
本颜色
void TMainWindow::Paint(HDC hdc,PAINTSTRUCT&.)
{ TEXTMETRIC tm; GetTextMetrics(hdc,&tm);
int charHeight = tm.tmHeight + tm.tmExternalLeading;
ReleaseDC(HWindow,hdc);
RECT r;
GetClientRect(HWindow,&r);
statusLineRectangle.left = r.left-1;
statusLineRectangle.top = r.bottom-charHeight-4;
statusLineRectangle.right = r.right+1;
statusLineRectangle.bottom = r.bottom;
SelectObject(hdc, CreateSolidBrush(STATUS __LINE__
BACKGROUND));
Rectangle(hdc,statusLineRectangle.left,
statusLineRectangle.top,
statusLineRectangle.right,
statusLineRectangle.bottom);
SetTextColor(hdc,STATUS __LINE__ TEXT);
TextOut(hdc,3,statusLineRectangle.bottom-charHeight-2,
message,strlen(message));
}
void TMainWindow::WMSize(TMessage& Msg)
{ int ClientWidth,ClientHeight;
int FrameWidth,FrameHeight;
FrameWidth=SIZEMSGW(Msg);
FrameHeight=SIZEMSGH(Msg);
TEXTMETRIC tm;
GetTextMetrics(hdc,&tm);
int charHeight = tm.tmHeight + tm.tmExternalLeading;
ReleaseDC(HWindow,hdc);
statusLineRectangle.left = -1;
statusLineRectangle.top = FrameHeight-charHeight-4;
statusLineRectangle.right = FrameWidth+1;
statusLineRectangle.bottom = FrameHeight;
ClientWidth=FrameWidth;
ClientHeight=FrameHeight-charHeight-4;
MoveWindow(ClientWnd->HWindow, 0, 0, ClientWidth,
ClientHeight, TRUE);
InvalidateRect(HWindow,&statusLineRectangle,0);
}
void TMainWindow::SetupWindow()
{ HMENU FrameMenu;
RECT R;
InitClientWindow();
RemoveClient();
FrameMenu = GetMenu(HWindow);
ClientWnd->ClientAttr->hWindowMenu = GetSubMenu
(FrameMenu,ChildMenuPos);
GetClientRect(HWindow,&R);
if (ClientWnd->Attr.X==CW_USEDEFAULT)
{ ClientWnd->Attr.X=R.left;
ClientWnd->Attr.Y=R.top+charHeight+4;
}
}

```

```

if (ClientWnd->Attr.W==CW_USEDEFAULT)
{ ClientWnd->Attr.W=R.right-R.left;
ClientWnd->Attr.H=R.bottom-R.top-charHeight-4;
}
TheSystem=new TSystemManager(this);
ClientWnd->Attr.Style|=WS_VSCROLL+WS_HSCROLL;
if (ClientWnd->Create())
TWindow::SetupWindow();
else
Status=EM_INVALIDCLIENT;
}
void TMainWindow::Message(char * string)
{ strcpy(message,string,sizeof(message));
InvalidateRect(HWindow,&statusLineRectangle,0);
}
TSystemManager::TSystemManager(TMainWindow * window)
{ mainWindow=window;
TheSystem = this;
}
void TSystemManager::StatusLineMessage(char * cp)
{ mainWindow->Message(cp);
}
void TMainWindow::WMSetFocus(TMessage&.)
{ TheSystem->StatusLineMessage("Please select a command");
}
void TMainWindow::WMKillFocus(TMessage&.)
{ TheSystem->StatusLineMessage("");
InvalidateRect(HWindow,&statusLineRectangle,0);
}
BOOL TMainWindow::CanClose()
{ delete TheSystem;
return TRUE;
}
class TAboutDialog : public TDialog
{ public: TAboutDialog(PTWindowsObject);
};
TAboutDialog::TAboutDialog(PTWindowsObject AParent)
TDialog(AParent,"ABOUT __ DIALOG")
{ TheSystem->StatusLineMessage("Press the OK button");
}void TMainWindow::About(TMessage&.)
{ GetApplication->ExecDialog( new TAboutDialog(this));
}class TMyApp : public TApplication
{ public: TMyApp ( LPSTR AName, HINSTANCE hInstance,
HINSTANCE hPrevInstance,
LPSTR lpCmdLine,int nCmdShow)
: TApplication(AName,hInstance,hPrevInstance,lpCmdLine,
nCmdShow){}
virtual void InitMainWindow();
}; void TMyApp::InitMainWindow()
{ MainWindow = new TMainWindow ( APPLICATION __ NAME,"
MENU");
}
int PASCAL WinMain ( HINSTANCE hInstance, HINSTANCE
hPrevInstance,
LPSTR lpCmdLine,int nCmdShow)
{ TMyApp MyApp(APPLICATION __ NAME,hInstance,hPrevInstance,
lpCmdLine,
nCmdShow);
MyApp.Run();
return MyApp.Status;}

```

SKILL

动态小字库的驻留及使用

薛 海

实

际应用中,我们经常需要读取汉字点阵信息并将其在屏幕上显示出来。目前我们采取的方法大致是:直接读取硬盘汉字库文件或将其制成小字库形式。前者汉字读取速度慢,对硬盘的寿命及汉字显示速度有一定的影响。而小字库虽没有上述问题但最终由于存入的字数有限使其应用较少。

为了克服以上问题,笔者研制了一套读取汉字点阵信息程序。它可随系统的不同情况将汉字点阵信息驻留到相应内存中,且不占基本内存或尽量少占基本内存。其读取汉字点阵速度及质量完全与汉字系统相当。本程序主要采用 XMS 技术、动态小字库技术以及与汉字系统软件接口等技术。现将有关技术介绍如下:

一、扩充内存管理规范 (XMS) 及其应用

1. XMS 规范

在扩充内存管理规范(XMS)中,扩充内存指 640K 以上的存储空间。

XMS 是从 Windows 3.0 开始引入的扩充内存使用规范标准,系统程序 HIMEM.SYS 便是该规范的管理程序。它可使用户象使用基本内存一样方便地使用扩充内存。XMS 提供了检测 XMS 驱动程序是否存在、XMS 程序控制功能地址、驱动程序信息、HMA 管理、A20 线管理、扩充内存管理及上位存储区管理等系统功能。这里因篇幅所限主要介绍扩充内存管理子功能,下面给出 XMS 类库及部分功能源代码。

//程序功能:扩充内存管理(采用 XMS V2.0 规范)

```
class XMS {
    struct XMSstatus {
        WORD version, revision;
        BOOL HMA __ exist;
    } status;
    void getXMSpoint(void); //取 XMS 驱动程序的
        //入口指针地址
    void (far * XMSCALL)(void); //XMS 中断调用
        //函数

public:
    struct { // 扩充内存块传送结构
        LONG bytes; // 0000H-0003H 需传送的 32 位
            //字节数
        HANDLE s __ handle; // 0004H-0005H 源块
            //句柄
        union { // 0006H-0009H 32 位源偏移值
            LONG s __ address;
            struct {
                int offset;
                int segment;
            } s __ segoff;
        } sourceaddress;
        HANDLE t __ handle; // 000AH-000BH 目的句柄
        union { // 000CH-000FH 32 位目的偏移值
            LONG t __ address;
            struct {
                int offset;
                int segment;
            } t __ segoff;
        } targetaddress;
    } XMSblock;
    struct memstruct {
        WORD max; //以 KB 计的最大自由扩充内存块
            //的大小
        WORD size; //以 KB 计的自由扩充内存的总的
            //大小
    } XMSmem;
    BOOL XMStest(void); //测试 XMS 是否安装, 安装在
        //为 TRUE
    void XMSstatus(void); //取 XMS 状态(版本号及 HMA
        //存在信息)
    void XMSavail(void); //查询自由扩充内存
    HANDLE XMSalloc(WORD); //从自由扩充内存池中分
        //配一指定大小内存块
}
```

```

BOOL XMSrealloc(HANDLE,WORD); //改变扩充内存块的大小
                                //(重新分配扩充内存)
BOOL XMSfree(HANDLE handle); //释放分配的扩充内存块
BOOL XMSlock(HANDLE handle); //锁住扩充内存块
BOOL XMS unlock(HANDLE); //扩充内存块解锁
BOOL XMSbackstatus(HANDLE,BYTE&,.BYTE&); //取扩充
                                //内存块的附加信息
BOOL XMSmove(void); //移动扩充内存块
BOOL HMAalloc(WORD); //请求高内存区 HMA(HMA 共 65520
                                //字节)
BOOL HMAfree(void); //释放高内存区
BOOL EnableA20(void); //全程启用 A20 线
BOOL DisableA20(void); //全程停用 A20 线
BOOL A20status(void); //检查 A20 线的状态(状态:0 失败,
                                //1 成功)
BOOL UMBAlloc(WORD UMBsize;WORD&,.WORD&); //申请
                                //上位存储块 UMB
BOOL UMBfree(WORD); //释放上位存储块 UMB
BOOL XMS::Init(void); //XMS 初始化函数(取 XMS 调用地址,
                                //扩充内存大小等)
};
//测试 XMS 是否安装,返回真假-----
BOOL XMS::XMStest(void);
{
    union REGS regs;
    struct SREGS sregs;
    //DOS 功能调用 INT2FH 功能 43H 子功能 00H(取 XMS 驱动
//程序的安装状态)
    //入口:AH=43H
    //    AL=00H
    //返回:AL=80H XMS 驱动程序已加载
    //    AL=00H XMS 驱动程序未加载
    regs.x.ax=0x4300;
    int86(0x2f,&regs,&regs);
    if(regs.h.al!=0x80) return FALSE;
    return TRUE;
}
//取 XMS 驱动程序的入口指针地址-----
//注:当应用程序要使用 XMS 调用前必须先取 XMS 驱动程序
//的入口指针地址!
void XMS::getXMSpoint(void)
{
    union REGS regs;
    struct SREGS sregs;
    //DOS 功能调用 INT2FH 功能 43H 子功能 10H(取 XMS 驱动程序
//的安装状态)
    //入口:AH=43H;AL=10H
    //返回:Es:BX Es XMS 驱动程序段址;BX XMS 驱动程序偏址
    regs.x.ax=0x4310;
    int86(0x2f,&regs,&regs,&sregs);
    FP __SEG(XMSCALL)=sregs.es;
    FP __OFF(XMSCALL)=regs.x.bx;
}
//取 XMS 状态(版本号及 HMA 存在信息)-----
//注:版本信息放在 status 结构中
void XMS::XMSstatus(void)
{
    //XMS 调用 AH=00H(取 XMS 版本号)
    //入口:AH=00H
    //返回:AX XMS 版本号;BX XMS 内部版本号
    //    DX HMA 存在标志(0 不存在,1 存在)
    __AH=0x00;
    XMSCALL();
    status.version=__AX;
    status.revision=__BX;
    status.HMA__exist=__DX;
}
//查询自由扩充内存-----
//注:有关内存大小信息存放 XMSmem 结构中
void XMS::XMSavail(void)
{
    //XMS 调用 AH=08H(取自由扩充内存)
    //入口:AH=08H
    //返回:AX 以 KB 计的最大自由扩充内存块的大小
    //    DX 以 KB 计的自由扩充内存的总的大小
    __AH=0x08;
    XMSCALL();
    XMSmem.max=__AX; //以 KB 计的最大自由扩充内存块的
                                //大小
    XMSmem.size=__DX; //以 KB 计的自由扩充内存的总的大小
}
//从自由扩充内存池中分配一指定大小内存块-----
//注:分配的内存应以 KB 为单位,若返回为句柄为 NULL,则
//表示分配失败
HANDLE XMS::XMSalloc(WORD memory __size)
{
    //XMS 调用 AH=09H(分配一指定大小内存)
    //入口:AH=09H;DX=以 KB 计的内存块大小
    //返回:AX 状态 0 失败,1 成功
    //    DX 分配块的句柄
    __AH=0x09; __DX=memory __size;
    XMSCALL()
    if(__AX)return(__DX)
    return(NULL);
}
//改变扩充内存块的大小(重新分配扩充内存)-----
BOOL XMS::XMSrealloc(HANDLE handle,WORD new __size)
{
    //XMS 调用 AH=0FH(重新分配扩充内存)
    //入口:AH=0FH
    //    BX=以 KB 计新的内存块大小
    //    DX=要改变大小的块的句柄
    //返回:AX 状态 0 失败,1 成功
    __AH=0x0F;
    __BX=new __size;
    __DX=handle;
    XMSCALL()
    if(__AX) return(TRUE)
    return(FALSE);
}
//释放分配的扩充内存块-----
//BOOL XMS::XMSfree(HANDLE handle)
{
    //XMS 调用 AH=0AH(释放已分配扩充内存)
    //入口:AH=0AH
    //    DX=要释放的块的句柄
    //返回:AX 状态 0 失败,1 成功
    __AH=0x0A;
    __DX=handle;
    XMSCALL()
    if(__AX) return(TRUE)
    return(FALSE);
}

```

```

}
//移动扩充内存块-----
BOOL XMS::XMSmove(void)
{
//XMS 调用 AH=0BH(移动扩充内存)
//入口:AH=0BH
//DS:SI 指向扩充内存块移动结构的指针(请参见 XMSblock 说明)
//返回:AX 状态 0 失败,1 成功
asm { push ds;
      __SI=FP __ OFF((LPINT) &XMSblock);
      __DS=FP __ SEG((LPINT) &XMSblock);
      __AH=0x0B;
      XMSCALL();
      asm { pop ds;
            if(__AX) return(TRUE);
            return(FALSE);
          }
//XMS 初始化函数(取 XMS 调用地址,扩充内存大小等)----
BOOL XMS::Init(void)
{if(! XMSstst()) return(FALSE); //判断 XMS 驱动程序是否存在
  get XMSpoint(); //取 XMS 调用地址
  XMSstatus(); //取 XMS 版本号(信息放在 status 中)
  XMSavail(); //取扩充内存大小(信息放在 XMSmem 中)
  return(TRUE);
}

```

2. 将汉字库读入 XMS

根据 XMS 规范提出的功能,我们可有效地将汉字库存入扩充内存中,并在需要时将其点阵信息提出显示。以下便是将汉字库读入扩充内存和读出的实用程序。

```

XMS HZK16XMS;
static BYTE dotbuffer[32]; //读出的点阵信息缓冲池
HANDLE handle; //XMS 中所申请 HZK16 的句柄
/* ----- */
//将汉字 16 点信息读入扩充内存,原理是先初始化 XMS 并判断扩充内存内存是否可容纳的下 HZK16。若容纳的下在 XMS 中分配内存并同时在常规内存中分配一个 32K 的内存以做与 XMS 的交换信息时的缓冲池,调用 XMS 的传送子功能将 HZK16 读入扩充内存。
/* ----- */
BOOL Load __ HZK16 __ XMS(void)
{
LONG l;
int Filehandle;
WORD avil __ size; HZK16 __ size;
unsigned nread;
LPSTR p;
if (! HZK16XMS.Init()) return(FALSE);
//将 HZK16 的长度转换为 KB 方式。
HZK16 __ size=(261696L+1024L)/1024L; //HZK16 长度为
//261696
if (HZK16XMS.XMSmem.max < HZK16 __ size)
return(FALSE);
if ((handle=HZK16XMS.XMSalloc(HZK16 __ size))==NULL)
return(FALSE);
if ((p=(LPSTR)malloc(0x8000))==NULL) { //0x8000=32K
  HZK16XMS.XMSfree(handle);
  return (FALSE);
}
if (__dos __ open ("HZK16",O __ RDONLY,&Filehandle)){

```

```

  puts("Errpr:Open file error!");
  exit(1);
}
HZK16XMS.XMSblock.s __ handle=0;
HZK16XMS.XMSblock.t __ handle=handle;
HZK16XMS.XMSblock.sourceaddress.s __ segoff.segment=
  FP __ SEG((LPSTR)p);
HZK16XMS.XMSblock.sourceaddress.s __ segoff.offset=
  FP __ OFF((LPSTR)p);
l=0;

do {
  __dos __ read(Filehandle,ptr,0x8000,&nread);
  if (nread>0){
    HZK16XMS.XMSblock.targetaddress.t __ address=1;
    HZK16XMS.XMSblock.bytes=nread;
    if (! HZK16XMS.XMSmove()) { //如 XMS 传送错误,则释放
      //XMS 句柄
      HZK16XMS.XMSfree(handle);return(FALSE);
    }
    l+=nread;
  }
  while (nread==0x8000);
  __dos __ close(Filehandle);
  return(TRUE);
}

/* ----- */
//从扩充内存中读取汉字 16 点阵信息,原理是:根据汉字的区位码计算出在扩充内存中的偏址,并调用 XMS 的传送子功能将点阵信息调入 dotbuffer 缓冲池中。
/* ----- */
void GetXMSDot16(WORD code)
{
LONG l; //l 为汉字字模偏址
l=((LONG)((((UINT)code)>>8)-0xal)*0x5e+((UINT)code&0xff)-0xal)<<5);
HZK16XMS.XMSblock.s __ handle=handle;
HZK16XMS.XMSblock.t __ handle=0;
HZK16XMS.XMSblock.targetaddress.t __ segoff.segment=
  FP __ SEG((LPSTR)dotbuffer);
HZK16XMS.XMSblock.targetaddress.t __ segoff.offset=
  FP __ OFF((LPSTR)dotbuffer);
HZK16XMS.XMSblock.sourceaddress.s __ address=1;
HZK16XMS.XMSblock.bytes=0x20
if (! HZK16XMS.XMSmove()) { //将扩充内存移到常内存
  puts("错误:读来自 XMS 的字库信息错误!");
  HZK16XMS.XMSfree(handle); //释放句柄
  exit(1);
}
}

```

二、动态小字库技术

所谓动态小字库技术就是将每次从硬盘中读出的汉字进行保留,每次显示时先在小字库中查找,若找不到就将此字从硬盘中读出并加入到小字库中。当小字库满时则将小字库中最久不用的字从小字库中删除。这样一直动态地维护着小字库。下面便是一组维护小字库的程序。


```
static FILE *HZK16 __P //汉字库文件指针
#define HzNum 100 //小字库汉字数为 100
int Hztime=0;
static struct {
    WORD HzCode; //汉字
    WORD UseCount; //最近一次时间计数
    BYTE Buffer[32]; //汉字点阵
} SLib[HzNum];
//读来自小字库的汉字点阵信息
void GetRAMDot 16(WORD code)
{
    int count __min,mini;
    register int i;
    if (Hztime>=60000) Hztime=0
    else Hztime+=1;
    count __min=60000; //为比较最小值而设的最初值
    for(i=0;i<=HzNum;i++){
        if (code==SLib[i].HzCode){ //在小字库中找汉字
            SLib[i].UseCount=Hztime; //时间记录更新
            __fmemcpy(LPSTR)dot buffer,SLib[i].Buffer,0x20);
            return;
        }
        //找最久未用汉字,其偏址在 mini 上反映
        if(SLib[i].UseCount<count __min){
            mini=i,Count __min=SLib[i].UseCount;
        }
    }
    SLib[mini].HzCode=code;
    SLib[mini].UseCount=1;
    //从硬盘中读字库
    fseek(HZK16 __P (LONG) (((code>>8)&0xff)-0xa1)*
    0x5e+(code&0xff)-0xa1)<=5,SEEK __SET);
    fread(SLib[mini].Buffer,1,0x20,HZK16 __P);
    __fmemcpy((LPSTR)dotbuffer,SLib[mini].Buffer,0x20);
    return;
}
```

三、汉字系统接口技术

以上两种方法适用于在无汉字系统条件下进行的。若当前已在汉字系统下,上述两种方法虽可以使用,但这样就没有充分利用系统资源。一般在进入汉字系统后,系统已将汉字库读入扩充内存或 RAM 中,并提供了相应软件接口。现以 2.13 系统为例介绍如下:

在 2.13 汉字系统中提供了若干系统接口,其中软中断 Int7F 便是专门为提取汉字 16 点阵信息而设置的。利用该接口系统可顺利的从字库中读出点阵信息。下面便是用 Int7F 中断提取汉字点阵信息一例。

```
//调用 INT7F 读入汉字点阵信息
void Int7F __Dot16(WORD code)
{
    LPSTR Buffer;
    struct REGPACKpreg;
    //Int7F 使用规则
```

```
//入口:DX= 汉字
//出口:DX:0000=汉字点阵信息数据地址
preg.r __dx=code;
intr(0x7f,&preg); //int7F
Buffer=(LPSTR)MK __FP(preg.r __dx,0x0000);
__fmemcpy((LPSTR)dotbuffer,Buffer,0x20);
}
```

使用本程序时应注意读汉字点阵信息模块(如 FILE2A.COM 等汉字程序)已加载至内存,否则将会死机。使用前最好判断汉字系统是否加载。其他汉字系统的点阵信息读取请参阅该汉字系统的使用说明。

四、汉字的驻留及读取

综上所述,我们可以将上述三种技术封装成一个函数。该函数可从由初始化函数自动判断得到的读取方式来读取汉字点阵信息。这样主要判断均由程序来作,用户只需调用即可。下面例出的即是初始化函数及读点阵信息函数。

```
XMS HZK16XMS;
static BYTE dotbuffer[32]; //读出的点阵信息缓冲区
HANDLE handle; //XMS 中所申请 HZK16 的句柄
BYTE ReadType; //0 Int7F 中断读;=1 小字库读;=2 XMS
读
/* ----- */
| 初始化读汉字模块;主要工作原理:首先判断是否进入了汉字 |
| 系统(即可通过 Int7F 的地址得出),若进入则设置读中断方式 |
| ,否则判断将汉字库读入扩充内存是否成功,若成功则设置读 |
| XMS 方式,否则设置为读小字库方式 |
/* ----- */
void HZInit(void)
{
    register int i;
    //判断 Int7F 是否存在,存在时其地址为:0F42:020A
    if ((FP __SEG(getvect(0x80))=0xF42)&&(FP __OFF(getvect
    (0x80))=0x020A)){
        ReadType=0; //设置中断读方式
        return;
    }
    if (!Load __Hzk16 __XMS()){
        if ((HZK16 __P=fopen("HZK16","rb"))==NULL){
            puts("Error:HZK16 can't open!");
            exit(1);
        }
        for (i=0;i<HzNum;i++){
            SLib[i].HzCode=0;
            SLib[i].UseCount=0;
        }
        ReadType=1; //设置小字库读方式
    };
    else ReadType=2; //设置 XMS 读方式
}
//关闭汉字模块(其主要功能为:关闭汉字句柄)
```

(下转第 44 页)

使用 ADS 为 AutoCAD 做二次开发

□ 阙俊林 王绍安

A

AutoCAD 是美国 Autodesk 公司推出的在微机
上运行 CAD 软件。它具有功能强,支持平台
多,版本升级快,便于二次开发的特点。
AutoCAD 软件包从 2.17R 开始,引入了 LISP
语言的功能。从 AutoCAD 11.0 开始增加了 C
语言开发环境,C 语言与 LISP 语言相比具有
速度快、保密性强、易于编程,特别是用于编
写大型系统,更能显示出它的优点,所以它一
推出就为广大程序员所喜爱。AutoCAD
11.0R 对编译环境要求非常严格。编译器只
能使用 Metaware High C 或者 Watcom C,
连接程序只能使用 Phar Lap 386 link. exe,
调试程序只能使用 Phar Lap 386 debug.
exe,所产生的程序扩展名为 .exp,只能在保
护模式下运行,不能装入扩展名为 .exe 的实
模式目标文件。从 AutoCAD 12.0R 开始增加
了 Borland C++ V3.0, Microsoft C, Zorth
C++ 这几种编译程序,这给编程人员带来
极大方便,因为这几种编译器是大家所常用
并且非常熟悉的。由于 AutoCAD 的特殊性,
AutoCAD 与 ADS 应用程序之间存在一个接
口问题,在做 ADS 应用时有一些特殊要求以
至初学者不知所措。本文就是针对这些初学
者,帮助他们快速入门。

一、ADS 应用程序的结构

一个 ADS 应用程序主要分为两大部分,
第一部分作用是与 AutoLISP 通信接口,第二
部分是用户自己编写的应用程序部分。下面
列出了一个 C 的源程序,它的 main() 函数表
现出了 ADS 应用程序的典型结构。

