

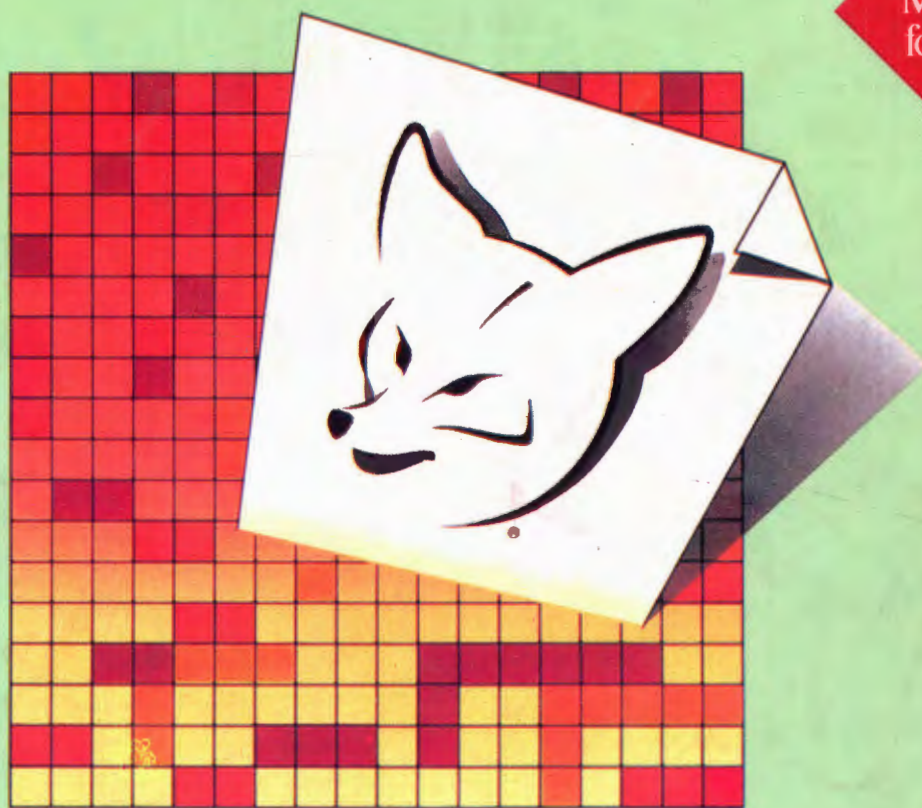
电脑编程技巧与维护

COMPUTER PROGRAMMING SKILLS & MAINTENANCE

1994 年 9 月



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



Microsoft **FOXPRO**
LIBRARY CONSTRUCTION KIT

For MS-DOS and 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。

☆ UCDS、dBASE IV 2.0中文版。

AUTO CAD 汉字环境等各种软件。

☆ 计算机技术资料。

地址:北京海淀路82号

电话:2567387、2562329、2565884、

2567819、2579873

信箱:北京8721信箱

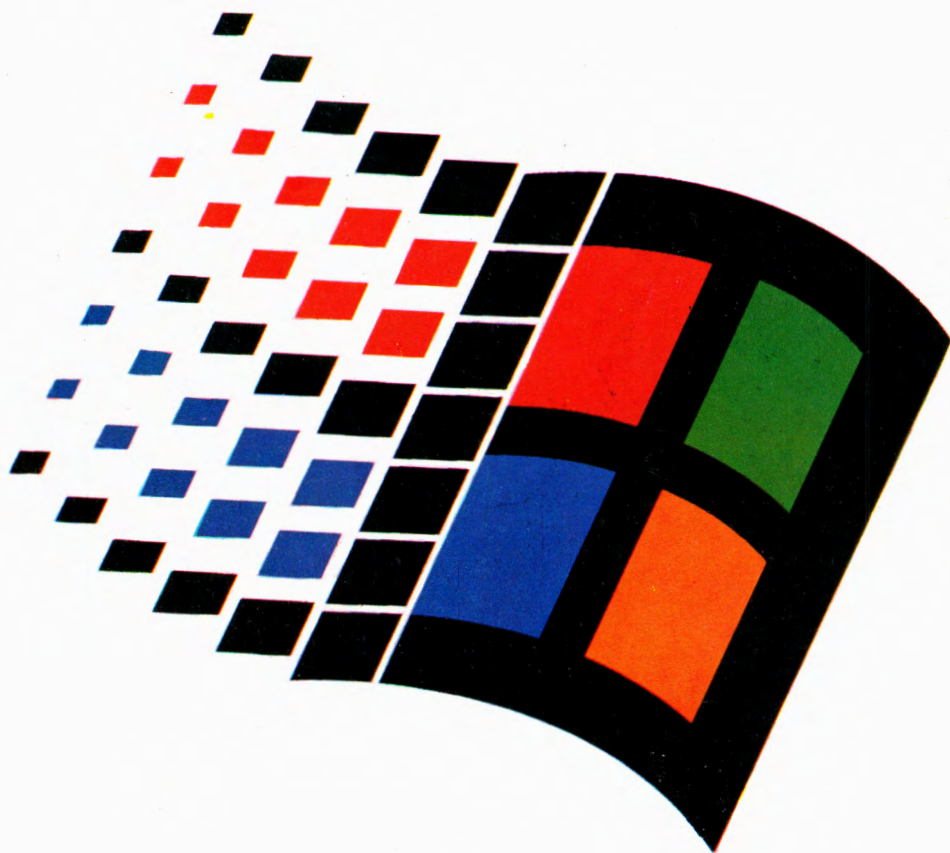
邮政编码:100080

电挂: 0755 传真: 2561057



北京希望

电脑公司



Microsoft® **WINDOWS NT**™

希望汉字系统

汉字系统的希望 UC DOS 3.0

为什么我们总是人们追赶的目标？

※ 支持直接写屏,英文制表符自动识别

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

※ 国内唯一真正实现零内存的汉字系统

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

※ 经香港金山公司授权,系统配备 WPS 进行文字处理

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

※ 真正实现网络共享

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

※ 完善的汉字输入方法

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

※ 强大的打印功能

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

※ 特殊显示功能

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

※ 彻底支持 DOS5.0、DOS6.0 和 DRDOS6.0

※ 彻底支持鼠标功能

※ 本系统以纯软件方式提供,是便携机用户的最佳选择

敬告用户:

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

北京希望电脑公司软件部

地址:北京 海淀路 82 号

信箱:北京 8721 信箱

邮编:100080

电话:2579873、8422024、

8422025

传真:2561057

上海希望电脑公司

地址:(200052)上海新华路 416 号

电话(传真):(021)2521656、

2569929

成都希望电脑公司

地址:(610015)成都市新南路

四维村街 6 号(磨子桥附近)

电话(传真):(028)5589787、

5556333

广州希望电脑技术公司

地址:(510620)广州天河体

育西路育蕾三街二号

电话:(020)7505151、7505152、

7505153

传真:(020)7500275

南京希望电脑技术公司

地址:(210005)南京市中山南路 105 号

电话:(025)4410794、4410549

传真:(025)4410548



即使您对计算机语言一窍不通,您立刻会成为编程大师;即使您已经是一名高级程序员,您立刻会成十倍地提高编程效率

《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,会成十倍地提高编程效率。《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:

可自动生成图形与文字“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罗其中。

七、“智能化”技术,独树一帜:

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

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

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

全国各地代理经销商名单(同时办理当地代理赠送业务)

单位名称	电话
安徽合肥安兴高科技开发公司	2649222
北京高林技贸公司	2543815
北京三江新技术产业公司	3816193
北京微宏电脑软件研究所	2560964
福建厦门四通公司	5088362
广东广州黄花岗产业区物科实业公司	3832081
广东广州瑞达信息技术服务部	3330898-3506
广东广州市海达科技有限公司	5515979-235
广东广州中兴现代办公设备公司	5515870
广东深圳爱华电子有限公司	3208546
广东深圳意百达科技有限公司	3333336
广东珠海海韵高科技研究所	8892736
广西南宁市数据设备公司	569162
海南省海口市安泰光电子实业有限公司	8662946
河北沧州创新电脑系统服务部	222871
河南开封师专电子技术中心	554795
河南洛阳市亚普技贸公司	426932
河南新乡神通计算机公司	220286
河南郑州艺高电脑新闻广告有限公司	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
江苏淮阴四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
辽宁锦州华桥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
陕西省仪口市电脑软件公司	219324
陕西西安市新航电子技术公司	4253439
上海海贝船舶技术研究所	4860037
上海两英家文经营部	2583576
上海中兴电脑经营服务公司	4395300-2402
上海市劳动局信息中心	3281394
四川攀枝花市金谷科技贸易发展公司	334187

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

特别注意
免费赠送

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,每单位限领一套,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。

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
邮费请另加特快专递费 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

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

顾问 郭平欣 刘洪昆

名誉社长 周明陶

社长 毕研元

副社长 袁保佑

总编 秦人华

执行总编 秦长久

副总编 王路敬 孙毅

编辑 开元 程钧 易言

艾蒙 田艾 肖贻

苏古 阮薇 温达

管逸群 业成

公关部主任 吕莉 伊梦

出版部主任 祁连顺

发行部主任 孙茹萍

编辑出版 《电脑编程技巧与维护》杂志社

地址 北京市劳动人民文化宫内

邮编 100006

编辑部电话 5123823

公关部电话 5232502

照排 《电脑编程技巧与维护》杂志社
电脑排版部

印刷 北京市地质矿产局印刷厂

订阅处 全国各地邮电局

国内总发行 山东省潍坊市邮电局

邮发代号 24-106

刊号 CN11-3411/TP

每期定价 3.80 元

出版日期 1994 年 9 月 8 日

敬告读者

1995 年报刊征订工作即将开始,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 3 期

(总第 3 期)

新技术追踪

DEC 扩展了它的客户机/服务器系列等 20 篇 (4)

软平台

扩展 Chicago 外壳 (8)

在 DOS 5.0、6.0 中运行 Novell 网上的 MIS 系统 (10)

实用简单的 PATH 编辑工具 (12)

FIND 命令的功能扩充及应用 (13)

也谈 DOS 重定向功能 (15)

图形图象处理

编写基于位平面的图象处理函数 (16)

在 DOS 下快速显示 SPT 图形 (22)

幻灯片文件的格式分析 (23)

中文幻灯片制作与放映 (26)

汉字处理

12×12 点阵汉字显示系统 (28)

PostScript 文件格式分析和调用 (35)

编程语言

一个实用的排版预处理程序 (39)

用 C 语言实现造型表功能 (40)

Visual C++ 的编程技巧——使用内置函数 (41)

用 Microsoft VB 编程 (43)

计算机安全

磁道接缝软件加密技术 (45)

征稿启事

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

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

在以上这些方面有经验所得的同仁请不吝赐稿。稿件一旦录用即寄样刊及稿酬。本刊每年都将举办优秀撰稿人评选活动,评选出的优胜者,本刊将免费赠送下一年的《电脑编程技巧与维护》杂志。来稿请寄(100006)北京市劳动人民文化宫内《电脑编程技巧与维护》编辑部。稿件请用稿纸书写整齐或打印出来,所附程序均请打印清楚。作者姓名、单位及通信地址、邮编请用整齐的字迹写在醒目的位置。来稿必须将有关的程序完整地附上。本刊鼓励软盘投稿(同时也请附上一份打印稿),磁盘投寄的稿件一旦录用,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

磁盘文件的彻底删除	(50)
子目录与文件的安全浅谈	(51)

数据库

FoxPro 应用系统中的错误捕获和处理	(52)
FoxBASE 数据库系统中一些不可忽略的功能和问题	(54)
汉字 FoxBASE+ 最小化的 DOSHELL	(56)
信息输入方式自动转换的实现	(57)
FoxBASE 数据库的同名字段加密法	(59)

软件维护

忘记 WPS 文件口令后的恢复技巧	(62)
为 DOS 系统增加一个模拟 TYPE 的外部命令	(65)

网络与通讯

使用 Access 2.0 的客户机/服务器特性	(66)
终端打印机接口程序的分析与设计	(68)
Novell 网服务器安装技巧	(71)
Novell 网络应用系统常见故障的维护技术	(72)

电脑博士信箱

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

电脑反病毒信息公告

反病毒升级信息表	(77)
----------------	------

本刊全部内容的版权,归本社所有,未经同意,不得转载。

新技术追踪

DEC 扩展了它的客户机/服务器系列

DEC 公司最近推出了几种新的增强的客户机/服务器产品。DEC 的新产品中包括一个新的多处理器服务器。这种服务器支持 4 个 Alpha CPU、Windows NT 下的 DCE(分布式计算环境——Distributed Computing Environment)软件、工作组(Workgroup)环境下的 X. 500 目录服务,以及 Pathworks LAN 操作系统和 IBM 连接软件的扩展。

DEC 产品中的 Digital 2100 Model A500MP 部门服务器是该公司此类产品中的主要产品,其代号为 Sable。2100 支持 4 个 190MHz Alpha 微处理器、2GB 内存及 170GB 的磁盘。据称,这种服务器支持两种总线结构:PCI(Peripheral Component Interconnect)和 EISA(Extended Industry Standard Architecture)。PCI 总线速率为 132Mbps 并支持三个扩展插槽,EISA 总线的速率为 33Mbps 支持八个扩展插槽。

这种服务器支持 OpenVMS、DEC OSF/1、Windows NT Advanced Server 操作系统,价格为 18900 美元。

Windows NT 下的 DCE 软件允许用户开发运行于 Windows NT 工作站及服务器(包括 DEC 的新 2100)上的分布式客户机/服务器应用软件。这些应用程序和其它的 DEC 应用程序可以互操作,而无论它们运行于那种平台上。Windows NT 下的 DCE Version 1.0 包括两部分:Digital DCE Applications Developers Kit(应用程序开发库)。前者运行于客户机工作站并支持全部的 DCE 服务,包括 RPC(远程过程调用)、计时、安全、线索和单元(Cell)目录;后者允许开发人员编写符合 DCE 规则的 Windows NT 应用程序。

被 Digital 的工作组称之为 InfoBroker 软件的新目录服务,为用户存取数据库的名称以及数据库和应用程序定位提供了一种方法。支持 X. 500 目录标准意味着 InfoBroker 用户能与支持 X. 500 的其他目录连接,从而扩展了资源定位能力。InfoBroker 包括两个软件分别运行于 OSF/1 服务器上 and Windows 客户机上。InfoBroker 服务器版本 1.0 存储用户信息和资源,它或作为一个 LAN 环境中的一个本地工作组目录,或作为一个整个 X. 500 目录服务的网关。InfoBroker 客户机版本 1.0 是一个 Windows 应用程序,允许用户读写、浏览及查寻 InfoBroker 服

务器或 X.500 整个目录的信息。这两种软件都支持 TCP/IP 和 DECnet 协议。

对于 Pathworks LAN 的用户,DEC 增加了一个开发者工具库,允许用户开发基于各种中间产品如 DCE、InfoBroker、Digital 的 ObjectBroker ORB 和 DECmessageQ 的分布式应用程序。

Digital 开放了 IBM SNA 连接软件、DEC SNA3270 应用服务程序(允许用户编写 OpenVMS 和支持 3270 数据流的 OSF/1 应用程序),这使得 Digital 系统上的用户通过网关可以连接到 SNA 网上,以共享 IBM 系统上的数据。

Microsoft 鼠标降价

Microsoft 鼠标(PS/2 及串行鼠标)的价格从 109 美元降低到 85 美元,Microsoft BallPoint 鼠标从 125 美元降低到 109 美元。鼠标器是随着软件的图形化发展而逐渐普及应用起来的计算机辅助输入设备,其市场前景是很有潜力的。

台湾 UMC 推出 486 兼容芯片

台湾半导体生产厂商的领导者 UMC (United Microelectronics Corp.) 公司今年上半年在市场上推出了 486 兼容的微处理器系列,这一系列称之为 U5S。U5S 系列有三个型号,其时钟频率分别为 25MHz、33MHz 和 40MHz,提供了工业标准的 486 功能,芯片的管脚和 Intel 的 486SX 和 486DX 的插座兼容。UMC 公司称,U5S 系列产品由于其专有的内部结构,在同样的时钟频率下操作,和同档次的 Intel486SX 相比性能约高 30%,同时功耗低。该公司还称在明年 5 月份前将推出 50MHz、60MHz 和 100MHz 的 486SX2、DX2 及 DX3 级 CPU。

Microsoft 公司销售 MS-DOS 6.22

Microsoft MS-DOS V6.22 采用 DriveSpace 磁盘压缩工具代替了 MS-DOS 先前版本中提供的 DoubleSpace。同时新版本的 MS-DOS 还提供了两

个工具,一个是为 MS-DOS2.11 或以上版本的用户提供的 Upgrade,另一个是 Set-Up 它允许现行的 MS-DOS6.0 或 MS-DOS6.2 的用户升级到 MS-DOS6.22。

即插即用 SCSI 1.0 规范完成

在没有 SCSI 1.0 规范之前,要增加 SCSI 外设的用户需要打开计算机,设置电路板的跳线及开关。而即插即用 SCSI 1.0 规范允许 SCSI 设备生产厂商生产在采用即插即用操作系统的情况下能自动重构的外设。这一规范设计了一个单独的 SCSI 电缆连接器以挂接 SCSI 设备,由于它采用单电缆使得用户选错电缆的可能降低到最小。为了不打开 PC 机或外设的外壳,SCSI 规范还定义了 SCSI 总线的自动电子设备(Automatic Electrical Termination)。规范规定了一个 ID 分配协议(Assignment Protocol),它能自动解决单个 SCSI 设备的地址。

SNA Server 协同产品推出

由 DCA (Digital Communications Associates) 和 OpenConnect Systems 推出的新产品扩展了 Microsoft SNA Server 用户的连接选择。DCA 即将推出的 DCA IRMA Workstation for Macintosh 版本得到增强以提供通过 Microsoft SNA Server 对 3270 的完整连接。最近推出的 OpenConnect/TN3270 Server 允许任何一个标准的 TN3270 客户机通过 Microsoft SNA Server 与 IBM 的大型机对话。

ODBC SDK 2.0 面市

ODBC (Open Database Connectivity) 接口允许开发者通过一个单独的标准 API 读写大数量的不同的关系型或非关系型数据库管理系统 130 多个不同的应用程序。驱动程序和数据库服务的提供者认知了 ODBC,并正在提供 140 多个兼容的应用程序和驱动程序。ODBC SDK 2.0 是为需要数据库读写的 C 语言应用程序的开发者设计的。ODBC SDK 2.0 包括下述几方面的增强:

支持 Win32 API 和 Windows NT; 光标库提供了对所有 ODBC 驱动程序的更一致的光标滚动功能; 支持所有新功能的样本程序库; 好的测试和调试工具; ODBC Desktop Database Driver, 其中包括 FoxPro、Access、dBASE、Paradox、Btrieve、Excel 和格式化文本文件的驱动程序; 用于用户驱动程序和系统配置的新的驱动程序安装库。

Lotus 发展多机种邮件系统

Lotus 正发展一名为“Lotus Communication Server”的产品, 这种产品能使现有的语音电话、传真机、掌上型计算机均具有与 cc:mail 电子邮件系统通信的能力。上述邮件通过 Lotus Communication Server 的处理信息可设定为自动和手动两种模式, 自动模式即将信息自动转换成节点前端可读取的文件格式, 如传真文件转换成图形文件、电话留言转换成语音邮件, 而当前端用户切换到手动状态时, 可设定为无线或传真状态。

GJN 公司推出集线器新产品

Grand Junctions Networks 公司推出了新的集线器产品 FastLink 10/100AG。这种集线器支持 24 个 10Mbps 交换(Switched)Ethernet 端口和一个共享 Ethernet 端口。在 100Mbps Fast Ethernet LAN 上, 可通过 4 个独立的端口读写服务器。

IBM 推出新集线器

IBM 推出的 Ethernet Streamer Switch 集线器提供了 8 个 Ethernet 连接端口, 提供全双工功能和 Etherstreamer 32 位 Ethernet 网卡使用可达 20Mbps 的传输速率。

Chicago 远程读写网络功能

Shiva 在开发远程读写网络功能方面将为 Microsoft 提供帮助, 这种远程读写网络的能力将集成

于 Chicago 之中, 支持 NetBEUI、IPX/SPX 和 TCP/IP 传输协议。Chicago 的拨号驱动程序将基于点对点(Point-to-Point Protocol)协议。

VB3.0 又增新功能

Visual Basic 3.0 为应用程序提供了读写以 Access1.1 格式存储的数据文件的能力, 但不能读写以 Access2.0 格式存储的数据文件。Compatibility Layer 软件突破了 Visual Basic3.0 程序采用 Access 2.0 的 Jet 2.0 数据库机读写数据的限制。

Windows NT 下的 Excel 和 Word

Excel 和 Word 利用 Windows NT 32 位结构的优点突破了 16 位 Windows 应用程序的一些限制。Excel 和 Word 的 32 位版本支持长文件名(高达 255 个字符)、用户设置和大文件处理。

Microsoft 推出中文版 Word

Microsoft 中文版 Word 5.0 是 Windows 环境下字处理应用程序, 它包括了同类英文版的所有功能, 支持 16 位双字节处理, 中文版 Word5.0 for Windows 增加的新功能包括拖放功能。此外, 制表也非常容易, 具有完善的中文界面, 支持汉字的下拉式菜单, 在中文 Windows3.1 下的用户, 可以非常自由地在 Word 的文件或 Excel 电子表格中, 将汉字及英文文本最大化, 中文版 Word 和 Excel 与 Microsoft 的 Office 的其它应用程序可以联合使用。这些应用程序包括 Microsoft PowerPoint 和 Microsoft Mail。

Windows 环境下的在线会议系统 TeamTalk

TRAX 公司推出的 TeamTalk 在广域网络上联接时, 可将任意服务器作为暂存装置, 同时支持

MAPI(Messaging API)、VIM(Vendor Independent Messaging)、MIME(Multipurpose Internet Mail Extensions)等多种电子邮件系统,以利于不同操作系统电子邮件的各种用户实时或分时方式的在线会议,TeamTalk与现有的由IBM或Intel公司推出的产品有所不同,即在线支持的用户数不受限制。

HP 推出 DCE 工具库

HP公司最近推出了通过预定义对象加速编程的工具库。面向对象的DCE/9000建立于一个包含约20个预定义对象的C++类库。它是沟通对象和DCE应用程序的桥梁。通过预定义的对象,用户可以免去编写400个DCE应用程序接口的复杂工作。

Chicago 具有多种界面

实时多任务和具有OS/2、Motif、Mac及Windows NT用户界面是Microsoft Windows 4.0即Chicago的一个特点。Chicago是Microsoft公司基于新版本DOS(7.0)的窗口操作系统。Chicago和MS-DOS的关系和先前的Windows产品有所不同,Chicago的整个安装过程包括DOS在内都在Windows环境下进行。新的用户界面使得用户运行应用程序和应用程序之间的切换更加方便。Chicago中的用户界面是三维的,同时实现了Word 6.0及Excel 5.0中的动态对话框。Chicago采用与OS/2、Windows NT及UNIX同样的优先多任务(Preemptive multitasking),支持DOS、Windows和Windows NT的应用程序。新引进的即插即用的功能使得用户安装外设如CD-ROM演唱机(Player)更加方便。集成于Chicago中的OLE 2.0使得原有功能更快、更可靠。

挂接式打印机共享器问世

北京向宇计算机公司最近推出了一种新型的打印机共享器。这种共享器采用总线结构,突破了传统打印机共享器的一转几的概念。用户增加共享打印机的计算机时只需增加一条打印机共享电缆。向宇

打印机共享器由两部分组成:专有的和计算机并口相连的控制器和与打印机相连的标准电缆,全套共享器小巧、灵活无需专业知识即可安装。随着用户计算机数量的增加,用户只需单独增加一套打印机共享器。

一种新型的反病毒软件问世

最近北京市帝霸计算机技术研究所推出了一种新型的反病毒集成软件系统。该系统引进了开放式概念,使得计算机病毒的检测与消除更加方便、灵活。传统的反病毒软件对于新型病毒一般采用重新编制程序的方法,这种方法升级速度慢,对厂商的依赖性强。帝霸反病毒软件采用独特的数据结构的方法描述计算机病毒,通过软件自身的强大的库维护功能,用户可以随时对软件进行升级。为了更广泛地服务于用户,最近,本杂志社和该研究所合作,该软件的升级信息将通过本刊独家公布,本刊读者可通过本刊得到该软件的病毒检测升级信息,从而能快速检测国内新出现的计算机病毒。

Microsoft 让多种用户共享 OLE 对象

分布式OLE将集成于Windows NT的面向对象的新版本Cairo,Microsoft对于分布式OLE的目标是允许OLE对象被更多的用户共享,甚至非兼容系统间的用户。分布式OLE软件开发库将于今年晚些时候或明年初出台。

Microsoft相信,分布式OLE将改变应用程序创建和操作的方法。在Microsoft开发系统之下,应用可被分离成对象,每一个对象可以运行于不同的计算机上。一个计算机网络对用户而言就象一台计算机,而不用去管入网的计算机系统是什么。有了分布式OLE远程系统可被紧密地联系在一起;当加入新的数据时,系统可自动修改,如由于非本地子公司的数据库的变化本地总公司的电子表格数据会自动修改。

今天的OLE兼容系统可不加修改地利用分布式OLE的能力。为分布式OLE专门编写的应用程序可以利用一个分布式环境。

SKILL

扩展 Chicago 外壳

在

Microsoft Windows 的下一版本 Chicago 里,应用系统可使外壳在许多方面得以扩展。

扩展的外壳加强了外壳的基本功能,为处理文件对象或其他信息提供了灵活的选择。该功能使应用系统简化了浏览文件系统和网络的操作,同时用户在查找文件系统时,可以轻松地获得处理该文件的工具。

例如,一个扩展的外壳能给每一文件分配一个图标或为文件加特殊的命令。这些命令被加在外壳所显示的上下文菜单里(当用户通过右鼠标按钮选中对象时),当用户通过左鼠标按钮选中对象并打开文件菜单时,命令被加在文件菜单中。

Chicago 外壳扩展设计基于 OLE 2.0 版的 Component Object Model, Chicago 通过界面获得对象,应用系统以扩展外壳的 DLL 方式操作那些界面。类似于 OLE 2.0 里的 InProc Server DLL。

一、定 义

在涉及具体的扩展外壳之前,我们先来看一些重要的定义:

1. 一个文件对象是外壳中的一项。最熟悉的文件对象是文件和目录。然而,一个文件对象实际上可能不是文件系统的一个部分。例如:打印机、控制窗、网络组、服务器和共享器都是文件对象。

2. 文件类就是文件对象的类型。每个文件对象是一种文件类的成员,这个文件类也包括处理该类的文件和程序。例如:文本文件及 Microsoft Word 文件就是两例,每种文件类有特殊的扩展外壳。外壳在对象所处的文

件类的基础上安装这些扩展。

3. “处理器”即操作一个扩展外壳的程序。

二、登记一个扩展外壳

所有扩展外壳都登记于 Windows 的登记库。每个处理器(handler)都必须在登记库中的 HKEY _ CLASSES _ ROOT \Clsid 关键字下登记它的类标识符。Clsid 关键字是 DCE RPC 的唯一标识(GUID)如:{00020810-0000-0000-C000-000000000046}由 GUIDGEN 工具产生,通过该类型标识符,处理器增加一个 Inproc Server 32 以给出处理器的 DLL 的位置。

应用程序生成并维护如电子表格、文字处理、数据库等文件,数据库文件通常在登记处登记两项,相关文件项和关键字名。

相关文件项将文件扩展名和程序标识联系起来。例如:一个文字处理应用系统在 HKEY _ CLASSES _ ROOT 的下面做如下登记:

```
HKEY _ CLASSES _ ROOT
.doc = AWordProcessor
```

关键字的名称(.doc)代表文件扩展名,而关键字的值(AWordProcessor)标明了包含那些处理文件扩展名的程序的信息关键字名称。关键字名称值是第二个登记项,它由处理文件的程序生成。

登记处的外壳部分的命令加在与它的类型有关的文件的上下文菜单(Context menu)中。PrintTo 项在用户将文件调到打印机上时也常常被使用。为了避免与其他类发生冲突,你必须用真正的 GUID。

新的 Shellex 关键字包含有关外壳用来将一扩展外壳与一文件类型联系起来的信息。外壳也使用其他几个特殊关键字如 *、Folder、Drives、Printes 以及网络上的关键字以在 HKEY _ CLASSES _ ROOT 下查找扩展外壳。

1. * 关键字用来登记处理器,该处理器在生成任一文件或每个文件对象的上下文菜单或属性表(Property sheet)时将被外壳调用。

2. 外壳通过关键字 Folder 让应用程序为文件系统的目录登记扩展外壳。一个应用系统能向为 * 关键字登记处理器一样的方法登记上下文菜单处理器, Copy-Hook 处理器和属性表处理器。还有一个额外的处理器是拖放处理器,它只适用于关键字 Folder 和 Printers。

3. 关键字 Drivers 允许跟 Folder 关键字同样的登记,但只允许对根目录进行调用,例如:C:\。

4. 关键字 Printers 允许跟关键字 Folder 同样地登记,但使用额外的处理器来处理打印机事件:打印机的删除或移动(通过 Copy-Hook 处理器)、打印机参数(用参数表处理器和上下文菜单处理器)。