```
#include <stdio. h>
```

```
#include "adslib. h"
int funcload(void);
int funcunload(void);
int userfun(void);
/* MAIN—主函数 */
void main (int argc, char * argv[])
{
    short scode = RSRSLT; /* 有效结果码(缺省) */
    int stat;

    ads __init(argc, argv); /* 初始化与 AutoLISP 的通信界面 */
    for(;;){ /* 请求或者结果的“无限的”调度循环 */
        /* 用结果码调用 ads __link() 函数确立应用程序与 AutoLISP 之间的通信链路,如果失败,退出。 */
        if((stat=ads __link(scode))<0){
            printf( /* MSG1 */ "SQRT; bad status from ads __link\n", stat);
            fflush(stdout);
            exit(1);
        }

        scode=RSRSLT; /* 重设结果码 */
        /* 判断 ads __link() 函数返回的结果码 */
        switch(stat){
            case RQXLOAD: /* 调入和定义函数 */
                scode=funcload()? -RSRSLT: -RSERR;
                break;
            case RQXUNLD: /* 释放函数,你没必要释放 ADS 应用程序,当执行 LISP 函数(xunload)时它为你做这一切。 */
                scode=funcunload()? -RSRSLT: -RSERR;
                ads __printf( /* MSG2 */ "Unloading. \n");
                break;
            case RQSUBR: /* 执行应用程序的函数 */
                scode=dofun()? RSRSLT: RSERR;
                break;
            case RQSAVE: /* 通知 ADS 应用程序已经调用 AutoCAD 的 SAVE 命令,不要求这样做。 */
                break;
            case RQQUIT: /* 通知 ADS 应用程序已经调用 AutoCAD 的 QUIT 命令,不要求这样做。 */
                break;
            case RQEND: /* 通知 ADS 应用程序已经调用 AutoCAD 的 END 命令,不要求这样做。 */
                break;
```

```

default;
break;
}
}
/* FUNCLOAD—定义应用程序的函数。*/
static int funcload()
{
    if(ads __ defun("userfunc",0) == RTNORM){
        /* 定义外部函数。可在新函数前加前缀 C; 而定义成一个
        AutoCAD 命令。例如: ads __ defun("C:userfunc",0); */
        ads __ regfunc(userfunc,0); /* 登记函数 */
        return 1;
    }else
        return 0;
}

/* FUNCUNLOAD——退出已经定义的应用程序的函数。*/
int funcunload()
{ads __ undef(userfunc);
return RTNORM;
}

/* USERFUNC——当收到一个 RQSUBR 结果码时执行已经定义的应用程序的函数。*/
int userfunc()
{
    /* 用户的函数部份。*/
}

```

从以上例子我们可以看出一个 ADS 应用程序的执行过程: 1. 用 AutoLISP 的函数(xload)将应用程序装入; 2. ADS 应用程序通过调用函数 ads __ init() 初始化与 AutoLISP 通信; 3. ADS 应用程序用结果码 RSRSLT 调用 ads __ link() 函数确立应用程序与 AutoLISP 之间的通信链路; 4. ADS 应用程序调用函数 ads __ defun() 定义一个外部函数; 5. ADS 应用程序用结果码 RSRSLT 再一次调用 ads __ link() 函数。等待用户或 AutoLISP 请求, 并且执行相应过程, 直到程序被释放。

按照这些简要步骤我们可以了解到 ADS 应用程序就是 AutoLISP 的“奴隶”, 它通过调用函数 ads __ link() 等待 AutoLISP 指示直到 AutoLISP 要求它计算一个外部函数或执行别的动作。另一方面, 当 ADS 应用程序正在响应 AutoLISP 的一个请求的时候, AutoCAD 和 AutoLISP 都处于等待来自 ADS 库函数的请求状态, 此时它们不响应用户的任何输入。由于函数 ads __ link() 是重复调用的, 所以我们将它放在一个前面包含有处理不同的 AutoLISP 请求的 switch 语句的“无限”调度循环之中。

二、如何编写 ADS 应用程序

ADS 应用程序既不同于普通的 C 语言程序又不同于 AutoLISP 程序。ADS 应用程序需要通过 Au-

toLISP 与 AutoCAD 建立通讯链路。ADS 应用程序与 AutoLISP 之间的这种通讯/链路结构对各个应用程序而言均是相同的。也就是说 ADS 应用程序与 AutoCAD 之间的接口是一个相对固定的格式。我们在编写 ADS 应用程序时对接口部分并不要过于操心, 重点应放在自己应用程序部分, 为了方便地掌握和使用 ADS, AutoCAD 本身提供了一个名为 TEMPLATE.C 的模板文件。只要遵循以下步骤就能很快编写出一个 ADS 应用程序。

1. 将文件 template.c 拷贝到你的当前目录下, 并且换成应用程序名;
2. 将函数 adsfnc 换成应用程序函数名, 并且加以扩充, 以满足应用程序的需要;
3. 用编译程序编译、连接, 找出语法上的错误, 修改, 产生目标文件;
4. 进入 AutoCAD, 并且用 AutoLISP 函数(xload)将 ADS 应用程序装入;
5. 执行应用程序, 找出错误, 修改源程序, 重新编译连接, 找出语法上的错误, 修改, 产生目标文件。

从以上介绍可以看出编写一个 ADS 应用程序并不难, 难的是理解 ADS 接口部分。要掌握这一套方法一要多看一些例子, 二要多模仿例子编写一些程序。

三、一个实例

以下的程序用来将一个文本文件写入 AutoCAD 图形中, 它演示了一个 ADS 应用程序的完整结构。它的执行速度比用 AutoLISP 编写的同样作用的程序快许多倍。

```

#include<math.h>
#include<stdio.h>
#include<string.h>
#include"adslib.h"
static int loadfuncs();
static int fi(); /* ADS 应用程序名 */

/* MAIN—the main routine */
void main(int argc, char * argv[])
{
    int stat;
    short scode=RSRSLT;

    ads __ init(argc, argv);
    for(;;){
        if((stat=ads __ link(scode))<0){
            printf("TEMPLATE: bad status from ads __ link()=%d n",
            stat);
            /* Can't use ads __ printf to display
            this message, because the link failed */
            fflush(stdout);
            exit(1);}
    }
}

```

```
scode=RSRSLT;
switch(stat){
case RQXLOAD:
    scode=loadfuncs()? RSRSLT:RSERR;
    break;
case RQSUBR;break;
case RQXUNLD;break;
case RQSAVE;break;
case RQQUIT;break;
default;break;
}
}
/* LOADFUNCS——定义外部函数,并将其定义成一个 AutoCAD 命令 */
static int loadfuncs()
{
    if (ads __ defun("C:fi",0)==RTNORM){
        ads __ regfunc(fi,0);
        return 1;
    }else
        return 0;
}
/* fi——外部函数部份 */
static int fi(void)
{FILE * inputfile;
char * filename;
char charline[256];
int i,j,line=0;
ads __ real deltx, delty;
ads __ point tmp, tmp1,base, textpoint;
char ctmp1[]=" ",ctmp2[]=" ";
ads __ command(RTSTR,"cmdecho", RTSTR,"0",NULL);/* 将命令 cmdecho 置为 OFF */
ads __ command(RTSTR,"blipmode",RTSTR,"OFF",NULL);/* 将命令 blipmode 置为 OFF */
tmp[X]=0;tmp[Y]=0;tmp[Z]=0;
ads __ getstring(0,"\\nPlease input a name of file:",filename);/* 输入文件名 */
```

(上接第 41 页)

```
void HZClose(void)
{if (ReadType=2) HZK16XMS.XMSfree(handle);/*释放 XMS
//内存字库
if(ReadType=1) fclose(HZK16 __ P);/*关闭 HZK 库
}
//读汉字点阵信息,点阵信息存入在 dotbuffer 中
void ReadDot16(WORD code)
{if (ReadType=0) Int7F __ Dot16(code);
if (ReadType=1) GetRAMDot16(code);
if (ReadType=3) GetXMSDot16(code);
}
```

五、部分数据结构说明

由于本程序采用了 Windows 风格的命名方法重新定义了一此数据类型,现将其一些本程序用到的类型定义如下。详细说明请参阅 Borland C++ 3.1 的 WINDOW.H。

```
inputfile=fopen(filename,"rt");
if(inputfile==NULL){
    ads __ printf("\\n%s open fail, hit any key exit",filename);
    ads __ exit(1);
}
ads __ getpoint(NULL,"\\nPlease input base point:",base);/* 数字化文本基点 */
textpoint[X]=base[X];textpoint[Y]=base[Y];textpoint[Z]=base[Z];
ads __ getreal("\\nPlease input deltx:",&deltx);/* 输入字间距。 */
ads __ getreal("\\nPlease input delty:",&delty);/* 输入行间距。 */

while(! feof(inputfile)){
    fgets(charline,256,inputfile);
    line++
    j=strlen(charline);
    for(i=0;i<j;i++){
        if(charline[i]<0){
            ctmp2[0]=charline[i];
            ctmp2[1]=charline[i+1];
            i++;
            ads __ command(RTSTR,"text", RTPOINT, textpoint,
RTSTR, " ",RTSTR," ",RTSTR, ctmp2,0);}
        else
        {
            ctmp1[0]=charline[i];
            ads __ command(RTSTR,"text",RTPOINT,textpoint,RTSTR,
" ",RTSTR," ",RTSTR,ctmp1,0);
        }
        textpoint[X]=textpoint[X]+deltx;
    }
    textpoint[X]=base[X];
    textpoint[Y]=textpoint[Y]+delty;
    ads __ printf("\\nLine %d",line);/* 显示所输入的行号 */
}
ads __ retvoid();
return 1;
}
```

SKILL

```
typedef int BOOL
#define FALSE 0
#define TRUE 1
typedef unsigned char BYTE;
typedef unsigned short WORD;
typedef unsigned int UINT;
#ifndef STRICT
typedef signed long LONG;
#else
#define LONG long
#endif
#ifndef NULL
#define NULL 0
#endif
typedef char far * LPSTR;
typedef int far * LPINT;
#ifndef STRICT
typedef const void near * HANDLE;
#else /* STRICT */
typedef UINT HANDLE;
#endif /* ! STRICT */
```

SKILL

三次 Bezier 曲线的一种快速软件还原算法

□王建勇

由

于三次 Bezier 曲线具有对称性、凸包性、几何不变性、灵活性等诸多良好的数学特性,使得它为外形设计、曲线拟合提供了一种较为完善的工具。近几年来,随着计算机技术的不断发展,三次 Bezier 曲线的应用范围变得越来越广泛了。在作为国际工业标准的新一代电子排版语言 PostScript Language 中,有关三次 Bezier 曲线的操作符 Curveto 等构成了该语言中描述图形的重要手段。现在随着人们对文字质量追求的不不断提高,越来越多的曲线字库采用三次 Bezier 曲线描述其字体轮廓。我们在编制一般的图形软件时也常用到 Bezier 曲线,但采用常规算法还原 Bezier 曲线时,其还原精度和速度均难以保证。本文采用分割作图法作为曲线的还原算法,具体实现过程中选取恰当的终止条件,利用浮点数的 IEEE 标准 754 格式巧妙地回避了浮点数乘、除法运算,大大提高了三次 Bezier 曲线还原输出的精度和速度。

一、三次 Bezier 曲线的分割作图法

一般来说,给定一个特征多边形就可唯一确定一条三次 Bezier 曲线的形状。假设特征多边形起点为 Q_0 、终点为 Q_3 、两个控制点分别为 Q_1 和 Q_2 ,那么三次 Bezier 曲线的数学形式可表示如下:

$$P(t) = (1-t)^3 \cdot Q_0 + 3t(1-t)^2 \cdot Q_1 + 3t^2(1-t) \cdot Q_2 + t^3 \cdot Q_3 \quad (0 \leq t \leq 1)$$

由于上式含有大量的乘、除法运算,一般 Bezier 曲线的还原算法并不采用上式,而是利用下面介绍的分割作图法。三次 Bezier 曲线分割作图法的步骤如下:

1. 选取参数 $t_s (0 < t_s < 1)$,下面过程可求得曲线上当参数 $t = t_s$ 时的点,如图 1 所示:

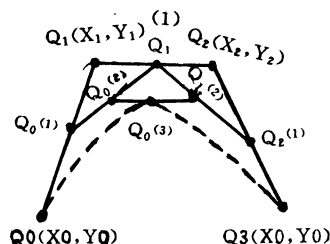


图 1 三次 Bezier 曲线的分割作图法

(1)从特征多边形 $Q_0 Q_1 Q_2 Q_3$ 的第一条边开始,以 $t_s = (1 - t_s)$ 的比例在条边上取一点,得到新的点 $Q_0^{(1)} Q_1^{(1)}$ 和 $Q_2^{(1)}$;

(2)然后,以相同的比例在两条边 $Q_0^{(1)} Q_1^{(1)}$ 和 $Q_1^{(1)} Q_2^{(1)}$ 上各取一点,得到 $Q_0^{(2)} Q_1^{(2)}$;

(3)最后在剩下的一条边 $Q_0^{(2)} Q_1^{(2)}$ 上以相同的比例取得一点 $Q_0^{(3)}$,该点就是曲线上当 $t = t_s$ 时的点。

2. 在求出 $Q_0^{(3)}$ 后,由 $Q_0 Q_0^{(1)} Q_0^{(2)} Q_0^{(3)}$ 和 $Q_0^{(3)} Q_1^{(2)} Q_2^{(1)} Q_3$ 又构成了两个新的特征多边形,然后对这两个特征多边形重复 1 的操作,直到所求曲线满足精度要求为止。

上述分割作图法实际上是一个递归过程,当满足用户的精度要求时,就终止递归调用。终止条件多种多样,较常用的是点 $Q_1^{(1)} Q_0^{(3)}$ 间的距离小于某个常数,即当满足此条件时,就直接用直线段 $Q_0 Q_0^{(3)}$ 和 $Q_0^{(3)} Q_3$ 近似代替曲线段 $Q_0 Q_0^{(3)}$ 和 $Q_0^{(3)} Q_3$ 。

二、对上述还原算法的分析

下面我们对分割作图法进行分析,选取适

当的参数 t 值和终止条件以提高曲线还原速度。

1. t 值选取

在计算机科学中,许多问题的有效解决通常和 2 的幂有关。本文中在采用分割作图法还原 Bezier 曲线时,参数 t 取为 2^{-1} ,即 $t = \frac{1}{2}$ 。下面分析一下这样取值的优点。由上述作图过程看出,当 $t = \frac{1}{2}$ 时,求点 $Q^{(3)}$ 的过程主要包括 6 次求中点的操作,例如 $Q_1^{(1)}$ 为点 Q_1 和 Q_2 的中点,点 $Q_1^{(1)}$ 的坐标: $x_1^{(1)} = (x_1 + x_2)/2$, $y_1^{(1)} = (y_1 + y_2)/2$,若坐标值为整型,那么除以 2 的操作可转换为右移一位的操作;若坐标值取浮点类型,从本文后面的分析中可知浮点数除以 2 的运算可转换为一次简单的减法运算,由于浮点数除法的时间复杂度是减法运算的数十倍,因此参数 t 取为 $\frac{1}{2}$ 能使还原速度有数量级的提高。

2. 递归过程的终止条件

由上面的分割作图法可知,还原曲线的过程是递归的,在用程序实现算法时应选择合适的递归终止条件。选择终止条件的标准应当力求在满足精度要求的基础上提高曲线还原速度。当选取点 $Q_1^{(1)}$ 和 $Q_0^{(3)}$ 间的距离小于某一常数 δ 作为终止条件时,能保证曲线处处均匀还原,但还原速度必然受影响,因为求距离 l 时涉及到浮点数乘法和求平方根运算。当然把终止条件转换为 $l^2 < \delta^2$ 就可消除求平方根运算,但浮点数乘法运算依然存在,势必影响曲线还原速度。因此,终止条件中尽量不含有时间复杂度太大的浮点数乘、除法等运算。本文中选取的终止条件为 $(|x_0 - X_0^{(3)}| < \delta) \&\& (|y_0 - y_0^{(3)}| < \delta)$,这个条件中只牵扯减法运算,对还原速度不会有太大影响,而且已经过程序验证。此控制条件亦能较好地保证曲线还原的精度要求。

三、三次 Bezier 曲线还原算法的具体实现

在进行了上述分析后,可以采用 C 语言及部分 80386 汇编代码具体实现曲线的还原算法。

1. 消除浮点数除以 2 的运算

Microsoft C6.0 支持 float 数据类型,它和 IEEE 标准 754 格式中的 8087 short real 格式完全相同,其格式为:

±	Biased exponent (7FH)	Significand
31 30	23 22	0

由上述格式看出, float 类型数值的最高位是符号位,此位为 1 表示负数,否则为正数。指数以无符

号形式存储,从指数值中减去偏移量 127 即得到真正的指数值。IEEE 标准 754 格式尾数的存储形式为大于等于 1 且小于 2 的二进制小数,对用 float 类型,最高有效位是一个隐含的 1,因此它的尾数实际长为 24 位,最高有效位并不存在于内存中。由于 float 型数值的尾数存储形式为二进制小数,所以一个浮点数除以 2 可被其指数值减 1 的操作所代替,这样一次浮点数除法变一次减法运算,时间复杂度有了数量级的降低。程序清单中的 addzdiv() 子过程就是实现将两个浮点数相加,然后除以 2 的操作,它用 80386 汇编代码实现。采用 80386 汇编代码有一个好处:一个 float 型数值占 4 个字节,恰好可放在一个 32 位长的通用寄存器中,可减少许多不必要的操作,这样做在某种程度上也会提高还原速度。

2. 用递推过程代替递归

当特征多边形的顶点坐标值较大时,若采用递归过程还原曲线,递归深度将会很大,栈中除了要保存过程参数外,还保存返回地址等现场内容,这样会占用很大的栈空间。若采用递推实现,则会明显减少存储空间的开销。又因为递推过程不含调用过程本身的操作,不必进行现场保护,这样也会提高处理速度。

下面给出源程序清单,过程 bezier() 由给定的特征多边形顶点坐标和精度控制参数 flatness 还原出相应的 bezier 曲线。参数 flatness 可控制曲线的光滑程度,其值越小,还原出来的曲线越光滑,但速度相应地变慢。该程序已调试通过并得到了应用,读者可将此过程加入自己的图形软件中直接加以利用。

源程序清单如下:

```

/* 程序 Bezier 所需头文件及宏定义 */
#include <stdio.h>
#include <graph.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#define ox 125
#define oy 125
/* 画 Bezier 曲线的子程序 */
void bezier(float p0, float p1, float p2, float p3, float p4, float p5,
            float p6, float p7, float flatness);
/* 实现两个浮点数的平均数,该子程序用汇编语言编写 */
float add2div(float, float);
/* 演示主程序:用 4 条 Bezier 曲线画一个以(ox,oy)为圆心的圆 */
void main(void)
{
    __setvideomode(__VRES16COLOR);
    __setcolor(14);
    bezier(ox, (oy+125), (ox+69), (oy+125), (ox+125), (oy+69),
           (ox+125), oy, 1);
    __setcolor(13);

```

```

bezier((ox+125),oy,(ox+125),(oy-69),(ox+69),(oy-125),
      ox,(oy-125),1);
__setcolor(12);
bezier(ox,(oy-125),(ox-69),(oy-125),(ox-125),(oy-69),
      (ox-125),oy,1);
__setcolor(11);
bezier((ox-125),oy,(ox-125),(oy+69),(ox-69),(oy+125),
      ox,(oy+125),1);getch();
__setvideomode(__DEFAULTMODE);
return;
}
/* (p0,p1). (p2,p3). (p4,p5). (p6,p7)分别为第一、第二、第三、第
四个控制点,flatness 为精度控制参数 */
void bezier(float p0,float p1,float p2,float p3,float p4,float p5,
           float p6,float p7,float flatness)
{ int x0,y0,x1,y1;
  float p[150];
  register int i;
  p[0]=p6;p[1]=p7;p[2]=p4;
  p[3]=p5;p[4]=p2;p[5]=p3;
  p[6]=p0;p[7]=p1;i=7;
  x0=p[6];
  y0=p[7];
  do
  { p[i+5]=p[i-1];
    p[i+6]=p[i];
    p[148]=add2div(p[i-3],p[i-5]);
    p[149]=add2div(p[i-2],p[i-4]);
    p[i-5]=add2div(p[i-5],p[i-7]);
    p[i-4]=add2div(p[i-4],p[i-6]);
    p[i+3]=add2div(p[i-3],p[i-1]);
    p[i+4]=add2div(p[i-2],p[i]);
    p[i-3]=add2div(p[i-5],p[148]);
    p[i-2]=add2div(p[i-4],p[149]);
    p[i+1]=add2div(p[i+3],p[148]);
    p[i+2]=add2div(p[i+4],p[149]);
    p[i-1]=add2div(p[i+1],p[i-3]);
    p[i]=add2div(p[i+2],p[i-2]);
    if((fabs(p[i+5]-p[i-1])>=flatness)|| (fabs(p[i]-p[i+6])>
      =flatness))
    i+=6;
  } else
  { x1=p[i+3]; y1=p[i+4];
    __moveto(x0,y0);
    __lineto(x1,y1);
    x0=p[i+1];
    y0=p[i+2];
    __lineto(x0,y0);
    x1=p[i-1];
    y1=p[i];
    __lineto(x1,y1);
    x0=x1;
    y0=y1;
    if((fabs(p[i-1]-p[i-7])<flatness)&&(fabs(p[i]-p[i-6])
      <flatness))
    {
      x1=p[i-3];
      y1=p[i-2];
      __moveto(x0,y0);
      __lineto(x1,y1);
      x0=p[i-5];
      y0=p[i-4];

```

```

__lineto(x0,y0);
x1=p[i-7];
y1=p[i-6];
__lineto(x1,y1);
x0=x1; y0=y1; i-=6;
}
} while((i>1)&&(i<=139));
return;
}
/* 该子程序用于实现将两个浮点数相加并除以 2 的操作 */
.model medium
.386
.data
buf dd ?
.code
public __add2div
f1 equ [ebp+6]
f2 equ [ebp+10]
__add2div: enter 0,0
  mov eax,f1
  mov ebx,f2
  cmp eax,0
  jf f1y
  jz f1zero
  cmp ebx,0
  jl f1nf2y
  jz f2zero
  mov ecx,eax
  mov edx,ebx
  shr ecx,23
  shr edx,23
  sub cl,dl
  je wsxjq
  ja tzf2
  xor cl,0fffh
  inc cl
  and eax,0ffffffh
  or eax,800000h
  shr eax,cl
  mov ecx,ebx
  or ecx,800000h
  and ecx,0ffffffh
  add ecx,eax
  cmp ecx,1000000h
  jb xresult
  shr ecx,1
  add ebx,800000h
xresult: and ebx,0ff800000h
  and ecx,7ffffffh
  add ebx,ecx
  sub ebx,800000h
  mov eax,ebx
  mov buf,eax
  mov ax,offset buf
  leave
  retf
tzf2: and ebx,0ffffffh
  or ebx,800000h
  shr ebx,cl

```

```

    jmp wsxj
wsxjq: or ebx,800000h
    and ebx,0ffffffh
wsxj: mov ecx,eax
    or ecx,800000h
    and ecx,0ffffffh
    add ecx,ebx
    cmp ecx,1000000h
    jb result
    shr ecx,1
    add eax,800000h
result: and eax,0ff800000h
    and ecx,7fffffh
    add eax,ecx
    sub eax,800000h
    mov buf,eax
    mov ax,offset buf
    leave
    retf
f1nf2y: mov ecx,eax
    mov edx,ebx
    and edx,7ffffffh
    mov ebx,edx
    cld
    cmp ecx,ebx
    jg jssz
    jl jssf
    jz zresult
f1y: cmp ebx,0
    jl f1yf2y
f1yf2n: mov edx,ebx
    mov ecx,eax
    and ecx,7ffffffh
    mov eax,ecx
    cld
    cmp eax,ebx
    jg jssf
    jz zresult
jssz: shr ecx,23
    shr edx,23
    sub cl,dl
    je rresult
    ja ttzf2
    xor cl,0ffh
    inc cl
    mov edx,eax
    mov eax,ebx
    mov ebx,edx
ttzf2: or ebx,800000h
    and ebx,0ffffffh
    dec cl
    shr ebx,cl
    mov ecx,eax
    or ecx,800000h
    and ecx,0ffffffh
    shl ecx,1
    sub ecx,ebx
    cmp ecx,1000000h
    jb bb
    shr ecx,1
    jmp bbb

```

```

bb: sub eax,800000h
    cmp ecx,0
    jz zresult
tzws: cmp ecx,800000h
    jnb bbb
    shl ecx,1
    sub eax,800000h
    jmp tzws
bbb: and ecx,7fffffh
    and eax,0ff800000h
    add eax,ecx
    jmp zzresult
rresult: or ebx,800000h
    and ebx,0ffffffh
    mov ecx,eax
    or ecx,800000h
    and ecx,0ffffffh
    cmp ecx,ebx
    jz zresult
    jnbe dd1
    mov edx,ecx
    mov ecx,ebx
    mov ebx,edx
dd1: sub ecx,ebx
nnn: cmp ecx,800000h
    jnb ddd
    shl ecx,1
    sub eax,800000h
    jmp nnn
ddd: and ecx,7fffffh
    and eax,0ff800000h
    add eax,ecx
zzresult: sub eax,800000h
    or eax,80000000h
    mov buf,eax
    mov ax,offset buf
    leave
    retf
zresult: mov eax,0
    mov buf,eax
    mov ax,offset buf
    leave
    retf
jssf: shr ecx,23
    shr edx,23
    sub cl,dl
    je frresult
    ja fttzf2
    xor cl,0ffh
    inc cl
    mov edx,eax
    mov eax,ebx
    mov ebx,edx
ftt2f2: or ebx,800000h
    and ebx,0ffffffh
    dec cl
    shr ebx,cl
    mov ecx,eax
    or ecx,800000h
    and ecx,0ffffffh

```

```

shl ecx,1
sub ecx,ebx
cmp ecx,1000000h
jb fbb
shr ecx,1
jmp bbb
fbb: sub eax,800000h
     cmp ecx,0
     jz fresult
ftzws: cmp ecx,800000h
      jnb fbbb
      shl ecx,1
      sub eax,800000h
      jmp ftzws
fbbb: and ecx,7ffffh
      and eax,0ff800000h
      add eax,ecx
      jmp fzzresult
fresult: or ebx,800000h
        and ebx,0ffffh
        mov ecx,eax
        or ecx,800000h
        and ecx,0ffffh
        cmp ecx,ebx
        jz fresult
        jnb fdd1
        mov edx,ecx
        mov ecx,ebx
        mov ebx,edx
fdd1: sub ecx,edx
fnnn: cmp ecx,800000h
      jnb fddd
      shl ecx,1
      sub eax,800000h
      jmp fnnn
fddd: and ecx,7ffffh
      and eax,0ff800000h
      add eax,ecx
fzzresult: or eax,80000000h
          sub eax,800000h
          mov buf,eax
          mov ax,offset buf
          leave
          retf
fzresult: mov eax,0
          mov buf,eax
          mov ax,offset buf
          leave
          retf
f1yf2y: mov ecx,eax
        mov edx,ebx
        shr ecx,23
        shr edx,23
        sub cl,dl
        je ffwsxjq
        ja fftzf2
        xor cl,0ffh
        inc cl
        and eax,0ffffh
        or eax,800000h
        shr eax,cl
    
```

```

mov ecx,ebx
or ecx,800000h
and ecx,0ffffh
add ecx,eax
cmp ecx,1000000h
jb xrst
shr ecx,1
add ebx,800000h
xrst: and ebx,0ff800000h
      and ecx,7ffffh
      add ebx,ecx
      sub ebx,800000h
      mov eax,ebx
      or eax,80000000h
      mov buf,eax
      mov ax,offset buf
      leave
      retf
fftzf2: and ebx,0ffffh
        or ebx,800000h
        shr ebx,cl
        jmp ffwsxjq
ffwsxjq: or ebx,800000h
        and ebx,0ffffh
ffwsxj: mov ecx,eax
        or ecx,800000h
        and ecx,0ffffh
        add ecx,ebx
        cmp ecx,1000000h
        jb rst
        shr ecx,1
        add eax,800000h
rst: and eax,0ff800000h
     and ecx,7ffffh
     add eax,ecx
     sub eax,800000h
     or eax,80000000h
     mov buf,eax
     mov ax,offset buf
     leave
     retf
f1zero: cmp ebx,0
        jz f1f2zero
        sub ebx,800000h
        mov buf,ebx
        mov ax,offset buf
        leave
        retf
f2zero: cmp eax,0
        jz f1f2zero
        sub eax,800000h
        mov buf,eax
        mov ax,offset buf
        leave
        retf
f1f2zero: mov eax,0
          mov buf,eax
          mov ax,offset buf
          leave
          retf
end
    
```

立体三维菜单的 C 语言实现

□ 马 勇

在

计算机信息管理系统中,菜单做为人机交换信息的重要窗口越来越受到重视。菜单技术随着计算机应用的不断深入正逐渐朝立体方向发展,为克服二维立体菜单的不足,于是立体三维菜单就产生了。

要想在屏幕上以逼真的效果显示三维图形就必须了解它在光线照射下四边颜色的变化情况(假定三维图形的颜色为白色)。当光线从左上角以 45 度平射时,其上边和左边则反射成白色;其面则呈浅灰色;其下边和右边则呈现深灰色。当它凹进时,其四边的最外部就要呈现黑色,凸起时,其四边的最外部又恢复到以前的色彩。在屏幕上显示立体三维图形的关键在于如何反映其四边的颜色,为此我们用 C 语言的 line() 和 setcolor() 编写了反映四边变化的函数 windowss(x1,y1,x2,y2,m)。其中 x1,y1,x2,y2,分别为窗口左上角和右下角坐标,m 为图形凹与凸的选择项,m=2 时为凹进,其余为凸起。

源程序清单如下:

```
#include <stdio.h>
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>
#include <bios.h>
#include <dos.h>
#include <process.h>
int inkey();
void windows();
main()
{int gd=VGA,gm=VGAHI;int key,kkey,i;
initgraph(&gd,&gm,"");
setfillstyle(SOLID __FILL,LIGHTMAGENTA);
bar(1,1,getmaxx(),getmaxy());
setfillstyle(SOLID __FILL,LIGHTGRAY);
bar(40,40,225,225);windows(40,40,225,225,1);
```

```
moveto(42,42,);outtext("welcome to use the menu");
moveto(90,75);
outtext("1.ucdos 3.0");winsows(45,68,72,88,1);
moveto(54,75);outtext("1");
moveto(90,110);
outtext("2.ucdos 213");windows(45,103,72,123,1);
moveto(54,110);outtext("2");moveto(90,145);
outtext("3.spdos 3.0");windows(45,138,72,158,1);
moveto(54,145);outtext("3");moveto(90,180);
outtext("4. exit to dos");windows(45,173,72,193,1);
moveto(54,180);outtext("4");kkey=0;
for(;;){key=inkey();
if(key!=='r')
{if(key=='1') windows(45,68,72,88,2);
else if(key=='2') windows(45,103,72,123,2);
else if(key=='3') windows(45,138,72,158,2);
else windows(45,173,72,193,2);
kkey=key;
cobtinue;}
else
{if(kkey=='1'||kkey==0){system("ucdos");exit(1);}
else if(kkey=='2'){system("213");exit(1);}
else if(kkey=='3'){system("spdos/vga");exit(1);}
else exit(0);}}
int inkey(void)
{int key,lo,hi;
key=bioskey(0);lo=key&0x00ff;hi=key&0x00ff>>8;
return((lo==0)? hi+256;lo);}
void windows(x1,y1,x2,y2,m)
int x1,x2,y1,y2,m;
{int i;
if(m==2){setcolor(DARKGRAY);
line(x1-2,y1-2,x2+2,y1-2);
line(x1-2,y1-2,x1-2,y2+2); delay(200);}
setcolor(WHITE);
for(i=1;i<3;i++)line(x1-i,y1-i,x2+i,y1-i);
for(i=1;i<3;i++)line(x1-i,y1-i,x1-i,y2+i);
setcolor(DARKGRAY);
for(i=1;i<3;i++)line(x2+i,y1-i,x2+i,y2+i);
for(i=1;i<3;i++)line(x1-i,y2+i,x2+i,y2+i);}
```

该程序在 GW0540、AST386、Compaq 480/33 机上通过,用 Turbo C 2.0 编制。

SKILL

Turbo C/C++ 编程技巧—— 按键的读取

□ 夏泰真 梁彦忠

键

盘是最普通的输入管道,如何读取按键在程序设计中至关重要,尤其是设计带有分支和菜单的程序。目前,这方面的知识在各报刊中介绍的不多。鉴于这种情况,笔者将在实践中经常使用的三种键码读取方法介绍给大家,供读者参考。本文给出的子程序勿需更改,可象 Turbo C/C++ 的库函数一样直接调用。为了便于说明我们先介绍一下基本概念:

扫描码:当我们按下某一键时,键盘上的集成电路就将该键的扫描码送往主机,由主机做进一步的处理。扫描码的特性是键盘上每个位置按键都对应一个扫描码,对于字母键来说,大小写字母对应同一扫描码,当我们不分大小写读键时,可使用扫描码。注意,因 Shift,Alt,Ins 等几个键单独使用的机会较少,故无扫描码。

ASCII 码:一般 PC 机有 256 个(包括组合键),除功能键和控制键的 ASCII 码为 0 外,其余任意按键都对应一个 ASCII 码,对于字母键,大小写有不同的 ASCII 码。

扩展码:一些功能键如 F1 至 F12 和控制键因无 ASCII 码,而是用扩展码来表示它们。

弄清键的不同码值在程设计中非常必要。下面分别介绍三种读取键码的方法。

1. 用 getch() 读键

Turbo C/C++ 提供了一个库函数 getch(),可利用它来读取按键的 ASCII 码和扩展码。getch()在头文件 conio.h 里,无入口参数,返回值为—整数。

当按下一普通字符键时,getch()返回 ASCII 码;当按下一复合键时,getch()先返回 0,接

着再调用一次返回其扩展码。虽然调用两次,实际只需按键一次。下面是子函数的源代码:

```
int getch1()
{
    int key;
    key=getch();
    if(key==0)return(getch());
    return(key);
}
```

先用 getch()取得按键,若是普通字符,返回 ASCII 码,若是功能键或控制键则返回扩展码。其调用示例如下:

```
#include<stdio.h>
main() {
    int i;
    i=getch1();
    switch(i)
    {
        case 59:printf("F1 键");break;
        case 65:printf("A ");break;
        default;break;
    }
}
```

2. 用 bioskey() 读键

bioskey()读键的功能比较强大,除了返回 ASCII 码和扩展码外,还可以用于检查一些特殊键的按键状态如 Shift,NumLock 和 Alt 键等。

函数源代码如下:

```
int getch2()
{
    int key,key0,key1;
    while(bioskey(1)==0)
    {
        key=bioskey(0);
        key0=key&0x00ff;
        key1=(key>>8);
        key=bioskey(2);
    }
}
```

用 bioskey(0) 测试是否有按键,然后用 bioskey(0)读键,通过右移和 & 运算分别取出按键的扩展码和 ASCII 码。最后用 bioskey(2) 检查特殊键的按键状态。特殊键的按键处理方法如下:

```
if(key&0x80) 则 Ins 键被按下;
if(key&0x08) 则 Alt 键被按下;
if(key&0x10) 则 ScrollLock 键被按下;
if(key&x04) 则 Ctrl 键被按下;
if(key&x40) 则 CapsLock 键被按下;
if(key&0x01) 则 Right-Shift 键被按下;
if(key&0x02) 则 Left-Shift 键被按下;
if(key&0x20) 则 NumLock 键被按下;
在具体程序设计中可直接利用这些处理方法。
```

3. 用 int86() 读键

这是读键功能最强大的一种方法,除了可返回 ASCII 码和扩展码外,还可以用于返回汉字内码。这在编写处理有关中英文的程序时,非常有用。函数源代码如下:

```
int getkey3()
{
```

(上接第 23 页)

```
main()
{
    char c;
    unsigned v;

    gotoxy(26,2);printf("1-----dos");
    gotoxy(26,3);printf("2-----ucdos");
    gotoxy(26,4);printf("3-----super dos");
    gotoxy(26,5);printf("-----");
    gotoxy(26,6);printf("please select(1-3):");
    gotoxy(36,6);
    c=getch();
    while(c<'1' || c>'3') {
        gotoxy(36,6);
        c=getch();
    }
    pokeb(0x40,0x1a,peekb(0x40,0x1c));
    v=0x1e;
    outportb(0x40,0x1a,0x1e);
    switch (c) {
        case '1':break;
        case '2':
            pokeb(0x40,v,'c');v+=2; if (v>0x3c) v=0x3c;
            pokeb(0x40,v,',');v+=2; if (v>0x3c) v=0x3c;
            pokeb(0x40,v,0x0d);v+=2; if (v>0x3c) v=0x3c;
            pokeb(0x40,v,'u');v+=2; if (v>0x3c) v=0x3c;
            pokeb(0x40,v,'c');v+=2; if (v>0x3c) v=0x3c;
            pokeb(0x40,v,0x0d);v+=2; if (v>0x3c) v=0x3c;
            pokeb(0x40,v,'c');v+=2; if (v>0x3c) v=0x3c;
            pokeb(0x40,v,'d');v+=2; if (v>0x3c) v=0x3c;
            pokeb(0x40,v,'\\');v+=2; if (v>0x3c) v=0x3c;
            pokeb(0x40,v,0x0d);v+=2; if (v>0x3c) v=0x3c;
```

```
union REGS reg;
reg.h.ax=0;
int86(0x16,&reg,&reg);
return(reg.h.ax);
}
```

这里利用 Turbo C/C++ 提供的软中断接口库函数 int86() 的 16 号子函数。关于 int86(), 可参考有关的 Turbo C/C++ 使用手册。其调用示例如下:

```
main()
{
    union REGS reg;
    reg.h.ax=getkey3();
    .....
}
```

利用 getkey3() 的返回值可进行不同的处理。判断方法:

如 reg.h.ah=0, 则 reg.h.al 一定为汉字内码;

如 reg.h.ah!=0, 则 reg.h.al、reg.h.ah 均返回 ASCII 码;

如 reg.h.al=0, 则 reg.h.ah 返回扫描码;

综上所述,三种读键方法各有优缺点。实践中可根据需要选择或结合使用。

SKILL

```
break;
case '3':
    pokeb(0x40,v,'c');v+=2; if (v>0x3c) v=0x3c;
    pokeb(0x40,v,',');v+=2; if (v>0x3c) v=0x3c;
    pokeb(0x40,v,0x0d);v+=2; if (v>0x3c) v=0x3c;
    pokeb(0x40,v,'x');v+=2; if (v>0x3c) v=0x3c;
    pokeb(0x40,v,'s');v+=2; if (v>0x3c) v=0x3c;
    pokeb(0x40,v,0x0d);v+=2; if (v>0x3c) v=0x3c;
    break;
default:break;
}
pokeb(0x40,0x1c,v);
outportb(33,0x80);
return 0;
}
```

其中 uc 和 xs 分别是起动 UCDOS 和 SuperCC DOS 的批文件。

本程序用 Borland c++ 3.1 编写,在 AST PII e86/33 DOS 3.3 下通过运行。

SKILL

最少的投资
最佳的方案
最多的组合

挂接式打印机共享器

北京向宇计算机公司

地址:北京复兴路甲 20 号 27 分号 312,314

邮编:100036 电话:8263441,2063366 呼 1511

一种简便而实用的局域网安全策略

□晏章军 王建群

U

NIX 系统是目前非常流行的多用户操作系统,它为用户提供了访问机器的多种途径,也为用户之间和多机之间通信提供了多种工具,极大地方便了用户的使用,特别是较好地满足了计算机网络环境下不同用户、不同机器之间的数据通信的需求,这也是 UNIX 系统倍受用户青睐的一个重要原因。

正因为 UNIX 系统良好的开放性以及提供了众多通信工具,使得 UNIX 系统的安全性较难得到保障。许多未授权用户出于种种目的,常常能利用 UNIX 系统本身的优点打入计算机系统,或者蓄意破坏,或者窃取重要的数据和程序等。即使合法用户,也会发生由于误操作等原因不慎破坏系统数据的现象。在计算机网络环境下,系统的安全性显得尤为重要,非法用户只要侵入一台机器,就会对其他联网机器造成很大威胁,因此,系统管理员需要制定一个好的系统安全性策略,严格控制系统各类用户的使用权限,最大限度地保证系统的安全,特别是我们越来越依赖于 UNIX 机器中的文件和数据时,系统的安全性也随之变得越来越重要。

一、UNIX 系统的一般安全保护措施

UNIX 系统本身也提供了不少安全性保护方法,基本上都是通过软件实现的,其核心是每个用户的注册标识(login id)和口令(password),但由于各种原因,用户的口令有时会被入侵者通过各种途径获得,从而顺利地进入系统,为了保证系统不易被他人入侵和提高系统的坚固性,一般可采取以下方法:

1. 起复杂的用户口令并定期更换用户口

令是提高系统安全性的有效方法。

2. 超级用户口令一定要严格保密,只有系统管理员等重要的系统管理人员才能知道。因为超级用户拥有的权限太大。当然,超级用户口令也要经常改动。

3. 充分利用 UNIX 的文件保护功能,对不同的文件实现不同级别的保护。有些重要文件或数据可设为只有超级用户才能修改。

4. 在每个用户的 profile 文件中,规定用户的不同权限。例如有些用户只能查询系统。做到系统的各个功能对用户透明,不同的用户可执行不同的系统功能。

以上方法虽然可有效地提高 UNIX 系统的安全性,但经常更换用户口令无疑给用户带来使用上的不便,而且这些方法也未从根本上防止非法用户的侵入。系统管理员只能在入侵系统并对系统进行不同程度的破坏后才能察觉到系统的安全性得到威胁,这对一些对安全性要求较高的部门当然是不行的。我们在实践中探索出了一条简便而实用的局域网安全策略。经过半年多的运行,证明行之有效。它不但可防止非法用户入侵系统,而且如有非法用户企图进入系统时,它可记录这些非法用户的登录情况,以便系统管理员随时监测系统的安全情况。

二、主要思想

这种安全性方法的主要思想是巧妙地将软件(用户的注册标识符)和硬件设备(用户注册终端)结合起来。通过构造一个安全级别表,来规定不同用户所能注册的终端。这种方法对提高局域网环境下的系统安全性很有效。特别

是对工作场地较为封闭、终端较为固定,对安全性要求较高的专用环境尤为适用,象银行、金融等部门。因为使用这种方法后,即使入侵者探得用户的标识和口令,由于他使用的终端不在用户标识所能使用的范围之内,所以也就无法侵入系统,从而提高了系统的安全性。

笔者的主要思想的实施步骤为:

1. 设计一个安全级别检查表文件。表中规定了所有已知的用户标识分别能使用的终端(给出终端号)。

2. 设计一个程序,用来完成安全检查表的检查功能。每次用户开机注册时,取得用户的注册标识及所注册终端的终端号,然后执行检查程序,与安全检查表中的内容对照,符合条件的返回值为0,否则直接在程序中杀死其进程,使其强行退出,并且在日志文件中记录非法用户的注册时间,注册失败提示,注册标识,注册终端的终端号以及注册的主机名称,以便系统管理员随时探知非法用户的情况。

3. 在系统的.profile文件最后增加下面几条语句:

```
/usr/etc/check
```

假设检查程序为check.c,放在子目录/usr/etc下,执行格式为:

```
check
```

因为每个用户一开机,系统便自动执行.profile文件来设置系统的运行环境,所以把检查程序,嵌入.profile格式中,既可在非法用户进入系统之前将其拒之门外,又使它无法探得自己被拒绝的原因。

三、具体实现方法

以上是笔者实现的安全方法的主要思想,它的具体实现包括两方面的内容:

1. 安全检查表文件的设计

安全检查表文件内容的设计是我们整个安全方法的关键。在设计的安全检查表文件中,有两种关键变量:一是用户注册标识,二是注册终端的终端号。如何制定安全检查表,一般有以下三种方案:

(1)以用户注册标识为主。即在表中先列出每个用户注册标识,随后列出它们分别所能使用的终端的终端号。这样做较符合人的思维习惯。

(2)以注册终端的终端号为主。即在表中先列出终端号,再列出能使用该终端的用户注册标识符。

(3)第三种方法是把前两种方法结合起来,一般情况以用户注册标识为主,而对于象console(主控

制台)及远程终端,则采用以注册终端的终端号为主的方法。这种方法可灵活处理不同情况,经过比较,笔者决定选用此方法。

当然,不同系统的系统管理员可根据自己所处的开发环境,灵活设计安全检查表的内容,核心是建立起用户注册标识与终端号之间的联系。

下面结合笔者的系统,给出安全检查表的一种设计:

```
[user1 user2 user3 (tty1 tty2 &.[user4 (tty5 &.[user5 user6 user7 user8 (tty8 tty9 &.{console tty10 (* &.[user9 user10 (*  
=  

```

该表较好地体现了我们第三种方案的思想。

(1)以“[”号开头的检查条件是以用户标识为主,后跟“(”作为分隔符,限制了前面那些用户所能注册的终端号。“&.”作为各个不同检查条件的分隔符。也就是[user1 user2 user3 (tty1 tty2 &表示用户标识为user1,user2,user3的用户只能在终端号为tty1和tty2的终端上注册。

(2)以“{”号开头的检查条件是以用户注册的终端号为主,后跟“(”作为分隔符,限制了前面那些终端上所能注册的用户标识。“&.”作为各个不同检查条件的分隔符。也就是{console tty10 (* &表示终端号为console和tty10的终端能注册所有用户。

(3)以“=”号表示文件的结束。

以上检查表中使用的各种分隔符是为了便于处理安全检查表的内容。因为安全检查表是整个安全方法的核心,所以,为了提高它的保密性,还可在表中采用一些省略表示方法。象“*”号表示任意的用户注册标识或任意终端号。

另外,在安全检查表的设计中,分隔符的选择一定要注意尽量避免使用UNIX系统本身的专用控制字符,以免程序控制中发生一些莫名其妙的错误。

2. 检查程序的设计实现

设计好检查表后,可用C语言编写安全检查表的检查程序。这里就不举例了。

四、结束语

以上是笔者在实际工作中探索的一种局域网系统安全策略。笔者系统的环境是:主机为Motorola8640,内装UNIX System V。主机所在地与工作地构成Ethernet局域网。由于各个用户具有的软、硬件环境各异,这里只提供一思路,希望会对同行有所启迪。

SKILL

应用 C 语言制作软盘密钥的小技巧

□徐 鹏

随

着计算机应用技术的发展,软件产品的不断丰富,软件开发者为了保护自己开发出来的软件的潜在市场,维护自己的利益,开始研究软件的加密技术。目前加密技术发展较快,应运而生出许多软盘加密和软件加密的方法和技巧。这些方法不外乎对软盘加密、只对软件加密和制做软件狗等。这些方法只有那些对计算机软、硬件相当精通的操作人员和编程人员才能实现,对一般的工程应用软件的开发人员就难以实现了。因此,人们常常希望能有简便易行的方法对软盘和软件进行加密,使得非法用户无法拷贝、查看和运行被加密的程序。这里笔者介绍一种应用 C 语言制作软盘密钥的小技巧,该技巧是将软盘加密和软件加密结合起来,克服了单一加密方式容易被人解密的缺陷,即使解密者知道了加密方式,若不知道软盘和软件的密码也很难解密,既要知道软盘的密码,又要知道软件的密码,这种双重解密的难度比单一的解密难度大得多,下面将这种软盘密钥的制作技巧介绍给大家。

一、设置软盘密码

大家知道,格式化过的软盘被划分为许多磁道和扇区,所有扇区被分为四个部分:引导扇区、文件分配表、文件目录表、数据区,各部分的起始位置是固定的,一种加密处理方式是通过软件改写某一部分的信息,使软盘正常使用功能失效。当改过来后就恢复其正常使用的功能,这样就达到了磁盘加密的效果。另一种加密方式就是采用绝对的磁盘读写的方式,在软盘的某一扇区、某一磁道上写

密码,这种用绝对磁盘读写的方式所写的密码忽略磁盘的逻辑结构,并不关心文件、FAT 和目录,既不能用 dir 显示、又不会被 copy 到另一张软盘,只有通过绝对磁盘读的方式,才能找到密码。这种方式破译密码难度较大,如果不事先知道密码写的位置,所读出来的一串密码是很难分辨出其中哪一串是你所要知道的密码,从而达到了软盘加密的目的。下面是应用 Borland C++ 的功能函数 abswrite() 函数建立的设置软盘密码的 C 程序:

源程序清单如下:

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
int main()
{
    int sector;
    int drive,nsets,
    char buf[512];
    printf("Please Insert Your Key Into Drive A and Press any
        Key\n");
    getch();
    printf("Please Input Disk cod:");/* 输入磁盘密码 */
    gets(buf);
    drive=1; /* 要写的磁盘驱动器号 0=A,1=B */
    nsets=1; /* 要写的扇区数 */
    sector=0; /* 要写的起始扇区号 */
    if(abswrite(drive,nsets,sector,&buf)!=0)
    {
```

最少的投资
最佳的方案
最多的组合

挂接式打印机共享器

北京向宇计算机公司

地址:北京复兴路甲 20 号 27 分号 312,314

邮编:100036

电话:8263441,2063366 呼 1511


```

PerrorC"Disk Problem";
exit(1);}
printf("Disk Cod is %s ",buf);
printf("Disk cod writed Ok\n");
return(0);
}

```

二、软件加密

这里对软件加密采用一种比较简单的方法,其原理是用二进制方式打开文件,对所读出的文件内容与密钥进行一次逻辑异或运算,然后再写入文件,从而达到加密软件的目的。密钥可以是字符或字符串、数字、汉字等任意一种方式的组合,长度是命令行可接受的长度,解密时,重复加密时的过程,即可解密。

源程序清单如下:

```

#include <stdio.h>
int KeyCod(int FileNum,char * FileName[]);
main()
{
    char * FileName[];
    char secret[20];
    printf("Please Input Filename:");
    gets(FileName[0]);
    printf("Please Input secret:");
    gets(FileName[1]);
    KeyCod(2,FileName);
}
/* 软件加、解密程序 */
int KeyCod(int FileNum,char * FileName[])
{
    FILE * fp1, * fp2;
    int t,i=0,j;
    char c[4];
    if(FileNum<2)exit(0);
    if(strcmp(FileName[0],FileName[2])==0)strcpy(c,"r+b");
    else strcpy(c,"w+b");
    if((fp1=fopen(FileName[0],"r+n"))==NULL)exit(0);
    if(FileNum>=3)fp2=fopen(FileName[2],c);
    else fp2=fopen(FileName[0],"r+b");
    j=strlen(FileName[1]);
    t=getc(fp1);
    while(t!=EOF)
    {
        t^= FileName[1][i];/* 密码与文件进行逻辑异或运算 */
        putc(t,fp2);
        i++;
        if(i==j)i=0;
        t=getc(fp1);
    }
    return(0);
}

```

三、密钥制作

当设置完软盘密码及对软件进行完加密处理后,就可以建立密钥程序,密钥程序是被加密后的软

件控制驱动程序。用户只有执行密钥程序才能执行软件,非法用户没有密钥程序和已设置密码的软盘是无法执行软件的。密钥程序设计的原理及步骤为:检查磁盘是否在规定的驱动器中、判断软盘密码是否为软盘的加密密码、对软件进行解密、执行软件、重新对软件进行加密处理。

源程序清单如下:

```

/* 制作的软件密钥程序 */
#include <stdio.h>
#include <conio.h>
#include <process.h>
#include <dos.h>
#include <string.h>
int KeyCod(int FileNum,char * FileName[]);
char * KeyNum[]={"clock.exe","d123"};
/* clock.exe 软件名,d123 软件密码 */
int main()
{
    int drive,nsets,sector;
    char buf[512];
    printf("Please Insert Your Key Into Drive B\n");
    printf("Press any Key,Continue. ....\n");
    getch();
    drive=1; /* 要读的磁盘驱动器号 0=A,1=B */
    nsets=1; /* 要读的扇区数 */
    sector=0; /* 要读的起式扇区号 */
    if(absread(drive,nsets,sector,&buf)!=0) {
        perror("Disk Problem\n");
        exit(1);
    }
    printf("%s\n",buf);
    getch();
    if(strcmp(buf,"No. F0011")==0)/* 读出的磁盘密码与设置的密码比较 */
    printf("This is Key. Ok!");
    KeyCod(2,KeyNum);/* 软件解密 */
    system(KeyNum);/* 执行软件 */
    KeyCod(2,KeyNum);/* 软件重新加[必 */
}
else{
    printf("This disk is not Key!!! \n");
    printf("Press any Key,Exit\n");
    getch();
    exit(0);
}
return(0);
}

```

SKILL

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

SXD 系列打印机共享器

成功实现 FoxPro/Windows 环境下的打印技术

□刘耀东

本

文给出的结果是在实际应用 Windows 打印管理器时产生的。笔者并未对打印管理器进行特别检测。此外,除非特别指出,无论是对 2.5 版还是 2.5a 版的 FoxPro/Windows 而言,所有的结果都是一样的。

一、背景知识

DOS 保留如下设备名: AUX、COM2、COM3、COM4 串行口 (AUX 与 COM1 相同) PRN, LPT1, LPT2, LPT3 并行口 (PRN 与 LPT1 相同);

CON 控制台(屏幕和键盘) NUL 无效端口这些设备名也预留了文件名,因此,如果执行如下 DOS 命令:

```
ECHO HELLO>LPT1
```

“HELLO”一词将出现在打印机上,而且并不创建名为 LPT1 的文件。若按下面形式来重写命令,则结果是一样的。

```
ECHO HELLO>c:\mydir\lpt1.dum
```

这时并无文件在 C 盘上生成,输出将定向至打印机。FoxPro 象其他使用标准 BIOS 例程的应用一样将完成同样的工作。如下述命令:

```
LIST TO FILE e:\mydir\lpt1.dum
```

将数据直接输送到并行口。发出如下命令会得到同样的结果。

```
SET PRINTER TO e:\mydir\lpt1.dum
LIST TO PRINTER
```

本文所包括的实例使用了某些 Report/Label, 名字如下: DOSDEF, 只有 DOS 对象所包含的 Report/Label 文件。WINDEF 包含 Windows 对象的 Report/Label 文件。

DOS 对象也可包含在该文件中,换言之,Report 或 Label 文件可经由 TRANSPORT 程序实现转换。

必须指出的是:所谓“PDSETUP 处于激活状态”是指由 FoxPro/DOS 下的 GENPD-APP 所产生的打印机驱动程序设置。有几种激活 PDSETUP 的方法:

1. 经由 FoxPro/DOS 文件的打印机设置选择菜单来选择 PDSETUP。

2. 键入 SET PDSETUP TO <something> 这里 <something> 为非空或非零值。

3. 键入 _PDSETUP = <something> 这里 <something> 为非空或非零值。

4. 系统配置文件 CONFIG.FP 包 PDSETUP = <something> 这里 <something> 为非空、非零、非“-”值。

5. 启动 FoxPro 时, CONFIG.FP 文件中并不包含 PDSETUP 设置行,但在从前的 FoxPro 会话中设置了某一个 PDSETUP 做为缺省值。

6. 在设计 FoxPro 下的 Report 或 Label 输出时,发出了带 PDSETUP 子句的 REPORT 或 LABEL 命令且检查了 “[] Printer Driver Setup” 检查框,并选择了某一 PDSETUP 设置。

二、输出命令

下述命令提供了指向文件或打印机的子句:

```
LIST...
DISPLAY...
DIR...
```

TYPE...
SELECT...

LIST TO FILE <destination>

FoxPro/Windows 向<destination>按字符方式进行打印。若激活了 PDSETUP 设置,则打印有效并格式化输出结果。如:

```
* The following outputs directly to the parallel port.
LIST TO FILE LPT1
* So does the following command
LIST TO FILE c:\subdir\lpt1.dum
* The following outputs to the designated file name.
LIST TO FILE e:\test\myfile.txt
```

在 Foxpro/DOS 下结果正常。然而,在 FoxPro/Windows 下使用下述命令可在打印机上按字符模式快速输出结果。

```
LIST TO FILE LPT1.
```

SET PRINTER TO<destination>

LIST TO PRINTER

或

SET PRINTER TO <destination>

SET PRINTER ON

LIST

取决于 PDSETUP 是否处于激活状态。若已激活,则 FoxPro/Windows 通过使用 PDSETUP 设置,按照字符模式将打印结果输出到<destination>。然而,若未激活 PDSETUP 设置,则结果取决于<destination>。若<destination>以字符“PRN”或“LPT”开始,则 FoxPro/Windows 忽略 destination 选项,通过使用 SET PRINTER FONT,将为当前活动的 Windows 打印机驱动设备定义的端口进行打印输出。反之,则以字符模式向<destination>进行打印。下面的伪代码将说明得更为清楚:

```
IF PDSETUP Active
    Print to <destination> in character mode
    using the PDSETUP
ELSE
    IF <destination> starts with "PRN" or "LPT"
        ignore the destination and instead print to
        the port defined for the currently active
        windows printer driver with the output being
        printed in the current SET PRINTER FONT
```

(In other words, the output is graphically created as a rule)

```
ELSE
    print to <destination> in character mode
ENDIF
ENDIF
```

如:

```
* The following will output to the LPT2 port
* in character mode using the PDSETUP
* The following will output to the LPT2 port
```

```
* in character mode using the PDSETUP.
__PDSETUP="My LaserJet Setup"
SET PRINTER TO LPT2
LIST TO PRINTER
* With the PDSETUP still active, the following
* will print to the file name in character
* mode using the PDSETUP.
SET PRINTER TO C:\mydir\lpttest.dum
LIST TO PRINTER
* With no PDSETUP active, the following will
* ignore the LPT2 port and instead print to the
* port defined for the current windows printer
* driver using the current SET PRINTER FONT
__PDSETUP="" && Turn off PDSETUP
SET PRINTER TO LPT2
LIST TO PRINTER
*
* Again, with no PDSETUP active, the following
* will ignore the SET PRINTER TO file name and
* instead print to the port defined for the
* current windows printer driver using the
* current SET PRINTER FONT
SET PRINTER TO c:\mydir\lpttest.dum
LIST TO PRINTER
* The following (still with no PDSETUP active)
* will print to the file, since the file name
* doesn't start with "LPT."
SET PRINTER TO c:\mydir\test.txt
LIST TO PRINTER
```

若未激活 PDSETUP 设置,则不可能将打印输出送往以“PRN”或“LPT”开头的文件中。通过使用 To FILE 子句,可以克服这一障碍。FoxPro/DOS 的表现则不一样。若命名用了 SET PRINTER TO PRN (或 LPT1、LPT2、LPT3) 命令,则输出指向并行端口,而不管文件名是否以“PRN”、“LPT”开头都当做合法的文件名,并将输出结果定向至这一文件。

REPORT FORM dosdef

LABEL FORM dosdef

REPORT FORM dosdef 和 LABEL FORM dosdef 命令全按照与输出命令 (LIST, DISPLAY, 等等) 相同的方式运行。这是由于 Report/Label 的定义仅包含 DOS 对象的缘故。如果 Report/Label 并无为其内部定义的 PDSETUP, 且用户发出了带 PDSETUP 子句的 REPORT/LABEL 命令, 则 FoxPro 关闭

FBASE —— DOS 下的最佳
多媒体数据库平台

新未来电子技术公司

电话: 2573355 - 271, 272

地址: 清华东门清华园宾馆一层

任何当前活动的 PDSETUP, 然后 FoxPro/Windows 将利用当前活动的 Windows 打印机驱动程序打印输出结果。

REPORT FORM windef

LABEL FORM windef

REPORT FORM windef 和 LABEL FORM windef 命令按照包含 Windows 对象的 Report/Label 定义运行。笔者将以 REPORT 命令做为例子, LABEL 将按照同样的方式执行。

REPORT FORM windef TO FILE

<destination>

在 FoxPro/Windows 上用当前激活的 Windows 打印机驱动程序将打印结果输出至 <destination>, 打印报表格式文件指明的报告及其全部对象(换言之, 按图形方式输出)。如果包括 PDSETUP 子句, 则忽略之。如果发出命令时, PDSETUP 已被激活, 则为支持 Windows 打印驱动程序, 将忽略 PDSETUP 设置, 例如:

* The following outputs directly to the parallel port.

REPORT FORM windef TO FILE LPT1

* The following outputs to the designated file.

* The PDSETUP clause is ignored.

REPORT FORM windef TO FILE c:\test\myfile.txt PDSETUP

SET PRINTER TO <destination>

REPORT FORM windef TO PRINTER

或

SET PRINTER TO <destination>

SET PRINT ON

REPORT FORM windef

除非忽略 <destination>, 否则将按照 SET PRINTER TO FILE 相同的方式执行。FoxPro/Windows 假定如果欲将打印输出送往打印机, 则使用由当前激活的 Windows 打印驱动程序所定义的打印机, 并且不理睬 SET PRINTER TO 命令中定义的 <destination> 选项, 例如:

* The following ignores the LPT2 port and prints

* to destination defined by the current windows

* printer driver

SET PRINTER TO LPT2

REPORT FORM windef TO PRINTER

* The following ignores the SET PRINTER TO

* file name and instead prints to the

* destination defined by the current Windows

* printer driver The active PDSETUP is ignored

SET PRINTER TO c:\mydir\reportfile.txt

ET PDSETUP TO "My LaserJet Setup"

REPORT FORM windef TO PRINTER

若想打印包含 Windows 对象的 Report/Label 定义, 则不能通过 SET PRINTER TO 命令指定输

出结果的接收站。可通过使用 TO FILE 子句来克服这一缺点, 或者通过定义全部合适的、针对不同端口的、满足打印输出需要的 Windows 打印机驱动程序, 以便在打印前可以选择所需要的一个驱动程序。

SET PRINTER ON ...? /??

SET DEVICE TO PRINTER ...@SAY

SET PRINTER ON 和 SET DEVICE TO PRINTER 命令的作用方式和前面已论及的 LIST TO PRINTER 命令一样, 除非已经激活 PDSETUP 设置。

IF PDSETUP Active

IF this is the very first bit of output

IF printing with @SAY

IF SET PRINTER is ON

Print to <destination> in character mode

ELSE

Ignore the destination and instead

Print to the port defined for the

currently active Windows printer driver

with the output being printed in the

current SET PRINTER FONT

(In other words, the output is graphically created as a rule)

ENDIF

ELSE

Continue printing in the mode designated

by the first bit of output

If printing with ? /?? and printing in

character mode, then the PDSETUP is used

ENDIF

ELSE

IF <destination> starts with "PRN" or "LPT"

Ignore the destination and instead

print to the port defined for the

currently active Windows printer driver

with the output being printed in the

current SET PRINTER FONT

(In other words, the output is graphically created as a rule)

ELSE

Print to <destination> in character mode

ENDIF

ENDIF

在发出 SET PRINTER TO 命令关闭打印机工作之前, 用户不会看到任何输出结果。应当注意的

FBASE —— DOS 下的最佳

多媒体数据库平台

新未来电子技术公司

电话: 2573355 — 271, 272

地址: 清华东门清华园宾馆一层

是,如果 PDSETUP 已经激活,@SAY 命令并不执行任何与其有关的函数(FoxPro/DOS 下情况也是如此)。

? /?? 命令将执行_PDRIVER 和 PDOBJECTO 函数,该函数带有做为参数传递的 STYLE 子句。PDADVPRPT()也可以被? /?? 命令调用(并且?也将使 PDLINEND()和 PDLINEST()成为可执行的。

当按照字符模式使用?? /? 和@SAY 打印时,所有? /?? 命令的输出结果似乎汇集在一个存储区,而@SAY 的输出结果似乎保留在另一个存储区域。当发出 SET PRINTER TO 来关闭打印机工作时,? /?? 存储区域的输出结果得以释放,随后是释放@SAY 存储区域的输出结果。

尽管存在如上情况,? /?? 命令仍将正确更新 PROW()和 PCOL()的值。

考虑如下情况:

```
SET PRINTER TO myfile.txt
SET PRINTER ON
SET DEVICE TO PRINTER
?
?? "This is on row " + LTRIM(STR(PROW()))
@ 3,3 say "Row 3 Column 3"
?? "I'm on row " + LTRIM(STR(PROW()))
?
?? "Now I'm on row " + LTRIM(STR(PROW()))
@ 6,10 say "Row 6 Column 10"
?? " This is on row " + LTRIM(STR(PROW()))
?
?? "Now I'm on row " + LTRIM(STR(PROW()))
@ 8,12 say "Row 8 Column 12"
SET DEVICE TO SCREEN
SET PRINTER OFF
SET PRINTER TO
RETURN
```

在 FoxPro/DOS 下,程序将给出如下输出结果:

Actual Row	
0	
1	This is on row 1
2	
3	Row 3 Column 3 I'm on row 3
4	Now I'm on row 4
5	
6	Row 6 Column 10 This is on row 6
7	Now I'm on row 7
8	Row 8 Column 12

在 FoxPro/Windows 2.5 下运行同样的程序,将输出如下杂乱无章的数据:

Actual Row	
0	
1	This is on row 1 I'm on row 3
2	Now I'm on row 4 This is on row 6
3	Now I'm on row 7

4	
5	Row 3 Column 3
6	
7	Row 6 Column 10
8	Row 8 Column 12

FoxPro/Windows V 2.5a 有所不同,但仍不正确:

Actual Row	
0	
1	This is on row 1 I'm on row 3
2	
3	Row 3 Column 3
4	Now I'm on row 4 This is on row 6
5	
6	Row 6 Column 10
7	Now I'm on row 7
8	Row 8 Column 12

实际上,可以通过单独使用@SAY 语句打印来解决这一问题。只要将程序中“?”均用“@PROW() + 1,0SAY 来替换,“??”用“PROW(),PCOL() SAY”来替换即可。

在按字符模式打印时勿将? /?? 和@SAY 混合使用! 若未激活 PDSETUP,则不能将打印输出到文件名以 PRN 或 LPT 开头文件中,或直接输出到并口。

???