三、五方面的外壳扩展

五个外壳的扩展有:

1. 上下文菜单处理器

一个应用系统通过一个上下文菜单处理器 I-ContextMenu 来达到这样的效果;当用户通过鼠标右按钮选择一个文件对象时可以向外壳所显示的下拉菜单中增加菜单项。它的作用是为这个文件类型自动增加动词。

在登记处的应用系统信息域里通过关键字 Shellex 登记上下文菜单处理器。ContextMenu-Handlers 列出包含每个上下文菜单的 Clsid 的子关键字名。

2. 拖放处理器

拖放处理器实现 IContextMenu 接口。实际上,拖放处理器就是一个简单的上下文菜单处理器,它改变用户以鼠标右键拖放文件对象时弹出的菜单。因为这类菜单是拖放菜单,在菜单中增加项的外壳扩展称之为拖放处理,拖放处理器同上下文处理器工作方式完全相同。

3. 图标处理器

Chicago 外壳允许应用程序自己改变文件类型的表征图标。用户可以两种方式文件定义图标。

第一种方式是(最简单的)为所有类型文件定义

一种图标。定义时可在登记库中增加一个 Default Icon 关键字,关键字的值规定了图标表示的可执行命令。如:

DefaultIcon=C:\Mydir\Myapp.exe,1

这与 Windows 3.1 处理缺省图标是一致的。使用一个类图标的优点是不需要编程;外壳为这一类显示图标。

Chicago 为关键字 DefaultIcon 赋一新值:%1。该值表明这一类型的每一文件都可以有不同的图标。应用系统必须为文件类型提供一个图标处理器并给应用程序关键字 Shellex 增加另外一项:IconHandler,一个应用系统只能有一个 IconHandler 项。关键字 IconHandler 的值就表示了图标处理器的 Clsid。要得到标准化的图符,应用系统必须提供一个能执行 IExtractIcon 的精简图标处理器。

4. 属性表处理器

外壳得以扩展的另一途径是通过使用规范化的属性表。当用户选择一个文件的属性时,外壳便显示一个标准的属性表。

如果被登记的文件类型有一个属性处理器,外壳就会让用户读写处理器所提供的额外表。属性表处理器执行 IShellPropSheetExt 接口。

属性表处理器以 Shellex 关键字登记,PropertySheetHandlers 关键字下列出了所有子关键字(它们包括了每个上下文菜单处理器的 Clsid)。

5. Copy-Hook 处理器

我们可登记一个 Copy-Hook 处理器,它在移动、复制、删除、改名一个文件对象时便会被 Chicago 调用。Copy-Hook 只是对文件夹操作作核对,Chicago 收到核对信息后,再去作实际操作(移动、复制、删除、改名)。Chicago 并不通知 Copy-Hook 操作的结果,因而 Copy-Hook 无法监控文件夹对象的状态。

一个文件夹对象可以有多种 Copy-Hook 处理器,而 Copy-Hook 处理除了标准的 IUnknown 方法外,还有另一种方法 CopyCallBack。

Chicago 在拷贝、移动、更名或删除一个文件夹对象之前调用 CopyCallBack。这个调用返回一个整数值,代表 Chicago 是否应完成操作。Chicago 将调用每一个被登的 Copy-Hook 处理器直到要么所有处理器都被调用,要么任一处理器返回 IDCANCEL。处理器也可返回 IDYES 表明操作要执行,IDNO 表明操作不执行。

(本文英文资料由 Microsoft 提供)

在 DOS 5.0、6.0 中运行 Novell 网上的 MIS 系统

□曹国钧

M

S-DOS 5.0、6.0 是美国 Microsoft 公司推出的优秀 DOS 操作系统,它的丰富内存管理命令使广大计算机用户爱不释手,因此,大多数用户已将自己的 DOS 操作系统升级为 DOS 5.0 或 DOS 6.0。但是,Novell 网在 DOS 5.0、6.0 上不能直接运行,致使有的用户仍然使用 MS-DOS 3.30 或 3.31 版本。在 MS-DOS 3.X 版本中运行 Novell 网,若调入汉字系统和 MIS 系统,则往往出现“内存空间不够”等情况。笔者根据在 Novell 网络中开发汉字 MIS 系统的经验,提出以下四种方法解决在 MS-DOS 5.0、6.0 下运行 Novell 网的问题。

1. 将 NetWare 3.11 版本中的 SHELL 程序 NET3.COM 或 NET4.COM 在 MS-DOS 5.0、6.0 中模拟成 3.X 和 4.X 的运行环境

MS-DOS 5.0、6.0 中提供了模拟版本号的驱动程序 SETVER.EXE,在 CONFIG.SYS 中增加一行:

```
DEVICE (HIGH) = C:\DOS5\
SETVER.EXE
```

在命令行上键入:

```
C:\NET>SETVER NET3.COM 3.31
或 C:\NET>SETVER NET4.COM 4.01
```

重新启动 DOS 系统后,则在 MS-DOS 5.0、6.0 上就能运行 NET3.COM 或 NET4.COM。

2. 修改 NET3.COM 或 NET4.COM, 使之对版本号进行检查

以 NET3.COM 为例,在命令行发出如下命令:

```
C:\NET>DEBUG NET3.COM
-N NET3.COM
-L 100
-r
AX=0000 BX=0000 CX=A4AA DX=0000 SP=FFFE
BP=0000 SI=0000 DI=0000
DS=0C54 ES=0C54 SS=0C54 CS=0C54 IP=0100
NV UP EI PL NZ NA PO NC
0C54:0100 E99FA4 JMP A5A2
-S 0 A5AA B4 30 CD 21
;查找版本号检测程序段
0C54:8D5D
-U 8D5D 8D72
0C54:8D5D B430 MOV AH,30
0C54:8D5F CD21 INT 21
;版本号检查功能调用
0C54:8D61 3C03 CMP AL,03
;是 3.X 版本吗?
(*)0C54:8KD63 740D JZ 8D72
0C54:8D65 BA0995 MOV DX,9509
0C54:8D68 90 NOP
0C54:8B69 B409 MOV AH,09
0C54:8D6B CD21 INT 21
;Incorrect DOS Version 信息显示
0C54:8D6D B8024C MOV AX,4C02
0C54:3D70 CD21 INT 21
0C54:8D72 8AC4 MOV AL,AH
-A 8D63 ;修改(*)行的内容,
0C54:8D63 JMP 8D72
0C54:8D65
-N NET5__6.com
-W
-Q
```

经以上修改的 NET __ 6.COM 就能直接运行于 MS-DOS 5.0、6.0 系统环境中。不过,用 NET3.COM 修改后的 NET5 __ 6.COM 的工作站 SHELL 有一个缺陷,就是在 Novell 网中执行 DIR 命令时会出现死机的现象,这是因为 DOS 3.X 与 DOS 5.0 有差异之故,用户可用运行于 DOS 4.X 上的 NET4.COM 进行修改,其修改方法类似于 NET3.COM。

3. 使用 DR-DOS 6.0 提供的 NETX.COM

在 DR-DOS 6.0 系统 2# 盘的子目录 NetWare 中有 NETX.COM 文件,它可直接在 DOS 5.0、6.0 上正确运行。形成在工作站上使用的 IPX.COM,必须用该子目录中的 IPX.OBJ 来连接,否则,将会出现不可预测的结果。

4. 使用 Windows 3.1V 提供的 NETX.COM

在 Windows 3.1V 的源盘 2# 中有 NETX. __CO 文件,该文件提供 Novell 网能直接运行于 MS-DOS 5.0 以上版本环境中的功能。因 NETX. __CO 是压缩文件,不能直接使用,应用 MS-DOS 5.0 提供的解压程序 EXPAND.COM 将 NETX. __CO 解开,即:

```
C:\NET>EXPAND A:\NETX. __CO
NET5 __6.COM
```

则 NET5 __6.COM 就能运行于 DOS 5.0、6.0 的 Novell 网的环境中。

若工作站点在 Novell 网中运行 Windows 系统,则站点中的 IPX.COM 必须用 Windows 系统 2# 盘中的 IPX.OBJ 来生成连接,否则,将会出错或死机。

一般地,用 FoxBASE+或 FoxPro/LAN 开发的汉字 MIS 系统需要较大的内存空间,但启动 Novell 网需 60KB 的内存,若汉字系统(包括一些辅助程序和实用程序)占用 150KB 以上,则留给应用程序的空间仅 400KB,这不能满足应用程序的要求。用 DOS5.0 对系统进行合理配置后,可留给应用程序 560KB 的内存空间。下面就是一种优化系统配置程序(386 以上微机、4M 以上内存、EGA/VGA 显示器)。

(上接第 22 页)

```
if(( spt = fopen( spt__name, "rb" )) != NULL)
{ fseek( spt, 34L, SEEK__SET );
  fread( &size, sizeof(size), 1, spt );
  size.x /= 8;
  picture = ( unsigned char * )malloc( size.x * size.y );
  if( picture == ( unsigned char * )0)
    return 0;
  fseek( spt, 64L, SEEK__SET );
  fread( picture, 1, size.x * size.y, spt );
  return 1;
}return 0;
}

void show__spt( int x, int y, int w, int h )
{ int l, i, t, lt;
  setvideomode( PIXEL );
  outp( 0x3c4, 0x02 ); outp( 0x3c5, 0x0f );
  if( w > 80 ) w = 80;
  if( h > 640 ) h = 640;
  for( l = 0; l < h; l++, y++ )
```

CONFIG. SYS 的内容:

```
DEVICE=C:\213\ANSI.SYS
DEVICE=e:\cgj\wins\himem.sys
device=E:\CGJ\WINS\EMM386.EXE I=B000-B7FF
FRAME=E000 NOEMS

dos=high,umb
SHELL=C:\COMMAND.COM/E,400/P
DEVICEhigh=D:\GYS\SMARTDRV.SYS 1024
FILES=30
buffers=10
STACKS=0,0
```

该配置可留给用户约 620KB 内存空间,640—1024KB 之间可用 200KB,可实现汉字系统的“零内存”的想法(笔者已将 CCBIOS 2.13H、SPDOS 5.0—6.0F 汉字系统全部装入 UMB 中)。这样,在 Novell 网络中运行 MIS 系统就不会出现“内存不够”的情况。这是使用了 MS-DOS 5.0、6.0 的丰富内存管理命令的结果。

因为 Windows 具有多任务处理应用程序和控制微机所有内存空间的能力,因此,要从根本上解决 MIS 系统运行时出现“内存不够”的问题,可以在服务器或站点中运行 Windows 3.1,用户将汉字系统和 MIS 系统全部装入扩展内存(XMS)或扩充内存(EMS)中运行(注意:因 SPDOS 6.0F 采用 286 方式编程,故它不能直接在 Windows 中运行,可让 SPDOS6.0F 在进入 Windows 之前运行,其他可放到 Windows 中运行),把它们作为 Windows 的一个任务,用户的工作只是设 Windows 应分配多少 XMS 或 EMS 给汉字系统和 MIS 应用程序,其余的内存协调与管理由 Windows 完成,用户无需为内存不够而费脑筋。

SKILL

```
{ t = size.x * y + x;
  lt = l * 80;
  for( i = 0; i < w; i++ )
    video[lt++] = picture[t++];
  }getch();
  setvideomode(CHAR);
}

void close__spt(void)
{ free(picture);
  fclose(spt);
  size.x = 0; size.y = 0;
}

int setvideomode(int mode)
{ union REGS regs;
  regs.h.ah = 0;
  regs.h.al = ( unsigned char )mode;
  int86( 0x10, &regs, &regs );
  return mode;
}
```

SKILL

实用简单的 PATH 编辑工具

□ 臧峥嵘

D

OS 为了能使用户在任意目录下都可执行程序,提供了一个 PATH 环境变量,当在命令行下执行程序时,如果当前目录下没有该程序,DOS 就到 PATH 环境变量定义的目录中去寻找,直到找到并执行该程序或所有目录都已找遍为止。对于常用程序的目录可在 AUTOEXEC. BAT 程序中预先设置,但有时为了某些目的,常常需要暂时在 PATH 中增加或删除某个目录,此时 DOS 就显得很不方便,虽然可用 C 语言等写一个程序,但毕竟繁琐,有无更简单的办法呢?本文介绍一个用批处理实现的例子,一则供大家参考,二则也想说明批处理程序有相当强的功能。

该程序使用方法如下:

PE+目录 在 PATH 尾部增加目录

PE& 目录 在 PATH 首部增加目录

PE-目录 从 PATH 中删除目录

请看程序清单,增加目录部分比较简单,只要在相应位置加上指定目录即可,而删除目录就很复杂。由于用户在命令行输入要删除的目录时有可能用的是小写字母,所以首先要将小写字母转换成大写字母,这是通过将该目录设为 PATH 的值来实现的,该值被暂时放在变量 PE __ DROP 中,接下来程序调用自己,并将 ' 作为参数 1,原来的环境变量作为参数 2 传给自己。因此第二次执行 PE. BAT 时,程序就直接跳到标号 DoDrop 处继续执行。在删除目录时,程序利用了 DOS 命令行参数的一个特性,这就是除了用空格分开参数外,也可以使用分号达到同样的目的,这一点很重要。基于这一特性,程序使用 SHIFT 命令依次取 PATH 中的目录,并与

要删除的目录进行比较,如果不是要删除的目录则仍旧放在 PATH 中,否则就从 PATH 中去掉该目录,一直到 PATH 中的目录都比较完为止。这时在 PATH 中就不再含有要删除的目录了。

由于程序在进行删除操作时可能会用到较大的环境变量空间,因此建议在配置文件 CONFIG. SYS 中加入下列命令行:

```
SHELL=C:\DOS\COMMAND.COM
C:\DOS /p /e:1024
```

以确保环境变量不会溢出。

批处理程序清单如下:

```
@echo off
if '%1'==' ' goto DoDrop
echo.
if '%1'=='+' goto Help
if '%1'=='&' goto AddEnd
if '%1'=='&' goto AddBegin
if '%1'=='-' goto Drop
goto Help
:AddEnd
PATH=%PATH%;%2
echo %2 added at end of PATH.
goto END
:AddBegin
PATH=%2;%PATH%
echo %2 added at begin of PATH.
goto END
:Drop
set '=%PATH%
PATH=%2
set PE __ DROP=%PATH%
:Now variable PE __ DROP contains %2 capitalized
PATH=% %
set '=%
set PE __ REM=
:Restart, passing PATH as a parameter
%0 '%PATH%
```

(下转第 27 页)

FIND 命令的功能扩充及应用

□贺烈华

D

OS 中的 FIND 命令是一个过滤程序,它可以在一组文件中筛选一个字符串,如要在一个文件中查找一个字符串是很方便的。但由于不能使用通配符 * 和?,在查找数量较多的文件时,需将文件名一一列出,颇感不便;倘若要从一组文件中查找一组字符串时,FIND 命令就更显得先天不足,力不从心。那么,能否寻找一种比较方便的方法来查找一组字符串呢?

如果将需查找的一组字符串放在一个文件内,然后把这个文件改写成一批文件,使其依次按各字符串对目标文件执行 FIND 命令,并把结果写到一个文件中去,那末我们就可达到对某一文件过滤一组字符串的目的。把文件名作为变量,在循环中调用这个批文件,则可对一组文件过滤一组字符串,并且可处理带通配符的文件名。

具体做法如下:

1. 为了便于把存放字符串的文件改写成批文件,每一字符串均以单引号为前导符、双引号为结束符。如要在文件中查找 IF、ELSE、ENDI 等字符串,假定这些字符串存放在文件 CF 中,则上述字符串应在文件 CF 中写成:

'IF'

'ELSE'

'ENDI'

2. 对一个文件过滤一组字符串,若某行同时含有几个被查找的字符串时,过滤结果则出现重复。为了避免重复,可先对上次过滤结果进行一次预处理,利用 FIND /V 功能将本次要过滤的字符串从上次结果中去掉,然

后再对目标文件进行本次过滤。

例如要从文件 FILE 中查出含 IF、ENDI 等字符串的语句,由于语句 ENDIF 既含 IF 字符串,又含 ENDI 字符串,因此过滤结果含有重复语句。假设先查出 FILE 文件中含 IF 的语句,结果写在 F3 文件中,在过滤 ENDI 字符串时执行如下命令:

```
FIND/V "ENDI" F3>F4
```

```
FIND "ENDI" FILE>>F4
```

```
COPY F4 F3
```

则可保证过滤结果在 F3 文件中的唯一性。

3. 对一个文件每次过滤一组字符串,匹配的语句按文本文件中字符串出现的次序排列,即首先显示第一个字符串在文件中的所有语句,再显示第二个字符串的匹配语句……,依次类推。为了使结果按目标文件中的相对位置排列,可使过滤结果带上行号,然后按行号排序,便能得到正确的结果。尽管 FIND 命令使用 N 参数便能显示行号,但我们不能用 FIND /N 来获得行号。因为 SORT 命令是按字符的 ASCII 码而不是按数字的值排序的,因此通过 FIND /N 过滤后程序中的第 2 行行号为[2],排序后它将排列在所有行号以 1 开头(比如第 1000 行行号为[1000])的后面。经过多次摸索,发现使用 EDLIN 的显示行号排序,可得到正确结果。编写文件 BF5 内容如下:

```
1,9999P
```

```
E
```

键入命令 EDLIN FILE<BF5>FILE.L,可使文件 FILE.L 成为带行号的 FILE 文件的副本,供筛选、排序用。

4. 对存放字符串文件的改写可由 EDLIN 命令来实现。编写文件 BF4 内容如下:

```
1,99R'^Z CALL BF2"
```

E

其中 ^Z 表示按住 Ctrl 键的同时按 Z 键。以 BF4 作为 EDLIN 的输入,使得改写工作自动进行,毋须人工干预。

5. 扩充 FIND 功能的批文件命名为 BF.BAT, 其使用格式为: BF FILE0 FILE1 FILE2.....

其中:FILE0 为存放字符串的文件,FILE1、FILE2 为一组目标文件,可含通配符(程序清单附后)。FIND 命令的功能经过扩充,在编程中有许多用处:

(1) 检查语句是否匹配

FoxPLUS 的源程序要求 IF、ELSE、ENDIF、DO WHILE、ENDD 等语句匹配。如要检查 MAIN.PRG 文件中 IF 语句是否配对,可使用 BF 批命令。首先要编写一文本文件,如命名为 IF,内容为:

```
'IF"
```

```
'ELSE"
```

```
'ENDI"
```

然后键入命令:

```
BF IF MAIN.PRG
```

即显示 MAIN.PRG 所有含 IF、ELSE、ENDI 的语句,且带有行号,由此可方便地检查 IF 语句是否配对。如需对所有的 PRG 文件进行检查,只需键入:

```
BF IF *.PRG
```

同样可检查 WHILE、CASE 语句的配对情况,只要按要求编写不同的文本文件即可。如要检查语句嵌套是否正确,可把 IF、WHILE、CASE 语句的关键字编到同一文本文件中。

(2) 检查数据库调用情况

编写文本文件 USE:

```
'SELE"
```

```
'USE"
```

```
'CLOS"
```

键入命令:

```
BF USE *.PRG
```

可检查各程序中各数据库的调用情况。这对于应用系统较大、源程序繁多,这些源程序又是由几个人合作编写的情况下很有使用价值。

(3) 了解子程序的调用情况

FoxPLUS 使用 DO 语句调用子程序,过滤一个字符串,只要用 FIND 命令即可。但考虑到输入文件名方便,将 DO 字符串编入文本文件 DO 中,键入命令:

```
BF DO *.PRG
```

可把所有含 DO 字符串的语句全部显示出来,但结果中含有 WHILE 和 CASE 等语句,显然不符合要求。因此须对结果进行二次过滤,把 WHILE、CASE 语句去掉。这可通过管道命令来实现:

```
BF DO *.PRG|FIND/V "WHIL"|FIND/V  
"CASE"|FIND/V "ENDDO"|FIND/V "  
DOHI">FD0
```

应用系统中子程序的相互调用情况便记录在文件 FD0 中。

(4) 提供编制流程图的依据

如果把上述文件综合到一个文件 F3 中,使用 BF 命令,便可得到一个包括各分支结构、循环结构、数据库及子程序调用在内的文件,作为编制流程图的依据。F3 文件内容如下:

```
'DO"
```

```
'IF"
```

```
'ELSE"
```

```
'ENDI"
```

```
'CASE"
```

```
'OTHE"
```

```
'ENDC"
```

```
'ENDD"
```

```
'SELE"
```

```
'USE"
```

```
'CLOS"
```

BF.BAT 源程序清单如下:

```
echo off  
del f1  
del f2  
del f3  
copy %1 bf2.bat>f7  
edlin bf2.bat<bf4>f7  
: start  
  shift  
  if "%1"==" goto end  
  for %%i in (%1) do call bf1 %%i  
  goto start  
:end  
type f1|find/v "-----" |more
```

bf1.bat 源程序清单如下:

```
echo %1>f3  
copy %1 f5>f7  
edlin f5<bf5>f2  
call bf2  
type f3|sort>>f1
```

bf3.bat 源程序清单如下:

```
find/v %1 f3>f4  
find/n %1 f2>>f4  
copy f4 f3
```

上述程序在 MS-DOS3.3 下运行通过。 SKILL

也谈 DOS 重定向功能

□冯殿秀

贵

刊《创刊号》软平台栏目中“巧用 DOS 的 I/O 重定向功能 10 例”一文介绍:如果想要记录每次使用计算机的日期和时间,以便加强用机管理,采用在自动批处理文件 AUTOEXEC.BAT 中加入下列命令的方法即可实现:

- (1)ECHO+>temp. abc
- (2)ECHO+>>temp. abc
- (3)TYPE temp. abc|DATE>>ABC
- (4)TYPE temp. abc|TIME>>ABC
- (5)DEL temp. abc

其中,第一行和第二行将执行 DATE 和 TIME 命令时要手工键入的“回车”重定向到一个临时文件 temp. abc 中,第三行和第四行则利用 DOS 管道功能将 DATE 和 TIME 命令的输出信息重定向到文件 ABC 中,这样在每次启动计算机以后可以执行命令 C: TYPE ABC 来查看每次使用计算机的日期和时间。

笔者认为,上述程序执行到(3)时,计算机将处于等待状态,停步不前。计算机给出屏幕不显示的信息:

```
Current date is Mon 8-08-1994
Enter new date (mm-dd-yy):
```

因为该命令只是把正文“DATE”追加到 ABC 中,等待输入新日期。此时键入 Ctrl-Break 计算机中断。键入“Y”,停止批处理文件。键入“N”,计算机执行下一条批处理命令。为了解决这一问题,要在该管道中增加一个筛选程序 MORE:

```
ECHO|MORE|DATE>>ABC
```

当这样使用时,MORE 命令会把一个回

车符加到上述 DATE 命令的输出之后,这是因为 DATE 命令并不将执行情况显示在屏幕上。这样,文件 ABC 就追加了如下的正文:

```
Current date is Mon 8-08-1994
Enter new date (mm-dd-yy):
```

然后,继续执行下一条批处理命令。上述程序执行到(4)时,也会遇到同样问题,采取同样措施处理即可。

综上所述,加在 AUTOEXEC.BAT 中的程序应改成如下形式:

```
ECHO OFF
ECHO+>temp. abc
TYPE temp. abc>>ABC
TYPE temp. abc|MORE|DATE>>
ABC
TYPE temp. abc|MORE|TIME>>
ABC
DEL temp. abc
ECHO ON
```

即可实现记录每次使用计算机的日期和时间,以加强计算机管理的目的。以上所述仅作抛砖引玉,以供同仁磋商。

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

编写基于位平面的图象处理函数

□牟达森

尽

管目前常用的编程语言如 C、Pascal 等都具备了处理图形的功能,并有丰富的函数库,但它们都没有提供基于位平面的操作函数。对位平面的操作要涉及 EGA/VGA 的内部结构,复杂而易出错,可又是经常要碰到的,比如在一些游戏程序中,常要对某一个或几个位平面的独立操作来实现动景与静景的分离,以达到逼真的效果。如果把对位平面的操作写成通用的函数放到一起,组成一个函数库,既可大大减少重复劳动又可避免差错。基于此,笔者谈谈基于位平面的图象处理函数的编写和使用,并给出应用实例:单一位平面上图象块的获取(GetBitImage)和重现(PutBitImage),程序被写成 C 语言调用方式,改变参数传递接口,也可供其他语言调用。

一、EGA/VGA 图形控制系统

1. 操作方法

EGA/VGA 所有的图形功能都是通过一系列寄存器来控制的。EGA/VGA 有三大控制器:时序控制器(地址端口 3C4H,数据端口 3C5H)、图形控制器(地址端口 3CEH,数据端口 3CFH)和属性控制器(地址端口 3C0H,数据端口 3C1H),此外还有 CRT 控制器和通用寄存器。对时序控制器和图形控制器访问的方法为:先向控制器的地址端口送出欲访问寄存器的索引号,然后向该控制器的数据端口写入或读出(EGA 的多数寄存器只能写不能读,而 VGA 的都可以读写)有关数据,而对属性控制器的读写操作则较复杂,它只占用一个 I/O 口地址,写到该端口

的值是索引号还是属性数据是根据 EGA/VGA 内部的一个触发器(Filp-Flop)的状态来决定的。通过对端口 3BAH(单色)或 3DAH(彩色)进行 I/O 读(IN)可使该触发器指向索引寄存器,紧接着一个端口输出指令、输出索引号,触发器随之翻转指向数据寄存器,接着再用一个 OUT 指令就可把数据送到相应寄存器中,之后,触发器在索引寄存器和数据寄存器之间重复翻转。

2. 部分寄存器的功能介绍

(1) 图形控制器

2 号颜色比较(Color Compare)寄存器和 7 号颜色无关(Color don't Care)寄存器:当选用读模式 1 时,Color Compare 寄存器的值将与水平相邻的八个像素值比较,相同为 1,不同为 0,结果送到主机中。而通过设置 7 号寄存器的值可使全部或部分显示平面的数据参加比较,当该寄存器低 4 位中的某一位为 0 时,表示相应的那个位平面在比较中不予考虑(参见图 2)。

3 号数据移位/功能选择(Data Rotate/Function Select)寄存器:该寄存器包括两个独立的部分:位 0~2 为 Data Rotate,其值决定处理器把数据写到显示平面前右移的位数。而位 4、3 为 Function Select,在 EGA/VGA 的内部有一个 32 位的锁存器(latch)用于暂时保存从四个显示平面中读出的 4 字节数据(每平面 1 字节),CPU 在往显存的某地址写数据时,EGA/VGA 先把该地址的像素搬到 latch 中,然后与 CPU 送来的数据进行 Function Select 所指定的操作,结果再送回显存。位 4、3 的值所对应的操作如下表:

位 4	位 3	功 能
0	0	取代(Replace)
0	1	与(AND)
1	0	或(OR)
1	1	非(NOT)

4 号读位面选择寄存器(Read Map Register): 此寄存器值决定对哪一个位面进行操作, 此寄存器仅用于读模式 0(参见图 1)。

5 号模式寄存器(Mode Register): EGA/VGA 有 2 种读模式(模式 0 和 1)和 3 种写模式(0、1、2), 它们是相同的, 而 VGA 还多一个写入模式(模式 3)。该寄存器的位 3 指定读模式, 而位 1、0 则选择写模式。

8 号位屏蔽(Bit Mask)寄存器: 显示存储区分为 4 个位平面, 每个位平面的每个像素由 1 位表示, 显存中一个位平面中一字节数据代表 8 个水平相邻的像素。高位对应右边像素, 低位对应左边像素。为了写入 8 个像素中的一个或几个而不影响另外的像素, 可通过设置该寄存器相应位为 0 来屏蔽掉不想改变的位。

(2) 时序控制器

2 号映射屏蔽(Map Mask)寄存器: 该寄存器是实现单个位平面操作至关重要的一个。我们知道, EGA/VGA 由四个显示页面组成, 每个页面有相同的存储地址, 写入其中的数据可能写入任何一个联合显示页中。Map Mask 寄存器的设置就是为了使写入的数据能送到指定的页面中, 该寄存器的低 4 位分别对应于 4 个页面, 当某位的值为 0, 相应的页面就被屏蔽。

3. 读写模式

要对显存进行特定的操作, 首先要选择适当的读取和写入模式。下面就与本文有关的二种读模式(模式 0、1)和一种写模式(模式 0)加以说明。

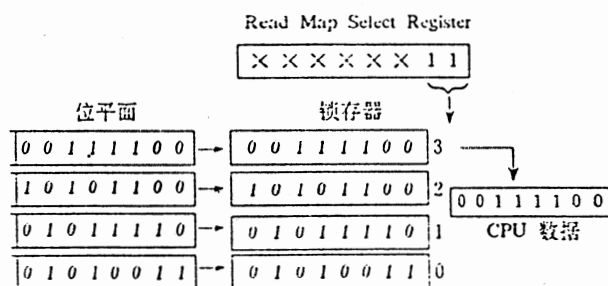


图 1 EGA/VGA 读取模式 0

(1) 读取模式 0

这种模式下, CPU 在读取时, 显存中的 8 个像素的数据(4 个字节)先被送到 EGA/VGA 内带的 32 位锁存器中, 然后根据 Read Map Select 寄存器(04h)的值选一字节到 CPU 的寄存器中。其过程具体如图 1 所示。