??? 命令引入到 FoxPro 语言中,是为了使用户直接将输出的结果送往打印机,而不用加入 PROW()或 PCOL()。事实上,用户不必使用 SET PRINT 将打印机置为 ON。

通常,该命令关闭当前的打印工作,然后开始自己的打印工作,按字符模式将输出结果指向 SET PRINTER TO(destination)。然而,如果你正在向文件中输出,如果先于??? 之前曾经执行过任一@SAY 命令,则??? 的输出结果和所有随后的? /?? /??? 命令的输出结果都将消失。例如,运行如下称为 TESTQM3.PRG 的程序如下:

```
* TESTQM3.PRG
PARAMETERS the __pdsetup,the __dest,atsay __flag
SET PDSETUP TO the __pdsetup
SET PRINTER TO &the __dest
SET PRINTER FONT "Arial",30
SETPRINTER ON
IF atsay __flag
    SET DEVICE TO PRINTER
    @ 1,0 SAY "Here is an @SAY in row 1"
ENDIF
? "Test Line 1"
??? "Now printing a triple"
```



```
WAIT WINDOW "Waiting 5 seconds . . ." TIMEOUT 5
? "Test Line 2"
??? "Another triple"
? "test Line 3"
?
IF atsay __ flag
    @6.0 say "Another @SAY in row 6"
    SET DEVICE TO SCREEN
ENDIF
SET PRINTER OFF
SET PRINTER TO
RETURN
```

例如：

```
DO TESTQM3 WITH "","LPT1" ,.F.
```

将输出：

```
Test Line 1 (Printed in 30-point Arial)
*** PAGE BREAK ***
Now Printing a triple (Printed in raw character mode)
Test Line 2Another triple
Test Line 3
```

```
DO TESTQM3 WITH "Epson","LPT1" ,.F.
```

将按字符模式输出：

```
Test Line 1Now Printing a triple
Test Line 2Another triple
Test Line 3
```

```
DO TESTQM3 WITH "","MYFILE1.TXT" ,.F.
DO TESTQM3 WITH "Epson","MYFILE1.TXT" ,.F.
```

均将按照字符模式输出：

```
Another triple
Test Line 3
```

```
DO TESTQM3 with "","MYFILE2.TXT ADDI" ,.F.
DO TESTQM3 with "Epson","MYFILE2.TXT ADDI" ,.F.
```

也都将按照字符模式输出：

```
Test Line 1Now printing a triple
Test Line 2Another triple
Test Line 3
```

现在让我们引入@SAY 命令，执行：

```
DO TESTQM3 WITH "","LPT1" ,.T.
```

将输出：

```
Here is an @SAY in row 1 (Printed in 30-point Arial)
Test Line 1
*** PAGE BREAK ***
Now Printing a triple (Printed in raw character mode)
Test line 2Another triple
Test Line 3
Another @SAY in row 6
```

```
DO TESTQM3 WITH "Epson","LPT" ,.T.
```

按字符模式输出：

```
Test Line 1Now printing a triple
Test Line 2Another triple
Test Line 3
Here is an @SAY in row 1
Another @SAY in row 6
```

```
DO TESTQM3 with "","MYFILE3.TXT" ,.T.
DO TESTQM3 with "Epson","MYFILE3.TXT" ,.T.
DO TESTQM3 with "","MYFILE4.TXT ADDI" ,.T.
DO TESTQM3 with "Epson","MYFILE4.TXT
ADDI" ,.T.
```

在 FoxPro/Windows 2.5 中，将按字符模式输出：

```
Test Line 1
Here is an @SAY in row 1
```

```
ANother @SAY in row 6
```

在 FoxPro/Windows 2.5 a: 中将按字符模式输出：

```
Here is an @SAY in row 1
Test Line 1
```

```
Another @SAY in row 6
```

唯一前后一致，工作正常的情形是使用 PDSETUP 的与? /?? 和@SAY 一起使用的??? 命令，在向端口（而不是文件）打印时工作无异常情况。然而，由于使用了 PDSETUP，打印属于字符模式，? /?? 和@SAY 在相互关联时并未正确打印在按 ADDITIVE 模式向文件打印输出时，唯独和? /?? 命令一起工作的??? 命令工作正确。当用 PDSETUP 向端口输出时，FoxPro/Windows 将??? 命令仅做为 DOS 特征来处理。

换页命令 EJECT

北京市威特电子技术公司

威特家用电脑系列
教育系列软件

电话：2558868, 2577999

预置 `_PADVANCE="FORMFEED"`, EJECT 命令基本上完成一个 `? CHR(12)` 操作。在 Windows 打印驱动程序下运行时, EJECT 一切工作正常。然而, 在你激活 DOS PDSETUP 时只是在关闭打印工作前执行换页命令 EJECT, 否则该命令不起作用。实际上, 它似乎在某一内部缓冲区中, 在开始一新的打印工作时, 方开始输出结果, 如:

```
_PADVANCE="FORMFEED"
SET PDSETUP TO "Epson-10cpi"
SET PRINTER TO MYFILE1.TXT
EJECT
SET PRINTER TO
```

执行完上述指令后, MYFILE1.TXT 为空, 再如:

```
SET PRINTER TO MYFILE2.TXT
SET PRINTER ON
?
SET PRINTER OFF
SET PRINTER TO
```

MYFILE 2.TXT 包含一个正常进程, 跟着回车和由 `? 命令` 产生的换行。若需要进行换页, 比较好的方法是执行如下命令:

```
?? " "
```

紧随在 EJECT 之后加入, 这会强迫字符进行输出。既然换页执行一个 `? CHR(12)`, 由于混用 `? /??` 和 `@SAY` 命令会产生问题若想按字符模式打印, 则不应将 EJECT 命令和 `@SAY` 命令混用。首先打印所有的字符, 然后是全部 `@SAY` 语句。在执行 `@SAY` 输出时, 建议用 `@0,0` 来代替 EJECT。

若 `_PADVANCE="LINEFEEDS"`, 则若想按字符方式打印一切均将化为泡影。有时在一行内执行 10 次如下命令, 会得到 10 种不同的文件内容。

```
_PADVANCE="LINEFEEDS"
SET PDSETUP TO ""
SET PRINTER TO MYFILE.TXT
EJECT
SET PRINTER TO
```

有时打印工作关闭的很快, 以便所有的换行操作进入文件中。

三、结 论

若想在 FoxPro/Windows 下按字符方式进行可靠的打印请遵守如下规则:

1. 使用 PDSETUP, 若不使用 PDSETUP。
2. 创建和编辑 FoxPro/DOS 下的全部报告和标签文件, 且不要对其进行转换。当然, 若想让你的报告按图形方式而不是字符方式打印, 则无论如何要在 FoxPro/Windows 下创建和编辑报告和标签文件

并且使用传送器。

3. 在输出结果时, 请勿混合使用 `? /??` 和 `@SAY` 命令, 打印工作中, 要么全用 `? /??` 或全用 `@SAY` 命令。

4. 就换页操作 EJECT 而言, 要紧跟着发出 `?? " "` 以确保输出字符。如果正在用 `@SAY` 命令进行输出, 则执行一个 `@0,0`, 而不是一个换页命令 EJECT。

5. 若想使用 `???` 命令, 则只要连同 `? /??` 命令一起发出 `???` 命令, 在同样的打印工作中勿用 `@SAY` 命令。

6. 如何 `???` 命令, 在 SET PRINTER TO destination 命令, 确保使用关键字 ADDITIVE

7. 若想用 `? /??` 或 `@SAY` 命令将数据输出到打印机上, 一定要发出 SET PRINTER TO 命令, 以便 FoxPro/Windows 关闭打印工作, 并在物理上开始在打印机上或文件上进行输出。

总而言之, 使用 PDSETUP 设置, 将解决大部分字符为基础的打印问题。

(本文英文资料由 Microsoft 提供) **SKILL**

《FoxPro 编程经验与技巧》

由本刊编辑出版的《FoxPro 编程经验与技巧》专集已经出版。该专集整理了由 Microsoft 公司提供的“FoxPro 中的数据加密”、“编写可重用代码”、“组合屏幕以提高开发速度”等 16 篇有关 FoxPro 的编程技术和技巧的文章, 该专集内容新、技术先进、实用性强, 可供我国广大 FoxPro 编程人员参考、借鉴, 甚至专集中的程序, 用户可以拿来即用。

该专集 16 开 94 页, 售价 9.00 元, 欲购买此专集的读者请邮局汇款(另加 10% 邮费)至 (100006) 北京市劳动人民文化宫内《电脑编程技巧与维护》杂志社发行部 孙如萍收, 联系电话: 5123823。

北京市威特电子技术公司

**威特家用电脑系列
教育系列软件**

电话: 2558868, 2577999

提高 FoxPro 多平台应用的速度

□ 常 久

F

oxPro 的最大的优点之一是它提供了适应于多种平台的版本,如 DOS、Windows、Macintosh 和 UNIX。通常在 FoxPro 的一种平台上建立的应用也能应用于 FoxPro 的其他平台之上。FoxPro 提供了一个 Transporter 使得 FoxPro 不同平台的应用可以跨越。有了 FoxPro 的 Transporter,当用户改变一个平台的任何一个对象的代码段时,用户必须打开其他平台上的工程(Project)文件(而后打开屏幕文件),以使 Transporter 修改这个对象的代码段。因此,要修改一个在 FoxPro for windows 中开发的屏幕程序使之转换为 FoxPro for Macintosh 中的程序,则必须将这个程序移入一个 Macintosh 系统中,而后用 FoxPro for Macintosh 的 Transporter 转换即可。但如果有一个外部 Transporter 用以传输被修改的代码段,则代码的维护就更加简单。

一、关于工程表(Project Table)

应用程序传输的关键是 FoxPro 存储应用工程文件及表中的对象定义。用 USE 命令打开一个工程表(.PJX 文件)。FoxPro 用工程名和一个扩展名为 .DBF 的文件去 USE 一个文件,而后用 BROWSE 命令浏览这个表。对于屏幕上 Project Builder 列表中的每一项,在工程表中都有一个记录,菜单在 TYPE 字段中用一个单字母 M 代表;库用 L 代表;程序用 P 代表;数据库用 D 代表;索引用 I 代表。屏幕文件有两种记录:在 TYPE 字段中一个是 s 一个是 S。大写字母表示屏幕组的编译的 .SPR 文件,小写字母表示单个

屏幕表 .SCX 文件。用 BROWSE FOR "s" \$TYPE 可以识别及定位工程文件中的所有屏幕文件。每个记录中的 NAME 字段包含了相对于工程表位置的一个屏幕文件的位置信息。

二、关于屏幕表(screen Table)

选择另一个工作域,USE NAME 字段中的屏幕文件(.SCX 文件),用 BROWSE 命令浏览这个表,关键字段是:PLATFORM、UNIQUEID、OBJTYPE 和 TIMESTAMP:

PLATFORM 包括: DOS、Windows、Mac、UNIX。

UNIQUEID 是跨平台对象的唯一 ID,这简化了识别不同平台上同一对象的搜索策略。

OBJTYPE 用以识别对象的类型,如 5 是文本对象,11 是一个列表,12 是一个按钮(Push button)。

TIMESTAMP 是时间、日期的数字表示。

对于屏幕表中 Memo 字段代码的维护,重要的是 COMMENT、PICTURE、WHEN、VALID、ERROR、MESSAGE、SHOW、ACTIVATE、DEACTIVATE。Numeric 字段 WHENTYPE、VALIDTYPE、ERRORTYPE、MESSAGE、SHOWTYPE、ACTIVTYPE 及 DEACTTYPE 和备注(Memo)字段中的一致。FoxPro 用上述语句决定匹配备注字段的信息是一个过程还是一个表达式。在 OBJTYPE 15 的情况下,备注字段 NAME 及 EXPR 显示记录字段的类型。NAME 字段中的一个输入项指出是一个 GET 语句,EXPR 字段中的一个输入项指出是一个 SAY 语

句。

三、外部 Transporter—OT

这里给出的 OT 只传输对象自身, 一个功能强大的外部传输器能包括分别包含子备注字段 SETUPCODE 及 PROCEDURE 中的 Setup 和 Cleanup 代码段。因为每一个传输字段都有自己的特性, 用户最终想打开其他平台上的应用, 并用 Screen Builder 来管理对象。这种系统不适合于将新的平台特有的功能加入用户的应用程序中。对此, 用户应采用传统的方法打开工程文件并引用正规的 Transporter, 这样可以控制对对象的创建、确定大小定位。假如现行平台是主要的, 所有的代码段可以。SCX 的形式从这个平台拷贝到其他平台。这可以采用最简单的过程, 应该注意的是, .PJX 和 .SCX 文件对于应用程序是至关重要的, 应和相应的备注 .PJT 和 .SCT 文件一起备份。

那么 OT 是怎样工作的呢? 首先 OT 自动识别现行平台, 而后, 要求用户有一个工程文件, OT 依次打开工程文件并处理屏幕表。本文所附程序就是一个完整的 OT。对于每一个屏幕表, OT 用别名 FRTAB 打开它, 而后形成一个名为 TOTAB 的备份。OT 为对象扫描 FRTAB, 这些对象的 PLATFORM 和现行平台一致并且 OBJTYPE 在外部传输对象的列表中。OT 将在 TOTAB 中形成适合于不同平台的记录。OT 将备注字段从 FRTAB 表(原始 .SCX)传入 TOTAB 表中。如果 OBJTYPE 是 15 指明是 SAY/GET 字段, 通过拷贝 NAME 和 EXPR 字段的内容可改变这个字段。对于单个工程管理器(屏幕文件已变化并需要重新编译), 在过程表的屏幕记录中的 TIMESTAMP 字段必须改变, 最简单的解决方法是将其置 0。用户可使用 TRANSPRT. PRG 中的函数 STAMPVAL 计算 TIMESTAMP 的现行值。OT 也改变 TOTAB 表中外部对象记录中的 TIMESTAMP 字段以匹配 FRTAB 表中的该对象的 TIMESTAMP。这能防止在屏幕文件在不同平台上被打开时而被调用。最后记录被拷贝回原始的屏幕表 FRTAB 中, 而后关闭这两个表, 并删除磁盘上的 TOTAB。

源程序清单如下:

```
* -- Initialize Variables
PRIVATE ;
m.gcPjtFile, ;
m.gcScrnFile, ;
m.gcPlatform, ;
m.gnSlashPos, ;
```

```
m.gnColonPos, ;
m.gcPath
* -- Set Up
SET TALK OFF
SET SAFETY OFF
* -- Get the platform
m.gcPlatform = ;
IIF( __DOS, "DOS", IIF( __Windows, "Windows", ;
IIF( __MAC, "MAC", "UNIX" )))
* -- get the project Filename
m.gcPjtFile = GETFILE ( 'PJX', ;
'Select a Project to Transport:', ;
'TransPort')
IF EMPTY (m.gcPjtFile)
RETURN
ENDIF
* -- Parse the string to get the Application Path
m.gnSlashPos=RAT("\", m.gcPjtFile)
IF m.gnSlashPos <> 0
m.gcPath=LEFT(m.gcPjtFile, m.gnSlashPos)
ELSE
m.gnColonPos=RAT(":", m.gcPjtFile)
IF m.gnColonPos <> 0
m.gcPath=LEFT(m.gcPjtFile, m.gnColonPos)
ENDIF
ENDIF
SET DEFAULT TO &gcPath
* -- Open the Table
CLOSE DATA
SELECT 0
USE (m.gcPjtFile)
GO TOP
* -- Look for Screen, ".SCX" TABLES
SCAN FOR "s" $ TYPE
* -- Revise the TIMESTAMP in the Project Table
* -- to signal the project manager to rebuild
REPLACE timestamp WITH 0
STORE name TO m.gcScrnFile
SELECT 0
USE (m.gcScrnFile) ALIAS frtab
* -- Create a temporary table ALIAS totab
SELECT * ;
FROM (m.gcScrnFile);
WHERE ! DELETED();
INTO TABLE totab
[ INDEX ON uniqueid TAG id
* -- Look for Objects to transport
SELECT frtab
SCAN FOR;
PLATFORM=m.gcPlatform;
AND;
INLIST ( OBJTYPE, 5, 11, 12, 13, 14, 15, 16)
* -- Look for objects with the same "uniqueid"
IF SEEK (uniqueid, "TOTAB")
SELECT totab
SCAN WHILE uniqueid=frtab.uniqueid
IF PLATFORM # m.gcPlatform
* -- Transport the code
REPLACE comment WITH frtab.comment
REPLACE picture WITH frtab.picture
```

(下转第 66 页)

FoxPro 2.6 命令及函数

□常 久

F

oxPro 2.6 在 FoxPro2.5 版本的基础上,对一些命令和函数的功能都有所增强,同时还增加了一些新的函数和命令。本文就几个有代表性的函数和命令作一介绍。

1. OBJVAR()函数

OBJVAR()函数是 FoxPro __CUROBJ 系统内存变量的一个很好的补充。OBJVAR()能告知用户程序执行到了何处, __CUROBJ 告知 FoxPro 如何执行。在默认调用的情况下(没有参数),OBJVAR()简单地返回一个 READ 语句中现行活动的 GET 的名字。如果没有活动的 READ,OBJVAR()函数返回一个空串。这一函数有两个有用的功能:返回一个现行 GET 变量及其名称的表的别称,即,如果编辑一个内存变量,返回值则加一“M”。如果用户编辑一个表的字段,字段名以其表的别称后跟·分隔符为前导。

OBJVAR()有两个可选的数值参数,第一个参数可使用户决定包含于现行 READ 级的对象的名称。通过下面的程序用户可以建立一个 READ 中元素的数组。

```
FUNCTION num __objs
PRIVATE pt __cur __err, pt __xx
IF RDLEVEL() > 0 && we are in a read
m.pt __cur __err = ON('error')
ON ERROR * && turn off error trap
m.pt __xx = 1
DECLARE pt __array[1]
DO WHILE .NOT. EMPTY(OBJVAR(m.pt __xx))
DECLARE pt __array [m.pt __xx]
pt __array [m.pt __xx] = DBJVAR(m.pt __xx)
&& something there
m.pt __xx = m.pt __xx + 1
ENDDO
* * * reset the active error handler
IF .NOT. EMPTY(m.pt __cur __err) && something there
```

```
ON ERROR &pt __cur __err && reset
ELSE && nothing there
ON ERROR && reset FoxPro EH
ENDIF
ELSE
m.pt __xx = 0
ENDIF
RETURN(m.pt __xx - 1)
```

函数返回 READ 中的对象数。在一个 READ 之前运行上述程序可以给出上一个 READ 的有关信息。在确认 READ 执行时要关闭错误处理,因为 OBJVAR()有一个问题,在 READ 语句激活后需要这些信息,就必须把这些代码放在 READ 的 WHEN 语句中。如果不关闭 FoxPro 的错误处理器(error handler),在执行上述程序时,将得到 FoxPro 提示的错误信息:

"Invalid function argument value, type, or count"

当 m.pt __xx 增量超出 READ 语句的最后一个对象时,循环失败。因此在编程时通过屏蔽 FoxPro 的错误处理器并使用 ON ERROR * (FoxPro 的下一版本解决了这一问题)。

第二个参数允许用户扩展一个 READ 列表到所有的活动级(READ)。用户可以创建所有活动 READ 的映象(从 1 到 5),这在非模块化编程中是很有用的。

2. SET KEY TO

如果用户有一个活动的索引,这里假设索引关键字是姓名的组合,那么可以通过 SET KEY TO 'A' 得到所有姓中,以 A 开始的人,用 SET FILTER TO lastname = "A" 速度更快。SET KEY 是有用的语句,一般而言,用户

在自己的表上设置一个基本过滤器(Primary Filter),然后作一系列过滤器(Secondary Filter),当改变 SET FILTER TO 次过滤器时,用户可检索一个 SET KEY TO 的通用过滤器。

3. SET LIBRARY TO

FoxPro 2.6 有一个 SET LIBRARY TO 命令。通过这一命令用户可以增加 FoxPro 代码的一个过程文件到自己的应用程序中。如果采用 ADDITIVE 子句,用户也不会丢失已经打开的 FoxPro 库。现在我们看一下 FoxPro 搜索一个文件运行的次序,当通过 DO 命令执行一个过程或作为函数调用时, FoxPro 以下述顺序搜索这一过程:

- . 现行激活的程序文件;
- . 通过 SET PROCEDURE 打开的过程文件;
- . 从最近执行的程序文件到主程序,备份全部执行堆栈;
- . 通过 SET LIBRARY 打开的过程文件;
- . 磁盘上的程序文件。

FoxPro 根据调用的先后执行其发现的第一个文件,这样用户就可以构造自己的应用,所以可以用在调用顺序中较高的新过程替代一般的代码段。在应用过程文件时,可将 SET LIBRARY TO 中的过程文件作为一般函数,而后用 SET PROCEDURE

TO 过程文件适合自己的函数来代替他们。由于 SET PROCEDURE TO 文件被首先搜索,因此,用户规定的代码可以得到执行。SET LIBRARY 在设计系统时给用户提供了很大的灵活性。

4. TAGNO()和 TAGCOUNT()

这是 FoxPro 2.6 中提供的两个有用的新函数, TAGCOUNT()返回现行选中数据表中活动的索引标志(index tag)数。没有参数时返回活动索引的总数(包括.IDX 索引文件及.CDX 索引文件)。如果提供索引文件名,函数返回该文件中的索引数。

和 TAGCOUNT()一样, TAGNO()也和索引文件有关,如果提供标志名子, TAGNO()返回这一标志的索引数,如果标志不存在,函数返回 0,通过这种方法编程人员可得知一个标志是否存在。

上面介绍了 Foxpro 2.6 新增的命令及函数。Microsoft 的 FoxPro 和 dBASE IV 是兼容的,但 dBASE IV 应用到 FoxPro 2.6 还需要程序转换,如 FoxPro 2.6 没有的功能是 SYSPROC 过程文件,在 dBASE IV 中,用户可以装入一个特殊的过程文件作为 CONFIG. DB 启动代码的一部分,这一文件对于整个对话过程是开放的,不能改变。

(本文英文资料由 Microsoft 提供) SKILL

(上接第 64 页)

```
REPLACE whentype WITH frtab. whentype
REPLACE when WITH frtab. when
REPLACE validtype WITH frtab. validtype
REPLACE valid WITH frtab. valid
REPLACE errortype WITH frtab. errortype
REPLACE error WITH frtab. error
REPLACE messtype WITH frtab. messtype
REPLACE message WITH frtab. message
REPLACE showtype WITH frtab. showtype
REPLACE show WITH frtab. show
REPLACE activtype WITH frtab. activtype
REPLACE activate WITH FRTAB. activate
REPLACE deacttype WITH FRTAB. deacttype
REPLACE deactivate WITH FRTAB. deactivate
* --Include if, Object is SAY/GET
IF OBJTYPE=15
REPLACE name WITH frtab. name
REPLACE expr WITH frtab. expr
ENDIF
* -- We don't want to call the Transporter
* -- the next time the Screen is opened
* -- under a New Platform so we'll reset
* -- the timestamp equal to this platform
REPLACE timestamp WITH frtab. timestamp
ENDIF
ENDSCAN
ENDIF
```

```
ENDSCAN
* -- Empty the original Screen file
SELECT frtab
ZAP
USE
* -- Reorganize the temporary screen file
SELECT totab
* -- VERY IMPORTANT we must return the records
* in the original order of the file because
* this is the order of GETS and SAYS.
* To change the order of the records will
* affect this order. So we return to the
* physical order of the file.
SET ORDER TO
* -- put the records back into the Screen File
COPY TO (m. gcScrnFile)
* -- Clean Up
USE IN totab &&. close totab
DELETE FILE ("TOTAB.DBF") &&. The Table
DELETE FILE ("TOTAB.FMT") &&. The Memo File
DELETE FILE ("TOTAB.CDX") &&. the Index File
ENDSCAN
CLOSE DATA
RELEASE ALL LIKE m *
SET TALK ON
SET SAFETY ON
RETURN
```

(本文英文资料由 Microsoft 提供) SKILL

应用 FoxBASE+2.10 给管理系统增加计算器功能

□王红军

在

管理系统中,经常会出现这样的情况:进行数据录入或处理数据时,需要将数据进行加减乘除运算,而又没有预备计算器或者算盘,这时我们就要去找或者用笔算,既麻烦又浪费时间,能不能需要时在当前屏幕上实现计算器功能呢(比如 WPS、Windows 系统中提供的计算器功能)?笔者就是基于这种想法用 FoxBASE+2.10 设计了一个计算器程序实现了上述功能。

程序主要是利用 2.13H 汉字系统中的特殊显示功能处理屏幕,利用 FoxBASE 提供的 INKEY()函数,实现对键盘的输入接收处理。程序再根据输入字符的 ASCII 值,判断输入的是数字还是运算功能键,再作出相应的处理。提供的有效键如下:1~9 数字键;+、-、×、÷、=、A、S、Q、T、R、W、E、C、BackSpace、End,其中“A”是寄存器加,“S”是寄存器减,“Q”是求平方,“T”是求平方根,“R”是读寄存器,“W”是写寄存器,“C”是清寄存器,“E”是清零,“BackSpace”是删除刚输入的字符,“End”结束并退出计算器功能。

调用方法是在系统主控程序的开始加一条指令:

```
ON KEY=<数值> DO<命令模块>
```

其中数值是输入的热键的 ASCII 值,并且它可以重新定义 F1 键,ASCII 值是 315,命令模块是计算器处理模块的名称,本程序为 CALC. PRG。在系统运行的任意时刻键入热键,系统就会自动调用计算器处理。

程序中的 SWIN 和 LWIN 两个子程序用以保存和恢复调用计算器前的窗口屏幕画面(由于 FoxBASE 提供的 SAVE SCRE 和

RESTORE SCRE 命令在汉字图形方式不够理想,故笔者用汇编语言编写作为外部命令调用,读者可根据资料自行编写);TS. PRG 是局部区域填色程序;再者,为简单起见,程序将数值处理成两位小数,有兴趣的读者可将其修改为符合自己要求的小数位数。

本程序在 AST 386、SUN 386、VGA 卡、2.13H 汉字系统下调试通过。

源程序清单如下:

```

** 程序名:CALCULCTE. PRG                **
** 功 能:应用 FoxBASE+2.10 设计计算器 **
SET TALK OFF
SET ESCA OFF
! SWIN 0
@1,1 SAY CHR(14)+"R0]"
SET COLO TO RG/B
DO TS WITH 10,20,22,65,"W/R+"
DO TS WITH 11,19,24,63,"W/3+"
@1,1 SAY CHR(14)+"C15D176,378B353,198]"
@1,1 SAY CHR(14)+"C4D201,351B302,146]"
@1,1 SAY CHR(14)+"C15D383,232B113,17]"
@1,1 SAY CHR(14)+"C15D207,232B113,17]"
@1,1 SAY CHR(14)+"C4D202,250Y301]"
@15,28 SAY "END 返回 BACKSPACE 7 8 9"
@16,28 SAY "SQRT + * 4 5 6"
@17,28 SAY "CE MR - / 1 2 3"
@18,28 SAY "MC MA MS MW 0 . ="
DO WHIL. T.
@12,48 SAY ""
YS="0"
H=ROW()
L=COL()
JG=0.00
MEM=0.00
ZF=""
DO WHIL. T.
SET COLO TO G+/N
@12,26 SAY MEM PICT "@Z 999,999,999.99"
@H,L SAY JG PICT "@Z 999,999,999.99"
DO WHIL. T.
KEY=INKEY()
DO CASE

```

```

CASE CHR(KEY) $"0123456789"
  ZF=ZF+CHR(KEY)
  @H,L SAY VAL(ZF) PICT "@Z 999,999,999.99"
CASE KEY=127
  IF LEN(ZF)>0
    ZF=SUBSTR(ZF,1,LEN(ZF)-1)
  ENDIF
  @H,L SAY VAL(ZF) PICT "@Z 999,999,999.99"
CASE CHR(KEY) $"+-*/"
  IF YS="0"
    JG=VAL(ZF)
    FK=KEY
    ZF=""
    YS="1"
  ELSE
    DO CASE
      CASE CHR(FK)="+"
        JG=JG+VAL(ZF)
      CASE CHR(FK)="-"
        JG=JG-VAL(ZF)
      CASE CHR(FK)="*"
        JG=JG*VAL(ZF)
      CASE CHR(FK)="/"
        JG=JG/VAL(ZF)
    ENDCASE
    @12.48 SAY JG PICT "@Z 999,999,999.99"
    IF YS="1"
      ZF=""
      FK=KEY
    ENDIF
  ENDIF
CASE CHR(KEY) ="="
  DO CASE
    CASE CHR(FK)="+"
      JG=JG+VAL(ZF)
    CASE CHR(FK)="-"
      JG=JG-VAL(ZF)
    CASE CHR(FK)="*"
      JG=JG*VAL(ZF)
    CASE CHR(FK)="/"
      JG=JG/VAL(ZF)
  ENDCASE
  @H,L SAY JG PICT "@Z 999,999,999.99"
  ZF=STR(JG)
  YS="0"

```

```

CASE CHR(KEY)&."CcEeAaSsRrWwQqTt"
  JG=VAL(ZF)
EXIT
CASE KEY=6
  ! LWIN 0
  @1,1 SAY CHR(14)+"R1"
RETU
ENDCASE
ENDDO
DO CASE
CASE CHR(KEY)&."Aa"
  JG=JG+MEM
  MEM=JG
  ZF=STR(JG)
  YS="0"
CASE CHR(KEY)&."Ss"
  JG=MEM-JG
  MEM=JG
  ZF=STR(JG)
  YS="0"
CASE CHR(KEY)&."Cc"
  MEM=0
CASE CHR(KEY)&."Ww"
  MEM=JG
  ZF=STR(JG)
  YS="0"
CASE CHR(KEY)&."Rr"
  JG=MEM
  ZF=STR(JG)
  YS="0"
CASE CHR(KEY)&."Tt"
  JG=ROUND(SQRT(JG),3)
  ZF=STR(JG)
  YS="0"
CASE CHR(KEY)&."Qq"
  JG=JG**2
  IZF=STR(JG)
  YS="0"
CASE CHR(KEY)&."Ee"
  EXIT
ENDCASE
ENDDO
ENDDO
RETU

```

SKILL

(上接第73页)

4. 在 CS:09E1 处键入保留 INT 1H 的返回地址 CS:08ED 的程序。

```

:09E1 1E PUSH DS
:09E2 31C0 XOR AX,AX
:09E4 8ED8 MOV DS,AX
:09E6 A10400 MOV AX,[0004]
:09E9 2E CS:
:09EA A32331 MOV [3123],AX
:09ED A10600 MOV AX,[0006]
:09F0 2E CS:
:09F1 A32531 MOV [3125],AX
:09F4 1F POP DS
:09F5 C3 RET

```

5. 在 CS:0138 处增加 DEBUG 自身的长度,修改 CX 寄存器的值,用 DEBUG 的 W 命令存盘退出。

如果对 DOS3.3 提供的 DEBUG.COM 进行改造,G 命令处理以及 INT3 中断服务程序的程序清单,可通过执行下面几条命令得到。

```

C:\>COPY DEBUG.COM DEBUG
C:\>DEBUG DEBUG
-G 2EF7
-T
-U1182 1261
-U1112 1169
-U12BB 131A

```

SKILL

在 WPS 中实现自动存盘 AutoSave 功能

□曹国钧

W

PS 是微机上普及率很高的中文编辑软件,但由于 WPS 未提供定时存盘功能,致使许多没有 UPS 的用户在突然停电的情况下,或者 WPS 系统内部出现意外情况下,使正在编辑的文件来不及存盘而丢失。经常使用 WPS 的用户,都不同程度地遇到这样的问题。因此,为 WPS 实现定时自动存盘 AutoSave 的功能显得特别重要。笔者编制了一个内存驻留 TSR 程序 AUTOSAVE.COM,在 WPS 中实现了定时 AutoSave 功能,以解除用户使用 WPS 时的后顾之忧。

设计思想:

在 WPS 编辑过程中,只要过一段时间键入 Ctrl+KS,就可以将所编辑的文件存盘。根据这一原理,我们修改了 INT08H 中断向量,使之指向新的 INT08H,并驻留内存。新设计的 INT08H 设置了一个减法计数器 NUMBER,初值定为 1092(根据微机内部时钟的频率 18.2 次/秒换算为 1 分钟时间),当减为 0 时 NUMBER 又恢复为原值 1092,此时就将存盘热键 Ctrl+KS 的键盘扫描码送入键盘缓冲区,在外观上就相当于用户按了 Ctrl+KS 复合键,从而实现了在 WPS 编辑过程中的定时 AutoSave 功能。Ctrl+KS 是由 Ctrl+S 组成的, Ctrl+K 的键盘扫描码为 250BH, Ctrl+S 的键盘扫描码为 1F13H,在新的 INT08H 中将这两个键盘扫描码预送入键盘缓冲区中即可。

程序实现:

笔者编制了 WPS 的自动存盘程序 AutoSave.ASM,该程序设置存盘间隔为 1 分钟,用户可将程序中两处 1092 修改为其他数

值,从而改变存盘间隔。另外,为了避免 DOS 的重入问题,程序中修改中断所采用的是初级编程。

程序 AutoSave.ASM 输入后,可按下列方法生成 AutoSave.COM 文件:

1. 编译 AutoSave.ASM

C:\WPS>MASM AutoSave;

2. 连接 AutoSave.OBJ

C:\WPS>LINK AutoSave;

3. 转化 AutoSave.EXE 为 AutoSave.COM

C:\WPS>EXE2BIN AutoSave.EXE
AutoSave.COM

在运行 WPS 之前,先运行 AutoSave 程序,该程序驻留内存后仅占用 576 个字节,于是,在 WPS 编辑中就实现了定时存盘的功能。用户只管输入文字,而不用担心是否存盘,因为 AutoSave.COM 已为您做了定时存盘工作。

此程序对所有 WPS 的版本及具有 Ctrl+KS 存盘功能的应用软件都适用。另外,本程序对(*)行以下的三行语句稍作修改,即:

```
MOV WORD PTR [BX+4],3C00H;送 F2
MOV WORD PTR [BX+2],0020H;修改键盘缓冲区
                        ;尾指针
```

就能实现具有 F2 热键存盘功能的应用软件,例如 CCDE、Turbo 系列软件,PE2(PE3) 等的定时存盘。

源程序清单如下:

```
code segment
    assume cs:code,ds:code
    org 100h
begin:    jmp start
```

(下转第 22 页)

Xenix 下与终端有关的软故障及其排除

□唐北海 石学荣

随

随着微型计算机的发展,越来越多的微型机装配了 Xenix 多用户系统。由于在多用户状态下,众多用户同时运行相同或不同的多个任务,同时抢占系统的硬件和软件资源,难免不发生冲突,这样就会因进程的死循环而造成机器故障。其中,终端机的故障是最主要的且经常发生的,象终端的各种参数的变化,终端与主机的通讯参数及相关文件的破坏,终端的设备文件及特性文件和参量数据库等变化或损坏,用户的运行程序受损或错误操作等等,都会引起终端发生显示、打印混乱,或终端挂起或锁死的现象。在排除硬件故障之后,针对引起终端故障的原因不同,给出不同的维护方法。下面将具体介绍我们在工作中常会遇到的一些故障现象,及其这些故障的排除方法。

1. 终端开机无反应

打开终端机时,不管是激活还是卸下终端,一切都正常;但是当使用一段时间后,从键盘上键入的字符得不到回应。这种情况多是由于出错、程序提前终止或用户按了 BREAK 键等误操作引起的,可按以下步骤进行处理:

1) 分别按几次 Ctrl+Q 键、Ctrl+D 键或 Delete 键,它能使一些无反应的终端恢复正常。

2) 重新启动终端:热启动,同时按下 Ctrl+Alt+Del 键;冷启动,将终端机关闭后再打开;按几次回车键,看是否能恢复正常。通过重新启动以恢复终端的特性,使因各种原因引起的终端通讯或显示参数的混乱得以恢复正常。

3) 首先按 Ctrl+J 键,不管系统显示的错误信息,继续键入 STTY SANE,由于终端不显示键入的字符,所以输入时要小心一些,确保无误,然后再按 Ctrl+J 键,在按下此键后,终端恢复正常。

2. 终端不能发送

终端在业务菜单限制下工作,终端的业务菜单可以随终端的激活而弹出,随终端的杀死而关闭,但从终端键盘上输入的字符均被提示为错误。造成如此故障现象,一般是由于在激活终端时,有误操作过程,或是对受限业务菜单的应用系统进行不正确地更改系统时间和初始化,引起系统进程混乱,从而造成多用户卡的端口产生一种“损坏”的现象。对此故障处理时,我们可将终端与主机连接的端口更换,一切就恢复正常或者将主机关闭,重新开机,这样原来不能使用的端口就得以恢复了。

3. 终端不显示业务菜单

受业务菜单限制的终端在激活时,没有反应;当卸下时,终端回显杀死的信息,更换与主机的连接端口也不能恢复正常。由于终端能显示杀死的信息,说明终端与多用户卡是完好的,且与之有关的系统文件是完好的;而终端不能显示激活的菜单,一定是应用系统的业务程序被损坏,将终端业务菜单的激活程序重新拷贝,故障得以排除。

4. 一个终端的端口不能激活

对某一端口作激活操作时,终端不能激活且显示信息如下:

```
enable ttyxx
/etc/ttys updated
enable: Can't find ttyxx
```

或者显示:

```
enable ttyxx
```

```
/etc/ttys updated
```

```
getty:cannot open "ttyxx".error:6
```

但是,对其他端口操作时,一切都是正常的。由此,我们可以判断出是由于该端口的设备文件“ttyxx”的损坏或丢失,造成上述故障的。用命令“/etc/mknod file [option] major __ device minor __ device”,重构该设备文件后,即可排除故障。

5. 所有终端的端口均不能激活

对所有端口激活时,均显示如下信息:

```
enable ttyxx
```

```
/etc/ttys updated
```

```
getty:cannot open "ttyxx" error:6
```

检查文件/etc/inittab,发现对终端端口的设置正确;用 sysadm 系统管理菜单检查发现串行多用户卡已丢失。用命令“/etc/mkdev serial”将串行卡挂上,即可恢复正常。

6. 一个终端的 GETTY 进程错误

当某个终端被激活后,若主监视器上突然显示如下的错误信息:

```
/dev/ttyxx:getty keeps dyings -- there may be a problem.
```

同时,终端死机,并且用命令“kill -9”不能将 GETTY 进程杀死,重新生成。此时,可用命令“disable ttyxx”把终端卸下后,再用命令“enable ttyxx”激活,即能恢复终端功能。

7. 所有终端的 GETTY 进程错误

若系统所有终端在激活时,都显示下面错误信息:

```
/dev/ttyxx:getty keeps dyings -- there may be a problem.
```

并且,命令“ps -e”找不到 getty 进程,则说明系统文件 getty 已经损坏。可以将自制的引导盘用命令“/etc/mount /dev/fd096ds15/mnt”挂到硬盘上,以引导盘内的 getty 文件覆盖硬盘中的 getty 文件,便能将故障排除,重新启动,系统便恢复正常。

8. 终端的特性文件损坏

从终端登录时,显示错误信息:

```
cannot open termcap file.
```

用户从终端上登录时,系统为其打开终端的特性文件/etc/termcap,该文件是终端仿真类型功能数据库,为每个打开的终端提供功能设置。从提示信息看有三种情况:一是文件丢失,二是文件属性不对,三是文件损坏。若是丢失或损坏,可用自制的引导盘

上的 termcap 文件来恢复;若是属性不对,可用命令“chmod”改正,就能使终端功能恢复正常。

9. 终端的仿真类型不符

用户登录时,显示错误信息:

```
type ct100 unknown.
```

说明/etc/termcap 与/etc/ttytype 或 profile 中,对应的仿真类型不符。若是/etc/tty.type 或 profile 不对,可用 vi 编辑修改;若是/etc/termcap 文件内不含该终端类型,则从别的机器中将含有该终端仿真类型的 termcap 文件拷过来,就可排除故障。

10. 终端的通讯速率错误

激活终端时,若循环显示下列错误信息:

```
enable ttyxx
```

```
/etc/ttys updated
```

```
getty:unable to find min "/etc/gettydefs".
```

则说明/etc/ttys 中的通讯速率代码在/etc/gettydefs 文件内找不到。维护时,若错误在 ttys 文件中,可根据终端参数值修改通讯速率代码来恢复;若错误是因 gettydefs 损坏或对应通讯项残缺引起,可从引导盘上将 gettydefs 文件拷入硬盘排除故障。

11. 终端代替主机监视器

若主机监视器因故不能打开使用,我们可用串行口上的终端代替它。具体方法是:用 vi 修改/etc/default/boot 文件,于其文件尾加入一行命令“systty = 1”,这样重新启动机器,系统原来在主机监视器上显示的信息,均传送给终端“ttyla”,该终端就起主监视器的作用。若将加入的命令改为“systty = 0”,便可恢复主机监视器的功能。

12. 终端参数的维护

终端参数一般包括综合设置、显示设置、通讯设置、键盘设置、打印设置、汉字特性以及其他设置等等,当计算机系统没有问题,而终端却发生显示与打印混乱,或键盘击键速度发生变化以及键盘失效等现象时,均应对照原来的终端参数,检查终端的参数值是否改变,将不正确的设置改正,就能恢复终端功能。

SKILL

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

增强动态调试程序 DEBUG 的跟踪能力

□巴力登

众

所周知,DEBUG 调试程序是一个功能很强的工具软件,主要用途是调试汇编程序,配备了完整的跟踪命令集,DEBUG 在提供一个可控制的测试环境的同时提供了一个可控制的跟踪环境,为监视和控制程序的运行提供了充分的支持,是对一般加密软件进行分析解密的主要工具。因此也就出现了各种针对动态调试 DEBUG 而采取的一系列反动态跟踪技术措施。

反跟踪手段总是利用和围绕动态调试程序 DEBUG 的弱点和特性进行的。因此我们采取措施改变和增强 DEBUG 跟踪程序的功能,从而能达到跟踪被加密程序的目的,下面我们介绍一种即方便、又简单的修改 DEBUG 程序的方法,改进后的方法易于实现,同样也可用于其他版本下的 DEBUG 程序。当然也可用 SR.EXE 生成 DEBUG 程序的源程序进行修改扩充,增强 DEBUG 程序的反跟踪调试能力,增加许多必需的功能程序等。

DOS 版本 2.X—5.X 提供给用户的 DEBUG 程序其大小不一,虽然 DOS 版本升高,可其中的 DEBUG 程序改进不大,以 DOS2.X 版提供给用户的 DEBUG 为例(其它版本类似),具体修改步骤为:

1. 对于 DEBUG 中的 G 命令,不使用 INT3 中断,而用 INTF3 中断来代替,将 G 命令处理程序和 T 命令处理程序分开,修改后的 G 命令新的入口地址为 CS:2F80,作为单独的程序段附在 DEBUG 程序的后面。用 A 命令输入,并在 G 命令处理程序入口地址处 CS:0374,指向 CS:2F80,另外在 CS:08E6

处键入一条件转移指令 JMP 300F,用 DEBUG 的 A 命令键入以下 G 命令处理修改程序:

```
XXXX:2F80 E816DA CALL 0999
;将 G 后面的断点地址转换为实际地址断点数
:2F83 31BD XOR BX,BX;初始为 0
:2F85 BF8F2D MOV DI,2D8F;内存区起始地址
:2F88 E83FD3 CALL 02CA;断点地址?
:2F8B 741B JN 2FA8;转
:2F8D 8B2EF82A MOV BP,[2AF8]
;被跟踪程序的代码段地址(CS)
:2F91 E807D6 CALL 059B
:2F94 8915 MOV [DI],DX;保存断点的断址
:2F96 894502 MOV[DI+02],AX
;保存断点的偏移量
:2F99 83C706 ADD DI,+06;保存下一个断点地址
:2F9C 43 INC BX;断点数加 1
:2F9D 83FB09 CMP BX,+09;断点数超过 10 个?
:2FA0 75E6 JNZ 2F88;转
:2FA2 B84250 MOV AX,5024;送 AX
:2FA5 E97DD8 JMP 0825;转移
:2FA8 891E342D MOV [2D34],BX;保存断点数
:2FAC 89D9 MOV CX,BX;BX 送 CX
:2FAE E313 JCXZ 2FC3;无断点转
:2FB0 BF8F2D MOV DI,2D8F;取存放地址
:2FB3 1E PUSH DS;入栈保存
:2FB4 26 ES;取断点的段地址;偏移量到 DS:SI
:2FB5 C535 LDS SI,[DI]
:2FB7 83C704 ADD DI,+04
:2FBA A5 MOVSW;保存断点中的内容(1 字)
:2FBB C744 FECDF3 MOV WORD PTR [SI-02],
F3CD;在断点处填入 F3CD 的代码
:2FC0 E2F2 L00P 2FB4;处理下一个断点
:2FC2 1F POP DS;恢复 DS
:2FC3 C706362D0100 MOV WORD PTR [2D36],
0001;设标记
:2FC9 8B1E5D2B MOV BX,[2B5D]
;恢复被中断程序的 PSP 断
:2FCD B450 MOV AH,50
:2FCF CD21 INT 21
:2FD1 1E PUSH DS;设置 INT F3H 的向量
:2FD2 31C0 XOR AX,AX
:2FD4 8ED8 MOV DS,AX
```

```

:2FD6 C706CC03E608 MOV WORD PTR
    [03CC],08E6
:2FDC 8C0ECE03 MOV [03CE],CS
:2FE0 FA CLI ;设置 INT 23H 向量
:2FE1 C7068C00E108 MOV WORD PTR [008C],08E1
:2FE7 8C0E8E00 MOV [008E],CS
:2FEB 1F POP DS
:2FEC BCE22A MOV SP, 2AE2
    ;SP 指向保存内存区的首地址
:2FEF 58 POP AX ;恢复各寄存器的值
:2FF0 5B POP BX
:2FF1 59 POP CX
:2FF2 5A POP DX
:2FF3 5D POP BP
:2FF4 5D POP BP
:2FF5 5E POP SI
:2FF6 5F POP DI
:2FF7 07 POP ES
:2FF8 07 POP ES
:2FF9 17 POP SS
:2FFA 8B26EA2A MOV SP,[2AEA]
    ;恢复被调试程序的栈指针
:2FFE FF36FC2A PUSH [2AFC]
    ;被调试程序的标志 FLAG 进栈
:3002 FF36F82A PUSH [2AF8] ;被调试程序的 CS 进栈
:3006 FF36FA2A PUSH [2AFA] ;被调试程序的 IP 进栈
:300A 8E1EF22A MOV DS,[2AF2]
    ;恢复被调试程序的 DS
:300E CF IRET ;弹出三次进栈值后,控制转向被跟踪程序
:300F 87E5 XCHG SP,BP
    ;从 CS:08E6H 处转入到此处执行
:3011 FF4E00 DEC WORD PTR [BP+00]
    ;将栈中保存的被调试程序的 IP 值减 2
:3014 FF4E00 DEC WORD PTR [BP+00]
:3017 87E5 XCHG SP,BP
:3019 E9D1D8 JMP 08ED
    ;进入 INT 1H 的入口地址 CS:08EDH