(2) 读取模式 1

该模式下 CPU 读入数据的过程是这样的: 从显存移到锁存器中的 8 个像素值先和 Color don't Care 寄存器中的值的低四位相与, 结果再和 Color Compare AND Color don't Care 的值相比较, 相等为 1, 否则为 0(如下图所示)。

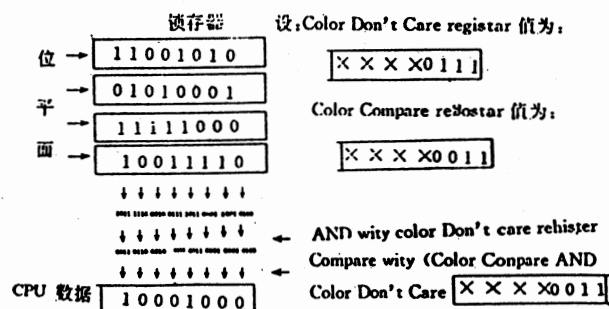


图 2 EGA/VGA 读取模式 1

(3) 写入模式 0

该模式下, CPU 写到显存的数据先根据 Data Rotate 的值右移, 右移后所得到的数据以 Function Select 指定的方式与锁存器中的数据进行操作, 结果根据 Bit Mask 值改变显存中指定的像素, 至于哪个或哪些位平面的内容被改变则由 Map Mask (02h) 寄存器的值来决定, 该过程如图 3 所示。

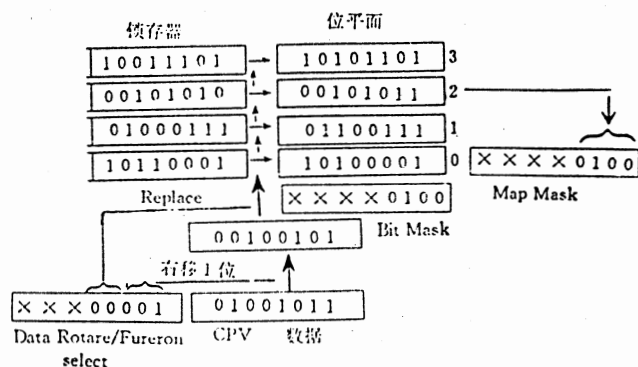


图 3 EGA/VGA 写模式 0

二、对位平面操作的实现

通过前面的叙述, 实现对位平面的操作就很清

楚了, BIOS 初始化时, 对时序控制器的 Map Mask 寄存器的位 0~3 置成 1111b, 这样, 4 个位平面同时可写, 如果只让位 3~0 的某一位为 1, 可实现对该位所对应的那个位平面写入而不影响其他位平面。要读取某一位平面的数据, 利用 EGA/VGA 的读取模式 0 (参见图 1) 可轻易地实现。

三、实 例

1. 参数的说明

buf: 象素块在内存中的首地址;

PlaneNum: 所要读取的位平面(0~3);

RMW __MAPMask: 高字节送到 DataRotate/Function Select 寄存器, 控制更新方式; 低字节送到 Bit Mask 寄存器, 取值 8、4、2、1 可实现单一位平面的写入。

2. 图象数据存放格式的说明

GetBitImage 获得的图象块数据在内存中的存放格式为:

第 0~1 字节: 图象块的高度(象素);

第 2~3 字节: 图象块的长度(字节);

第 4 字节: 每行最后一字节的掩码;

第 5 字节以后: 图象块数据。

本程序在 Turbo C 2.0, 386DX/40、VGA 卡的兼容机上通过。

源程序清单如下:

```

; *****
; * 程序名: CalcuAddr *
; * 功 能: 计算象素在 Video Buffer 中的地址 *
; * 调用方法: *
; * 入 口: AX:=Y 座标 *
; *          BX:=X 座标 *
; * 出 口: *
; * AH -- 没有经过移位的 BIT MASK *
; * BX -- 在 Video Buffer 中的偏移 *
; * CL -- 左移位数 *
; * ES -- Video Buffer 段值 *
; *****
BytesPerLine EQU 80 ; 屏幕逻辑宽
OriginOffset EQU 0 ; (0,0) 的偏移
VideoBufferSeg EQU 0A000h
__TEXT SEGMENT byte public 'CODE'
ASSUME cs:__TEXT
PUBLIC CalcuAddr
CalcuAddr PROC near
    mov cl,bl
    push dx
    mov dx,BytesPerLine
; AX := y * bytesPerLine
    mul dx
    pop dx
    shr bx,1
    shr bx,1
    shr bx,1 ; BX := X/8

```

```

    add bx,ax ; BX := y * bytesPerLine + x/8
    add bx,OriginOffset
; Bx := byte offset in video buffer
    mov ax,VideoBufferSeg
    mov es,ax
; ES:Bx := byte offset in video pixel
    and cl,7 ; cl := x&7
    xor cl,7 ; cl := number of bits to shift left
    mov ah,1 ; ah := unshifted bit mask
    ret
CalcuAddr ENDP
__TEXT ENDS
END

```

```

; *****
; * 函数名: GetBitImage *
; * 调用方法: *
; * void GetBitImage(int x0,int y0,int x1,int y1, *
; * char far *buf,char PlaneNum); *
; *****
ParaFrame struc
BasePtr DW ?
RetAddr DW ?
x0 DW ?
y0 DW ?
x1 DW ?
y1 DW ?
buf DD ?
PlaneNum DB ?
DB ?
ParaFrame ends

Rows EQU word ptr [bp-2]
RowLen EQU word ptr [bp-4]
BytesPerRow EQU 80

```

```

__TEXT SEGMENT byte public 'CODE'
ASSUME cs:__TEXT
EXTRN CalcuAddr:near
PUBLIC __GetBitImage
Para EQU [bp]

```

```

__GetBitImage PROC near
    push bp
    mov bp,sp
    sub sp,4
    push ds
    push si
    push di
; 计算位块的长、宽
    mov ax,Para.x1
    sub ax,Para.x0
    mov cx,0ff07h
; 计算每行最后一个字节的 Bit Mask
    and cl,al
    xor cl,7
    shl ch,cl
    mov cl,ch
    push cx
    mov cl,3 ; 计算行长(字节数)
    shr ax,cl
    inc ax
    push ax
    mov ax,Para.y1 ; 计算象素块的高

```

```

        sub ax,Para.y0
        inc ax
        push ax
;建立目标和源地址
        mov ax,Para.y0
        mov bx,Para.x0
        call CalcuAddr
        xor cl,7
        push es
        pop ds
        mov si,bx ;DS:SI -> video buffer
        les di,Para.buf
;ES:DI -> buffer in system RAM
;建立 5 字节的位块头
        pop ax
        mov Rows,ax
        stosw ;byte 0-1: 块长
        pop ax
        mov RowLen,ax
        stosw ;byte 2-3: 块高
        pop ax
        mov ch,al
        stosb ;byte 4: 末字节 Bit Mask
;设立图形控制器的参数
        mov dx,3ceh
        mov ax,0005
        out dx,ax
        mov ah,Para.PlaneNum
        mov al,04
;拷贝到系统 RAM 中
        out dx,ax ;选择一位平面
Loop01: mov bx,RowLen
        push si
Loop02: lodsw
        dec si
        rol ax,cl
        stosb
        dec bx
        jnz Loop02
        and es:[di-1],ch ;mask last byte in row
        pop si
        add si,BytesPerRow ;ds:si->下一行始址
        dec Rows
        jnz Loop01 ;读入下一行
        pop di
        pop si
        pop ds
        mov sp,bp
        pop bp
        ret
__GetBitImage ENDP
__TEXT ENDS
END
; * * * * *
; * 函数名: PutBitImage *
; * 功 能: 把 EGA/VGA video RAM 中指定区域的位 *
; * 块拷贝到系统 RAM 中 *
; * 调用方法: void PutBitImage(int x,int y,char far *buf, *
; * int RMW-PlaneMask); *
; * * * * *
ParaFrame struc

```

```

BasePtr DW ?
RetAddr DW ?
x DW ?
y DW ?
buf DD ?
BitPlaneMask DB ?
RMWbits DB ?
ParaFrame ends
Rows EQU word ptr [bp-2];位块的行数
RowLen EQU word ptr [bp-4];位块的行长
RowCounter EQU word ptr [bp-6];行计数器
StartMask EQU word ptr [bp-8];第一字节的掩码
EndMaskL EQU word ptr [bp-10];倒数第二个字节的掩码
EndMaskR EQU word ptr [bp-12];最后一个字节的掩码
BytesPerRow EQU 80;视屏缓冲区的逻辑宽
__TEXT SEGMENT byte public 'CODE'
        ASSUME cs:__TEXT
        EXTRN CalcuAddr:near
        PUBLIC __PutBitImage
Para EQU [bp]
__PutBitImage PROC near
        push bp
        mov bp,sp
        sub sp,12
        push ds
        push si
        push di
;计算有关起始地址:
        mov ax,Para.y ;ES:BX -> (x,y)所在字节的地址
        mov bx,Para.x
        call CalcuAddr
        inc cl
        and cl,7;
        mov di,bx
;ES:DI -> 为 video Buffer 中(x,y)点的地址
        lds si,Para.buf
;DS:SI -> 系统 RAM 首地址
;读取位块头,并设定有关变量
        lodsw
        mov Rows,ax
        lodsw
        mov RowLen,ax
        lodsb
        mov ch,al ;ch 中存放首字节的 Bit Mask
;设置图形控制器的参数
        mov dx,3ceh
        mov ah,Para.RMWbits
        mov al,3
        out dx,ax
        mov ax,0805h
;AH=00001000b(读模式 1,写模式 0)
        out dx,ax
        mov ax,0007h;设定 Color Don't Care 寄存器值为 0
        out dx,ax ;CPU 读到的永远为 0ffh
        mov ax,0ff08h
        out dx,ax
        mov di,0c4h
        mov ah,Para.BitPlaneMask
        mov al,02
        cmp cx,0ff00h

```

;CH 为末字节的 Bit Mask 值,CL 为向左的移位数,如果
;CH=0FFH,CL=00H,则首尾均无零头,若有零头则转

jne L13

;处理首尾像素均在字节边界的情况(无零头)

out dx,ax ;使一个位平面可写

mov cx,RowLen

mov bx,Rows

Loop11: push di

push cx

Loop12: lodsb

and es:[di],al

inc di

loop Loop12

pop cx

pop di

add di,BytesPerRow

dec bx

jnz Loop11 ;处理下一行像素

jmp Exit

;处理首末有一个不齐或都不齐的情况

L13: push ax ;ax 中存放着 Map Mask 的值

mov bx,0ffh

mov al,ch

cbw

cmp RowLen,1

jne L14 ;行长大于一个字节则转

mov bl,ch

mov ah,ch

xor al,al

L14: shl ax,cl ;AH: 末第二字节的 Bit Mask

shl bx,cl ;BL: 末字节的 Bit Mask

mov bl,al ;BH: 首字节的 Bit Mask

mov al,8

mov EndMaskL,ax

mov ah,bl

mov EndMaskR,ax

mov ah,bh

mov StartMask,ax

mov bx,RowLen

pop ax

;逐行写像素

out dx,ax ;使一个位平面可写

mov dl,0ceh

mov ax,Rows

mov RowCounter,ax ;初始化行计数器

;写本行第一个字节

L15: push di

push si

push bx

mov ax,StartMask

out dx,ax ;设定第一个字节的 Bit Mask

lodsw

dec si

test cl,cl ;首像素在字节边界?

jnz L16 ;不则转

dec bx

jnz L17 ;不为 0 则转至少有 2 个字节情况

jmp short L18 ;转行长只有一或二个字节的情况

L16: rol ax,cl ;把像素移到合适位置

and es:[di],ah

inc di

dec bx

L17: push ax ;ax 中为第一个字节的像素

mov ax,0ff08h

out dx,ax

pop ax

dec bx

jng L18

;转行长只有 1-2 个字节的情况

;写每行中间的像素

cli

Loop21: and es:[di],al

inc di

lodsw

dec si

rol ax,cl

dec bx

jnz Loop21

sti

;处理行末的像素

L18: mov bx,ax

mov ax,EndMaskL

out dx,ax

and es:[di],bl

mov ax,EndMaskR

out dx,ax

and es:[di+1],bh

pop bx

pop si

add si,bx

pop di

add di,BytesPerRow

dec RowCounter

jnz L15

mov dl,0c4h

;恢复图形控制器和时序控制器的缺省值

Exit: mov ax,0f02h

out dx,ax

mov dl,0ceh

mov ax,0003h

out dx,ax

mov ax,0005

out dx,ax

mov ax,0f07h

out dx,ax

mov ax,0ff08h

out dx,ax

pop di

pop si

pop ds

mov sp,bp

pop bp

ret

_PutBltImage ENDP

_TEXT ENDS

END

/* */

/* 目的:演示位平面操作函数的 C 语言调用方法. */

/* 说明:本程序计算位平面图象块的大小直接用 C 中 */

/* 的 imagesize 函数,所分配的内存量有浪费. */

/* 请读者自己编写用于位平面的大小计算函数 */

/* BitImageSize. */

/* */

```
#include <graphics.h>
#include <alloc.h>
void GetBitImage(int x0,int y0,int x1,int y1,char far * buf,
    char PlaneNum);
void PutBitImage(int x,int y,char far * buf,
    int RMW __ PlaneMask);
main()
{
    int gm,gd,i;
    unsigned long size;
    char far * buf1;
    gd=VGA; gm=VGAHI;
    initgraph(&gd,&gm,"e:\\tc");
    bar(00,00,150,80);
    setcolor(3);
    settextstyle(TRIPLEX __ FONT,HORIZ __ DIR,3);
    outtextxy(10,10,"Mou DaSen");
    size = imagesize(10,10,200,100);
    buf1 = farmalloc(size);
    getimage(2,2,10,10,buf1);
    GetBitImage(0,7,144,45,buf1,0x03);
    cleardevice();
    outtextxy(6,380,"Bit Plane Demo");
    settextstyle(DEFAULT __ FONT,HORIZ __ DIR,1);
    outtextxy(32,410,"N I N G B O");
    outtextxy(25,430,"—— Mou Dasen");
    line(0,0,639,0);
    line(639,0,639,479);
    line(639,352,0,352);
    line(0,479,0,0);
    line(190,0,190,479);
```

```
line(0,479,639,479);
setfillstyle(BKSLASH __ FILL,8);
bar(200,200,600,250);
bar(405,0,455,198);
setfillstyle(LTBKSLASH __ FILL,4);
bar(200,300,600,350);
bar(300,0,350,198);
setfillstyle(SOLID __ FILL,15);
bar(200,100,600,120);
PutBitImage(20,0,buf1,0x1008);
PutBitImage(20,100,buf1,0x1004);
PutBitImage(20,200,buf1,0x1002);
PutBitImage(20,300,buf1,0x1001);
for (i=0;i<51;i++) {
    PutBitImage(i * 6,0,buf1,0x1802);
    delay(40);
    PutBitImage(i * 6,0,buf1,0x1802);
}
PutBitImage(306,0,buf1,0x1002);
for (i=0;i<100;i++) {
    PutBitImage(306,i * 4,buf1,0x1808);
    delay(40);
    PutBitImage(306,i * 4,buf1,0x1808);
}
PutBitImage(306,400,buf1,0x1008);
free(buf1);
getch();
closegraph();
return;
}
```

SKILL

在 DOS 下快速显示 SPT 图形

□ 颜 飞

在

DOS 下显示 SPT 图形,通常使用画点的算法,速度太慢,对于较大幅面的图形要花很多时间。提高显示的唯一办法是将 SPT 图形数据直接写入 VGA 的图形缓冲区。

SPT 图形处理系统的文件有几种存储格式,本文使用其 SPT 非压缩格式。这种格式的文件前 64 个字节为文件头,文件头之后是图形数据,每个字节代表图形横向的 8 个点。

文件头结构如下:

- 0—15 字节:版本名称;
- 28—31 字节:作者姓名;
- 34—35 字节:版面宽度(8 的倍数);
- 36—37 字节:版面高度;
- 38—64 字节:不详。

VGA 的图形缓冲区在 640×480 16 色模式下分成 4 个位平面,每个位平面代表一种色彩,故可同时使用 4 种颜色。VGA 的 4 个位平面相对于 CPU 的起始地址是相同的,为了控制从 CPU 写入的数据置入哪一个位平面,必须正确地设置 VGA 的图形屏蔽寄存器(Map Mask Register)。该寄存器的地址是 3C5H,以地址为 3C4H 的寄存器的第一位为索引,其 0—3 位分别对应 4 个位平面,某位置 1,则相应的位平面开启,置 0,则相应的位平面关闭。向 VGA 的图形缓冲区直接写入数据的方法是:首先,将 3C4H 置为 02H,再给 3C5H 赋值(该值即为显示颜色),然后便可以向 VGA 图形缓冲区写入数据。为了提高显示速度,应将整个图形一次读入内存。

以下是快速显示 SPT 图形的程序,使用

Microsoft C 6.0 编译,其执行文件仅 8.5KB。

在主频 16MHz 的 286 微机上显示一幅 640×480 的图形,只是一瞬间的事情。若使用 Microsoft C 的 __setpixel 函数则需要 20 到 30 秒的时间,且可执行文件至少在 20KB 以上。

源程序清单如下:

```
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <malloc.h>

FILE * spt;
unsigned char * picture;
char __far * video = (char __far *)0xa0000000;

struct {
    int x, y;
} size;

enum VIDEO __MODE { CHAR = 0x03, PIXEL = 0x12 };
int open __spt( char * spt_name );
int setvideomode( int mode );
void close __spt( void );
void show __spt( int x, int y, int w, int h );

main( int argc, char * argv[] )
{
    if( argc != 2 )
    {
        printf( "you forgot to enter the filename. \n" );
        exit( 0 );
    }
    if( ! open __spt( argv[1] ) )
    {
        exit( 0 );
    }
    show __spt( 0, 0, size.x, size.y );
    close __spt();
}

int open __spt( char * spt_name )
{

```

(下转第 11 页)

幻灯片文件的格式分析

□张春明 张 翅

利

用 AutoCAD 软件提供的幻灯片制作功能,可以将屏幕上已绘好的图形制成一幅幅幻灯片存入幻灯片文件中。当需要时,再调用 AutoCAD 软件提供的幻灯片放映功能快速地再现幻灯片上的图形。连续地观察一组幻灯片,可以产生完整的幻灯效果。幻灯片对于模拟某个实验过程、制作教学演示片等具有十分重要的意义。然而上述幻灯片的制作和放映均须在 AutoCAD 软件支持下进行,若要在自己开发的程序中加入幻灯片功能,则必须对幻灯片文件的格式有一定的了解。本文将详细介绍幻灯片文件的格式,并给出在程序中直接调用幻灯片文件生成幻灯片的方法。

一、幻灯片文件的格式

幻灯片文件由文件头和一至多个数据记录组成。

1. 文件头的格式

不同版本的幻灯片文件其文件头的格式略有不同,表 1 和表 2 分别给出了 AutoCAD9.0 以前版本和以后版本幻灯片文件的文件头格式。

表 1 AutoCAD9.0 版本以前幻灯片文件的文件头格式

偏移 (十六进制)	长度 (字节数)	内 容
00	17	文件标志,即“AutoCAD Slide”,0D,0A,1A,00(十六进制)
11	1	类型标志,当前置为 56(十六进制)
12	1	层次标志,当前置为 1
13	2	图宽 X,即图形区宽度—1(以像素点为单位)
15	2	图高 Y,即图形区高度—1(以像素点为单位)
17	8	图形区高宽比(水平尺寸/垂直尺寸,以英寸为单位)
1F	2	硬件填充标志,当前置为 0 或 2(此值无实际意义)
21	1	过滤标志,此值无实际意义

表 2 AutoCAD9.0 版本以上幻灯片文件的文件头格式

偏移 (十六进制)	长度 (字节数)	内 容
00	17	文件标志,即“AutoCAD Slide”,0D,0A,1A,00(十六进制)
11	1	类型标志,当前置为 56(十六进制)
12	1	层次标志,当前置为 2
13	2	图宽 X,即图形区宽度—1(以像素点为单位)
15	2	图高 Y,即图形区高度—1(以像素点为单位)
17	4	图形区高宽比(水平尺寸/垂直尺寸,以英寸为单位)
1B	2	硬件填充标志,当前置为 0 或 2(此值无实际意义)
1D	2	测试码,当码值为 1234(十六进制)时表示文件中所有的双字节数都是高位字节在后(Intel 8086 系列 CPU),否则表示低位字节在后

2. 数据记录的格式

紧跟在文件头之后是一至多个数据记录,数据记录的数目取决于幻灯片上图形的复杂程度。每个数据记录的开头是一个双字节数,其中高位字节代表记录的类型(高位字节一般位于双字节数的第二个字节,对于 AutoCAD9.0 以后版本的幻灯片文件,可以

通过检测文件头中的测试码来确定双字节数中高位字节的实际位置)。根据记录的类型,可以确定该记录由几个字节组成以及各自代表什么意义,如表 3 所示。需要说明一点,幻灯片文件中的所有坐标均以屏幕上的象素为单位,并规定图形左下角坐标为(0,0)。

表 3 幻灯片文件中数据记录的格式

记录类型 (十六进制)	长度 (字节数)	意义	说 明
00~7F	8	向量	当高位字节的值在 00~7F 之间时,表示该记录描述的是一个向量。其中记录开头的双字节数为向量起点的 X 坐标,随后的 3 个双字节数依次为向量起点的 Y 坐标、终点的 X 坐标和终点的 Y 坐标。起点作为图形的最后一个点保存,以便随后的向量参照使用
80~FA	—	未定义	保留为将来使用
FB	5	偏移向量	当高位字节的值为 FB 时,表示向量的起、始点坐标均以该点距离保存的最后一点的偏移量(-128~+127)来表示。具体地说,该记录开头的双字节数中的低位字节,以及随后的 3 个字节分别为向量起点的 X、Y 坐标和终点的 X、Y 坐标的偏移值。调整后的起点作为图形的最后一个点保存,以便随后的向量参照使用
FC	2	文件结束	低位字节总是 00
FD	6	实填充	低位字节总是 00,随后的两个双字节数分别为要填充的多边形的一个顶点的 X 和 Y 坐标。多边形的每一个顶点都需要用这样的一个记录来描述。多个顶点记录按顺序出现,便构成了一个多边形实填充序列。当 Y 坐标的值为负数时,表示该记录为实填充序列的开始记录或结束记录。在开始记录中,X 坐标的值代表随后的顶点记录个数
FE	3	端点向量	将保存的最后一一点作为向量的起始点,再将该记录开头的双字节数中的低位字节及随后的一个字节分别作为向量终点的 X、Y 坐标距最后一一点的偏移值(-128~+127)。调整后的终点作为图形的最后一个点保存,以便随后的向量参照使用
FF	2	设置颜色	该记录用于设置新的颜色,随后画的向量都将使用低位字节指定的新颜色

二、调用幻灯片文件的方法

为了更加形象地说明调用幻灯片文件的方法,笔者用 Turbo C 编写了一段调用幻灯片文件生成幻灯片的程序。该程序可以处理 AutoCAD9.0 以前版本制作的幻灯片文件,但只要将程序中的文件头结构稍加修改,便可以处理 AutoCAD9.0 以后版本制作的幻灯片文件。

源程序清单如下:

```
#define TRUE 1
#define FALSE 0
#include<stdio.h>
#include<stdlib.h>
#include<graphics.h>

struct slideheader {
    char IDstring[17];
    char TypeID;
    char LevelID;
    int HighXdot;
    int HighYdot;
    double Aspectratio;
    int Hardware;
    char Filter;
} header;

union word __or__ byte {
```

由于 AutoCAD 软件中定义的颜色号与 Turbo C 中定义的颜色号有所不同,因此,我们对颜色号进行了适当的转换处理,使之能够准确地反映原图的色彩。另外,在 AutoCAD 软件中坐标原点位于屏幕的左下角,而在 Turbo C 中坐标原点位于屏幕的左上角,若直接利用原来的坐标,则必然会得到倒立的图象。针对这个问题,笔者在本程序中也作了相应的处理。

```
struct {
    char lowbyte;
    unsigned char rectype;
} bytes;
int word;
} recfield;

FILE * fp;
int fromx, fromy, tox, toy, lastx, lasty, newcolor;
int endoffile=FALSE;

main(int argc, char * argv[])
{
    int driver=DETECT;
    int mode=0;
    int colors[]={
        (BLACK, RED, YELLOW, GREEN, CYAN, BLUE, MAGENTA, WHITE);
    if(argc!=2){
```

```

printf("\nInvalid number of parameters! \n");
printf("\n(C)Copyright Northeastern University March 1994");
printf("\n Programmer:Zhang Chunming and Zhang Chi");
printf("\n\n Format: VSLIDE <slide filename>\n");
exit(1);
}
if((fp=fopen(argv[1],"rb"))==NULL){
    printf("\nCannot open Slide file %s! \n",argv[1]);
    exit(1);
}
fread(&header,sizeof(header),1,fp);
if((strcmp(header.IDstring,"AutoCAD Slide\r\n\x1A")!=0){
    printf("\nIllegal Slide file! \n");
    exit(1);
}
if(header.LevelID != 1){
    printf("\nIncorrect Version! \n");
    exit(1);
}
initgraph(&driver,&mode,"");
do{
    fread(&recfield,sizeof(recfield),1,fp);
    switch(recfield.bytes.rectype)
    {
        case 0xFF:
            newcolor=recfield.bytes.lowbyte;
            if(newcolor<=7)
                newcolor=colors[newcolor];
            setcolor(newcolor);
            break;
        case 0xFE:
            commonendpoint();
            break;
        case 0xFD:
            solidfill();
            break;
        case 0xFC:
            endofile=TRUE;
            break;
        case 0xFB:
            offsetvector();
            break;
        default:
            if (recfield.bytes.rectype<0x80)
                vector();
            break;
    }
} /* end of switch */
}while(endofile != TRUE);
/* end of do-while */
getch();
fclose(fp);
restorecrtmode();
}
/* end of main */

int vector()
{
    fromx=recfield.word;
    fread(&fromy,sizeof(int),1,fp);
    fread(&tox,sizeof(int),1,fp);

```

```

    fread(&toy,sizeof(int),1,fp);
    line(fromx,header.HighYdot-fromy,tox,header.HighYdot-toy);
    lastx=fromx;
    lasty=fromy;
}
/* end of vector */

int offsetvector()
{
    char offx1,offx2,offy1,offy2;
    fromx=lastx+recfield.bytes.lowbyte;
    fread(&offx1,sizeof(char),1,fp);
    fread(&offx2,sizeof(char),1,fp);
    fread(&offy1,sizeof(char),1,fp);
    fread(&offy2,sizeof(char),1,fp);
    fromy=lastx+offx1;
    tox=lastx+offx2;
    toy=lasty+offy1;
    line(fromx,header.HighYdot-fromy,tox,header.HighYdot-toy);
    lastx=fromx;
    lasty=fromy;
}
/* end of offsetvector */

int commonendpoint()
{
    char offx;
    tox=lastx+recfield.bytes.lowbyte;
    fread(&offx,sizeof(char),1,fp);
    toy=lasty+offx;
    line(lastx,header.HighYdot-lasty,tox,header.HighYdot-toy);
    lastx=tox;
    lasty=toy;
}
/* end of commonendpoint */

int solidfill()
{
    int i=0,j=0,nums,dumb,shape[30];
    fread(&nums,sizeof(int),1,fp);
    fread(&dumb,sizeof(int),1,fp);
    /* Y is negative */
    while(j++<nums)
    {
        fread(&dumb,sizeof(int),1,fp);
        /* skip Solidfill ID,00 FD */
        fread(&shape[i++],sizeof(int),1,fp);
        fread(&shape[i++],sizeof(int),1,fp);
        shape[i-1]=header.HighYdot-shape[i-1];
        /* reverse Y coordinate */
    }
    /* end of while */

    fread(&dumb,sizeof(int),1,fp);
    /* Y is negative */
    fread(&dumb,sizeof(int),1,fp);
    fread(&dumb,sizeof(int),1,fp);
    setfillstyle(SOL_ID__FILL,newcolor);
    fillpoly(nums,shape);
}
/* end of solidfill */

```

中文幻灯片制作与放映

□董占山

C

CDOS2.13H 汉字系统的特殊显示功能可以显示 24 点阵的汉字和字符,以及在屏幕上画点、线、矩形和圆等,给我们提供了制作中文幻灯片的基本条件。下面我们简要介绍中文幻灯片的制作与放映技术。

一、幻灯片制作技术

一般来说,制作中文幻灯片要考虑两个方面,即文字内容与色彩调配。文字内容是指字体与字形,色彩包括前景色、背景色、标题文字色和内容文字色等,这些功能 CCDOS2.13H 汉字系统均提供了。制作中文幻灯片的方法如下:首先,在 CCDOS2.13H 汉字系统下,用 WS 按 2.13H 汉字系统的要求编辑幻灯片的内容,然后用 TYPE 命令将编辑好的文件显示在屏幕上,直到满意为止。而后用《2.13 系列汉字系统用户手册》上提供的屏幕图象文件的直接存取程序(p. 205-219),将屏幕上显示的幻灯片存入指定的文件,备放映幻灯片时使用。

二、幻灯片放映技术

放映幻灯片就是要在计算机屏幕上按一定的次序将制作好的幻灯片显示出来,象幻灯机一样,每放一张暂停一下,可前进、可后退,也可立即停止。笔者用 Turbo Pascal 编写了一个幻灯片放映程序 SLIDE.PAS(源程序附后),可实现上述思路。该程序使用的功能健为:Home 显示第一张,PgDn 显示下一张,PgUp 显示上一张,End 停止。

放映幻灯片的步骤:

1. 建立幻灯片顺序文件:这是一个文本

文件,每一行包括一个幻灯片,由幻灯片文件名和显示背景色号组成,用逗号隔开。格式如下:

```
DEMO1.SLD
DEMO1.G,1
DEMO2.G,4
DEMO3.G,1
DEMO4.G,0
DEMO5.G,2
```

背景色号 0、1、2、3、4、5、6、7 分别代表黑色、红色、绿色、黄色、蓝色、品红色、青色和白色。背景色是通过 ANSI.SYS 的扩展功能实现的,所以在 CONFIG.SYS 中应当有 DEVICE=ANSI.SYS 的命令,且 ANSI.SYS 应是 CCDOS2.13H 提供的。

2. 放映幻灯片:在西文 DOS 系统下,执行幻灯片放映程序 SLIDE,格式:

```
SLIDE DEMO1.SLD
```

上面介绍的方法在 GW0520CH、GW286B 等长城机上运行通过。

源程序清单如下:

```
{ $M $1000.0, $1000 }
program chinese _ slide _ for _ GW0520CH;

uses dos;
type
  sldtyp=record
    flnm:string[12];
    color:byte;
  end;

var
  flnm:string;
  count,i:word;
  sld:array[1..100]of sldtyp;
function readkey:char;
```



```
inline( $b8/$00/$07/$cd/$21):
```

```
procedure ReadSlideFile;
```

```
var
```

```
  s1:string;
```

```
  fl1:text;
```

```
begin
```

```
  assign(fl1,flnm);
```

```
  reset(fl1);
```

```
  if ioresult < 0 then
```

```
    begin
```

```
      writeln('File not found');
```

```
      halt;
```

```
    end;
```

```
  i:=0;
```

```
  repeat
```

```
    inc(i);
```

```
    readln(fl1,s1);
```

```
    sld[i].flnm:=copy(s1,1,pos(' ',s1));
```

```
    delete(s1,1,pos(' ',s1));
```

```
    val(s1,sld[i].color,count);
```

```
  until eof(fl1);
```

```
  close(fl1);
```

```
end;
```

```
procedure PlaySlide;
```

```
var
```

```
  ch:char;
```

```
begin
```

```
  count:=1;
```

```
  repeat
```

```
    ch:=readkey;
```

```
    if ch=#0 then
```

```
      begin
```

```
        ch:=readkey;
```

```
        if ch=#71 then count:=1;
```

```
        if ch=#73 then
```

```
          if count>1 then
```

```
            count:=count-1;
```

```
        if ch=#81 then
```

```
  if count<=i then
```

```
    count:=count+1
```

```
  else ch:='E';
```

```
  if ch=#79 then ch:='E';
```

```
  if ch in [#71,#73,#81] then
```

```
    begin
```

```
      exec(getenv('COMSPEC')).'/C vsp'+sld[count].flnm);
```

```
    case sld[count].color of
```

```
      0:writeln(chr($1b)+'[1;30;40m');
```

```
      1:writeln(chr($1b)+'[1;31;41m');
```

```
      2:writeln(chr($1b)+'[1;32;42m');
```

```
      3:writeln(chr($1b)+'[1;33;43m');
```

```
      4:writeln(chr($1b)+'[1;34;44m');
```

```
      5:writeln(chr($1b)+'[1;35;45m');
```

```
      6:writeln(chr($1b)+'[1;36;46m');
```

```
      7:writeln(chr($1b)+'[1;37;47m');
```

```
    end;
```

```
    writeln(chr($1b)+'[2J');
```

```
  end;
```

```
end;
```

```
until ch='E';
```

```
end;
```

```
procedure ClearScr;
```

```
inline( $b8/$03/$00 {mov ax,03}
```

```
  /$cd/$10); {int 10h}
```

```
procedure GetComLineParameter;
```

```
begin
```

```
  flnm:='';
```

```
  if paramcount=1 then
```

```
    flnm:=paramstr(1);
```

```
end;
```

```
begin
```

```
  GetComLineParameter;
```

```
  if flnm='' then exit;
```

```
  ReadSlideFile;
```

```
  PlaySlide;
```

```
  ClearScr;
```

```
end.
```

SKILL

(上接第 12 页)

```
:DoDrop
```

```
set PATH=
```

```
:DropLoop
```

```
shift
```

```
if '%1'=='' goto DoneDrop
```

```
if '%1'=='%PE __DROP%' goto DidRemove
```

```
if '%PATH%'==' ' PATH=%1
```

```
if '%PATH%'=='%1' goto DropLoop
```

```
PATH=%PATH%;%1
```

```
goto DropLoop
```

```
:DidRemove
```

```
set PE __REM=YES
```

```
goto DropLoop
```

```
:DoneDrop
```

```
if '%PE __REM%'==' ' echo %PE __DROP% not found.
```

```
if '%PE __REM%'==' YES' echo %PE __DROP% removed from
PATH.
```

```
set PE __DROP=
```

```
set PE __REM=
```

```
goto END
```

```
:Help
```

```
echo PE option directory
```

```
echo option:
```

```
echo +:Adds directory at end of PATH.
```

```
echo &.:Adds directory at begin of PATH.
```

```
echo -:Drops directory from PATH.
```

```
echo Note that the PATH must not be empty.
```

```
:END
```

```
path
```

SKILL

12×12 点阵汉字显示系统

□吴文江 郑群 陆佳明

本

文介绍了 12×12 点阵汉字的生成方法及显示技术,可广泛用于不需要汉字输入的应用系统中,以提高应用系统的字符分辨率及视觉效果。

一、12×12 点阵汉字的生成

目前市场上流行的汉字系统的显示字库大都为 16×16 点阵的。当需要设计精巧的界面时,这种字体便略显笨拙。我们经过大量实验,发现利用吴晓军的 24 点阵楷体字库 HZK24K 可生成令人满意的 12×12 点阵的字库。

字库生成的中心思想就是把原字模的 2×2 点压缩为一个点,方法是将四个点的值(1 表示有,0 表示无)相加,当其和大于或等于 2 时,我们便在 12×12 点阵中指定对应位为 1,否则为 0。实现时需注意将原 24×24 点阵转置以生成按行排列的显示字库。对 HZK24T 也需同样处理,并加在前面,这样形成的字库文件即为按国标码排列的字库。字模按行排列,每行占一个半字节(12 位),每个字模占 18 个字节,见程序 2。由于嵌入 286 指令以提高速度,故我们使用 Borland C++ 3.1 编译系统的 286 选项(用户编译时一定要加上)。程序 2 中生成的 12 点阵字库名为 12.PNT(汉字字库)和 12.ASC(西文字库)。

二、12×6 点阵西文字库的生成

为了使西文字符为汉字的一半,所有西文字符号(32~126 ASCII 码)都需重置,原理与汉字相近,只是把 HZK24T 中 10 区的 94

个字模再横向压缩 2:1,即横向连续两点相加为大于等于 1 时便置 12×6 字符相应位为 1,否则为 0。为快速起见,每个字符占 12 个字节(每个字节低两位无用),这样形成的文件 12.ASC 即为 ASCII 码 21H~7EH 的可显示字符库,前面再加一个全为 0 的字模,即为 ASCII 码 20H(空格)。

三、汉字的显示

显示汉字的关键在于位屏蔽技术,也就是说,与要显示的水平位置有关。为此,我们可先把 18 字节点阵的 12×12 点阵字模(12×6 也一样)变为一个二维数组 `buffer[3][13]`,其中 `buffer[i][0]` 为对应第 *i* 个字节的位掩码,其余 12 个字节为该列字模,当位掩码为 0 时,它们无意义。这样,我们就可以快速显示 12×12 点阵汉字(也包括 12×6 西文字符)了,见程序 1。

四、字库管理技术

如果每个字符都要从磁盘上读取,无疑是浪费了时间,也消耗了磁盘。而把整个字库都放到内存中又减少了应用程序可用的内存空间。为此,我们可先将应用程序源代码中的汉字字符提取出来,做成一个小字库和一个索引字库,启动程序时可先将它们装入内存,显示时可先查索引库,再根据索引地址将字模取出。这对一般应用来说较为合适(整个字库约为八千字,而一般应用只用不到一千个汉字)。每个汉字加索引只占二十个字节,500 个汉字的应用程序只需 10K 内存空间即可。用这种方法显示汉字速度极快。为应用程序生成的汉

字库名为 12.CHN,索引库名为 12.INX,见程序 3。

源程序清单如下:

/* 程序 1: DEMO. C 这是 12 * 12 点阵汉字的演示程序 */

```
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#include <alloc.h>
#include <string.h>
#include <conio.h>

#define TOO __LONG__ STR      -1
#define FILE __NOT__ FOUND    -2
typedef unsigned char BYTE;
typedef unsigned int WORD;
int PrepareData(void);

void DrawOneLine(void);
void Draw 12 __ 12char( WORD Left,WORD Top,BYTE foreground,
    BYTE background);
void ChineseConvert(int offset,char far * ptr);
void EnglishConvert(int offset,char far * ptr);
int Displayer(BYTE * str,int x,int y,const int limit,

int forecolor,int backcolor);
void DisplayOneChinese(BYTE * ptr,int x,int y,
    int forecolor,int backcolor);
void DisplayOneEnglish(BYTE * ptr,int x,int y,
    int forecolor,int backcolor);
WORD GetChinesePosition(BYTE c1,BYTE c2);
BYTE TempData[18],buffer[3][13],
    EnglishData[94 * 12], * ChineseData;
WORD * ChineseIndex,DataNumber,DataLength;

BYTE chin[2][30]={
    "欢迎使用 12 * 12 点阵汉字系统",
    " DEMO/MAKE __ LIB/CREATE"
};

main()
{
    int gdriver=VGA,gmode=VGAHI,i;
    int errorcode; //Modified
    if(registerbgidriver(EGAVGA—driver)<0)
        return(-1);
    initgraph(&gdriver,&gmode,""); //modified
    errorcode=graphresult();
    if (errorcode!= grOk)
        /* an error occurred */
        {
            printf("Graphics error: %s\n", grapherrormsg(errorcode));
            printf("Press any key to halt:");
            getch();
            return -1;
        }
    /* return with error code */
    if(! PrepareData())
    {
        for(i=0;i<2;i++)
            Displayer(chin[i],100,100+i * 20,20,7,0);
    }
}
```

```
getch();
closegraph();
}

fcloseall();
return(0);
}
```

/* 函数 PrepareData 将应用程序使用的专用汉字库及其索引字库和西文字库读入内存 */

```
int PrepareData(void)
{
    FILE * stream;
    if((stream=fopen("12. ASC","rb"))==NULL)
        return(FILE __NOT__ FOUND);
    fread(EnglishData,sizeof(char),1128,stream);
    /* EnglishData holds the ASCII code 20h—7eh fonts */
    fclose(stream);
    if((stream=fopen("12. INX","rb"))==NULL)
        return(FILE __NOT__ FOUND);

    fread(&DataNumber,sizeof(int),1,stream);
    ChineseIndex=farmalloc(DataNumber * sizeof(int));
    fread(ChineseIndex,sizeof(int),DataNumber,stream);
    /* ----- */
    fclose(stream);
    DataLength=DataNumber * 18;
    if((stream=fopen("12. CHN","rb"))==NULL)
        return(FILE __NOT__ FOUND);
    ChineseData=farmalloc(DataLength);
    /* ----- */
    fread(ChineseData,sizeof(char),DataLength,stream);
    fclose(stream);
    return(0);
}
```

/* 函数 Draw12 __ 12char 在(Left,Top)处以给定的前景/背景显示一字符,字模取自 buffer[3][13],为提高速度,程序中嵌入了汇编码 */

```
void Draw12 __ 12char( WORD Left,WORD Top,
    BYTE foreground,BYTE background)
{
    WORD s __ SEG,s __ OFF,video __ offset;
    BYTE fore,back,both;
    s __ SEG=FP __ SEG(buffer);
    s __ OFF=FP __ OFF(buffer);
    video __ offset=(Left>>3);
    video __ offset+=Top * 80;
    both=foreground & background;
    fore=foreground & (~both);
    back=background & (~both);

    push es
    pusha
    mov ax,s __ SEG
    mov ds,ax
    mov ax,0xa000
    mov es,ax
    mov si,s __ OFF
    mov dx,0x3ce
    lodsb
    mov ah,al
```

```

mov al,8
out dx,ax
mov cx,12
mov di,video__offset
mov dx,0x3c4
mov bh,fore
mov bl,back

```

Draw12 __ 12Char001:

```

asm mov ah,both
DrawOneLine();

```

```

asm {
    add di,0x50
    loop Draw12 __ 12Char001
    mov dx,0x3ce
    lodsb
    mov ah,al
    cmp al,0
    jnz Draw12 __ 12Char090
    jmp Draw12 __ 12Char100
}

```

Draw12 __ 12Char090:

```

asm {
    mov al,8
    out dx,ax
    mov cx,12
    mov di,video__offset
    inc di
    mov dx,0x3c4
    mov bh,fore
    mov bl,back
    mov cx,12
}

```

Draw12 __ 12Char002:

```

asm mov ah,both
DrawOneLine();
asm {
    add di,0x50
    loop Draw12 __ 12Char002
    mov dx,0x3ce
    lodsb
    mov ah,al
    mov al,8
    cmp ch,0
    jz Draw12 __ 12Char100
    out dx,ax
    mov cx,12
    mov di,video__offset
    inc di
    inc di
    mov dx,0x3c4
    mov bh,fore
    mov bl,back
}

```

```

mov cx,12
}

```

Draw12 __ 12Char003:

```

asm mov ah,both
DrawOneLine();
asm add di,0x50
asm loop Draw12 __ 12Char003

```

Draw12 __ 12Char100:

```

asm {
    mov dx,0x3ce
    mov ax,0xff08
    out dx,ax
    mov dx,0x3c4
    mov ax,0xf02
    out dx,ax
    popa
    pop es
    pop ds
}

```

/* 函数 DrawOneLine 显示一像素行中的八个点(一字节)

AL=要显示的字节,
BH=前景独有的位平面,
BL=背景色独有的位平面,
AH=前景色和背景色共有的位平面 */

void DrawOneLine(void)

```

{
    asm {
        lodsb
        push di
        push ax
        mov ah,bh
        mov al,2
        out dx,ax
        mov al,byte ptr es:[di]
        pop ax
        mov byte ptr es:[di],al
        not al
        push ax
        mov ah,bl
        mov al,2
        out dx,ax
        mov al,byte ptr es:[di]
        pop ax
        mov byte ptr es:[di],al
        mov al,2
        out dx,ax
        mov al,0ffh
        mov ah,byte ptr es:[di]
        stosb
        pop di
    }
}

```

return;

```

}

/* 函数 EnglishConvert 将西文 12 * 6 点阵(12 字节)存储数组转换后
放入缓冲区 buffer[3][13] */
void EnglishConvert(int offset, char far * ptr)
{
    int s __SEG, s __OFF, d __SEG, d __OFF, i;
    BYTE first, second;
    s __SEG = FP __SEG(ptr);
    s __OFF = FP __OFF(ptr);
    d __SEG = FP __SEG(buffer);
    d __OFF = FP __OFF(buffer);
    first = (offset & 0x7);
    asm { push ds
        push es
        pusha
        mov ax, s __SEG
        mov ds, ax
        mov ax, d __SEG
        mov es, ax
        mov si, s __OFF
        mov di, d __OFF
        mov cl, first
        mov ax, 0xfc00
        shr ax, cl
        mov es, [di].ah
        mov es, [di+13].al
        mov second, al
        xor al, al
        mov es, [di+26].al
        inc di
    }
    for(i=0; i<12; i++)
    {
        asm {
            lodsb
            mov ah, al
            mov cl, first
            shr ax, cl
            mov al, ah
            stosb
        }
    }
    if(second)
    {
        asm inc di
        asm mov si, s __OFF
        for(i=0; i<12; i++)
        {
            asm {
                lodsb
                mov ah, al
                mov cl, first
                shr ax, cl
                stosb
            }
        }
    }
    asm { popa
        pop es
        pop ds
    }
}

```

```

}
return;
}

/* 函数 ChineseConvert 将汉字 12 点阵(18 字节)存储数组转换后放
入缓冲区 buffer[3][13] */
void ChineseConvert(int offset, char far * ptr)
{
    int s __SEG, s __OFF, d __SEG, d __OFF, i, j=0;
    BYTE first;
    s __SEG = FP __SEG(ptr);
    s __OFF = FP __OFF(ptr);
    d __SEG = FP __SEG(buffer);
    d __OFF = FP __OFF(buffer);
    first = (offset & 0x7);
    asm { push ds
        push es
        pusha
        mov ax, s __SEG
        mov ds, ax
        mov ax, d __SEG
        mov es, ax
        mov si, s __OFF
        mov di, d __OFF
        mov cl, first
        mov ax, 0xfff0
        shr ax, cl
        mov es, [di].ah
        mov es, [di+13].al
        mov ax, 0xf000
        mov cl, 8
        sub cl, first
        shl ax, cl
        mov es, [di+26].ah
    }

    j++;
    for(i=0; i<6; i++)
    {
        asm { mov di, d __OFF
            add di, j
            lodsb
            mov ah, al
            mov al, ds:[si]
            mov cl, first
            ror ax, cl
            mov es, [di].ah
            mov es, [di+13].al
            mov es, [di+26].ah
        }
    }
    j++;
    asm {
        mov di, d __OFF
        add di, j
        lodsb
        mov ah, al
        lodsb
        mov cl, 4
        shl ax, cl
        mov cl, first
        ror ax, cl
    }
}

```



```

mov es:[di],ah
mov es:[di+13],al
mov es:[di+26],ah
}
j++;
}
asm{popa
    pop es
    pop ds
}
}

/* 函数 Displayer 以给定的前景/背景色(x,y)
   显示一 12 点阵字符串 */
int Displayer(BYTE *str,int x,int y,const int limit,int forecolor,
    int backcolor)
{ BYTE c1,c2;
  WORD x1,strlen,i,offset;
  x1=x;
  strlen=strlen(str);
  if(strlen>limit)
    strlen=limit;
  if((strlen*6+x1)>=640)
    return(TOO_LONG_STR);
  for(i=0; i<strlen; i++) {
    c1=*(str+i);
    c2=*(str+i+1);
    if(c1<=0x7E && c1>=0x20)
      { offset=(c1-32)*12;
        DisplayOneEnglish(EnglishData+offset,x1,y,forecolor,
          backcolor);
        x1+=6;
      }
    else
      if(c1>0xa0 && c2>0xa0)
      {
        offset=GetChinesePosition(c1,c2);
        offset*=18;
        DisplayOneChinese(ChineseData+offset,x1,y,
          forecolor,backcolor);
        x1+=12;
        i++;
      }
  }
  return(0);
}

```

/* 函数 GetChinesePosition 计算一汉字在索引库中的位置 */
 WORD GetChinesePosition(BYTE c1,BYTE c2)

```

{
  WORD s __SEG,s __OFF;
  s __SEG=FP __SEG(ChineseIndex);
  s __OFF=FP __OFF(ChineseIndex);
  asm {push es
    push cx
    push di
    mov ax,s __SEG
    mov es,ax
    mov di,s __OFF
    mov ah,c1
    mov al,c2

```

```

    mov cx,DataNumber
    push cx
    repnz scasw
    pop ax
    sub ax,cx
    dec ax
    pop di
    pop cx
    pop es
  }
  return(__AX);
}

```

```

void DisplayOneChinese(BYTE *ptr,int x,int y,
    int forecolor,int backcolor)
{
  ChineseConvert(x,ptr);
  Draw12_12char(x,y,forecolor,backcolor);
}

```

```

void DisplayOneEnglish(BYTE *ptr,int x,int y,
    int forecolor, int backcolor)
{
  EnglishConvert(x,ptr);
  Draw12_12char(x,y,forecolor,backcolor);
}

```

/* 程序 2: MAKE __LIB.C 此程序在 24 点阵汉字库 HZK24K 和 HZK24T 基础上生成 12 点阵字库 12.PNT 和 12.ASC */

```

#include <stdio.h>
#include <conio.h>
typedef unsigned char BYTE;
typedef unsigned int WORD;
void Converter12(char *s,char *d);
void ConverterC(void);
void ConverterW(void);
void Reverse(char *str1,char *str2);
BYTE s[72],ss[72],d[24];
BYTE result __c[18];
/* Contains the result Chinese font of 12*12 */
BYTE result __w[12];
/* Contains the result English font of 12*6 */
BYTE f[4]={0x80,0x20,0x8,0x2};
BYTE g[6]={0x80,0x40,0x20,0x10,0x8,0x4};
BYTE bitmap[8]={0x80,0x40,0x20,0x10,0x8,0x4,0x2,0x1};

```

```

main()
{ FILE *stream0,*stream1,*stream2,*stream3;
  int i;
  if((stream0=fopen("HZK24K","rb"))==NULL) // Modified
  { printf("\nCannot open the file HZK24K! \n");
    return(1);
  }
  if((stream1=fopen("HZK24T","rb"))==NULL) //Modified
  { printf("\nCannot open the file HZK24T! \n");
    return(1);
  }
  if((stream2=fopen("12.PNT","wb"))==NULL)
  { printf("\nError on openning file 12.PNT for writing! \n");
    return(1);
  }
}

```

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，微电脑靠其内部的储能电容通常能维持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
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

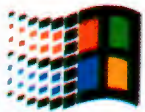




Microsoft WIN32

SOFTWARE DEVELOPMENT KIT

Tools for Writing 32-Bit Applications for Windows.



MICROSOFT
WINDOWS[®]
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 Windows[™] and Windows NT[™]


```

if((stream3=fopen("12. ASC", "wb"))==NULL)
{ printf("\nError on openint file 12. ASC for writing! \n");
return(1);
}
for(i=0;i<12;i++)
result __ w[i]=0;
fwrite(result __ w, sizeof(char), 12, stream3);
fseek(stream1, (94 * 9 * 72), SEEK __ SET);
for(i=0;i<94;i++)
{ fread(s, sizeof(char), 72, stream1);
Reverse(ss, s);
Converter12(ss, d);
ConverterW();
if(! fwrite(result __ w, sizeof(char), 12, stream3))
{ printf("\nError on writing data to file 12. ASC\n");
break;
}
}
fseek(stream1, 0, SEEK __ SET);
while(fread(s, sizeof(char), 72, stream1))
{ Reverse(ss, s);
Converter12(ss, d);
ConverterC();
if(! fwrite(result __ c, sizeof(char), 18, stream2))
{
printf("\nError on writing data to file 12. PNT\n");
break;
}
}
while(fread(s, sizeof(char), 72, stream0))
{ Reverse(ss, s);
Converter12(ss, d);
ConverterC();
if(! fwrite(result __ c, sizeof(char), 18, stream2))
{ printf("\nError on writing data to file 12. PNT\n");
break;
}
}
return(0);
}

```

/* 函数 Reverse 将 24 点阵字库转置 */

```

void Reverse(char * str1, char * str2)
{
WORD i, j, ik, i1, jk, j1;
for(i=0;i<72;i++)
* (str1+i)=0;
for(i=0;i<24;i++)
{ ik=i&0x7;
i1=i>>3;
for(j=0;j<24;j++)
{ jk=j&0x7;
j1=j>>3;
if(* (str2+j * 3+i1) & bitmap[ik])
* (str1+i * 3+j1) |= bitmap[jk];
}
}
}

```

/* 函数 ConverterC 将 12 点阵汉字字模从 24 字节压缩为 18 字节 */

void ConverterC(void)

```

{
int i, j=0;
BYTE tchar;
for(i=0;i<24;i++)
{
* (result __ c+j)=* (d+i);
i++; j++;
* (result __ c+j)=* (d+i);
i++;
tchar=(* (d+i));
* (result __ c+j) &= 0xf0;
* (result __ c+j) |= (tchar>>4);
tchar<<=4;
i++;
j++;
tchar|=(* (d+i)>>4);
* (result __ c+j)=tchar;
j++;
}
}

```

/* 函数 ConverterW 将 12 点阵西文字模从 24 字节压缩为 12 字节 */

```

void ConverterW(void)
{
int i, j;
for(i=0;i<12;i++)
result __ w[i]=0;
for(i=0;i<12;i++)
{
for(j=0;j<4;j++)
if(d[i * 2] & f[j])
result __ w[i] |= g[j];
for(j=4;j<6;j++)
if(d[i * 2+1] & f[j-4])
result __ w[i] |= g[j];
}
}

```

/* 函数 Converter12 将 24 点阵汉字字模从 72 字节压缩为 24 字节的 12 点阵字模 */

```

void Converter12(char * s, char * d)
{ int i, j, k, t;
for(i=0;i<12;i++)
{
d[i * 2]=0; d[i * 2+1]=0;
for(j=0;j<3;j++)
{
t=0;
for(k=0;k<8;k+=2)
{ if(s[i * 6+j] & bitmap[k])
t++;
if(s[i * 6+j] & bitmap[k+1])
t++;
if(s[i * 6+j+3] & bitmap[k])
t++;
if(s[i * 6+j+3] & bitmap[k+1])
t++;
if(t>=2)
{ if(j==0)
d[i * 2] |= bitmap[k/2];

```

```

if(j==1) d[i*2]=bitmap[4+k/2];
if(j==2) d[i*2+1]=bitmap[k/2];
    }
    }
}
}

/* 程序 3:CREATE.C 此程序用于生成应用程序专用的汉字库及其索引库 */
#define TOTAL __NUMBER 1000

#include <io.h>
#include <stdio.h>
#include <fcntl.h>
#include <dos.h>

typedef unsigned char BYTE;
typedef unsigned int WORD;
WORD GetChinesePosition(BYTE c1,BYTE c2);
WORD Length,DataNumber,DataLength,
    ChineseIndex[TOTAL __NUMBER];
BYTE TempData[18];
FILE * Handle;

main(int argc,char * argv[])
{
    FILE * stream,* streamDATA;
    BYTE c1,c2;
    WORD temp,i;
    unsigned long int ltemp;
    if(argc!=2)
    {
        printf("\n\nThe input format is CREATE FILE __NAME. \n");
        return(0);
    }
    if((Handle=fopen(argv[1],"rb"))==NULL)
    {
        printf("\n\nNo %s file. \n",argv[1]);
        return(0);
    }
    Length=0;
    ChineseIndex[0]=0;
    while(Length<TOTAL __NUMBER)
    {
        fread(&c1,sizeof(char),1,Handle);
        if(c1==0)
            break;
        if(c1>247 || c1<161)
            continue;
        fread(&c2,sizeof(char),1,Handle);
        if(c2>247 || c2<161)
            continue;
        // Modified
        /* if(GetChinesePosition(c1,c2)<=Length+1) Modified <=
        Change to >=continue; */
        ChineseIndex[Length+1]=(unsigned int)c1*256+c2;
        //Modified c1<<8
        Length++;
    }

```

```

ChineseIndex[0]=Length;
stream=fopen("12.INX","wb");
if(stream==NULL)
{ printf("\n\nError on open 12.INX file. \n");
return(0);
}
fclose(Handle); //Modified Add Line
fwrite(ChineseIndex,sizeof(int),Length+1,stream);
fclose(stream);
streamDATA=fopen("12.CHN","wb");
if(streamDATA==NULL)
{ printf("\n\nError on 12.CHN file. \n");
return(0);
}
stream=fopen("12.PNT","rb");
if(stream==NULL)
{ printf("\n\nError on open 12.CHN file. \n");
return(0);
}
for(i=0;i<Length;i++)
{ temp=ChineseIndex[i+1];
c1=(temp/256)-161;
c2=temp&0xff-161;
ltemp=(unsigned long int)(94*c1+c2)*18;
fseek(stream,ltemp,SEEK __SET);
fread(TempData,sizeof(char),18,stream);
fwrite(TempData,sizeof(char),18,streamDATA);
}
printf("\n\nThe total Chinese characters are %d.",Length);
return(0);
}
/* 计算一汉字在索引库中的位置 */
WORD GetChinesePosition(BYTE c1,BYTE c2)
{
    WORD s __SEG,s __OFF;
    s __SEG=FP __SEG(ChineseIndex);
    s __OFF=FP __OFF(ChineseIndex);
    asm { push es
        push cx
        push di
        mov ax,s __SEG
        mov es,ax
        mov di,s __OFF
        mov ah,c1
        mov al,c2
        mov cx,Length //Modified
        inc di // Modified Add Line
        inc di // Modified Add Line
        push cx
        repnz scasw
        pop ax
        sub ax,cx
        dec ax
        pop di
        pop cx
        pop es
    }
    return(__AX);
}