```

2. 对于 INT1H 来说由于是由硬件实现的,无法用其他中断替换,因此程序修改中,采取在执行 T 命令之前,先保护 INT1H 向量区中原先内容,并在跟踪执行完毕进入 INT1H 处理时恢复其原先内容。以下是 T 命令的修改程序。

```

XXXX:0863 E83301 CALL 0999 ;T 命令参数处理
:0866 E861FA CALL 02CA
:0869 E8BCFC CALL 0528
:086C BA0100 MOV DX, 0001
:086F 7206 JB 0877
:0871 B90400 MOV CX,0004
:0874 E8A5FC CALL 051C
:0877 8916362D MOV [2D36],DX
:087B E804FD CALL 0582
:087E C706342D0000 MOV WORD PTR [2D34],0000
:0884 800EFD2A01 OR BYTE PTR [2AFD],01
    ;设置单步中断陷阱
:0889 8B1E5D2B MOV BX,[2B5D]
:088D B450 MOV AH,50
:088F CD21 INT 21
:0891 1E PUSH DS ;设置中断 INT 1H 的向量
:0892 33C0 XOR AX,AX

```

```

:0894 8ED8 MOV DS,AX
:0896 C7060400ED08 MOV WORD PTR [0004],08ED
:089C 8C0E8E00 MOV [0006],CS
:08A0 FA CLI ;设置中断 INT 23H 的向量
:08A1 C7068C00E108 MOV WORD PTR [008C],08E1
:08A7 8C0E8E00 MOV [008E],CS
:08AB E83301 CALL 09E1 ;保留 INT 1H 的返回地址
:08AE 90 NOP
:08AF 90 NOP
:08B0 90 NOP
:08B1 90 NOP
:08B2 90 NOP
:08B3 90 NOP
:08B4 90 NOP
:08B5 1F POP DS
:08B6 BCE22A MOV SP,2AE2 ;设置 DEBUG 堆栈指针
:08B9 58 POP AX ;由栈中弹出用户程序的各寄存器的值
:08BA 5B POP BX
:08BB 59 POP CX
:08BC 5A POP DX
:08BD 5D POP BP
:08BE 5D POP BP
:08BF 5E POP SI
:08C0 5F POP DI
:08C1 07 POP ES
:08C2 07 POP ES
:08C3 17 POP SS
:08C4 8B26EA2A MOV SP,[2AEA] ;恢复 SP
:08C8 FF36FC2A PUSH [2AFC]
    ;被调试程序的标志 FLAG 进栈
:08CC FF36F82A PUSH[2AF8] ;被调试程序的 CS 进栈
:08D0 FF36FA2A PUSH[2AFA] ;被调试程序的 IP 进栈
:08D4 8E1EF22A MOV DS,[2AF2] ;恢复调试程序的 DS
:08D8 CF IRET ;中断返回
:08D9 E8D7F9 CALL 02B3
:08DC E8C3FD CALL 06A2
:08DF EB9D JMP 087E
:08E1 83C406 ADD SP,+06
:08E4 EB07 JMP 08ED
:08E6 E92627 JMP 300F ;转入 G 命令处理返回地址

```

3. 恢复 INT1 的返回地址,在 CS:0972 处键入一条无条件转移指令 JMP 09B1,恢复 INT 1H 的返回地址,并开放键盘中断,使封锁键盘中断的反跟踪措施失效。

```

:09B1 1E PUSH DS ;入栈保存 DS
:09B2 31C0 XOR AX,AX ;AX=0
:09B4 8ED8 MOV DS,AX ;DS=0
:09B6 ZE CS:
:09B7 A12331 MOV AX,[3123] ;恢复 INT 1H 的返回地址
:09BA A30400 MOV [0004],AX
:09BD 2E CS:
:09BE A12531 MOV AX,[3125]
:09C1 A30600 MOV [0006],AX
:09C4 E421 IN AL,21 ;开放键盘中断
:09C6 24FD AND AL,FD
:09C8 E521 OUT 21,AL
:09CA 1F POP DS ;恢复 DS
:09CB E93AF8 JMP 0208
    ;去显示 DEBUG 提示符,等待输入新的命令

```

(下转第 68 页)

多频显示器的特点及维修(一)

□柳永林

当

今人类发展到信息社会。计算机的迅速发展和广泛应用使其在科技领域以及国民经济各行各业都占有重要地位。计算机在我国的总装机量已有一百多万台。显示器是计算机的重要外部设备,是不可缺少的配套产品。随着计算机的发展以及各应用领域对显示系统要求地不断提高,显示器已从黑白两色显示发展到多种彩色,从 TTL 信号(最多 64 色)发展到模拟(Analog)信号(从理论上无限多色彩),以及低辐射、超静电技术。多频两用(TTL、VGA)显示器在电路方面是最复杂,技术上最先进的显示器,它的功能齐全,可在全频段范围内工作,适用于各种显示方式,可称为万能显示器。多频显示器具有以下几个特点:

1. 行频同步系统采用了频率自动跟踪技术

行同步信号经过整形处理,送到频率/电压转换器,将行同步信号的频率变化变成直流电压变化,去控制行振荡器(压控振荡器)的频率变化,使行振荡频率和相位能自动地与行同步信号同步。

2. 行逆程谐振电容自动切换

随着显示方式的变化,行同步信号的频率随着变化,行扫描频率自动跟踪,行扫描逆程时间要相应变化,而行逆程时间还与行逆程谐振电容有关,关系式如下:

$$T_H = T_s + T_r$$

$$T_r = \pi L_y \cdot C_r$$

式中: T_H : 行扫描周期

T_s : 行扫描正程时间

T_r : 行扫描逆程时间

L_y : 行偏转线

C_r : 行逆程电容谐振电容

3. "S"失真自动矫正电容的自动切换

由于显象管荧屏不是呈球面形状,而是个曲面或平面,因此电子束扫描角速度和线速度在屏幕中间与边缘是不一样的,这样屏幕所显示的图象就出现"S"形失真如图 1 所示:

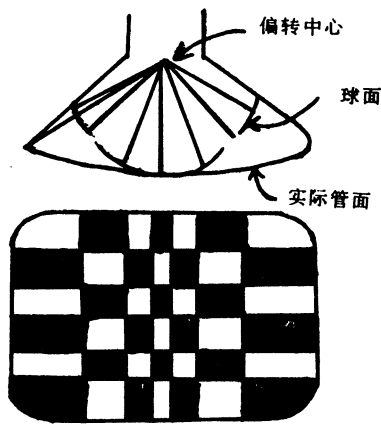


图 1 "S"形失真示意图

为了减小失真就利用电容器的频率特性,将电容器串联在偏转线圈电路中,电容器的容量大小要适当,太小起不到矫正的作用,容量太大矫正过头效果也不好。要随着扫描频率变化而变化。

4. 场幅自动控制

多频显示器能与任何显示卡联机,每一种显示卡都对应一种显示方式,显示方式不同分辨率不同。场幅的大小不同,如果不加以控制,垂直幅度就可能出现超满幅或幅度太小的现象,这样既不规范也不好看,因此必须自动加以调整。

5. 显象管高压自动稳定

由于多频显示器扫描频率是随显示方式变化而变化。行输出管集电极反峰电压的大小是与扫描频率有关系的,其经验公式如下:

$$V_{cp} = [3.14/2(Th/Tr - 1) + 1] * E$$

式中:

V_{cp} 为集电极反峰电压

Th 为行扫描周期

Tr 为行扫描逆程时间

E 行输出电源电压

由关系式可看出,要保证行输出管集电极反峰电压稳定,在行扫描频率变换的情况下,就要适当调整集电极电源电压 E 。显象管阳极高压与集电极反峰电压是成正比的。因此要保证高压的稳定一要自动调整电源电压的大小,二要稳定电源电压 E 。高压稳定一般通过 $B+$ 电压控制,如图 2 所示:

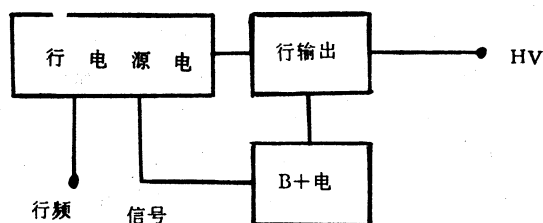


图 2 高压稳定原理框图

$B+$ 电压的形成:由行输出变压器次级取样的逆程脉冲,经整流滤波而得。有的行电源电压除了受 $B+$ 电压控制外,还受行频信号的控制。

由以上几个特点不难看出,多频显示器在技术上是先进的,在电路方面也是很复杂的,因此在维修时困难是比较大的。对此,笔者将维修实例以连载的方式介绍给广大同行,仅供参考。

例 1:

机型: CTX CC-1435 多频自动跟踪 TTL、VGA 两用彩色显示器。

故障现象: 加电指示灯亮,显象管灯丝亮,用手触摸屏幕没有高压静电反映,联机没有显示。故障分析与维修:多频显示器一般都采用“黑底”技术。所谓“黑底”技术就是在不联机时,显示器单独加电屏幕无光栅,而是黑底。那么,如何判断显示器是正常的呢?直接条件就是将显示器亮度电位器和对比度电位器调到最亮位置,可看到暗淡的光栅,管加速极电压调高些即可。如果看不见光栅可将显象管加速极电压调高即可。

间接条件有:

1. 指示灯要亮:指示灯电压由电源 12V 供电。

2. 显象管灯丝亮:灯丝电压由电源 6V 供电。

3. 显象管阳极高压正常:屏幕有高压静电反映。

4. 加电关机瞬间可听到偏转线圈磁场变化的声音。

以上判断条件该机均不能达到。

CC-1435 显示器的电源有五种电压输出,即:

6V:显象管灯丝电压

12V:指示灯,Q304,Q749,Q801-Q803,

Q805-Q821

24V:IC201(LM7830),Q201-Q205,Q301,

Q303-Q306,IC301-IC305

85V:Q302,Q309,Q310,行输出变压器高压包负端 2 脚

40-120V:行输出

根据故障现象判断行输出没有工作。怀疑行输出有问题。测量集电极电压为 250V,用指针表 1K 档或数字表二极管档检测行输出管的管极间电阻或极间电压,检查结果行输出管性能良好。测量电源输出插座 G 各脚电压 85V、24V、12V、6V,均正常,而行输出电源(40V-120V)有问题。行电源电压的产生和稳压电路是很复杂的。下面给出原理框图。

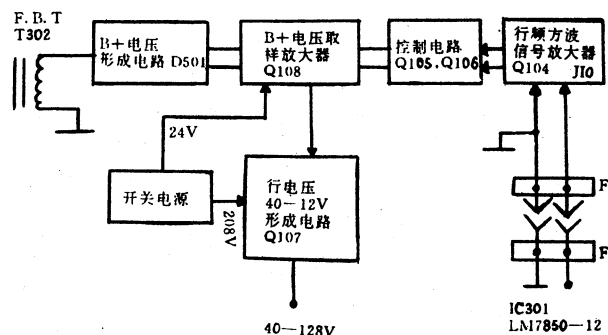


图 3 行输出电源原理方框图

框图里开关电源输出的 24V 作为控制电路的电源电压,205V 作为 40V-120V 形成电路 Q107 的电源电压。 $B+$ 电压的形成电路是将行输出 1 脚行逆程

香港金山公司
北京金山软件公司

经营:DEC、Hp 代理,方正系列汉卡

地址:100088 北京海淀知春路 22 号北京金山软件公司

电话:2049624,2040009-3035,2061869

传真:2061869

脉冲进行整流滤波得到B+电压。B+电压的变化与阳极高压的变化成正比,即与行输出电源电压的变化成正比。B+电压取样放大器由三极管Q108及周围阻容元件组成,其功能是将B+电压取样并放大送给行电源控制电路。行频方波信号由LM7850的12脚输出,通过阻容耦合加到Q104极,Q104将行频方波信号进行放大,通过电容C122加到控制电路,使控制电路的频率与行振荡频率同步。控制电路与Q107采用变压器耦合方式。Q107是它的激振荡器,其振荡频率与行振荡频率是同步的,当显示方式改变时,同步信号的频率跟着改变,Q107振荡频率就同步发生变化。行频脉冲经电感L104和电容滤波后输出,输出电压的高低与振荡频率有关,所以当显示方式变化时行输出电源电压就作相应变化。当电源电压不稳定时,由B+电压进行控制调整达到电压稳定的目的。

行输出集电极电压为205V,说明行输出没工作。检查行电源Q104—Q108及周围元件都是好的。为了验证电源是否正常,下面作排除试验:即将行输出电源断开(拔掉D插头),6V、12V、24V、85V输出插头仍插在显示器主板上,使其有关电路进行工作。在D输出插头上接上假负载(300—400/50W)加电试验,如图4所示:

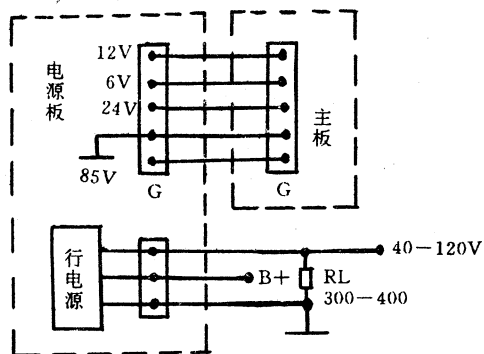


图4 行输出电源试验电路示意图

适当调整假负载电阻,加电试验结果输出电压为2.5V,但显示器主板上G插座上的电压均正常,拔掉G插座,使其空载,假负载上的电压仍为2.5V。换个好电源,按图4接好连线和负载,两种方法试验结果不变。由上述试验证明电源是没问题的。

用示波器观察行输出电源电压形成电路四个三极管Q104、Q105、Q106、Q107的集电极波形,观察结果没有波形,而且Q104基极,F插座也都没有行频脉冲波形,这说明故障在显示器主板上。继续用示波

器观看主板上F插座波形,仍然没波形,IC301 12脚还是没波形。故障一定在C301(LM7850)集成电路或外围元件上。测量芯片行振荡部分各脚电压,即1—12脚,结果7脚0.8V(正常值为6.4V),12脚为0V(正常值为1.8V)两脚都不正常。经认真查找发现电容C306有严重漏电现象,而电容容量正常。更换电容C306后加电,12脚电压和波形都正常。

将机器恢复正常,在不联机情况下加电,可听到行频“吱吱”叫,由行输出波形计算出行扫描周期为6uS、频率为14.7KHz,行电源电压为40.6V,联机时行不同步,说明行频偏低。将行频周期调到64uS时(频率为15.625KHz),与CGA卡联机行频同步而且很稳定,集电极电压为50V;当与EGA卡联机时,行周期为46uS,频率为21.74KHz,仍然同步,电源电压为63V;当与VGA卡联机时同样显示正常,电源电压为90V。

总结上述分析,由于电容器C306严重漏电,使行振荡12脚没有方波信号输出,造成行电源电压形成电路Q107停止工作,使行扫描各级电路不能工作,行电源处于空载状态。更换电容C306故障排除。接上假负载电源只输出2.5V。其原因由图5可以看出。

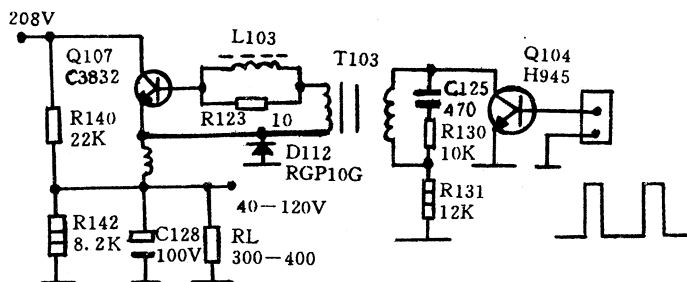


图5 行电源加假负载试验电路

SKILL

香港金山公司
北京金山软件公司

经营:DEC、Hp 代理,方正系列汉卡

地址:100088 北京海淀知春路22号北京金山软件公司

电话:2049624、2040009—3035、2061869

传真:2061869

实用文件分割程序一议

□温华波

关

于分割文件的程序,报章已曾见过,但用起来总觉得欠缺。本文与众不同之处如下:

1. 分割每一个小文件时,询问是否需要,若不要,则当前序号跳过继续进行。这样就能实现可从文件的中途开始。

2. 每个分割文件长度可变。而且中途可更改。这样可以适应手头各种磁盘。例如可以利用 1.2M 和 360 K 磁盘与及那些已经录有部分内容的磁盘,凑齐去分割大文件。

3. 原文件和分割后的文件不能硬性指定在同一驱动器内。若是如此,欲将硬盘内一个大文件要分割成装到软盘上的小文件则硬盘空间必须有比大文件大一倍的空间,以便装小文件,然后再逐一覆制到软盘,这点限制是不可忽视的,且既慢又麻烦。

4. 分割的长度应当有灵活性。有些分割程序指定非 360K 即 1.2M,本程序则无此限制。但当以“K”为单位时,会自行凑合到相近的 16 进数,例如你要设长度为 2K,程序自行定为 2048 字节。

5. 有时分割的目的只不过是为一篇文章分几部分以便分散处理(如阅读,修改),这时每个部分的末尾即不希望把一个句子截断,本文在有参数“/A”时,就会稍作延长,至该句遇到回车换行才截止。

刚才所述,直接分割转贮到软盘就会碰到转贮失败的问题。如果没有从任何中途开始及长度可更改的功能,则一旦失败必须从头再做,确实烦人。例如看起来转录很顺利,但当组合恢复时,有时中间某一张会发生读不出的错误。这一类情况实在太常见了,在本程序则只须跳过正常的文件号再执行一次即可。

可。即设定原分割长度,运行时当出现询问之时,不是要重录的文件号选“N”跳过(SKIP)即可。

以上这些功能,相信凡使用过的人均对其方便实用必有同感。也许有人问,既有 ARJ 之类可分卷压缩,还何需分割程序呢?正是上文所述的灵活性,加上直观性:即处理后的文件不存在 ARJ 格式结构,可随意阅读修改,甚至更改其长度,仍然是可合并的,且合并时勿须 ARJ 程序处理等等,这些优点即说明了其存在的必要。

最后必须讲讲怎样合并已被分解了的文件的方法。也不必专门设计一个合并的程序,直接利用 COPY 命令即可。

举例说:有三张软盘,分别录有 XY. 01, XY. 02, XY. 03 (这些 01, 02 等是本文分割时自动加上去的),现在要把它们组合到 C 盘,给一个新名字为 ALL. EXE,即 ALL. EXE 为大文件。步骤如下:(设 C 盘为当前盘)

1. COPY A:XY. 01 ALL. EXE

2. COPY ALL. EXE/B+A:XY. 02

3. 以后每次将 A 盘文件名后缀修改,重复此命令即可。

注意“/B”及“+”号都不能少,且不要加上目标文件名。

运行 CUT/? 即可得到使用方法提示。

```
;CUT
rwbuf      equ      800h
org         100h

start:
          jmp      short loc __ 1
flen __ h  dw         0
flen __ l  dw         0
```



```

bas __1      dw      0
bas __2      dw      8000h
sum __1      dw      0
sum __2      dw      0
data __1     dw      0
data __2     dw      0
addfno       db      0,0,0
data __4     db      0
hdl __1      dw      0
hdl __2      dw      0
add __2      dw      0
buf          db      20 dup(0)
kbuf         db      10 dup(0)
loc __1:
    mov     si,80h
    mov     al,[si]
    mov     dx,offset hlpwd
    cmp     al,4
    jb      to __4
    jmp     loc __10
to __4:      jmp     loc __4
jerr:       jmp     loc __3
loc __2:
    mov     si,81h
    inc     si
    mov     al,[si]
    cmp     al,20h
    jz      i21
    mov     dx,si
    mov     ax,3d00h
    int     21h
    jc      jerr
    mov     hdl __1,ax
    mov     ax,4202h
    mov     bx,hdl __1
    mov     cx,0
    mov     dx,cx
    int     21h
    jc      jerr
    mov     flen __h,dx
    mov     flen __l,ax
lp1:        call    dow
    mov     di,0
    mov     ax,4200h
    mov     bx,hdl __1
    mov     cx,sum __2
    mov     dx,sum __2
    int     21h
    jc      loc __3
    mov     ax,bas __2
    mov     data __2,ax
    mov     ax,bas __1
    mov     data __1,ax
    mov     ax,flen __h
    mov     cx,flen __l
    cmp     sum __1,ax
    ja      exrw
    jc      w1
    cmp     sum __2,cx
    jnc     exrw
w1:         cmp     si,0

```

```

jz          w2
call        sub __1
w2:         cmp     di,0
    jnz     exrw
    mov     ax,bas __2
    add     sun __2,ax
    mov     ax,bas __1
    adc     sum __1,ax
    mov     si,word ptr addfno
    mov     ah,[si]
    inc     si
    mov     al,[si]
    inc     ax
    cmp     al,3ah ;=9?
    jc      tolp1
    inc     ah
    mov     al,30h
    dec     si
    xchg    ah,al
    mov     [si],ax
    jmp     lp1
exrw:       mov     ah,3Eh
    mov     bx,hdl __1
    int     21h
    mov     dx,offset okwd
    jmp     short loc __4
loc __3:    mov     dx,offset data __5
loc __4:
    mov     ah,9
    int     21h
    int     20h
sub __1     proc
    near
    mov     ah,3Ch
    mov     dx,offset buf
    mov     cx,20h
    int     21h
    mov     hdl __2,ax
loc __5:
    mov     dx,data __1
    mov     ax,data __2
    mov     cx,0f000h
    cmp     dx,0
    jne     loc __6
    cmp     ax,0f000h
    jae     loc __6
    mov     cx,ax
loc __6:
    mov     di,0

```

UNIX 找龍馬

为您提供从 UNIX 原版软件到大型网络系统集成服务

地址邮编:北京海淀西操场一号天府饭店 428-433 室
电话号码:(01)2545980 2545981 2569499 2532515

```

sub     ax,cx
sbb     dx,0
mov     data__1,dx
mov     data__2,ax
mov     ah,3Fh
mov     bx,hdl__1
mov     bx,rwbuf
int     21h
cmp     ax,0
je      loc__7
cmp     ax,cx
je      subcon
mov     di,1
suncon: push ax
pop     cx
mov     ah,40h
mov     bx,hdl__2
mov     dx,rwbuf
int     21h
cmp     di,0
jnz     loc__7
jmp     short loc__5
loc__7: call sub__2
mov     ah,3Eh
mov     bx,hdl__2
int     21h
retn

sub__1  endp
sub__2  proc      near
mov     al,data__4
cmp     al,0
je      loc__ret__9
mov     ah,3Fh
mov     bx,hdl__1
mov     dx,rwnuf
mov     cx,1
int     21h
cmp     ax,0
je      loc__ret__9
push    ax
pop     cx
mov     ah,40h
mov     bx,hdl__2
mov     dx,rwbuf
int     21h
mov     ax,0c01h
int     21h
cmp     byte ptr [rwbuf],1Ah
je      loc__ret__9
cmp     byte ptr [rwbuf],0Ah
jne     lic__8
lic__8:
loc__ret__9: retn
sub__2  endp
filw2   proc
mov     si,[add__2]
inc     si
mov     al,[si]
cmp     al,20h
jz      if1
mov     cld