```

PostScript 文件格式分析和调用

□ 邵 刚

在

进行某些系统软件的开发时,有时会遇到显示或打印大汉字的问题。虽然许多汉字系统可以显示或打印大汉字,但一般汉字系统采用的是点阵字库,放大以后显示或打印会产生锯齿、比例失调等现象,且不可在自己的系统中直接打印或显示。方正 Super V、VI 汉卡采用了 PostScript 字体技术,它以三次曲线拟合汉字,因此无论放大字均不会变形。而在自己的系统中要利用这些字库,关键在于要知道字库的结构,以下就 PostScript 文件格式进行探讨说明。

一、字库文件格式

下面给出了 WPS 6.0 及方正 WPS NT 的软、硬字库结构。

软字库文件由两部分组成,前一部分为一个数组,每个元素占三个字节,一共 24 位,指出了对应的汉字在字库文件中的绝对偏移量。这三个字节进行了加密,设三个字节依次为 A、B、C,则解密算法为:

$$A = A \text{ 右循环移 } 3 \text{ 位}$$
$$B = B \text{ 左循环移 } 3 \text{ 位}$$
$$C = C - A - B$$

字库的第二部分为字模数据体,每个汉字的字模由第一部分数组中的索引值指定字模数据在文件中的绝对偏移量,相邻两个偏移量的差值便为字模数据的长度。在字库中的字模数据还不是真正的解释时用到的字模数据,因为该字模数据经过了压缩加密处理,在解释字模数据前再对其进行还原解密,这样一方面可以压缩字库、节省磁盘空间;另一方面,又可以起到加密的作用。

硬卡字库的结构基本上和软字库相同,但所有的字库同存在于一张硬卡上,因此在数据区中有一张硬卡字库的起始地址表,分别对应各种字体在硬卡上的起始地址,同时在硬卡上还存在着各个字体的字根字模库,同样也存在着一张地址表,指出各种字体的字根在硬卡上的起始地址。硬卡中的每个字库也由两部分组成,前一部分为指针数组,每一元素给出了对应汉字相对于字库首的偏移量,但这里的三个字节没有加密,后一部分给出了压缩加密过的字模数据。

二、数据解密算法及解密后的数据结构

加密数据的解码可由一个子程序来完成,它需要一个解密数据表,入口为加密数据的首地址。具体算法见本文所附程序中的有关解密部分。软字库和硬卡字库汉字字模经过解密以后存入目标数据缓冲区,解密后的字模结构如下:

标志字节,若干字节数据,标志字节,若干字节数据……

字模数组的长度存在 length 中,标志字节决定了跟在它后面的数据字节的个数和含义。字节数据为坐标采样点在基本字模上的坐标值。标志字节以及跟在它后面的数据的个数和含义如表所示。

当标志字节大于 8,它后面的参数描述的并不是该汉字的外轮廓线数据,而是组成该汉字的字根的编号,以及这个字根在该汉字中的位置和进行的放大缩小比例,对于宋体、仿宋体、楷体、黑体,字体比较正规,不连贯,因此,很容易拆成字根来存储。所以汉卡上同时又存

标志字节值	其后数据个数	含 义
0	2	基本坐标
1	1	上一点和本点之间连线为一条横线,因此本字节只是一个横坐标值
2	1	上一点和本点之间连线为一条竖线,因此本字节只是一个纵坐标
3	2	一个点的坐标,与上一点连成一条直线
4	4	两个点的坐标,和上一点共三点之间画一条曲线
5	6	三个点的坐标,和上一点共四点之间画一条曲线
6	4	两个点坐标,这两个点为矩形的两个顶点
7	2	保留
8-0FEH	4	字根编号和字根位置以及缩放比例
0FEH	5	字根编号和字根位置以及缩放比例
0FFH	5	字根编号和字根位置以及缩放比例

在一个字根库,其中存放了字根的轮廓描述数据,每一个字根都有一个编号,一共有七百多个字根。但对于行楷、魏碑、标宋、隶书字体书写的时候都是连笔,不能拆成字根来描述。当标志字节值在 8-0feh 之间时,标志字节值减 7,即为字根的编号;标志字节值为 0FEH 时,后面第 5 个字节的值加上 0F7H 为字根编号。标志字节值为 0FFH 时,后面第 5 个字节的值加上 01F7H 为字根编号。

三、程序实现及说明

由于篇幅的限制,这里只给出软字库的解密显示的具体过程,对于硬字库,只要将硬卡当中的数据映象读出,再解密显示即可。本文程序用 Borland C++ 实现,但读者很容易将其转化到其他系统上实现。

需要说明的是,在 showps 中,w 为显示字的宽度,h 为显示字的高度,spin 为显示字的旋转角度,gradient 为显示字的倾斜角度,color 为显示字的颜色,rec 决定是否画边框。

源程序清单如下:

```
#include <mem.h>
#include <math.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <iostream.h>
#include <graphics.h>

#define max(x,y) (x>y)? x:y
#define abs(x,y) (x>=y)? (x-y):(y-x)
#define PI 3.1415926
```

```
unsigned char buffer[2000]; //存放解码数据
long length; //存放解码数据的长度
```

```
class analysis {
    unsigned char buffer0[1000]; //存放原码数据
    unsigned char data,flag; //解码辅助变量
    int bp0,bp; //数据指针
    unsigned ror(unsigned char x);
    unsigned rol(unsigned char x);
    void decode(void);
    public: void pretation(char * code);
} first;

class showchinese {
    int nowx,nowy; //坐标辅助指针
    double rotsin,rotcos,rotsc,gradtg,gradtgx;
    int width,height; //字形变化变量
    void cubic __ interpolation(int x1,int y1,int x2,int y2,
        int x3,int y3);
    void fourth __ interpolation(int x1,int y1,int x2,int y2,
        int x3,int y3,int x4,int y4);
    void translate(int oldx,int oldy,int * newx,int * newy);
    public:
    void showps(int x1,int y1,int w=168,int h=168,int spin=0,
        int gradient=0,int color=10,int rec=1);
} second;

unsigned analysis::ror(unsigned char x)
{
    //循环左移
    return (((x&0x07)*0x20)|((x&0xf8)/0x08));
};

unsigned analysis::rol(unsigned char x)
{
    //循环右移
    return (((x&0xe0)/0x20)|((x&0x1f)*0x08));
};

void analysis::decode(void)
{
    //辅助解码函数
    flag=~flag;
    data=(flag)? buffer0[bp0]/16: buffer0[bp0++];
};

void analysis::pretation(char * code)
{
    //解码函数
    unsigned char temporary;
    unsigned char cipher __ table[]={2,1,1,2,4,6,4,2,5,5,5,0xfd,
        0x19,0x59,0,0xff
    };
    char * filename;
    FILE * fp;
    int loop;
    if (code[0]<0xd8) { //根据汉字内码确定打开文件
        length=((unsigned char)code[0]-0xb0)*0x5e+(unsigned
            char)code[1]-0xa1)*0x03;
        filename="WBDOT.PS1"; //缺省字库为魏碑字库
    }
    else {
        length=((unsigned char)code[0]-0xd8)*0x5e+(unsigned
            char)code[1]-0xa1)*0x03;
        filename="WBDOT.PS2"; //用户可根据实际需要修改
    };
    if ((fp=fopen(filename,"rb"))==NULL) {
        cout<<"\nThe file "<<filename<<" NOT found! \n";
    }
}
```

```

exit(0);
}
fseek(fp,length,SEEK_SET); //定位文件指针
fread(buffer0,6,1,fp); //读取原字模长度位移数据
length=(unsigned char)(buffer0[0]+buffer0[1]+buffer0[2])*
0x10000L+long(ror(buffer0[1])*0x100+rol(buffer0[0]));
//原字模位置
fseek(fp,length,SEEK_SET); //定位文件指针
length=long(rol(buffer0[3])-rol(buffer0[0])
+(ror(buffer0[4])-ror(buffer0[1]))*0x100); //原字模长
memset(buffer,0,500);
memset(buffer0,0,500);
fread(buffer0,length,1,fp); //读取原字模数据
fclose(fp);
bp=0;
bp0=0;
flag=0; //解码
decoding1:
decode();
temporary=data=int(data)&0x0f;
if (data<=0x06)
goto decoding3;
if (data<=0x08)
goto decoding4;
if (data==0x00)
goto decoding2;
temporary=0xff;
goto decoding1;
decoding2:
temporary=0xfe;
decoding3:
buffer[bp++] = temporary;
decoding4:
temporary=0;
loop=cipher[CD # 1]table[data];
for (;loop;loop--) {
decode();
temporary=data*16;
decode();
data=(data&0x0f)|temporary;
buffer[bp++] = data;
};
if (data) goto decoding1;
length=bp-3; //确定字模长度
};

void showchinese::cubic __ interpolation(int x1,int y1,int x2,int y2,
int x3,int y3)
{ //三点二次插值
int loop,x4,y4,k0;
double k1,k2,k3,k4,k5,k6,k7,k8,k9;
k0=(max(abs(x1,x2),abs(y1,y2))+(abs(x3,x2),abs(y3,
y2)))/4;
if (k0<=3) k0=3;
k1=double(1)/double(k0);
k2=double(x1+x3-2*x2)*k1*k1;
k3=double(y1+y3-2*y2)*k1*k1;
k4=k2+2*double(x2-x1)*k1;
k5=k3+2*double(y2-y1)*k1;
k6=2*k2;
k7=2*k3;

```

```

k8=double(x1);
k9=double(y1);
for (loop=1;loop<=k0;loop++) {
k8+=k4;
k9+=k5;
k4+=k6;
k5+=k7;
if (k8<=0) x4=0;
else
x4=(k8>0x7fff)? 0x7fff:unsigned(k8);
if (k9<=0) y4=0;
else
y4=(k9>0x7fff)? 0x7fff:unsigned(k9);
lineto(x4,y4);
};
};
void showchinese::fourth __ interpolation(int x1,int y1,int x2,int y2,
int x3,int y3,int x4,int y4)
{ //四点三次插值
int x5,y5,loop,k0;
double k1,k2,k3,k4,k5,k6,k7,k8,k9,k10,k11,k12,k13;
k0=(max(abs(x2,x3),abs(y2,y3))+max(abs(x2,x1),abs(y2,
y1))+max(abs(x3,x4),abs(y3,y4)))/4;
if (k0<=3) k0=3;
k1=double(1)/double(k0);
k2=double((x2-x3)*3+x4-x1)*k1*k1;
k3=3*double(x1+x3-2*x2)*k1*k1;
k4=double((y2-y3)*3+y4-y1)*k1*k1;
k5=3*double(y1+y3-2*y2)*k1*k1;
k6=k2*k1+k3+3*double(x2-x1)*k1;
k7=k4*k1+k5+3*double(y2-y1)*k1;
k8=6*k2*k1;
k9=6*k4*k1;
k10=k8+2*k3;
k11=k9+2*k5;
k12=double(x1);
k13=(double)y1;
for (loop=1;loop<=k0;loop++) {
k12+=k6;
k13+=k7;
k6+=k10;
k7+=k11;
k10+=k8;
k11+=k9;
if (k12<=0) x5=0;
else
x5=(k12>0x7fff)? 0x7fff:unsigned(k12);
if (k13<=0) y5=0;
else
y5=(k13>0x7fff)? 0x7fff:unsigned(k13);
lineto(x5,y5);
};
};
void showchinese::translate(int oldx,int oldy,
int *newx,int *newy)
{ //坐标变换函数
int k1,k2,k5,k6;
double k3,k4;
k1=(oldx-0xa8/2)*width/0xa8;
k2=(oldy-0xa8/2)*height/0xa8;
k3=(double)(k1+gradtg*k2)/gradtgx;

```

```

k4=(double)k2/gradtgx;
k5=(double)(k3*rotcos-k4*rotsin)/rotsc;
k6=(double)(k4*rotcos+k3*rotsin)/rotsc;
*newx=k5+width/2; *newy=k6+height/2;
};
void showchinese::showps(int x1,int y1,int w,
    int h,int spin,int gradient,int color,int rec)
{
    //显示内存中数据结果
    int x2,y2,x3,y3,x4,y4,loop=0;
    double rotsc1,rotsc2;
    width=w;
    height=h;
    gradtg=fabs(tan(gradient*PI/180));
    gradtgx=((double)width+gradtg*(double)height)/
        (double)width;
    rotsin=fabs(sin((double)spin*PI/180));
    rotcos=fabs(cos((double)spin*PI/180));
    rotsc1=(double)width*rotsin/(double)height+rotcos;
    rotsc2=(double)height*rotsin/(double)width+rotcos;
    rotsc=max(rotsc1,rotsc2);
    setcolor(color);
    setviewport(0,0,getmaxx(),getmaxy(),0);
    if (rec) rectangle(x1,y1,x1+168,y1+168);
    setviewport(x1,y1,x1+width,y1+height,1);
    while(loop<length) //解释字模
    switch (buffer[loop++]) {
        case 0: //定义基点
            nowx=buffer[loop++];
            nowy=buffer[loop++];
            translate(nowx,nowy,&nowx,&nowy);
            moveto(nowx,nowy);
            break;
        case 1: //画横线
            x1=buffer[loop++];
            translate(x1,nowy,&nowx,&nowy);
            lineto(nowx,nowy);
            break;
        case 2: //画纵线
            y1=buffer[loop++];
            translate(nowx,y1,&nowx,&nowy);
            lineto(nowx,nowy);
            break;
        case 3: //画直线
            x1=buffer[loop++];
            y1=buffer[loop++];
            translate(x1,y1,&nowx,&nowy);
            lineto(nowx,nowy);
            break;
        case 4: //三点二次插值
            x1=nowx;
            y1=nowy;
            x2=buffer[loop++];
            y2=buffer[loop++];
            x3=buffer[loop++];
            y3=buffer[loop++];
            translate(x2,y2,&x2,&y2);
            translate(x3,y3,&nowx,&nowy);
            cubic __ interpolation(x1,y1,x2,y2,nowx,nowy);
            break;
        case 5: //四点三次插值

```

```

            x1=nowx;
            y1=nowy;
            x2=buffer[loop++];
            y2=buffer[loop++];
            x3=buffer[loop++];
            y3=buffer[loop++];
            x4=buffer[loop++];
            y4=buffer[loop++];
            translate(x2,y2,&x2,&y2);
            translate(x3,y3,&x3,&y3);
            translate(x4,y4,&nowx,&nowy);
            fourth __ interpolation(x1,y1,x2,y2,x3,y3,nowx,nowy);
            break;
        case 6: //画一个框
            x1=buffer[loop++];
            y1=buffer[loop++];
            x2=buffer[loop++];
            y2=buffer[loop++];
            translate(x1,y1,&x1,&y1);
            translate(x2,y2,&x2,&y2);
            rectangle(x1,y1,x2,y2);
            break;
    };
};
int main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"");
    first.pretation("电");
    second.showps(152,0);
    first.pretation("脑");
    second.showps(316,0);
    first.pretation("编");
    second.showps(0,188,150,100,0,10,11,0);
    first.pretation("程");
    second.showps(100,188,150,100,0,20,12,0);
    first.pretation("技");
    second.showps(200,188,150,100,0,30,13,0);
    first.pretation("巧");
    second.showps(300,188,150,100,0,40,4,0);
    first.pretation("与");
    second.showps(200,300,100,100,0,0,15,0);
    first.pretation("维");
    second.showps(300,250,150,200,10,0,6,0);
    first.pretation("护");
    second.showps(400,250,150,200,20,0,14,0);
    getch();
    closegraph();
    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

一个实用的排版预处理程序

□葛晓滨

批

处理式的中西文排版软件《科印》在国内拥有众多的用户,该排版软件具有较强的排版控制功能,适用于普通类型的微机是一个很实用的排版软件,并且它的文字处理和排版处理工作可分离操作,使用户可以借助系统外部的字处理软件完成文稿的文字处理工作。但如同其他的批处理排版软件,《科印》采用了普通的 ASCII 码作为排版控制的命令符,如用“\$”表示排版自然段结束,用“&”表示取消段前空等,这就使得文稿中凡涉及到这些控制字符的文字部分必须采用纯中文的存储方式,否则在文稿排版输出时会产生混乱,直接影响排版效果。

在通常的文稿录入过程中,用户由于录入环境的限制或因时间上的要求,一般较难顾及这种要求,而将控制字符的转换工作留待录入阶段结束后完成,使得这个阶段工作量增大,而且极易产生错误。为此,笔者用 C 语言编制了一个实用的小程序:PROCESS. C,它可以代替人工简单快速地将文稿中所涉及到的排版控制符转换为文本字符,方便了《科印》用户的文字处理工作。

PROCESS. C 程序的编译环境为 Turbo C 或 Borland C++,经编译器编译生成排版预处理程序 PROCESS. EXE,用户可在文稿录入结束后,用 PROCESS 程序对文稿进行处理,将文稿中的控制字符转换为文本字符,同时,即使所处理的文稿中不含排版控制符,PROCESS 程序也不会对源文稿产生任何损害,用户可将 PROCESS 作为《科印》系统中的一个辅助的工具软件。

本程序的设计思想也可用于设计其他的

一些批处理排版软件的文稿预处理程序,如《华光》批处理排版软件等,用户只需将 PROCESS 程序中对应的排版控制符替换为相应的批处理排版软件的排版控制符即可。

源程序清单:

```
#include<stdio.h>
#define TRUE 1
#define FALSE 0
#main(int argc,char *argv[])
{ FILE *fp1,*fp2;
  int readbit,in,i;
  int table1[]={0x21,0x23,0x24,0x26,0x2a,0x2f,0x3f,0x40};
  int table2[]={0xa1,0xa3,0xa4,0xa6,0xaa,0xaf,0xbf,0xc0};
  clrscr();
  if(argc!=3){
    printf("\007 用法:%s 源文件名 目标文件名\n",argv[0]);
    exit(1);
  }
  if((fp1=fopen(argv[1],"r+b"))==NULL){
    printf("无法打开源文件:%s",argv[1]);
    exit(1);
  }
  if((fp2=fopen(argv[2],"w+b"))==NULL){
    printf("无法打开目标文件:%s",argv[2]);
    exit(1);
  }
  while((readbit=getc(fp1))!=EOF){
    in=FALSE;
    for(i=0;i<=7;i++){
      if(readbit==table1[i]){
        putc(0xa3,fp2);
        putc(table2[i],fp2);
        in=TRUE;
      }
    }
    if(! in)
      putc(readbit,fp2);
  }
  fclose(fp1);
  fclose(fp2);
  printf("\n 排版文件预处理成功!");
}
```

SKILL

用 C 语言实现造型表功能

□ 麦智祥

早

期使用过 GWBASIC 语言的人都知道造型表这个概念,通过它我们可对自定义的任意造型进行诸如缩放、旋转等各种特技显示,使设计出的界面更生动、友好。

造型表的实现方法是:预先定义好一些代码,其含义分别为往不同的方向移动画笔,并区分开落笔和抬笔的码。通过不同的代码描述,从而将造型在屏幕上勾画出来。对某造型,我们按定义将其编好码,然后给出缩放比例、显示角度等参数,再调用造型表显示程序即可将其显示出来。

本文给出了一个用 C 语言实现造型表程序及使用实例。该程序中预定义的造型编码为:0—7 代表的方向依次为上、右上、右、右下、下、左下、左、左上,8—15 的方向与 0—7 相同,但表示为抬笔。所有的造型编码定义完毕后,以 16 作为结束码,并放于一整形数组内调用显示程序。

源程序清单如下:

```
#include "graphics.h"
#include "math.h"

void shape(int p, int x, int y, int color, int ang, int scale);
main()
{
    int d=DETECT, m=0;
    int i, p[] = {0, 2, 4, 6, 16};
    initgraph(&d, &m, "");
    for(i=0; i<360; i+=5)
        shape(p, 100, 100, YELLOW, i, 30);
    getch();
    closegraph();
}

void shape(int p[], int x, int y, int color, int ang, int scale)
{
    int i, sinx, cosx, lastx, lasty;
```

```
float ang1;
i=0;
ang1=ang * 3.1415926/180;
sinx=sin(ang1) * scale;
cosx=cos(ang1) * scale;
setcolor(color);
moveto(x, y);
while(p[i] != 16)
{
    lastx=x; lasty=y;
    switch(p[i]) {
        case 0:
        case 8:
            x=lastx+sinx; y=lasty-cosx; break;
        case 2:
        case 10:
            x=lastx+cosx; y=lasty+sinx; break;
        case 4:
        case 12:
            x=lastx-sinx; y=lasty+cosx; break;
        case 6:
        case 14:
            x=lastx-cosx; y=lasty-sinx; break;
        case 1:
        case 9:
            x=lastx+.7071*cosx+.7071*sinx;
            y=lasty-.7071*cosx+.7071*sinx; break;
        case 3:
        case 11:
            x=lastx+.7071*cosx-.7071*sinx;
            y=lasty+.7071*cosx+.7071*sinx; break;
        case 5:
        case 13:
            x=lastx-.7071*sinx-.7071*cosx;
            y=lasty+.7071*cosx-.7071*sinx; break;
        case 7:
        case 15:
            x=lastx-.7071*cosx+.7071*sinx;
            y=lasty-.7071*cosx-.7071*sinx; break;
    }
    if (p[i]<8) lineto(x, y);
    else moveto(x, y);
    i++;
}
```

的宏,你也很容易忘记其副作用而导致一些错误。

三、内置函数

内置函数克服了上述缺点。定义内置函数的语法跟普通函数没什么两样,你可利用函数的所有东西:参数、局部变量、多条语句、控制结构等。

上例中的宏 max1 可以用内置函数很容易地重新改写:

```
inline int max1(int a,int b)
{return (a>b ? a : b);
}
```

内置函数同普通函数一样定义,他们同样遵循宏的一些规则。内置函数必须在使用之前事先定义,并且内置函数没有外部连接特性,每个源程序文件中都必须有其定义。内置函数最好在头文件中定义,一般而言是放在 #define 语言的位置,相反标准函数却不能在头文件中定义,因为在头文件中定义标准函数可能会出现重复定义的错误。

这些规则使我们觉得内置函数执行时很象宏。对一个标准函数调用,编译器肯定会产生一个 CALL 指令,因为标准函数会在其他地方有定义,编译连接时,编译器才会去检查函数存在与否。对内置函数,编译器每检查到文件名便需产生一些执行内置函数功能的指令。

另一个内置函数须在头文件中定义的原因是:在 C++ 语言中,在类定义中的函数都是内置的,因而函数定义并不象标准函数定义可以产生一个外连接,却成为类定义的一部分,如:

```
/* prog.h */
class CHore : public CAnimal
{
private:
    int Handicap;
    int Breed;
public:

//inline functions:
    int get __ handicap(void){return Handicap;}
    int get __ breed(void){return Breed;}

//not inline:
    int race __ the __ horse(long date);
};
```

上面的例子中,我们看出 C++ 内置函数的典型用法及一些要点。get __ handicap 和 C++ 中 get __ breed 是内置函数,没有外部连接特性,不会出现重复定义错。如果这些函数被定义成具有外部连接特性的全局函数,你便不能把这种类定义放在头文

件中,否则便会出现重复定义。相反函数 race __ the __ horse 不是一个内置函数,便只能在 .cpp 中定义一次,而不是头文件中。

```
/* module1.cpp */
int CHore::race __ the __ horse(long date)
{
...
}
```

从上面的例子也可看出,为什么内置函数在 C++ 语言中如此有用,因为它可以通过编写一个短的函数来存取私有数据。如,例中 Chouse,它可以包含私有数据和用以存取私有数据的函数。在优秀的 C++ 程序中,这种内置函数会很常见。

四、内置函数节省执行时间

用普通函数而非内置函数去存取数据会耽误很多执行时间。内置函数显然会提高执行速度。但对小函数来讲,内置函数会显著地减小文件的长度。正因为指令是内置的,编译器的优化例程可很好地对大量的程序代码(调用函数包括内置函数代码及其他代码)进行优化,而当仅有标准函数应用时,编译器分别优化每一个函数。

前面我们提到内置函数不存在外连接,而普通函数存在外连接(除非其定义为静态的)。一个重要的例外是使用编译器的 /Ob2 选项,可使 Microsoft C、C++ 编译器自动把函数变为内置的,而源码无须改变。有了 /Ob2 选项,普通函数不必移至头文件,即可在编译时变为内置的。

除非使用优化开关,否则语法和作用域和原有意义是一致的。但无论用什么选项,定义的内置函数不会有外连接而普通函数有外连接(不管是否用 /Ob2 选项)。

对是否用内置函数,除了效率外,我们还要考虑其他一些原因,有些函数并不适合于作为内置函数,如递归函数便不能完全定义为内置函数。无论如何,编译器以正确性为最后标准。对不能内置的函数,编译器会把函数定义为一个独立的过程并在适当时候插入 CALL 指令。

在 Microsoft 的 C 中可使用内置函数,这是 Microsoft 对 C 的扩充,因而只能用具有两条下划线的 __ inline 关键字才行。没有下划线的 inline 关键字只能在 C++ 中使用。

C++ 的特殊符号“;”是对软件高层和低层的综合考虑。内置函数在这方面起了很大的作用。

(本文英文资料由 Microsoft 提供) SKILL

Visual C++的编程技巧—— 使用内置函数

内

置函数(inline function)是一段看似函数、功能也象函数的程序代码序列,但实际上编译器并不把它看作是函数。内置函数不用CALL就能执行。

编译器在调用内置函数的过程体内产生一段代码序列。在某些情况下,内置函数的实现比标准的函数调用的实现效率还要高,但一个内置函数在调用时其语法和语义都会由编译器很好地维护着。

C++的一个优点是其执行和用法是两回事。C++的本质是先弄清楚一个对象如何被使用,再以最优的方式去实现它。内置函数就是C++这一特性的很好体现。

一、C语言的一个特色——宏定义

宏定义是C、C++语言为方便编程人员而引用的一个方法。编程人员在用C或C++语言编程时,对于一些操作经常用宏定义的方法来实现,从而大大地减少了编程人员的繁琐劳动。

1. 宏的使用

采用C语言编程的人经常用宏来定义一些函数。这种方法在执行时间比程序大小更重要的场合是很诱人的,在函数体很短的场合也是吸引人的。如:一个宏可以模拟一个max函数,这个函数返回两个参数中较大的一个:

```
#define max1(A,B)((A)>(B)? (A) : (B))
```

在这个例子中用宏定义的方法求两个数中较大的一个是合适的。C的预处理器在编

译时将把源程序中的所有max1过程以其定义代替。使用宏max1跟使用函数max1完全一致,但宏定义只影响现行编译,编译器并不把max1编译为一个公用符号。每一个引用max1的源文件都必须有max1宏的定义,这同具有外部连接特性的标准函数只须在整个系统中定义一次是截然不同的。

2. 宏的弱点

把宏max1同真正的函数调用相比较,我们就会发现宏有许多弱点:

(1)我们知道,在C语言的预处理阶段并没有类型检查。正是这种原因,宏max1能接受任何错误的变量。类型错误只能在后期编译或运行时发现,此时再检查程序的错误是困难的。因为不能深入宏定义内部集成调试程序(debugger)也不能为程序的查错带来多大帮助,宏定义的复杂逻辑(或许采用传统的操作符创建控制流)很难去查错。

(2)宏定义通常是实现一个表达式,此时,宏有一个返回值。但编译器并没对返回值做检查。

(3)正象max1一样,宏定义都需要额外的括号。否则,宏的结果很难预料。在子表达式A和B中的操作符比通过宏调用产生的操作符的优先级高。宏的最大缺点可能是,在宏定义里参数每出现一次即执行一次,这就意味着这可能带来副作用(如变量a++),这种副作用可能出现多次。

(4)如果宏的调用者并不知道宏是如何写的(如宏是在复杂的头文件中由第三方提供的),错误可能会出现得更频。即使你自己定义

用 Microsoft VB 编程

下

下面我们讲述六个方面的 VB(Visual Basic)编程技巧。

一、VB3.0 的兼容性模块

VB3.0 能存取 Microsoft Access1.1 的数据,VB3.0 兼容性模块可解决存取 Access 2.0 数据的问题。兼容性模块安装在 VB3.0 之上,但 VB.INI 和 SETUP.INI 需作适当修改。这个模块可通过以下几种方法得到:ODK、Microsoft Access 的开发者工具库、CompuServe。

Access 2.0 中的 SQL 语句中增加了几个新的保留字,因而如果你的现行 VB 程序中嵌入了 Access 的 SQL 语句,并且有字段名为 Note(Note 为关键字),你便需要修改你的 SQL 语句。只需在冲突的各字周围加上括号即可。

二、VB 中栈的实现

VB 没有指针类型并不意味着不能构造许多有意义的数据结构。下面的程序构造了一个跟踪鼠标的栈:

```
Module level dclarations
Dim MStack() as Integer 'Mouse stack
Dim MStackTOS as Integer 'TOS=TOP of Stack
Sub InitMouse()
    ReDim MStack(20) 'Initial Stack Size
    MStackTOS=-1 'Initial TOS Value
End Sub
Sub PushMouse (NewMValue as Integer)
    'Expand stack if necessary
    If MStackTOS=UBound(MStack) Then
        ReDim Preserve
        MStack(UBound(MStack)+10)
```

```
End If
'Store existing mouse cursor value
MStackTOS=MStackTOS+1
MStack (MStackTOS)=Screen.Mousepointer
'Change mouse cursor
Screen.MousePointer=NewMValue
End Sub
Sub Popmouse()
'POP value off stack
If MStackTOS>=0 Then
    Screen.MousePointer=
    MStack(mStackTOS)
    mStackTOS=mStackTOS-1
End If
End Sub
```

对鼠标栈进行初始化,只需调用 InitMouse,栈初始化后,便可用下列程序使用此栈:

```
PushMouse HOURGLASS
PopMouse
```

请注意 PushMouse 是如何在需要时对数组重新定义的,在 PopMouse 中加入类似的程序可减少数组的大小(维数)。

三、Combo Box 的技巧

在一个底色为灰色的表格(form)上设置一个 Combo Box (Style=0),怎样去掉编辑域和按钮之间的空白间隔呢?按以下步骤便可做到:

1. 在表格中设置一个 Picture Control;
2. 把它的背景色置为灰色;
3. 把它的 borderStyle 属性置为“0—None”;
4. 使它同空白地方一样大;
5. 把它挪至 Combo Box 之上。

四、在 List Box 中加入 Tabs

这个问题在 VB 程序中经常见到,问题是如何在 List Box 中显示长度相等的一串字符,下面的 Windows API 调用给出了答案。VB3.0 没有读行符,所以下面的程序在键入时必须一行接一行。

```
Declare Function SendMessage Lib "User"
    (ByVal hwnd As Integer, ByVal wMsg As
    Integer, ByVal wParam As Integer, lParam
    As Any) As Long
Const WM__USER = &H400
Const LB__SETTABSTOPS = (WM__USER + 19)
Const CH__tab = CHR$(9)
Sub SetTab()
    Dim temp As Long
    Dim NumOfTabs As Integer
    ReDim Tabs(3) As Long
    list1.AddItem "One" + CH__tab + "Two"
    NumOfTabs = 4
    Tabs(0) = 50
    Tabs(1) = 90
    Tabs(2) = 110
    Tabs(3) = 140
    temp = SendMessage (list1.hWnd,
        LB__SETTABSTOPS, NumOfTabs, Tabs(0))
    list1.Refresh
End Sub
```

建议用户多花时间去学习 Windows API,而这可从任何一本关于 VB 中的 Windows API 的书开始。

五、查询的优化

如果你需要对 Access 库做大量单库查询的话可以用"Seek"来代替 Microsoft Access 的 SQL 或 QueryDef,原因是"Seek"不需要复杂的查询处理,当然这适用于文件共享的数据库,如 FoxPro 和 Microsoft Access 而不适用于 SQL Server 或其他客户机/服务器下的数据库。

下面的程序在 486/66 上用了 406 秒:

```
Dim db As database
Dim qb As queryDef
Dim s As snapshot
Dim i As Integer
Dim x As Variant
Set db = OpenDatabase("C:\test.mdb")
Set qd = db.OpenQueryDef("FetchPrice")
For i = 1 To 10000
    qd!PartParam = RandomSearchstring()
    Set s = qd.CreateSnapshot()
    price = S!price
Next i
S.Close
qd.Close
db.Close
```

相反,下面一段程序只用了 20 秒,而它却完成了相同的功能。

```
Dim db As database
Dim t As table
Dim i As Integer
Dim a As Integer
Dim b As Integer
Dim c As Integer
Dim x As Variant
Set db = OpenDatabase("c:\test.mdb")
Set t = db.OpenTable("Parts")
t.Index = "Part #"
For i = 1 To 10000
    t.Seek = "RandomSearchString()"
    Price = t!price
Next i
t.Close
db.Close
End Sub
```

上面的程序在易读性、封装性、稳定性上有欠缺。SQL 在多表操作时有优势,Seeks 在单表简单查询时有优势。

六、平滑动画

为了实现动画,你可能欲购买 Windows 环境下的 Microsoft Video 但我们经常在无这些工具时,也能在封面或帮助文件上做些动画。下面是动画的步骤:

1. 在表格上置一个名为"picShow"的 Picture Control;

2. 在表格上再置一个名为"picDraw"的 Picture Control 把它的 autoRedraw 属性置为真,可见属性置为假要保证两幅图大小一样。下面的子程序 DrawFrame 画每一幅图:

```
Sub DrawFrame(pic as control, f as Frame)
End Sub
```

下面程序的结果便是一个简单的动画,因为用户每次只能看见一帧。

```
Declare Sub BitBlt Lib "GDI" (ByVal hDestDC As
    Integer, ByVal x As Integer, ByVal y As
    Integer, ByVal nWidth As Integer, ByVal
    nheight As Integer, ByVal hSrcDC As
    Integer, ByVal xSrc As Integer, ByVal Ysrc
    As Integer, ByVal dwRop As Long)
For Frame = 0 TO maxFrame
    PicDraw.Cls
    DrawFrame picDraw, Frame
    BitBlt picShow.hDC, 0, 0,
    picShow.Scalewidth,
    picShow.ScaleHeight,
    picDraw.hDC, 0, 0, &HCC0020
Next Frame
```

(本文英文资料由 Microsoft 提供) **SKILL**

磁道接缝软件加密技术

□林平卫

防

止非法拷贝技术是计算机安全技术特别是加密技术的一个重要组成部分,本文以磁道接缝为基础,全面详细地介绍了一种特殊的加密技术——磁道接缝软件加密技术。

一、综述

防拷贝软件的可靠软加密通常由下面三步组成:1. 寻找当今技术不可控制的可靠随机量,制成可读取的参量。2. 根据读取的参量值,制作建立在随机数基础上的密码。3. 在源程序中编制具有有效反跟踪措施的软件来读取参量的值和密码,根据两者是否符合来判断程序使用的合法性。

激光加密、掩膜加密盘中“孔”的位置在制作时是一不可控制的随机量,但是一旦制成,却又是一相对稳定可靠的参量,可用软件精确读取。磁道接缝在每次格式化时,接缝信息都会因驱动器转速极其微小的变化而有所不同,在软件读取时,其信息却是稳定、可靠、保持不变的。所有在制作时不可控制、制成后稳定可靠的量,具有不可复制性,原则上都可用来实现可靠加密。由于磁道接缝加密使用普通软盘在PC机上即可制作,具有简便、可靠的特点,是一种极好的加密方法。下面按软件加密的基本模式,来详述磁道接缝加密。

二、磁道接缝的制作

1. 磁道接缝的含义、位置和信息

我们知道一个磁盘从外到内可分为许多同心圆,称为磁道。由于磁道是圆形的,因此必然有头尾相接的地方,这种接缝处就包含了某种特殊的信息。我们可以利用这种不可

复制的信息来加密软件。这种基本思想在文献中有提及,但未能作深入细致的分析,有些提法也有不妥之处。笔者在对接缝作较深入的分析基础上,对自行研制的软件系统实现了可靠的加密。经过近三年在不同机型、不同版本DOS下运行的实践表明,磁道接缝加密稳定可靠。下面我们对磁道接缝作详细地分析。

一个磁道由若干个扇区组成,每个扇区大致可分为两个区域:识别区和数据区。一个磁道的开头处总是放若干字节的4E,而磁道的结尾处也有若干字节的4E。由于接缝的存在,当我们通过超长扇区的方法跨越接缝读磁道开头处若干字节的4E时,读得的内容是4E的变形或90(所有数据除特别注明外均为十六进制)的变形。当我们将若干字节的4E用二进制写出再任意切取连续的8位组成一字节,其值总是27、93、C9、E4、72、39、9C、4E,这8个数称为4E的变形。同样21、42、84、09、12、24、48、90是90的变形。4E变形的出现是由于写到最后一个字节时,破坏了磁道最开始字节的完整性,发生错位,使得字节重新组合。对于MFM制的磁盘信息编码,存放4E这个字节时需写入二个时钟位,若将数据位与时钟位颠倒(错1/4数据编码位),则读出的内容就是90。上述10个数出现的几率均等,为1/10。

2. 加密软盘的制作

PC机DOS中断13(或40)的05功能可用来格式化(规划)一磁道。中断13使用中断1E为磁盘机参数表指针。参数表内容通常如下:(0)DF步进电机加速和去载时间(ms);(1)02磁头加载时间(ms);(2)25磁头等待时间(ms);(3)02磁头数N,每扇区的字节数

为 80×2^N ; (4)0F 磁道的最后扇区号码或格式化时的扇区数; (5)1A 磁区间隙; (6)FF 资料长度; (7)54 规划所需磁隙长度; (8)F6 补白位元(格式化时填入数据区的内容); (9)01 磁头稳定时间(ms); (A)08 马达启动时间(ms)。

调用 INT 13 的 05 功能格式化磁道时: DL=驱动器号(0-3); DH=面号(0-1); CH=道号(0-27); AH=05。而规划时所需的识别数据则从 ES: BX 指定地址按磁道(T)、磁头(H)、扇区(S)、磁数(N)四个一组依序排列。为了格式化出一特殊的磁道,四个数的取值可从 0 到 FF 任意选取。

例如:将 ES:BX 指向单元内容放入 8F、73、B8、8A; 0F、00、B7、5F; 0F、0、F7、3; 0F、0、E9、06; 0F、0、E3、6; 将磁盘机参数的第 3 项有关扇区长度的磁数改为 3, 第 4 项扇区数改为 5, 而对第 7 项规划所需磁隙长度也可作适当修改。则:

```
MOV AH,05
MOV DX,0001
MOV CH,0F
INT 13
```

执行后,在第 0F 道 0 面得到 5 个扇区。第一个扇区无法正常读写,第 5 个扇区只有将磁盘机工作参数第 3 项 N 改为 6,并将 E3 这一扇区号送到 CL 中才能找到并读出数据。

```
MOV AX,0201
MOV DX,0001
MOV CX,0FE3
INT 13
```

则将 0F 道第 0 面第 5 个扇区的内容读到 ES: BX 指向的单元中。读出数据长度为 80×2^6 字节,读出内容已跨越了接缝,因此包含了接缝信息。把读出的内容显示在屏幕上或存入一文件中再编辑,就可得知接缝的详细状况。由于磁盘机转速有些变化,因此,同样的程序格式化一磁道后,不但接缝信息可能不一样,而且接缝的具体位置会发生几个字节的变化,特别是在不同驱动器中格式化后,位置变化甚至会达到十几、二十几个字节。

我们将接缝附近的数据显示出来,内容如下:

```
4E 4E 4E 4E 4E 4E 13 93 93 93 93 93
```

前面的若干 4E 显然是磁道尾部的内容,突然出现的 13 表示了接缝的具体位置,下面的若干个 93 是 4E 的变形,反映了接缝的状况。值得注意的是,当我们将盘放在另一驱动器中读取扇区内容并将接缝附近的数据显示出来时,可能会变成如下内容:

```
4E 4E 4E 4E 4E 4E 4F 93 93 93 93 93
```

就是说最后一个 4E 变成了 4F,而接缝处的 13 变成了 93。当我们格式化十几条磁道后,再在别的驱

动器中读取,发现经常有近一半的磁道发生了这种类似的变异现象。仔细研究不难发现,4E 变成 4F 完全是由于最后一位由 0 变为 1 所致,而接缝处的 13 变成 93 是高三位发生了变化。比较不同磁道的接缝处的数据可以看出:接缝处的字节的低 5 位总是与下面的几个字节的低 5 位一致。可以得出如下结论:接缝处有 4 位是不稳定的,下面几个字节在不同驱动器中读出时内容是稳定不变的。这是因为在 MFM 编码方式下,数据识别主要靠时间间隔来控制,存放的时钟位等信号对同步只起辅助作用。因此前面的混乱不会造成后面数据的任何变化。在读取接缝数据后,寻找接缝位置和信息时,只要将最后的 4F 与 4E 等同看待且取稍离接缝几个字节的数据作为接缝信息,就可得到满意结果。

我们知道:90 变形的出现是由于时钟位与数据位发生了颠倒所致,若正好处于临界状态,则有时读出的数据就成了 4E 的变形。实验发现,出现这种不稳定的几率极小,通常可以通过重读加以克服,更好的办法是用多磁道法来解决。把十几个磁道都进行加密,判断时只要有一半以上磁道合法就认为合法。这样即使有部分磁道坏了仍然可以用,对合法用户大大增加了可靠性,对非法用户也大大增加了破密和复制的难度。

三、磁道接缝信息的读取及密码的制作

在正确格式化特殊的磁道后,可用中断 13 的 02 功能读取包含磁道接缝位置 and 信息的这一特殊扇区的数据。中断 13 功能完成后,回送 AL 的一个字节给出了磁盘驱动器的状况。对于实际长度与标识的长度不一致的扇区,AL 第 4 位为 1 即 AL 为 10 表示 CRC 检验错是正常现象。有时也会出现 AL 为 09 的情况。经查阅较详细的资料发现,这是由于试图让 DMA 越过 64K(十进制)边界进行传输。为了克服这个问题,必须在相邻地方同时申请二个空间略多于实际放入数据个数的字符数组。将一个数组的起始地址的段地址变成 x000 的形式,此时的位移量若大于 C000,则传送 4000 字节的数据必然会跨越 64K(十进制)边界而出错,而另一组数组由于与第一个数组相邻而不会跨越。我们只要将 ES:BX 指向不跨越边界的数组即可。也可采用先用第一个数组,若出现 AL 为 09 的错误后再改用第二个数组的办法来解决。

在读取与接缝有关的数据后,找出接缝的位置和信息,经某种加密运算后,混杂在随机数中,存到磁道的某扇区中。判断程序只要读取与接缝有关的

数据及存到某扇区的加密数据,经比较就可判断出合法性。为保证接缝具体位置确实只与转速有关,应判定扇区之间间隙是否与设定的一致,防止解密者通过修改磁盘机工作参数中的第 7 项(规划所需的磁隙长度)来格式化盘,使判断程序误以为接缝位置正确。为保证可靠性,应采用多磁道加密法,加密的数据也应放入多个扇区中。

四、建立有效的反跟踪反破密措施

由于磁道接缝的不可复制性,破密只有通过跟踪窃取密码、修改代码、伪造数据来实现,因此必须建立有效的反跟踪、反破密措施。

我们通常采用编译型高级语言编制程序,因此判断程序可采用高级语言编制,可插入到原高级语言源程序中,经编译连接得到的可执行文件是一个不可分割的整体,不存在任何加密程序和真正程序分离的现象。编制的判断程序应对 A 或 B 驱动器中的 360K 或 1.2M 或 1.44M 盘均能自动适应,以方便使用。

商品化加密软件通常采用在 EXE 文件开头加一段有反跟踪措施的判断程序形成新的 EXE 文件,这与病毒感染 EXE 型文件类似。经精心研究后,就可编制出相应程序,象切除 EXE 型文件感染的病毒一样切除加密判断程序,还原出原 EXE 文件而实现解密。可以说有商品化加密软件就可能有相应的解密软件。商品化加密软件造成解开一个就能解开一类软件,对解密者极具诱惑力,易被高水平解密者选中而破解。对于在源程序中加密的软件,由于各个程序千差万别,解密者只能一个个解,某一个程序被高水平解密者选中而破解的可能性很小。对于能够编制加密软件的程序员来说,制作加密软盘和在源程序中编写带有反跟踪措施的判断程序都是比较简单。因此可靠反跟踪防解密必须是建立在自力更生的基础上,在源程序中加入必要的语句来实现。当然,在源程序加密的基础上,采用一些加密软件,对 EXE 文件再生成一保护壳效果会更好。

常用的反跟踪措施有:破坏单步中断 1、断点中断 3、显示中断 10、键盘输入中断 16、DOS 中断 21(在其暂不用时)等中断;采用时间计数器监视程序运行时间,若程序运行超出规定时间,则程序一定被跟踪,可产生错误数据,将跟踪者引入歧途。

利用系统的中断 40 代替中断 13 读软盘,并确保中断 40 向量指向不可写入的 EPROM,有效防止通过修改中断 13 内容来仿真磁盘接缝数据而破密的目的。通过随机数确定一可读扇区名来读取磁道

接缝数据,使得不可能通过伪造扇区放置伪造磁道接缝数据,因为任一磁道无法同时容纳多个大扇区。

对于放有磁道接缝信息密码的解密运算算法,由显式的常数,改为很多变量参与的较复杂的隐式运算,破密者即使能成功跟踪到此,从汇编代码中极难读懂真正的解密运算算法。

采用多磁道法后,要求判断程序对多道磁道逐一加以判断,记下成功的个数,到所需的个数后即认为合法,程序正常运行,否则返回 DOS。因此程序必定有两个极关键、极脆弱的地方:第一处在判定某一磁道是否成功的条件转移上;第二处在判定整体是否合法、程序是否运行的条件转移上。破密者篡改代码,使程序非法运行,通常从这两处下手。随机给出某一磁道不成功的数据,若返回结果表明该磁道成功,则程序第一处一定被篡改,可由变量记录,在程序运行到适当时候,对运行非法程序者给予惩罚;采用多个不同形式的变量计数,在程序第二处被篡改非法运行时,由于有些变量值不正常,在程序特定处判断后可给出错误结果或给予惩罚,使得被篡改的程序似乎正常运行,但有时却产生错误结果而毫无用处。通过修改代码能使程序运行,对 EXE 文件加密来说是解密的结束,而对在源程序中加密的程序来说,这只是解密的开始。

五、结束语

磁道接缝加密采用磁道接缝这种难以控制、不可复制的量作为加密的基础。对于将密码安装到硬盘上的加密,由于硬盘类型较多,磁盘接缝加密在硬盘上实现较为复杂,可采用如硬盘上存放某一文件的起始簇号这种较难控制的量作为加密的基础。

由于出发点的不同,选定作为加密基础的量可能不同,但是加密基本模式却相同:不可控制的随机量、相应的密码、在源程序中编制具有有效反跟踪措施的判断程序,这就是防拷贝软件的软加密的全部。从纯技术的角度讲,绝对可靠的加密是不存在的,只有技术和法律的结合,才能实现较可靠的加密。

采用磁道接缝加密法或同时采用磁道接缝和激光(掩膜)加密法,并在源程序中加入具有有效反跟踪措施的加密判断程序,放弃采用商品化的加密软件来对 EXE 型文件进行加密这一传统的不可靠方式,这就是笔者对实现软件可靠加密的看法。

```
#define TRCNUM 3
#define TWOTRNCM 6
#define MAXPLAC 0xf0
#define SECTLENT 0x400 /* 128*2^3 */
#include "stdio.h"
```

```

#include "dos.h"
#include "alloc.h"
#include "stdlib.h"
#include "time.h"
unsigned long r0,r2;
unsigned char * ads1e, * s2, * s3;
int high,sectnam,head,track __ add,disk,track,first __ track;
int right __ number,evtrak[TWOTRCNM];
int plac0[TWOTRCNM],plac2[TWOTRCNM],code0[TWOTRCNM],
    code2[TWOTRCNM];
int data[16]={0x27,0x93,0xc9,0xe4,0x72,0x39,0x9c,0x4e,
    0x21,0x42,0x84,0x09,0x12,0x24,0x48,0x90};
int mh2[TRCNUM]={0xa0,0x9f,0x970};
/* 360K disk plac0e */
int mh3[TRCNUM]={0x1a50,0x1a40,0x19d0};
/* 1.2M disk plac0e */
int mh4[TRCNUM]={0x2260,0x2240,0x21d9};
/* 1.44M disk plac0e */
int mh7[TRCNUM]={0x27,0x33,0x59};
int mh8[TRCNUM]={0xe6,0xf5,0xa3};
sect __ name[TRCNUM][5]={
    {0xf5,0x6c,0x38,0xb9,0xba},
    {0x69,0xe8,0x5f,0xb5,0x54},
    {0x32,0xd8,0xe3,0xa5,0x6e}};
union REGS rvs;
struct SREGS srvs;
main(int argc,char * argv[])
{
    unsigned c3,c4,c5,c6,c7,c8;
    if(argc>1&&(argv[1][0]!='B' || argv[1][0]!='b')) disk=1;
    /* disk in dirve B */
    else disk=0; /* format disk in drive A */
    s3=calloc(0x4100,1);
    if(s3==0){printf("no enough memory");return 0;}
    r0=s3;r0&=0xffff;r2=s3;r2=r2>>12;r0=r0+(r2&0xffff);
    if(r0>0xc000)
    {s2=calloc(0x4100,1);
    free(s3);
    if(s2==0){printf("no enough memory");return 0;}
    else s2=s3;
    ads1e=getvect(0x1e);
    printf("This is used to Mark disk. Press any key When Ready:\n");
    getch();
    high=1;first __ track=15;
    c3=* (ads1e+3);c4=* (ads1e+4);c5=* (ads1e+5);
    c6=* (ads1e+6);c7=* (ads1e+7);c8=* (ads1e+8);
    if(form0())
    {if(argc>1&&argv[1][0]<'Z')
        {if(format()) {savcd();chkcd();}
        else chkcd();}
    * (ads1e+3)=c3;* (ads1e+4)=c4;* (ads1e+5)=c5;
    * (ads1e+6)=c6;* (ads1e+7)=c7;* (ads1e+8)=c8;
    free(s2);
    return 0;
    }
    form0() /* check have dos formatted disk in drive ? */
    {
        int i,k;
        track=0;sectnam=1;* (ads1e+4)=9;
        for(k=0;k<10;++k)
        {rvs.x.ax=0;

```

```

            int86x(0x40,&rvs,&rvs,&srvs);
            r0=s2;
            srvs.es=r0>>16;rvs.x.bx=r0;
            rvs.x.dx=disk;
            rvs.x.cx=track * 256+sectnam;
            rvs.x.ax=0x0201;
            int86x(0x40,&rvs,&rvs,&srvs);
            if(rvs.x.cflag==0) return 1;}
            printf("Read trace 0 failure %x\n",rvs.x.ax>>8);
            return 0;
        }
        format()
        {
            int i;
            for(i=0;i<TRCNUM;++i)
            {if(formx(5)) return 0;}
            return 1;
        }
        formx(int sectnum)
        {
            unsigned char s[16][4];
            int i,j,k,i0;
            track=first __ track+track __ add;
            printf("format track: %d\n",track);
            * (ads1e+7)=mh7[track __ add];
            * (ads1e+8)=mh8[track __ add];
            for(i0=0,head=0;head<2;)
            {if(i0>10){printf("format failue\n");return 10;}
            for(i=0;i<sectnum;++i)
            {s[i][0]=track;s[i][1]=head;s[i][2]=sect __ name[track __
                add][i];
            if(high==2) s[i][3]=6;else s[i][3]=7;}
            s[0][3]=3;sectnam=sect __ name[track __ add][sectnum-2];
            * (ads1e+3)=3;* (ads1e+4)=sectnum;
            for(k=0;k<5;++k)
            {r0=s;srvs.es=r0>>16;rvs.x.bx=r0;
            rvs.x.dx=head * 256+disk;rvs.x.cx=track * 256+sectnam;
            rvs.x.ax=0x0501;int86x(0x40,&rvs,&rvs,&srvs);
            if(rvs.x.cflag==0) break;}
            if(k==5){printf("Format failure %x\n",rvs.x.ax>>8);return
                5;}
            k=read __ sect(0);
            if(k>0) return k;
            if(k<0){++i0;continue;}
            ++head;}
            ++track __ add;
            return 0;
        }
        read __ sect(int disp)
        {
            int i,j,k,plc,place;
            if(high==2){* (ads1e+3)=6;place=mh2[track __ add]+disp;}
            else
            {* (ads1e+3)=7;
            if(high==1) place=mh2[track __ add];
            else if(high==3) place=mh3[track __ add]+disp;
            else if(high==4) place=mh4[track __ add]+disp;}
            for(k=0;k<5;++k)
            {r0=s2;srvs.es=r0>>16;rvs.x.bx=r0;
            rvs.x.dx=head * 256+disk;rvs.x.cx=track * 256+sectnam;
            rvs.x.ax=0x0201;int86x(0x40,&rvs,&rvs,&srvs);

```

```

if(rvs.x.cflag&&(rvs.x.ax>>8)==0x10) break;}
if(k==5) {printf("Read failure %x\n",rvs.x.ax>>8);return 6;}
plc=SECTLENT+2;
if(s2[plc]! = 0x4e||s2[plc-1]==0x4e) return -9;
for(i=0; ++i) {if(s2[plc+i]! = 0x4e) break;}
if(i! = mh7[track __ add]) return -9;
if(high==1)
{for(plc=place;plc<place+0x2000; ++plc)
{if((s2[plc]&255)! = 0x4e) break;}}
if(plc<place+MAXPLAC) {high=2;return -2;}
if(plc>mh3[track __ add]&&plc<mh3[track __ add]+MAXPLAC)
{high=3;place=mh3[track __ add];}
else if (plc > mh4 [track __ add] &&plc < mh4 [track __ add] +
MAXPLAC)
{high=4;place=mh4[track __ add];}
else return -9;}
else
{for(plc=place;plc<place+MAXPLAC; ++plc)
{if((s2[plc]&255)! = 0x4e) break;}}
if(plc==place+MAXPLAC) return -9;}
if(s2[plc]&255==0x4f) ++plc;plc+=2;
for(i=1;i<10; ++i) {if(s2[plc+i]! = s2[plc]) break;}
if(i<10) return -9;
k=s2[plc]&255;
for(i=0;i<16; ++i) {if(k==data[i]) break;}
if(i==16) return -9;
plac0[track __ add * 2+head]=plc-place;
code0[track __ add * 2+head]=s2[plc];
return 0;
}
savcd()
{
int i,j,k,xor;
randomize();
while(1)
{for(i=0;i<1024; ++i) s2[i]=rand();
j=9+s2[5]%6;
s2[j]=high;
for(i=0;i<TWOTRCNM; ++i)
{k=j+8+s2[j+7]%11;
j=k+11+s2[k+6]%13;
xor=s2[j];
s2[k]=plac0[i]^xor;
k=j+5+s2[j+3]%19;
j=k+7+s2[k+4]%17;
xor=s2[j];
s2[k]=code0[i]^xor;
if(j>1020) break;}
if(i==TWOTRCNM) break;}
printf("save data in sect\n");
track=first __ track; sectnam=sect __ name[0][0];
* (ads1e+3)=3;savsec();
track=first __ track+1;sectnam=sect __ name[1][0];
* (ads1e+3)=3;savsec();
track=first __ track+2;sectnam=sect __ name[2][0];
* (ads1e+3)=3;savsec();
return 0;
}
savsec()
{
int k;
for(head=0;head<2; ++head)
{for(k=0;k<5; ++k)
{r0=s2;srvs.es=r0>>16;rvs.x.bx=r0;
rvs.x.dx=head * 256+disk;rvs.x.cx=track * 256+sectnam;
rvs.x.ax=0x301;int86x(0x40,&rvs,&rvs,&srvs);
if(rvs.x.cflag==0) break;}
if(k==5) {printf("Write failure %x\n",rvs.x.ax>>8);return 6;}
}
return 0;
}
chkcd()
{
int i,j,k,xor;
printf("Check formatted track\n");
track=first __ track; sectnam=sect __ name[0][0]; head=0; *
(ads1e+3)=3;
for(k=0;k<5; ++k)
{r0=s2;srvs.es=r0>>16;rvs.x.bx=r0;
rvs.x.dx=head * 256+disk;rvs.x.cx=track * 256+sectnam;
rvs.x.ax=0x201;int86x(0x40,&rvs,&rvs,&srvs);
if(rvs.x.cflag==0) break;}
if(k==5) {printf("Read failure %x\n",rvs.x.ax>>8);return 6;}
j=9+s2[5]%6;
high=s2[j];
for(i=0;i<TWOTRCNM; ++i)
{k=j+8+s2[j+7]%11;
j=k+11+s2[k+6]%13;
xor=s2[j];
plac2[i]=s2[k]^xor;
k=j+5+s2[j+3]%19;
j=k+7+s2[k+4]%17;
xor=s2[j];
code2[i]=s2[k]^xor;}
right __ number=0;for(i=0;i<TWOTRCNM; ++i) evtrak[i]=0;
track __ add=0;testx();
track __ add=1;testx();
track __ add=2;testx();
if(right __ number==TWOTRCNM) printf("OK\n");
else
{for(i=0;i<TRCNUM; ++i)
{for(j=0;j<2; ++j)
{if(evtrak[i * 2+j]==0)
printf("Track: %02d Head: %02d error\n",first __ track+i,j);}}
return 0;
}
testx()
{
int i,j,k;
track=first __ track+track __ add;
for(head=0;head<2; ++head)
{randomize();k=rand()%3+1;
sectnam=sect __ name[track __ add][k];
j=(3-k) * (SECTLENT+0x3e+mh7[track __ add]);
if(read __ sect(j)! = 0) continue;
i=track __ add * 2+head;
if(code2[i]==code0[i]&&plac2[i]==plac0[i])
{++right __ number;evtrak[i]=1;}}
return 0;
}
}

```

磁盘文件的彻底删除

□ 费洪晓

在

研究计算机系统的安全性时,为了防止泄密,用户必须删除明码文件(虽然密码学强调将密码文件重写在明码文件原位置上,但实现时往往有些繁杂)。通常的删除方法(如 DEL 命令、PCTOOLS)还不够保险,因为这些方法只是让用户无法通过文件控制表(FDT 和 FAT)访问文件的内容,而并没有真正抹去这些内容,甚至文件控制表的入口内容,除第一个字符外的文件名、文件大小、文件起始簇号、簇号链等等,都保持不变直到它被其他数据覆盖。

对于具有一定的微机知识和有关软件的窃密者来说,恢复用这些方法删除的文件是轻而易举的,如利用 PCTOOLS、DEBUG、CHKDSK 等软件。这无异于为他们打开了一个窃密的方便之门。

如果用户对文件具有保密要求,那么当他不再需要这些文件时,用普通的文件删除方法并不能彻底删除这些文件,为了防止这类泄密,笔者用汇编语言编制了一个实用程序 XDEL. ASM,它将用户不希望保存在磁盘上的文件用空白字符(ASCII 码值为 0)从头到尾重写一遍,再删除,从而彻底地删除了文件,安全可靠。

源程序清单如下:

```
code      segment para public 'code'
          org 100h
start     proc far
          assume cs:code,ds:code
          push ds
          xor ax,ax
          push ax      ;标准程序首部
          mov si,81h
          lea di,filename
```

```
mov cl,[si-1]
mov ch,0      ;命令行参数长度
mov al,[si]
cmp al,' '
jz s1
cmp al,9
jnz s2
inc si
dec cl
jmp s0      ;舍去命令名与参数间的分隔符
cid
rep movsb
;从 PSP 读文件名参数至缓冲区
lea dx,filename
mov ax,3d02h
int 21h      ;打开文件
jnb ok1
lea dx,errmsg      ;若打开不成功,出错处理
mov ah,9
int 21h
ret      ;主程序结束
ok1:      mov bx,ax      ;保存文件句柄
          lea dx,delhlp
          mov ah,9
          int 21h
          mov ax,0c07h
          int 21h      ;要求输入确认信息
          or al,20h
          cmp al,'n'
          jnz ok2
          mov ah,3eh
          int 21h
          ret      ;主程序结束
ok2:      mov cx,0
          mov dx,0
          mov ax,4202h
          int 21h      ;获得文件字节数
          mov cx,10
shift:    shr dx,1
          rcr ax,1
          loop shift
          inc ax      ;换算成文件所占簇数
```

(下转第 53 页)

子目录与文件的安全浅谈

□杨志翔

磁

盘经 DOS 格式化后分为四大区域:引导区,也称 BOOT 区;文件分配表区,也称 FAT 区;根目录区,也称 FDT 或 ROOT 区;数据区,也称 DATA 区。以上四区及其中的有关数据结构,形成了一定的磁盘及文件的管理格局。通过文件目录项及文件分配表实现了文件内容的定位。

1. 磁盘文件信息的定位

一般而言,用户对磁盘文件的操作通常由文件名来进行指定。DOS 则根据所给文件名对目录区进行扫描,查找相应的目录项,由相应目录项中的起始簇号定位文件内容的起始位置,再由文件分配表定位后读内容。因此对一个磁盘而言,其目录区是一个文件操作的关键部分,若该区内容被破坏,则将意味着磁盘文件的“丢失”。

2. 根目录和子目录

对一个特定磁盘而言,经 DOS 格式化后,其根目录区的位置和大小也就确定了。而对于子目录,DOS 将它作为一种“特殊文件”来对待。当在根目录下建立一个子目录时,DOS 在根目录区中同样为它建一个目录项,同时在数据区中为它分配一个簇的空间,作为该子目录区,并把该簇位置填入目录项中的起始簇号里。所以子目录的目录区是在建子目录时,由 DOS 在数据区中进行分配。在使用过程中,若子目录区用完后,DOS 再分配另外一个簇给该子目录。一般,子目录下的文件数目仅受盘空间的限制。

3. 恢复的基本思想

利用子目录区中的特征标志查找该区域,定位后记下该簇号。再建一子目录,将所

定位簇号填入新建子目录的起始簇号位置即完成了恢复工作。

利用工具软件 PCTOOLS 来完成的具体方法如下:

(1)特征标志的确定:若记得子目录下的一个文件名,则可用其作为特征标志。否则可用子目录下系统创建的两个特殊目录.和..作为特征标志。

(2)利用 PCTOOLS 的磁盘查找功能,扫描特征标志。注意:在输入特征串时为提高速度和准确度,可在.后加 7 个空格或在..后加 6 个空格键。找到后利用 E 键进入编辑/浏览状态,确定有关信息。

(3)目录区位置的确定:由特殊目录.的目录项中起始簇号位置中的值,即可知该目录区所在簇号。由特殊目录..的目录项中起始簇号位置中的值,可知上一级目录区所在的簇号,若该值为 0,则表示上一级为根目录。记录下.目录项的起始簇号值后,退出磁盘查找功能。

(4)利用 PCTOOLS 的目录管理功能,在根目录下建立一任意子目录。也可在 DOS 提示符下用 MD 命令来建立。

(5)恢复:利用 PCTOOLS 的磁盘浏览/编辑功能,读出盘根目录区,将所记录下的子目录区的起始簇号,填入新建子目录目录项中的起始簇号处即可。

由上所述可见,子目录的使用不仅仅是为了便于磁盘文件的组织和管理,在当前计算机病毒猖獗的情况下,从某种角度来看,子目录下的文件较根目录中更为安全。因此,即使在软盘上,也推荐大家使用子目录,以便在遇到类似问题时,用上方法进行恢复。

SKILL

FoxPro 应用系统中的错误捕获和处理

□ 罗 辉

任

任何一个系统,在设计和运行过程中总不可避免会隐含各种不可预知的错误,尤其是在软件设计日趋庞大的今天,隐含错误的几率越来越大。但只要我们能及时捕获到错误以便纠正,或当错误出现后能妥善处理,那么出现错误并不可怕。对错误的捕获和处理是应用系统设计中很重要的一部分。但是,君不见许许多多的应用软件中,错误的捕获和处理并不灵活,能提供的错误信息亦不完善,维护较困难。往往还出现这样的情况,当操作员发现一个错误,维护人员赶到现场时,屏幕上的错误信息却早已消失不见而无法维护!

如果你是一个 Xenix 系统管理员,你将总可以在 Xenix 中发现一些 core 和 errlog 之类的文件,Xenix 系统利用这些文件容纳系统的错误信息。从这些文件中常常可获得一些重要的维护信息。如果我们设计的软件也能提供这样一种错误信息的处理和保存能力:它能反映错误发生的详细情况,并能及时将之保存到一个错误信息文件中,将会给软件调试和维护提供很大方便。

本文设计了一个通用错误信息捕获和处理过程 errorproc,如果你是在 FoxPro 平台上开发软件,那么将之挂靠到你的应用系统中去将是一个很好的方法。当应用系统出现错误时,errorproc 过程能自动捕获引起错误的丰富详尽的环境信息,包括出错的文件名、程序行、错误类型、错误提示信息、当前数据库及编辑字段域,以及当前系统状态、内存变量等的取值情况,并保留在一个文件 ERROR.TXT 中。出错时,你可以现场查阅或打印当前的错误信息,根据出错现场状态即

可判断错误的出处和原因。或者,如果出错现场已被破坏,维护人员也可翻阅 ERROR.TXT 文件了解当时的出错情况。有了这样一个错误捕获和处理的自动过程,即可防止操作员没有详细记载现场而难以维护的不足。尤其在多用户环境下其优越性更加突出。本文所附程序 EXAMPLE.PRG 用以演示调用错误捕获和处理过程,其在 FoxPro 2.5 和华达汉字系统 HDDOS1.0 的运行环境下通过。

源程序清单如下:

```
set talk off
clear
on error do errorproc with LINENO(1),program()
select 1
use install
Other __ Process
&& 非法命令,由它激活错误捕获和处理过程
on error
return
procedure errorproc && 错误捕获和处理过程
parameter lineno,progfile
private errorno
on error && 暂时关闭错误陷阱
errorno=error()
do createrrfile && 形成错误信息文件
DEFINE WINDOW error ;
FROM INT((SROW()-15)/2),INT((SCOL()-50)/2);
TO INT((SROW()-15)/2)+16,INT((SCOL()-50)/2)+54 ;
TITLE "错误信息窗口" SHADOW DOUBLE ;
COLOR SCHEME 10
ACTIVATE WINDOW error
DO WHILE .T.
@ 3,2 GET choice ;
PICTURE "@ * H 重试;退出;挂起;忽略;显示;打印;清除";
SIZE 1,6,1 DEFAULT 1
@ 7,2 SAY "出错号:"+ltrim(str(errorno))
@ 7,20 SAY "出错行号:"+ltrim(str(lineno))
@ 8,2 SAY "出错程序号:"+progfile
@ 9,2 SAY "出错信息:"
```

```
@ 10.2 SAY message()
READ CYCLE
on error do errorproc with LINENO(1),PROGRAM()
&& 恢复错误陷阱
DO CASE
CASE CHOICE=1
    RELEASE WINDOW error
    RETRY
CASE CHOICE=2
    RELEASE WINDOW error
    QUIT
CASE CHOICE=3
    DEACTIVATE WINDOW error
    SUSPEND
    ACTIVATE WINDOW error
CASE CHOICE=4
    RELEASE WINDOW error
    EXIT
CASE CHOICE=5
    modi comm tmp. txt noedit
CASE CHOICE=6
    @12,2 GET dychoice pict "@ * H 打印当前信息;历史信息";
    size 1,16,1 default 1
    read
    dyfile=iif(dychoice#2,'tmp. txt','error. txt')
    if FILE((DYFILE))
        ! type &dyfile. >PRN
&&DOS 外部命令不支持 dyfile 等名字表达式
    ENDIF
    @12,2 clear to 12, 30
CASE CHOICE=7
    IF FILE('ERROR. TXT')
        ! DEL ERROR. TXT >NUL
    ENDIF
ENDCASE
ENDDO
RETURN
```

```
procedure createrrfile && 错误信息文件形成过程
set console off
set printer on
set printer to tmp. txt
? padc("错误汇报",78,'*')
? 'Date:' + dtoc(date()) + space(5) + 'Time:' + time()
?? space(5) + 'FoxPro 版本号:'
```

(上接第 50 页)

```
push ax
mov cx,0
mov dx,0
mov ax,4200h
int 21h ;文件指针指向文件头
pop cx
write: push cx
    lea dx,buf
    mov cx,1024
    mov ah,40h
    int 21h
    pop cx
    loop write ;重写整个文件(每次一簇)
```

```
?? version()
?
proccount=1
do while len(sys(16,proccount))<>0 && 嵌套调用的程序名
? iif(proccount=1,'主程序:',space(proccount-1)+'调用:')
?? sys(16,proccount)
proccount=proccount+1
enddo
?
? '错误号:' + ltrim(str(errorno))
? '错误信息:' + message()
? '引起错误的程序行号:' + str(lineno)
? '出错行语句:'
? message(1)
?
IF len(alltrim(dbf()))#0
    ? '当前工作区:'
    ?? select()
    ? '打开的数据库:' + DBF()
    ndxnum=1
    SCAN WHILE LEN(NDX(NDXNUM))#0
        ? NDXNUM
        ?? '索引文件名:' + NDX(NDXNUM)
        NDXNUM=NDXNUM+1
    ENDSCAN
ENDIF
tmp=ALLTRIM(VARREAD())
if LEN( tmp )<>0
    ? '当前的编辑域:' + tmp
    ? '其取值为:'
    ?? (tmp)
ENDIF
?
? '出错时的状态:'
list stat
?
? '出错时的内存变量或数组取值:'
list memo
? repl(' ',80)
! copy error. txt + tmp. txt error. txt >NUL
set printer to
set printer off
set console on
return
```

SKILL

```
mov ah,3eh
int 21h ;关闭文件
lea dx,fname
mov ah,41h
int 21h ;删除文件
ret ;主程序结束
fname db 32 dup(0) ;文件名缓冲区
buf db 1024 dup(0) ;文件重写内容缓冲区
delhlp db 0dh,0ah,' Do you really want to
delete? Type y or n$'
errmsg db 'File access error!!! $'
endp
code ends
end start
```

SKILL

FoxBASE 数据库系统中一些不可忽略的功能和问题

□朱 原

笔

者近些年来接触了一些 FoxBASE 数据管理信息系统软件,并且自己研制、开发了一些软件后发觉,有许多用户虽然运用了很长时间的 dBASE、FoxBASE 系统,但对其中许多命令、函数却仍然缺乏一定的了解,放弃了不少很巧妙的命令和函数功能,还忽略了一些容易出错的问题,造成软件执行时用户感到使用不够方便或者产生一些莫名其妙的错误。本文就此问题分述如下:

1. SELECT 0 命令

SELECT 命令在运用关系数据库系统时是用来选择当前工作区的,它的自变量(其范围为 1—10, FoxPRO 可以达到 25 个之多)就是系统所要指定的工作区。当其自变量为 0 时,则表示系统将选定最小号数的尚未使用的工作区作为当前工作区。例如:用户已在工作区 1—3, 以及工作区 3 上各自打开了一个文件,则此时无论系统处在哪一个工作区,只要执行命令 SELECT 0, 系统都会自动将工作区指定到 4 上。这对于多人一起编写一个系统,将要各自打开相应的数据库操作,且不影响其他数据库操作的程序时,是很有用的一条命令。有时,编程者要临时打开一个数据库进行操作,马上又要关闭,且与其他的工作区并不相关时,可以使用如下语句来执行命令:

```
LSELE = IIF (SELE() < 10, STR (SELE) (), 1), STR
(SELE(), 2)) &&. 检测当前工作区号
SELECT 0 &&. 选择尚未使用的最小号数的工作区
USE (文件名)
..... &&. 用户此时可进行各种必要的操作
USE
SELECT &LSELE
&&. 转回前面工作状态的工作区上
```

2. READKEY() 函数

在用 FoxBASE 编制系统时,用 @... PROMPT 和 MENU TO 命令组合来选择菜单界面是极为常见的一种方法。但这条命令要求用户必须通过选择菜单选择项中的退出项,才可以执行退出命令,与许多软件常用 Esc 键退出系统的方法不太一致,以至用户感觉不很方便。而采用读取用户退出全屏幕编辑命令时的按键值 READKEY() 函数则能通过 Esc 键来退出菜单:

```
@1.0 PROMPT '1, 选择一'
@2.0 PROMPT '2, 选择二'
@3.0 PROMPT '0, 退出'
MENU TO KEY
DO CASE
CASE KEY=3. OR. READKEY()=12
RETURN
CASE KEY=1
..... &&. 执行选择一
CASE KEY=2
..... &&. 执行选择二
ENDCASE
```

这样,用户无论是选择了功能 0, 或是按 Esc 键都可以执行退出系统的功能。

3. 函数 BOF()、EOF()、RECNO() 和命令 SKIP

当数据库指针在文件首部或是在尾部时,执行命令 SKIP 1/-1, 其所返回的记录号值与人们的一般想象不一样。假定数据库不为空,且数据库指针在文件首(BOF() 函数的返回值为 .T.) 时,用 RECNO() 函数来检测,其返回值为第一条记录的记录号,执行了 SKIP 命令后,RECNO() 函数值仍为该条记录的记录号;而数据库的指针指在文件尾部(Eof() 函数为 .T.) 时,函数 RECNO() 的返回值将

为数据库记录的总数加1,执行了命令 SKIP-1后,函数 RECNO()的返回值则变为最后一条记录的记录号了。因此,当用户使用函数 RECNO()测到的记录号与第一条记录的记录号相同时,并不能够确定指针是否已指在了该条记录上,而应当再用 BOF()函数来检测一下。

4. WAIT 命令

WAIT 命令在程序的编辑过程中广为用户所运用,但是,由于该命令要接受一个字符的输入,而且在屏幕上会显示出来(除了回车键),因此,该命令很容易破坏程序的屏幕定义格式,又因为其只有一个字符而常被编程者所忽略。解决的办法有二:一个是用 SET CONSOLE OFF 命令将输出到屏幕的信息全部屏蔽起来,WAIT 命令执行完毕后,再用 SET CONSOLE ON 命令恢复;二是改用 INKEY()函数,当该函数的自变量为0时,则其可以达到 WAIT 命令的同样功效,但其所接受的键盘输入字符却不会显示出来。

5. 运用代码录入数据时的一个编程技巧

运用代码来录入信息是简化大量文字性材料录入的一种简便方法,许多编程者对代码本身的处理,往往是另外开辟一个模块来对其进行追加、修改和删除等操作,而在靠代码进行主数据库信息录入时则无法对代码进行追加(代码的修改与删除在此不是必需的),然而,这却是极其需要的功能。因为,只有在主数据录入的过程中才能发觉未归入代码类的信息,在此时,编程者的处理往往是要求用户先退出主数据库的录入模块,改为进入代码数据的维护功能中去进行操作,修改完毕后,再返回主数据库的录入模块中继续该数据的录入。例如:某人事信息主数据库中,工作岗位是靠代码录入的,在代码库中尚未含有“打字员”的工作岗位代码时,在主数据库中录入某人工作岗位为“打字员”,则代码将无法录入,用户只好退出本主数据库的录入模块,进入到代码维护中去补充该代码,然后,重新再进入人事主数据库中去录入或修改该人信息。这给用户的录入工作造成麻烦。也有编程者要求用户将所有的未列入代码的信息均作为“其他类”,这显然不是一个好的解决办法。

实际上用户只要将 @...GET、READ 命令与 ACCEPT 命令结合起来就可以很好地解决这个问题,使用户在录入过程中,一旦发现有未曾编排过的代码,不必退出此处的录入系统,靠自定义函数 TEST()在本录入模块中直接就可以追加新的代码,

新的代码追加加入代码库后,用户就能录入该代码了,主数据库中的其他代码数据均可用同样方法完成。

例如,有一人事主数据库 BSK.DBF 需要录入一个长度为2的字符型字段——“工作岗位”(代码),工作岗位代码库为一已按代码字段 DM 建立索引的数据 DM.DBF 和 DM.IDX。DM.DBF 代码库中所包括的两个字段为:长度为2的字符型字段 DM(代码);长度为10的字符型字段 ZD(代码对照的意义)。

首先在录入主数据库数据的程序中必须有下列语句用来录入工作岗位的代码。

源程序清单如下:

```
set talk off
set stat off
clea all
set procedure to p2
use rsk
gzgw=space(2)
@ 2,10 say "工作岗位(代码):" get gzgw valid test(gzgw)
read
if gzgw # space(2)
    append blank
    repl 工作岗位 with gzgw
endif
use
set proc to
retu

procedure tese
parameters cdm
priv rowf.colf.key,lsele,bl1,bl2
rowf=row()
colf=col()
if len(trim(cdm))=0
    retu .f.
endif
lsele=iif(sele()<10,str(sele(),1),str(sele(),2))
sele 0
use dm inde dm
find &cdm
if eof()
    @ 22,0 say " "
    acce"请输入代码:"to bl1
    acce"字典:"to bl2
    if len(trim(bl1))>0 .or. len(trim(bl2))>0
        appe blank
        repl dm with bl1,zd with bl2
    endif
else
    @ rowf.colf say zd
endif
use
key=.t.
sele &lsele
retu key
```

SKILL

汉字 FoxBASE + 最小化的 DOSSHELL

□曹国钧

大

多数中西文软件,如 Turbo C、WPS、Windows 等都实现了 DOSSHELL 功能,但汉字 FoxBASE + (FoxPro)不具有 DOSSHELL 功能,这给用 FoxBASE + 编制应用程序的程序员带来诸多不便。若用 RUN/! 来直接实现 FoxBASE + 的 DOSSHELL 功能,则在该 DOSSHELL 状态下可用空间已很小,无实际利用的意义。事实上,通过 FoxPro (FOXGRAPH)软件包中所提供的精巧的 Fox 内存管理器 FOXSWAP.COM 与 RUN/! 命令配合使用,就能方便地实现 FoxBASE + (FoxPro)的最小化的 DOSSHELL 功能,而且,该 DOSSHELL 状态下可用内存大,故在 DOSSHELL 状态下仍能执行大型应用程序。下面就是在 MS-DOS 6.0 系统优化配置后 RUN/! 和 RUN/! FOXSWAP 的所占内存比较情况。

由该表可看出,用 RUN/! 命令实现的 DOSSHELL 下可用内存仅为 109KB,而用 RUN/! FOXSWAP 实现的 DOSSHELL 的可用空间为 483KB,在此环境下还可加载象 WPS 等大型软件。

在 FoxBASE + (FoxPro)内部执行下面的命令:

```
RUN/!FOXSWAP /NK COMMAND.
COM 或 COMM. BAT
```

就能实现汉字 FoxBASE + (FoxPro)的最小化的 DOS SHELL 功能。其中:

(1) COMM. BAT 为包含 COMMAND. COM 命令的批处理文件,其内容如下:

```
COMMAND
PROMPT (FOX SHELL) $P $G
```

RUN/! 与 RUN/! FOXSWAP 的
DOSSHELL 状态所占用的内存比较表

总空间735KB(1M 以下)	RUN/!	RUN/! FOXSWAP
MS - DOS 占用 (包括 COMMAND.COM)	16KB	16KB
设备驱动程序,如 ANSI. SYS, HIMEM. SYS, EMM386. EXE, DBLSACE. BIN, SETVER. EXE 等	51KB	51KB
SPDOS 汉字系统	132KB	132KB
DOS 实用程序,如 APPEND. EXE, SMRTDRV. EXE, DOSKEY. COM, MOUSE. COM 等	46KB	46KB
FOXPLUS 2.10 占用	378KB	4KB
FOXSWAP.COM 占用	00KB	16KB
DOSSHELL 应占用 COMMAND. COM	3KB	3KB
DOSSHELL 状态下可用内存	109KB	483KB

(2) 命令中参数 /N 或 /NK, 可以为 FoxBASE + 或 FoxPro 运行 COMMAND. COM 时指定所用的系统内存空间的大小, N 是为 RUN FOXSWAP 命令提供的 K 字节为单位的内存空间的字节数。

① 当 N = 0 时, FOXSWAP 使 COMMAND. COM 可用的内存达到最大, 即 FOXSWAP 为 RUN 命令提供了系统所能提供的最大内存空间。实际上, 该命令的默认值为 0。

② 当 N > 0 时, FoxBASE + 或 FoxPro 将完成: 将 FoxBASE + 系统的所有缓冲区的内容快速地写回磁盘, 并在磁盘上将形成类 __T

(下转第61页)

信息输入方式自动转换的实现

□张华建

在

FoxBASE 数据库编程中,经常要求使用者做各类信息的输入工作,系统使用者也随各种输入信息不同进行输入方式的转换,通常这些操作都要在用户的人工干预下完成。那么能否在应用系统编写中根据录入信息的需要来完成录入方式的自动转换呢?本文针对这个问题,利用 FoxBASE 与汇编语言的接口技术,通过汇编语言编程来模拟键盘按键操作顺利地实现了这种设计思想。

在中文 DOS 下,所有通过键盘的输入的信息利用中断写入 DOS 在内存的键盘缓冲区中,再利用中断 16H 从缓冲区取出按键信息,并根据不同的信息进行相应处理。该缓冲区 DOS 定义在内存 40:1EH~40:3DH 中,40:1AH 与 40:1CH 分别存放该键盘缓冲区首尾指针。本方法就是通过向缓冲区中写 Alt-F1~F9 按键的扫描码与 ASCII 码并改变缓冲区首尾指针来完成人工按键模拟的。与此同时,由于考虑到在各种不同输入方法下 Capslock 键的状态也十分重要,故在编写 Change. bin 的同时另外附加一个 Capslock. bin,可根据不同输入法对 Capslock 键的要求进行“开”、“关”处理。在完成以上程序的编辑之后,经汇编、连接,然后用 EXE2BIN 把它们转换成. bin 文件即可使用。

调用方法:先用 LOAD 命令把 Change. bin 与 Capslock. bin 分别装入内存,然后用 call...with(参数)进行调用。对于 change,其参数必须为“1”~“9”(分别对应于按 Alt-F1~F9)字符之一,若为一变量参数,则须为一字符型变量;Capslock 调用方法同 Change,其参数“1”与“0”分别执行 Capslock 键的

“开”与“关”操作。例:

```
•A="6"
•LOAD change
•LOAD Capslock
•CALL change with A(或"6") ;拟按 Alt-F6键
•CALL Capslock with "1" ;Capslock 键使能
•RELEASE MODULE CHANGE
•RELEASE MODULE CAPSLOCK
```

该方法笔者在联想 CCDOS、FoxBASE 2.00 下使用,效果甚佳。

源程序清单如下:

```
;capslock. asm
capslock segment
assume cs: capslock
org 0
begin: push cx ;保护环境
        push bx
        push dx
        push ds
        push es
        push di
        push si
        cmp bx,0
        jz return
        mov cx,bx
looping: mov al,[bx]
        cmp al,'1'
        jz loop1
        cmp al,'0'
        jz loop1
        inc bx
        loop looping
        jmp return
loop1:  mov bx,40h
        mov ds,bx
        mov si,17h
        mov ah,[si]
        cmp al,30h
        jz loop2
        or ah,40h
        jmp loop3
loop2:  and ah,0bfh
```

```

loop3:      mov [si],ah
return:     pop si      ;恢复环境
            pop di
            pop es
            pop ds
            pop dx
            pop bx
            pop cx
            retf
capslock    ends
            end begin

;Change.asm
Change      segment
            assume cs:change,es:change
            org 0h
start:      push cx
            push bx
            push dx
            push ds
            push es
            push di
            push si
            cmp bx,0    ;是否带参数
            jz exit    ;不带则结束
            mov cx,bx   ;参数个数送 cx
            mov ah,0
sta__looping: mov al,[bx] ;取一 ASCII 码参数
            cmp al,'1'
            jb con__looping
            cmp al,'9'
            jbe out__looping
con__looping: inc bx
            loop sta__looping
            jmp exit

out__looping: mov cl,8
            sub al,30h
            shl ax,cl
            add ax,6700h ;转换参数
            cli
            push di
            mov bx,40h ;模拟键盘按键
            mov ds,bx
            mov di,1ah
            mov word ptr[di],1eh
            inc di
            inc di
            mov word ptr[di],22h
            inc di
            inc di
            mov [di],ax
            inc di
            inc di
            mov word ptr[di],3920h
            pop di
            mov ah,0
            int 16h
            sti
exit:        pop si
            pop di
            pop es
            pop ds
            pop dx
            pop bx
            pop cx
            retf ;适用于 mase 5.0 以上
change      ends
            end start

```

SKILL

.....

(上接第73页)

电源没有打开或者没有装纸,程序会出错中断,或者认为已送出打印而修改数据,但实际却没打印成功。

这种情况的处理比较简单,只要采用打印机出错子程序进行处理。程序在要求打印之前,都设置 ON ERROR 语句,检测打印机是否正常,若不正常则提示信息,跳过打印程序段,对数据库不作处理。

2. 工作站一使用共享打印机打印汉字就死机。

我们知道,网络工作站打印与单机打印环境有所不同,大部分单用户汉字系统及很多网络汉字系统的打印机驱动程序是直接调用中断或写入打印端口模式,这样 NetWare 的 CAPTURE 就截获不到本地打印端口信息,并发生内部冲突死机。解决方法只能是修改打印驱动程序或使用支持网络共享打印的汉字操作系统。

3. 网络打印环境设计混乱。

共享打印机是共享资源最为重要的内容之一,NetWare 支持用户在网络共享打印机上打印文本,

并提供了为管理打印和打印作业所设计的丰富的菜单实用程序和运行命令。Novell 网允许一台共享打印机为一个或多个队列服务,一个对列也可以为两个以上的共享打印机服务。对一般的管理应用系统来说,只需一到二个打印队列即可。为了现场便于使用和维护,建议将打印环境设计为一个打印机只服务于一个打印队列,然后为每一个队列指定其用户以及用户们的打印优先权,定义打印格式。

五、结束语

计算机网络应用系统完全不同于单机、单用户的应用系统,对网络维护及其故障排除提出了更高的要求。只有重视网络的维护,才能确保网络的正常运行。同时还应建立一整套严格的管理措施,一个网络系统,机器分布广、操作人员多,必须有健全的规章制度,才能有效地防止计算机病毒的侵入以及减少各种网络故障的发生,最大限度地发挥网络系统的优势。

SKILL

FoxBASE 数据库的同名字段加密法

□黄焕如

众

所周知,建立 FoxBASE 数据库时不能含有相同的字段名,但是在某些特殊的应用中,需要在同一数据库中设置相同名字的字段。例如建立一简单数据库 LS.DBF,并输入几条记录:

•USE LS

•display structure

数据库结构: E:\JT\LS.DBF

数据记录数: 3

最新更改日期 :02/19/93

字段	字段名	类型	宽度	小数
1	ABC1	数值	10	2
2	ABC2	数值	10	2
3	ABC3	数值	10	2
4	ABC4	数值	10	2
5	ABC5	数值	10	2
6	ABC6	数值	10	2

** 总和 **

61

三个记录中各字段的值分别如下:

ABC1	ABC2	ABC3
1254.20	1367.20	2349.32
3572.45	2367.34	2390.12
2356.70	3415.30	2379.34

ABC4	ABC5	ABC6
2340.12	1265.30	
3456.30	2357.30	
3467.23	2356.78	

然后利用 DEBUG 或 PCTOOLS 等工具软件,将字段名 ABC5改成 ABC2,这样该数据库具有两个相同的字段名 ABC2。

•!DEBUG LS.DBF

—E1A0 “ABC2”

—W

—Q

对于相同字段名,FoxBASE 是如何处理

默认的字段的呢?以 REPLACE 命令为例,略见一斑。

•USE LS

•REPL ALL ABC6 WITH ABC2

•LIST

则三个记录中各字段的值分别为:

ABC1	ABC2	ABC3
1254.20	1367.20	2349.32
3572.45	2367.34	2390.12
2356.70	3415.30	2379.34

ABC4	ABC5	ABC6
2340.12	1265.30	1265.30
3456.30	2357.30	2357.30
3467.23	2356.78	2356.78

读者不难看出,FoxBASE 取靠后边的同名字段为其默认字段,实际参与计算或替换,忽略了其前面的字段。

根据上述原理,在数据库中选择一重要数据的字段(如上例中 ABC2)为同名字段,在前面的同名字段中设置真实的数据,在后面的同名字段中设置虚假的数据,然后在数据库中备份区域(如 DBF 文件中第29~32字节)设置口令加密,解密时询问口令,若回答不正确,则将数据库中后一字段名改成和前一字段名完全相同,统计时将使用虚假的数据,结果完全错误;若回答正确,则将数据库中后一字段名改成和前一字段名不相同,统计时将使用真实的数据,结果正确。

同名字段加密法和传统的数据库加密法相比有其独特之处,如果输入口令不正确,并不退出或死机,而是调换成对另一字段的操作,将使非法用户误认为一切正常,而实际上将出现完全错误的结果。至于同名字段中数据

的误差范围和错误程度是用户应妥善考虑的,为了避免用户在交互操作中发现相同的字段名,可在字段名字的选择和字段名的排列上做得隐蔽一些,并且将口令字进行某种变换然后写入数据库,保密性能更好。

程序一 dbfkl. bin 可为任何 FoxBASE 数据库设置口令字。使用方法:

- load dbfkl
- call dbfkl with<数据库文件名>

程序二 dbfls. bin 是针对上面举例的数据库(ls. dbf),将字段名 ABC5改成 ABC2的程序,用法同上。实际上只要修改程序中相关数值,就能适用于任何数据库的同名字段名的修改,详见程序中的注释。程序三为示范程序,并且已经为数据库 LS. DBF 设置了口令,显然对于口令的正确与否,计算的结果并不相同。

源程序清单如下:

;程序一

;DBFKL. BIN 数据库加口令

;编译链接后用 EXE2BIN 程序转换