```

```

if2:    lodsb
        cmp     al,'/'
        jnz     lp201
        dec     si
        jmp     exf2
lp201:  cmp     al,20h
        jz      exf2
        stosb
        cmp     al,'.'
        jnz     lf2
lp20:   lodsb
        cmp     al,'/'
        jne     no2f
        dec     si
        jmp     lp21
no2f:   cmp     al,20h
        jnz     lp20
lp21:   dec     di
exf2:   mov     al,'.'
        mov     [di],al
        mov     al,'0'
        inc     di
        mov     word ptr addfno,di
        mov     [di],al
        inc     di
        inc     al
        mov     [di],al
        dec     si
        ret

file2   endp
dow     proc
mov     si,word ptr addfno
mov     al,[si]
inc     si
mov     ah,[si]
mov     si,offset nowd
mov     [si],al
inc     si
mov     [si],ah
mov     dx,offset askwd
mov     ah,9
int     21h
mov     ax,0c01h
int     21h
mov     si,0
cmp     al,'n'
jz      isno
cmp     al,'N'
jz      isno

```

UNIX 找龍馬

为您提供从 UNIX 原版软件到大型网络系统集成服务

地址邮编:北京海淀西操场一号天府饭店 428-433 室
电话号码:(01)2545980 2545981 2569499 2532515

```

        cmp     al,'/'
        jne     notno
lpkey:   mov     ax,0a06h
        mov     dx,offset kbuf
        mov     di,dx
        mov     [di],al
        int     21h
        mov     si,offset kbuf+2
        call    noap
        cmp     dx,0
        je      notno
        mov     ax,0e07h
        int     10h
        mov     [80h],di
        jmp     lpkey
notno:   mov     si,1
isno:    mov     ax,0e0dh
        int     10h
        mov     al,0ah
        int     10h
        ret
dow      endp
noap     proc
;enter:si->input buf.out bas __ 1,2
        mov     cx,0Ah
        mov     ax,0
        mov     bx,0
        mov     dx,0
loc __ 15: mov     bl,[si]
        cmp     bl,4Bh ; 'K'
        je      loc __ 16
        cmp     bl,'k'
        jz      loc __ 16
        cmp     bl,30h
        jb      loc __ 17
        cmp     bl,3Ah
        jae     loc __ 17
        sub     bl,30h ; '0'
        cmp     dx,0
        jne     l19
        mul     cx
        add     ax,bx
        inc     si
        jmp     short loc __ 15
loc __ 16: mov     cx,400h
        cmp     dx,0
        jne     l19
l161:    mul     cx
loc __ 17: mov     bas __ 1,dx
        mov     bas __ 2,ax
        mov     dx,0
l19:     ret
noap     endp
loc __ 10: mov     si,81h
loc __ 13: inc     si
        mov     al,[si]
        cmp     al,0Dh
        je      to19
        cmp     al,20h
        jne     loc __ 13
        mov     add __ 2,si;1st file

```

```

        mov     byte ptr [si],0
loc __ 14: inc     si ;for 2nd file
        mov     al,[si]
        cmp     al,0Dh
        je      loc __ 20
        cmp     al,'/'
        jne     cock
        dec     si
        jmp     jcal
cock:    cmp     al,20h
        jne     loc __ 14
jcal:    call    file2
loc __ 18: inc     si
        mov     al,[si]
        cmp     al,'/'
        je      ispr
        cmp     al,0Dh
        je      loc __ 20
        cmp     al,20h
        je      loc __ 18
        jmp     loc __ 19
loc __ 20: jmp     loc __ 2
ispr:    inc     si
        mov     ax,[si]
        cmp     al,'a'
        je      isa
        cmp     al,'A'
        jne     noa
        mov     al,1
        mov     data __ 4,al
l101:    jmp     loc __ 18
noa:     call    noap
        cmp     dx,0
        je      l101
loc __ 19: mov     dx,offset data __ 6
        mov     ah,9
        int     21h
        int     20h
hlpwd    db      'Usage: CUT infile outfile /len[K][ /A]',
             0Dh,0Ah,0Ah
        db      'len=1-65535', 0Dh
        db      '[ /A] use as textfile(file end intact)'
        db      0Dh,0Ah,24h
data __ 5 db      'FILE NOT FOUND! ',7,0Dh,0Ah,24h
okwd     db      'OK! ',0Dh,0Ah,'$'
data __ 6 db      'PARAMETER ERROR! ',7,0Dh,0Ah,24h
askwd    db      '[|n|/].[/]input new ten. [n]skip. ,File #'
nowd     db      30h,30h,'... $'

```

SKILL

UNIX 找龍馬

为您提供从 UNIX 原版软件到大型网络系统集成服务

地址邮编:北京海淀西操场一号天府饭店 428-433 室
电话号码:(01)2545980 2545981 2569499 2532515

DOS 与 Windows



空字符 NUL 有什么妙用?



将键盘操作的输出送到 DOS 的空设备可以避免屏幕杂乱,此外,空字符 NUL 还有其它用途。可以使用 DOS 的假定功能,假定所有的磁盘都有一个名为 NUL 的文件,以判定一个系统中是否存在一个特定的硬盘。例如,下面这个批处理文件 DRVTEST. BAT,接受一个盘符做为输入,并且提示你这个盘是否存在。DRVTEST. BAT 的内容如下:

```
@ECHO OFF
IF EXIST %1\NUL ECHO
DRIVE %1 EXISTS
IF NOT EXIST %1\NUL E-
CHO DRIVE %1 DOES NOT
EXIST
```

可以使用 EDIT,EDLIN 或者一个字处理软件来输入上面三行,以建立一个 DRVTEST. BAT 文件,然后,在 DOS 提示符下,可以通过输入一个命令,命令格式是在 DRVTEST 后面加上一个盘符,如:DRVTEST C: 来检测一个硬盘驱动器是否存在。

在这个例子中,DOS 用 C: 代替 %1 参数,实际上就将批处理文件的第二行换成 如下的一行:

```
IF EXIST C:\NUL ECHO
DRIVE C: EXISTS
```

DRVTEST. BAT 对软驱不起任何作用,因为如果驱动器中没有软盘,DOS 将提示 如下错误信息:

```
Not ready reading drive A(或
B)
```

Abort,Retry,Fail?

那么批处理文件就根本没有机会检查 NUL 是否存在。当执行一个特殊的操作,需要让操作者选择硬盘驱动器时,这个批处理文件做为一个大批处理文件的一部分是非常有用的。使用这个技巧得到一个系统的硬盘驱动器的完整的目录。下面的这个批处理文件 DRVINV. BAT,可以完成这件工作。

```
DRVINV. BAT 的内容如下:
@ECHO OFF
ECHO HARD DRIVES ON
THIS SYSTEM:
```

```
FOR %%D IN (C: D: E: F:
G: H: I: J: K:
L: M: N: O: P: Q: R: S:
T: U: V: W: X:
Y: Z:) DO IF EXIST %%D
\NUL ECHO %%D
```

建立和保存这个批处理文件后,使用这个命令时,只需在 DOS 提示符下输入:

DRVINV

但要注意:批文件中的第三行非常长,当你输入这一行时,在括号中的内容要确保每两个盘符间都要输入空格。



在安装一些应用软件时,常会遇到安装程序改变 AUTOEXEC. BAT 文件内容的情况。如何才能保存 AUTOEXEC. BAT 的内容,使之不被更改而正常运行?



有一些程序在执行安装过程中,通常会更改 AUTOEXEC. BAT 文件的内容,这对你曾花费了很长时间来编写 AUTOEXEC. BAT,使它在每次开机时精确地执行你所让它完成的工作来说,那么这种由于安装引起的更改是挺让人头疼的。

这里有一个比较简便的方法来保护你的 AUTOEXEC. BAT 文件不会被意外地更改,操作步骤如下:

(1) 拷贝当前的 AUTOEXEC. BAT 文件到一个新的文件里,比如可以叫做 STARTUP. BAT:

```
C > COPY AUTOEXEC.
BAT STARTUP. BAT
```

(2) 建立一个新的 AUTOEXEC. BAT 文件,文件的内容为:

CALL STARTUP

在这种设置下,AUTOEXEC.BAT 只是运行 STARTUP 文件,这样,即使程序改变了 AUTOEXEC.BAT 的内容,最初的设置还是被保存下来了。安装完程序后,可以用一个文本编辑器将 AUTOEXEC.BAT 文件中存放的重要语句转移到 STARTUP.EXE 文件中即可。

? 如何利用 DOS 5 快速格式化磁盘?

! 用 DOS 5 快速格式化软磁盘,需要在调用 FORMAT 命令时加上 /Q 和 /U 参数,操作格式如下:

FORMAT A: /Q /U

这个命令告诉 DOS 要快速无条件地格式化一张 A 驱动器中的软盘。

利用这个命令,可以很快地格式化一张新盘或重新格式化一张已使用过的旧盘,但是格式化完后却不能再使用 UNFORMAT 命令了。因为加了 /U 参数,DOS 将不会在格式化时保存磁盘上的信息。当然,对新盘来讲,磁盘上没有任何信息,所以用无条件参数 /U 来格式化它们是安全的。

使用 FORMAT 命令,还可以选择 /F:size 参数,指定了一张磁盘的容量,DOS 将无条件地格式化这张磁盘。

? 我想在列硬盘上的文件目录时,能够把文件和子目录分别列出,并且能够按一定的顺序排列吗?如何能够达到这个目的?

! 在 DOS 5 中,如果你使用如下 DIR 命令:

DIR /O:N

那么硬盘上的文件和子目录将以字母升序的顺序排列,这样虽

然好,但是还有一个问题,那就是所有的文件和子目录是混杂在一起排列的,如下面这个例子:

```
<DIR> 08-31-92 9:14a
<DIR> 08-31-92 9:14a
FIG4-2 TXT 443 09-01-92 8:14a
RPMSWAP TMP 2162688 08-31-92 1:41p
VOODOO <DIR> 08-31-92 9:14a
XENO <DIR> 08-31-92 2:54p
6 file(s) 2163131 bytes
29315072 bytes free
```

在硬盘上存有大量的文件和子目录的情况下,使用 DOS 5 的 DIR 命令,可以将所有的子目录按字母顺序排列在前,将所有的文件按字母顺序排列在子目录的后面,操作如下:

DIR /C

那么上面例子中的文件及子目录将会作如下排列:

```
<DIR> 08-31-92 9:14a
<DIR> 08-31-92 9:14a
VOODOO <DIR> 08-31-92 9:14a
XENO <DIR> 08-31-92 2:54p
FIG4-2 TXT 443 09-01-92 8:14a
RPMSWAP TMP 2162688 08-31-92 1:41p
6 file(s) 2163131 bytes
29315072 bytes free
```

如果盘上的文件和子目录数量多,可以再加 /P 参数,分屏浏览文件目录了。

? 怎样合理使用 PATH 命令?

! 如果 PATH 语句需要 DOS 寻找几个驱动器和成百个文件,那么计算机的运行速度将会大大降低,因此要保证 PATH 命令的整齐,短小。PATH 命令行最长只能为 127 个字符。

DOS 只在寻找可执行文件时

才查找路径设置中是否指明,不能使用 PATH 来查找数据文件。所以,最好将应用软件的可执行文件放入一个目录中,将数据文件放入另外一个或更多的目录中。例如,你最好将 QORDPERFECT 装入 C:\WP51,而将它的一些文档文件装入 C:\WP51\DOCS;然后在 PATH 中指定 \WP51 目录。格式如下:

PATH = C:\DOS;C:\Windows;C:\WP51

设置 PATH 时,要仔细考虑所涉及到的目录的顺序。DOS 根据 PATH 指定的路径从所列的一串目录的最左端开始寻找文件,找到后调用这个文件。应将最常用的目录放在最前面,且要保证在 PATH 语句中间不包括任何空格符号。因为 DOS 在遇到空格时将停止读取目录。

此外,在 PATH 中列出的经常使用的目录,可以再建立一个简单的批文件,用来转换程序的目录并且在从提示符下激活批文件时启动应用程序。这样,你就不必记住用来启动程序的最初命令,就不会将机器的运行速度降低下来。

例如,如果你使用一个 SPREADSHEET 程序,名字叫做 Crunch,调用可执行文件,CRYPTIC.EXE,那么建立如下的批文件,名字叫做 CRUNCH.BAT:

CD C:\CRUNCH
CRYPTIC

将 CRUNCH.BAT 装入 BATCH 目录,这个目录应该包含在 PATH 语句中,以便你可以在任何目录下运行所有的批文件。

这样,只要敲入 CRUNCH 即可启动程序,并且可以自动切换到应用程序所在的子目录。

顺便提醒一点,当你清除 PATH 语句时,将 Windows 程序

从 PATH 中挑出(但是留下 Windows 自身)。Windows 用 WIN.INI 来保存能找到其程序的轨迹。

? DOS 5 中有许多命令在 DOS 6 中没有了,这些命令还有用吗?

! DOS 6 中丢失了许多在以前的 DOS 版本中可以见到的命令,如果你的旧批处理文件中使用了旧版中的命令,就要确保没有从 DOS 子目录删除掉这些文件。不过不必担心,DOS 6 的安装程序在安装过程中不会删除或覆盖旧的程序。

在 DOS 6 丢失的命令中, BACKUP, ASSIGN, COMP, EDLIN, EXE2BIN, GRAFTABL, JOIN, RECOVER 和 MIRROR 这些命令可能大部分你都不需要了,除了 MIRROR,这个命令包括一个 /PARTN 参数,让你可以保存一份关于硬盘分区信息的副本在你的系统启动软盘上,如果在硬盘分区表中发现了错误,可以用备份的这个分区表恢复使驱动器能够正常工作。如果 DOS 6 是你使用的第一个 DOS 版本,或者你跳过了 DOS 5,直接升级到 DOS6,可以找一份 MIRROR.EXE 的复制本(也包括其他命令),然后将它拷贝到 DOS 子目录中。

? 如何利用删除命令,将指定的文件保存,而将指定文件以外的文件全部删除?

! 我们平时使用 DEL,可以直接删除某一类特定的文件,但是想要用 DEL 删除除某一类特定文件外的其他文件,就不能直接达到目的了。

下面这个 DELBUT.BAT 文件可将使这种删除工作变得简便易行,文件内容如下:

```
@ECHO OFF
ATTRIB *.* -R
:START
IF %1P==P GOTO DELETE
ATTRIB %1 +R
SHIFT
GOTO START
:DELETE
ECHO Y|DEL *.* > NUL
ATTRIB *.* -R
```

这个批处理文件将可以做到删除指定某一类文件外的所有其他文件的目的。

使用这个程序时,敲入 DELBUT 后,后面要带上你不想删除的文件的文件名,可以是一个,也可以是多个,文件名也可以用通配符来表示,代表一类文件。在每个文件名或有通配符的文件名之间要用一个空格来分隔。

例如,想从当前目录中删除除 MENU.EXE 和以 .COM, .DOC 为后缀的文件以外的所有文件,可以敲入如下命令:

```
DELBUT MENU.EXE *.
COM *.DOC
```

这一程序将首先关闭当前目录中所有文件的只读属性,然后再将在命令行中指出的文件(也就是你想删除的文件)的属性置为只读属性。当批处理程序执行到语句:

```
ECHO Y|DEL *.* >
NUL 时,
```

DOS 将删除没有标记为只读的所有文件,并且将 DOS 的信息重定向到 NUL 设备以使它们不在屏幕上显示出来。最后,DELBUT.BAT 将关闭没有被删除的文件的只读属性。

同时需要指出和说明的是这个程序有一个缺陷:它更改了被只读属性所保护的文件的属性。为保证这些文件不被删除,在调用 DELBUT 时需要指定这些文件名。在执行完 DELBUT 后,如果你想保护这些文件不被修改或删除,就需要使用 ATTRIB 命令将它们

标记为只读文件。

? 如何方便地从 DOS 状态退回到 Windows?

! 当你从 Windows 3.1 中退到 DOS 提示符下时,会看到一条提示信息,底色是黑色,字符颜色是白的,告诉你退回到 Windows 时敲"EXIT"。但是,在你敲入几个 DOS 命令后,这个提示会被从屏幕中上卷走,因此会很容易忘记你是从 Windows 下调用的 DOS 状态,并且忘记从 DOS 下返回到 Windows 需用什么命令。

为了避免这样的情况发生,可以在 AUTOEXEC.BAT 文件中加入如下一行 WINPMT 命令:

```
SET WINPMT = $e[s $e[1;1f $e[0;45;37;1m $e[K $d
Windows' DOS Prompt.
```

```
Type EXIT to return to Win-
dows. $v $ — $e[0m $e[u $p
$g
```

这个提示命令将显示出一条紫色亮条在屏幕的顶行。亮条中的信息(包括星期,日期,提示内容和当前的 DOS 版本号)以白色字符出现,这些信息在执行 DOS 命令过程中不会随屏幕内容上卷而消失,始终出现在屏幕上方,一直到退回到 Windows。

要想使这个命令行之有效,在输入上面那行命令时必须小心谨慎,确保正确使用大小写字符和上位键,下位键字符,输入的内容要与上面命令行中所显示的完全一致。同时,必须在 CONFIG.SYS 文件中调用 ANSI.SYS 文件,因为 DOS 依赖这一程序中断这样一个工作序列。例如,在 C 盘 DOS 子目录中存在 ANSI.SYS 文件,正确的 CONFIG.SYS 文件中的命令调用如下所示:

```
DEVICE = C:\DOS\ANSI.
SYS
```

SKILL

帝霸计算机反病毒服务网 专用软件升级信息表

基本描述

库类别: 0

病毒名称: AAV

攻击对象: 2

病毒长度: 2020

第(1)基址描述

4/2/0:8/2/FF00

第(1)特征描述

1/196/2E. C6. 6. 62. 2. FF. E8. 90. 2. 2E. 80. 3E. F. 1. 6D

基本描述

库类别: 0

病毒名称: AAV

攻击对象: 2

病毒长度: 0

第(1)特征描述

0/196/2E. C6. 6. 62. 2. FF. E8. 90. 2. 2E. 80. 3E. F. 1. 6D

病毒分析:

该病毒发现于北京地区。它被命名为 AAV。这是一种文件型病毒,只传染 COM 型文件。该病毒活动时驻留在内存低端,它截获 DOS 中断 INT 21H 以获取系统控制权。当系统时间大于等于 1994 年 5 月,在每个小时的 30 分 45 秒之后,病毒开始发作。屏幕上将显示如下信息: THIS FILE IS SAFE. THANKS FOR USE AAV IDEARS AUTO _ ANTI _ VIRUS SOFTWARE GROUP AAV MARK:4540055520。当已感染该病毒的程序再被加长时,运行该病毒程序将显示如下信息: THIS FILE MAY BE INFECTED WITH VIRUS TO KILL

VIRVS YOU CAN REINSTALL THIS FILE。本病毒无明显的恶性破坏作用,当其传染 C:\COMMAND.COM 时,病毒程序将强行占用硬盘的 0 柱面 0 道 2~14 区,有可能对硬盘造成严重破坏,尤其是当硬盘分区表感染有启动型病毒时,分区表被彻底破坏的可能性非常大。另外,在高版本 DOS 下 AAV 传染 Command.com 文件时,会破坏 Command.com 文件,使系统无法启动。特提醒广大读者注意。

注:AAV 病毒由于有两种感染文件机制,因而为了快速准确有效的识别该病毒,需要建立二个数据库项(如上所示),请广大用户仔细准确的录入。

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。
3. 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。
4. 每月免费赠送一本《电脑编程技巧与维护》杂志。
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,网员可获优先和优惠。

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。

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。

2. 每年本网发表的软件升级版本,均免费赠送一套。
3. 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。
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。

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。

致读者

自本刊今年7月创刊至今,已有四期杂志同我们亲爱的读者见了面。《电脑编程技巧与维护》这样一本实用性和技术性都很强的杂志在我国可说是首创,虽然四期杂志的出版过程是极为短暂的,还仅仅是开始,但令我们欣慰的是,我们已拥有了一批支持、帮助我们的读者朋友,真心的教诲、诚恳的建议、热切的期望……使得这样一个刚刚创办的杂志其栏目更加丰富、内容更加充实。在他们的鼓励和帮助下,我们解决了初创阶段一个又一个的困难,同时也坚定了我们办好这样一种为读者提供实用技术的杂志的信心。

杂志是广大读者的杂志,我们认为杂志的办刊宗旨和方针应该体现广大读者的心愿,为此我们在创刊的开始就细心聆听广大读者的建议,并因此而丰富了本刊现有的栏目和内容,同时从本刊的创刊号开始就刊登读者反馈信息表。在几个月的时间中我们陆续收到了来自全国各地的读者来信数千封,在认真地拜读了每一封来信之后,我们总结出了有代表性的意见和建议,并将在我们今后的编辑出版工作中逐步得到实施。本刊为了感谢广大读者的热心帮助和支持,决定从这些读者来信中选出20名对本刊建议中肯、有代表性的读者作为本刊的第一届“热心读者”,对这些“热心读者”本刊将免费赠阅1995年全年的杂志。

当然,关心本刊的读者远不止这20名,本刊不能一一致谢,只能在我们未来的编辑出版过程中本着“实用第一、智慧密集”的办刊方针为广大读者提供更多更好的实用文章,同时将以读者为上帝而全心全意地提供服务,以此回报广大读者对我们的厚爱。最后真诚地希望广大读者对本刊继续给予指导。

《电脑编程技巧与维护》杂志编辑部

1994年10月

本刊第一届“热心读者”

获奖名单:

- | | |
|-----|-------------------|
| 徐京 | 北京,中国资源卫星应用中心 |
| 林钧礼 | 北京,北师大计算中心 |
| 阎鸿邦 | 北京,国家环保局海洋处 |
| 宁培松 | 北京,煤炭部信息中心 |
| 谭为 | 北京,光大国信北京系统集成联合公司 |
| 高淑莲 | 内蒙古,包头城建技校 |
| 巴力登 | 新疆,新疆工学院 |
| 杨洁 | 湖南,永州市三中 |
| 赵亚均 | 浙江,新昌县 |
| 米明伟 | 黑龙江,省农机总公司 |
| 王秀琦 | 黑龙江,哈尔滨市科技情报研究所 |
| 黄树可 | 广东,惠东县盐洲中心小学 |
| 谭浩通 | 广东,广州华南理工大学 |
| 管旭东 | 陕西,西安交通大学 |
| 唐晓东 | 陕西,西安第四军医大学 |
| 马建华 | 河北,省电子技术研究所 |
| 曹国柱 | 河北,石家庄军械工程学院 |
| 包敢 | 江苏,省委党校图书馆 |
| 吴建齐 | 江苏,睢宁县苏塘乡粮油管理所 |
| 李学春 | 山东,东营石油大学 |

《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	

《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	

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)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"/> 趋势与展望 | <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"/> 硬件维护 | <input type="checkbox"/> 实用软件 |
| <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"/> 新技术追踪 | <input type="checkbox"/> 趋势与展望 | <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"/> 软件维护 | <input type="checkbox"/> 硬件维护 | <input type="checkbox"/> 实用软件 | <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_____