```
code segment byte
    assume cs:code
    org 00h
start proc far
    jmp begin
ds __ ptr dw 0000h
handle dw 0000h ;文件句柄
pawo db 28d dup(?)
paword db 4d dup(?)
sus __ msg db '数据库加口令成功!',24h
not __ msg db '非 DBF 文件!',24h
ask __ msg db '请输入口令(4位):',24h
begin:
    push ds
    pop ax
    mov ds __ ptr,ax
    push cs
    pop ds
    push cs
    pop es
    push ds
    mov ds,ds __ ptr
    mov dx,bx ;ds,dx 指向文件名
    mov ax,3d02h
    int 21h ;打开文件读/写
    pop ds
    mov handle,ax
    mov bx,ax
    mov dx,offset pawo
    mov ah,3fh
    mov cx,0020h
    int 21h ;读前20H 内容
    mov si,offset pawo
    mov ax,[si]
```

```
or al,80h
cmp al,83h
jne ndb ;是否是 DBF 文件
jmp kwo
```

ndb:

```
mov dx,offset not __ msg
mov ah,9
int 21h
jmp exit
```

kwo:

```
mov bx,handle
mov ax,4200h
mov cx,0
mov dx,0
int 21h ;移动指针到文件头
mov dx,offset ask __ msg
mov ah,9
int 21h
mov cx,4
mov si,offset paword
```

read:

```
mov ah,7
int 21h
not al ;简单变换
mov [si],al
inc si
loop read
mov ah,40h
mov dx,offset pawo
mov cx,0020h
mov bx,handle
int 21h ;写文件
mov dx,offset sus __ msg
mov ah,9
int 21h
```

exit:

```
mov bx,handle
mov ah,3eh
int 21h
ret
```

start

```
endp
```

code

```
ends
```

```
end start
```

;程序二:

;UNDBFKL. BIN 数据库同名字段加密编译链接后用 EXE2BIN 程序;转换 LS. DBF ABC5修改成 ABC2地址00A0H

```
code segment byte
    assume cs:code
    org 0h
start proc far
    jmp begin
ds __ ptr dw 0 ;AX
handle dw 0 ;文件句柄
buff1 db 1ch dup(?) ;文件头
buff2 db (5-1)*20h+4 dup(?) ;5表示 ABC5
    是第5字段
buff3 db 8 ;ABC5
old __ pw db 4 dup(?) ;原口令
new __ pw db 4 dup(?) ;新口令
zd1 db 'ABC2',0,0,0,0 ;修改后字段名
```

```

zd2      db 'ABC5',0,0,0,0    ;原字段名
ask __msg db '请回答口令:',24h
not __dbf db '不是 DBF 文件!',24h
begin:
    push ds
    pop ax
    mov ds,__ptr,ax
    push cs
    pop ds
    push cs
    pop es
    push ds
    mov ds,ds,__ptr
    mov dx,bx
    mov ax,3d02h    ;打开文件读写
    int 21h
    pop ds
    mov handle,ax
    mov bx,ax
    mov dx,offset buff1
    mov ah,3fh
    mov cx,5*20h+8h    ;5表示 ABC5是第5字段
    int 21h
    mov si,offset buff1
    mov ax,[si]
    or al,80h
    cmp al,83h    ;是否是03或83
    jne no __dbf
    jmp n0
no __dbf: mov dx,offset not __dbf
    mov ah,9
    int 21h
    jmp exit
n0:      mov si,offset buff2
    mov di,offset old __pw
    mov cx,4
    cld
rest:    lodsb
    not al
    stosb
    loop rest
    mov dx,offset ask __msg
    mov ah,9
    int 21h    ;输入口令
    mov cx,4
    mov si,offset new __pw

read:    mov ah,07
    int 21h
    mov [si],al
    inc si
    loop read
    mov si,offset old __pw
    mov di,offset new __pw
    mov cx,4
    repz cmpsb    ;检查口令
    je n1
    mov si,offset zd2
    jmp n3
n1:      mov si,offset zd1
n3:      mov di,offset buff3
    mov cx,8
    repz movsb
    mov bx,handle
    mov ah,42h
    mov al,00h
    mov cx,0000h
    mov dx,0000h
    int 21h
    mov ah,40h
    mov dx,offset buff1
    mov cx,5*20h+8h    ;5表示 ABC5是第5字段
    mov bx,handle
    int 21h
exit:    mov bx,handle
    mov ah,3eh
    int 21h
    ret
start    endp
code     ends
end start

;程序三:
;LOAD UNDBFKL
CALL UNDBFKL WITH 'LS.DBF'
USE LS
SUM ALL ABC2 TO NU
?NU
* * 口令正确 NU=5979.38
* * 口令错误 NU=7149.84
    
```

SKILL

(上接第56页)

__0000X. TMP(X 为执行 DOSSHELL 的次数,X=1,2,3,...)的临时文件,该文件在运行其他实用程序时不能删除,否则,将不能正确返回到 FoxBASE+ 中。

下面是笔者编制的一个简单的 DOSHELL. PRG 程序,在 MS-DOS 5.0 以上版本中运行。

```

1:SET STAT OFF
2:SET TALK OFF
3:SET COLOR 7/0
4:CLEAR
5:@12,0,SAY'退出 DOSSHELL 状态,请按 EXIT 返回汉字
    
```

```

FoxBASE+'
6:RUN FOXSWAP COMMAND /K prompt(FOX SHELL)$P$G
7:CHEAR
8:RETURN
    
```

其中第6行可替换成:FOXSWAP COMM. BAT,这样,DOSHELL. PRG 程序在任何 MS-DOS 版本中均能运行。执行 DOSHELL. PRG 程序后,将显示新的 MS DOS 提示符(FOX SHELL)D:\CGJ\FOX>,表示现在处于 FOX 的 DOSSHELL 状态下,并可执行大型应用程序。

SKILL

忘记 WPS 文件口令的恢复技巧

□王东辉

W

PS 是继 WS 编辑软件之后的又一深受用户欢迎的文字处理系统。金山公司为该系统设计了友好的用户界面,丰富的编辑打印功能。尤其在文件的加密上,更是花费了功夫。但是,密文必须要有密码。一旦文件合法用户忘记了密码,或者误操作,输入不明字符,原文件就无法使用,造成损失。

为了恢复文件,笔者分析了 WPS 的密码体系,获得成功,现提出以供参考。

WPS 文件不同于其他编辑软件的是:在正文前有一个长 1024 字节的文件头。包括文件的控制信息,其中与密码设置有关的字段为:

(1)文件头偏移 0002H 处的两个字节放着文件长度(不包括文件头)。这也说明 WPS 处理文件正文的长度最大为 65535 字节。

(2)在偏移 02DDH 处是八个字节的密码。但密码已经过处理,而不是密码原代码。其处理过程为:设一个字节的密码原代码为 ZHXL,将 ZH 和 XL 高低半字节交换,然后分别求反,再合二为一,即为已加密的原码。而密码不够八个字节的,系统自动以 00H 补足。

(3)偏移 0400H 处为文件正文(已被加密)

WPS 对正文的加密,是简单的将密码(02DDH)与正文进行异或,八个字节一个循环,直到文件结束。

通过上面的分析可知,WPS 加密采用了两次加密过程:不仅加密密码而且加密原文件。如果只是简单除去 02DDH 处的密码,仍然无法得到原文件。不过,了解了加密过程,

我们就可以采用两方法进行解密:

1. 破译密码

按 WPS 处理密码原代码的过程逆向处理,就可获得密码原代码。这种方法简便快捷,而且不“破坏”密码。但有些原代码为不可见字符,输入时会有麻烦。

2. 直接恢复原文件

我们可以编个小程序,截获 02DDH 处的密码,然后根据文件长度,将 0400H 处的原文件与密码再次异或,就可恢复原文件,但必须将 02DDH 处的密码填为 00H。这时,再用 WPS 调用该文件,就可不经输入密码而直接得到原文件。如果想再加密,可以重新设置密码。

本文所附程序 JM _ WPS. ASM 就是根据上面思想用汇编语言编写的。程序经过汇编、连接后生成 EXE 文件,然后用 EXE2BIN 命令转化为 COM 文件。使用时,根据提示输入 WPS 原文件后,就可生成同名的已恢复的原文件。该程序可在 PC/XT、286、386 等多种机型上使用。

源程序清单如下:

```
code      segment
          assume cs:code,ds:code,es:code
          org 100h

start:

          jmp begin

flag      db 0
handle    dw 0

num       dw 0
mm        db 8 dup(0),0ah,0dh,'$'
jamifile  db 32 dup(0)
buffers   db 4000h dup(0)
sinfo     db 'Please input jamifile: ','$'
helpmsg   db 'Usag:JM _ WPS filename ',13,10,'$'
```

```
errmsg1 db 'Requires DOS 3.0 or higher ',13,10,'$'
ferr0 db 'File not be opened !','$'
ferr1 db 'File read error !','$'
ferr2 db 'File write error !','$'
ferr3 db 'File closed error !','$'
fine db 0dh,0ah,'----- OK -----','$'
begin:
```

```
push cs
pop ds
push cs
pop es
cld
mov si,81h ;判断命令行是否带
;参数"/?"
```

```
call scanhelp
jnc checkver
mov ah,9
mov dx,offset helpmsg
int 21h
mov ax,4c00h
int 21h
```

```
checkver:
mov dx,offset errmsg1 ;检测 DOS 版本
mov ah,30h
int 21h
cmp al,3
jae checkfile
jmp err
```

```
checkfile:
mov dx,offset sinfo ;获取 WPS 文件名
mov ah,9
int 21h
mov si,offset jamifile
call get__name
push cs
pop ds
mov dx,offset jamifile ;打开 WPS 文件
mov ax,3d00h
int 21h
mov dx,offset ferr0
jc err
mov handle,ax ;保存文件句柄
mov bx,ax
mov cx,1024 ;将 WPS 文件头(1024
;字节)读入缓冲区
```

```
lehead
mov ax,3f00h
mov dx,offset buffers
int 21h
mov dx,offset ferr1
jc err
mov si,offset buffers ;获取 WPS 正文长度
; (以字节为单位)
mov ax,[si+2]
mov num,ax
mov bx,handle ;关闭 WPS 文件
mov ah,3eh
int 21h
mov dx,offset ferr3
jc err
mov dx,offset jamifile ;重新打开 WPS 文件
mov ax,3d00h
int 21h
```

```
mov dx,offset ferr0
jc err
mov handle,ax
mov cx,num+1024 ;根据正文及文件头长度,
;将文件调入缓冲区
```

```
mov bx,handle
mov ah,3fh
mov dx,offset buffers
int 21h
mov dx,offset ferr1
jc err
mov bx,handle
mov ah,3eh
int 21h
mov dx,offset ferr3
jc err
mov si,offset buffers ;获取密码(8 个字节)
mov di,offset mm
add si,2ddh ;将密码首址放入 SI
mov cx,8
jmp loopp
```

```
err:
mov ah,9
int 21h
mov ax,4c01h
int 21h
```

```
loopp:
mov al,[si]
mov [di],al
mov al,0
mov [si],al ;将原密码置 0 以除去
;密码
```

```
inc si
inc di
dec cx
jnz loopp
mov cx,num ;对 WPS 正文进行解密
push cx
mov si,offset buffers
add si,1024
```

```
loop1:
mov cx,8
mov di,offset mm
```

```
loop2:
mov al,[si]
mov bl,[di]
xor al,bl ;通过异或操作产生原
;代码
```

```
mov [si],al
inc si
inc di
dec cx
cmp cx,0
jnz loop2
pop cx
cmp flag,1
jz exit
sub cx,8
push cx
cmp cx,8
```

```

ja loop1
mov flag,1
mov di,offset mm
jmp loop2

exit:
push cs
pop ds
mov dx,offset jamifile ;以原文件名产生新
mov ax,3c00h ;文件长度变为 0
mov cx,0
int 21h
mov dx,offset ferr0
jc err
mov bx,ax ;将修改后的缓冲区
mov ax,num ;内容写入新文件
add ax,400h
mov cx,ax
mov ax,4000h
mov dx,offset buffers
int 21h
mov dx,offset ferr2
jc err
mov bx,handle ;关闭该文件
mov ax,3e00h
int 21h
mov ah,9 ;响铃,同时给出成功
;信息

mov dx,offset fine
int 21h
mov ax,0e07h
mov bx,0
int 10h
mov ah,4ch
int 21h
ret

scanhelp proc near
push si ;保存 SI
scanloop:
lodsb ;获取字符,若为回
;车,
;退出

cmp al,0dh
je scan__exit
cmp al,'?' ;若不是“?”循环
jne scanloop
cmp byte ptr[si-2], '/' ;若不是“/”循环
jne scanloop
add sp,2 ;清栈
sub si,2 ;修正 SI
stc ;置标志退出
ret

scan__
exit:
pop si ;出栈 SI
clc ;清标志退出
ret
scanhelp endp
get__ near proc
name
push si ;保存文件名缓存区
;指针

mov ah,3 ;获取当前光标位置
mov bx,0
int 10h
push dx

lop:
mov ah,0
int 16h ;等待字符输入
cmp al,0dh ;若是回车,退出;若
;有光标键按下,列数
;减 1,缓存指针减 1
jz ext

cmp ah,4bh
jnz next
pop dx
dec dl
dec si
jmp next1

next:
cmp ah,4dh ;若左光标键按下,列
;数减 1,缓存指针加
1
jnz next2
pop dx
inc dl
inc si

next1:
push dx ;调用函数以置光标
;新位置
call setpos
jmp lop

next2:
mov [si],al ;若是文件名,在回显
;的同时,存入文件名
;缓冲区

mov ah,0eh
mov bx,0
int 10h
inc si
pop dx
inc dl
push dx
call setpos
jmp lop
pop dx

ext:
pop si
ret

get__
name
setpos proc near ;置光标位置
push dx
mov ah,2
mov bx,0
int 10h
pop dx
ret
setpos endp
code ends
end start

```


为 DOS 系统增加一个模拟 TYPE 的外部命令

□ 万其应

T

YPE 命令是 DOS 中的一条内部命令, 主要用来浏览文本文件的内容。笔者在使用 TYPE 命令时, 发现它有一点缺陷, 当文本文件的内容超过一屏时, 它不能分屏显示, 必须使用 Pause 键或 Ctrl-S 键使之暂停显示, 以供阅读。由于按 Pause 或 Ctrl-S 键的时间不同, 可能想阅读的东西待按键时已经不在屏幕上了, 尤其是在阅读的文件很长时, 这一缺陷表现得更为突出。为此, 笔者用 Turbo C2.0 编写了一个小程序 CTYPE.C, 通过编译、连接生成 CTYPE.EXE 可执行文件, 再把 CTYPE.EXE 放置在 \DOS 子目录下, 实现了模拟内部命令 TYPE 的功能, 达到了分屏显示的目的。每屏显示的行数自定, 在屏幕上方显示文件名, 下面显示页号, 按任意键显示下一页, 按 ESC 键返回 DOS 状态。

CTYPE 同 DOS 其他外部命令一样使用, 在 DOS 提示下, 键入:

```
[D:][PATH]CTYPE [D:][PATH]
FILENAME[.EXT] N
```

其中各项为 CTYPE.EXE 所在的路径、所阅读的文件路径和文件名、每屏显示行数等。如: C>\DOS\CTYPE\WQY\README.TXT 20 即为以每屏 20 行阅读子目录 WQY 下的 README.TXT 文本文件。

本程序用 Turbo C2.0 下编写, 适用于中英文环境, 可提供中文和英文两种提示, 还可根据需要舍去中文或英文提示。

本程序在东海系列、联想系列、AST、COMPAQ 等机型上运行通过。

源程序清单如下:

```
#include "stdio.h"
```

```
main(int argc, char * argv[])
{
    int i, pg, nub, flag, page;
    int c;
    char buf[132];
    FILE * fp;
    if(argc != 3)
    {
        printf("use: %s %s %s\n", "CTYPE", "filename",
            10);
        printf("使用方法: %s %s %s\n", "CTYPE", "文件名",
            "每屏显示行数");
        exit(1);
    }
    if((fp = fopen(argv[1], "r")) == NULL)
    {
        printf("%s not found\n", argv[1]);
        printf("文件: %s 没找到\n", argv[1]);
        exit(2);
    }
    nub = atoi(argv[2]);
    flag = page = 1;
    while(flag)
    {
        system("cls");
        printf("\t\t %s \t\t\n", argv[1]);
        printf("\t %s \t\n", "=====");
        for(i = 1; i <= nub; i++)
        {
            if((fgets(buf, 132, fp) == NULL)) exit(3);
            else printf("\r%s", buf);
        }
        pg = page;
        page++;
        printf("\n%s %d--", "--PAGE ", pg);
        printf("\t-第%d页--PAGE (按任意键继续, Esc 返回
            DOS)\r", pg);
        printf("\007");
        while(bioskey(1) == 0);
        c = bioskey(0) & 0xff;
        if(c == 27) exit(3);
    }
    fclose(fp);
}
```

SKILL

使用 Access2.0 的客户机/服务器特性

Microsoft Access2.0 提供了新的客户机/服务器功能,这大大方便了那些欲把 Access 应用系统移植到客户机/服务器环境下的开发人员。这些新功能包括:新的 SQL 执行(Pass-through)方法(在服务器中直接执行已存好的过程);改善了数据库机(以 ODBC 来存取数据,不管这些数据是什么格式,也不管他们存放在什么地方)。

开发人员可用 Access 2.0 来升级已有的应用系统,也可以新创建客户/服务器模式下的应用系统。

一、基本设计原则

首先,我们先提出以下几条原则以提高程序执行效率(在文件服务器和客户/服务器环境中都适用):

1. 设计的所有表格,在打开时都不要从服务器中直接读取数据,建议在表格中增加一个命令按钮来专门读取数据。把表格的 allow editing 属性置成 data entry 即可。

2. 使用“快照记录集(Snapshot-type Recordset)”对象,如果结果中仅包含少数字段并且不包括 OLE 对象或大的 MEMO 字段,这样你便无需直接去修改服务器中的关系表。

3. 尽量减少服务器上的 Combo Box 列表中的项数。一般情况下,最好使用来源于“快照记录集”对象填充 Combo Boxes。

4. 如果列表中的数据不经常变化,可在本地保存一个服务器上的表的副本,如果在本地表中再建上索引,列表选择会更快。副本

的更改最好对用户透明。

5. 如果应用系统需对多组数据进行比较,便需把从服务器中获取的数据存在本地表中。开发者让用户能以适当方式来存取这些数据。

二、使用 Access 的界面

为了有效地操作 Access 2.0 的用户界面,最好不要去大改 ODBC 所附的大表中的记录指针的位置。可使用新提供的 SQL pass-through 选项来执行存储和基于服务器的过程。这些程序直接通过名字来调用,节省了时间,也减少了网络传输量。在 2.0 版中,所有 Access 对象都可通过 SQL pass-through 查询来直接查询数据。

三、服务器性能和可靠性的提高

下面的改进和提高,可能涉及到对 Access 源码的修改,MSACC 20. INI 和应用系统本身的. INI 文件的修改,以及在服务器数据库上创建额外的存储过程。

如果你的服务器支持 time stamp 数据类型(时间标记,以进行版本控制),便可在所有服务器表中增加一个 timestamp 字段。有了这一字段,服务器在修改或删除一条记录时便会对这一字段进行核查,一旦其值有变化,修改或删除操作便会被中止。

在 Microsoft Access 2.0 中,可用事务机制来对多个数据库对象进行集中修改。但是,即使你的服务器能处理嵌套的事务,也最好不要在 SQL 调用中嵌入 BeginTrans 语句,因为他们相互冲突。

当你使用事务机制处理表时,Jet 一次仅处理一条记录。因而,你可以执行一些服务器不支持的集中操作。一般情况下,集中操作用 SQL 做要快一些。如果需要一批事务(如订单处理系统),最好用 Access 的 BeginTrans, CommitTrans, Rollback。

如果不希望屏幕实时反映数据的变化,可把 ODBC 的刷新间隔置为最大(3600 秒),方法是设置 Access 的 Application.SetOption。用户中止应用系统时,需把它设回默认值(1500 秒)。间隔越短,实时性越强。

“快照记录集”对象默认情况下是可双向浏览记录的。如果只需单向浏览记录,可把 DB _ FORWARDONLY 选项同 OpenRecordset 的选项变量作或运算。这样“快照”效率会很高,因为结果数据无需拷贝到一个数据缓冲区中。

在 MSACC20.INI 的 [ODBC] 一项中增加一项 TraceSQLMode = 1 (在自己的 .INI 文件中增加亦可,检查对 ODBC 数据进行操作的 SQL 语句。SQL 语句存在 Access 安装目录或当前目录下的 SQLOUT.TXT 文件中。

对经常使用的数据最好放在本地以避免网络堵塞。如果本地数据存储异常频繁,可使用 Access 2.0 的缓冲管理功能。用 CacheSize 和 CacheStart,来存储频繁使用的数据(在本地内存中),再用 FillCache 来将服务器中的数据填进缓冲区。本地的数据存取操作可加速表格显示。由服务器来处理传统的 Select、Append、Delete、Update 语句可避免网络堵塞,加快执行速度。唯一的例外是处理那些异构的多表联接运算,ANSI SQL 无法表达的 SQL 语句。服务器在这种情况下把查询分解为本地和网络两部分以提高执行速度。

四、服务器锁定和错误处理

服务器负责对表进行页级(Page-Level)或记录级锁定。对所有表格,RecordLocks 属性必须置为 No Locks 对服务器上的表,Microsoft Access 并不对记录加锁除非是修改操作。

我们可用传统的 Access 错误捕捉机制来处理服务器上产生的错误信息,并可用文字处理工具来编辑错误信息。

五、客户机/服务器下的修改事务

在经过 NumberOfUpdateRetries 定义的数个重

试后,一个修改操作可能会最终失败。如果你想让用户能重试、放弃、存储当前数据,可把事务放在一个循环中。但是经过 ODBC 送到服务器上的事务不能嵌套。

通过批量事务可向日志文件中追加记录。修改服务器上的表时,务必做一致性检查。为了达到最佳速度,在打开记录集时可选用 DB _ APPENDONLY 选项。

六、管理服务器连接

在构造数据库服务器时,我们需要估计一下服务器能同时支持多少个连接:

1. 如果服务器的光标交付动作是去关闭所有活动的语句(没返回结果的查询),诸如插入、修改、删除等,服务器须支持多个并行连接以保持数据一致。如果服务器不允许一个连接中有多个活动语句,并行连接的数目取决于应用系统创建的并行活动语句的数目。这两点就决定了服务器的连接方式。

2. Microsoft Access 2.0 对“动态集”对象(由少于 100 条记录的查询而创建)只需一个连接。“快照”不允许修改因而也只需一个连接。

七、结束语

用 Microsoft Access 2.0 把应用系统从文件服务器移植到客户机/服务器上,可采用下面七个步骤:

1. 创建服务器上的数据库。
2. 以 ODBC Administrator 来创建 ODBC 资源,以 ODBC Administrator 系统来选择 ODBC 的驱动程序。
3. 整理已有的 Access 数据库。可用 Microsoft Access 2.0 的 Database Documentor 工具来打出一份 .MDB 文件中的所有库的属性。
4. 把所有库移至服务器上。为了加快速度,可用 Microsoft Access 的 File Export 命令来完成这一工作。
5. 给服务器上的表增加索引,无索引的表服务器不能处理。
6. 对服务器的表做一致性限制。可用刚生成的数据库文档来增加默认值、域一致性等限制。
7. 连接上服务器。最容易的方法是用数据库中的 File Attach 命令。

SKILL

终端打印机接口程序的分析与设计

□唐兆海 石学森

U

NIX/Xenix 是一种多用户操作系统,终端用户在系统中使用终端打印机是比较广泛的。而怎样安装与配置终端打印机呢?终端打印接口程序的工作原理是什么?如何进行接口程序的设计?等,现将笔者对终端打印接口程序的一点粗浅认识介绍给同仁,希望对大家能有所帮助。

一、终端打印接口程序的分析

在 UNIX/Xenix 系统中,每个加进假脱机打印系统内的打印机,均需为其编制一个接口程序,否则,该打印机就不能工作;虽然系统内已提供了一些常见的打印机接口程序供其选择,但有时为了特殊的需要,不得不舍弃这些已有的接口程序而另行编制特殊用途的打印接口程序,因此我们懂得打印机接口程序的编制原理,明白其设计方法,是很重要,也是很有用的。

在多用户系统中,所有加在假脱机打印系统内的打印机的接口程序,均存于系统假脱机打印目录/usr/spool/lp/interface下,当假脱机打印的调度程序调用某一打印机接口程序时,其命令格式为:interface/ P.request -id name title copies options files,其中:interface/ 为接口程序;P. 打印机的名称;request-id 由 LP 返回的请求号;name 由发出该打印请求的用户注册名;title 由用户说明的可选择标题;copies 被打印文件要打印的份数;options 由用户说明的某打印机的列表;files 以空格间隔的要打印的文件名组。

以上命令格式所引用的参数,在 shell 语言里,从左到右分别对应于位置参数 \$0、

\$1、\$2、\$3、\$4、\$5……。当假脱机打印的调度程序调用打印接口程序时,接口程序将按照命令行上的给定参数进行格式化输出,其将标准输入定为/dev/null,把标准输出与标准错误输出都定向到打印机设备上;但是我们使用每个加到假脱机打印系统中的终端打印机打印时,不是直接传送数据终端打印机的,而是借助于终端机作为桥梁,间接地向终端打印机发送打印指令,因此,终端打印机的接口程序与主机系统打印机的接口程序就略有不同。

以下是笔者在 ZJ 386/33 上 Xenix System V 2.32 中,为某个端口(设为 tty5h)上的仿真型号为 ct100 的终端机编制的打印接口程序(如程序 1)。

1. 变元处理部:从第 1 行到第 7 行,程序将命令行中前面的五个位置参数分别赋给相应的变量,接着再把这些位置参数左移出去,以保证命令行中的位置参数仅包含要打印的文件名组。

2. 终端设定部:从第 8 行到第 12 行,以 stty 命令对终端进行设定,使接口程序具备合适的 stty 方式,这一点在配置终端打印机时,尤其重要,不可缺少。接下来的两个“echo”语句,是设定 ct100 仿真终端为透视打印方式,为主机进行直接打印作准备。

3. 前页处理部:从第 13 行到第 48 行,打印的是有关这次打印请求的用户信息,一般称之为前导页,在系统内若用户不是很多时,就没有必要打印。该段处理过程为:首先判断命令行参数 \$5 是否为“-ob”,若是则不打印前导页内容,否则在第 22 行到第 27 行采集/etc/systemid 文件含有的运行 Xenix 系统的

二、接口程序的简化

三、接口程序的 C 语言实现

四、公用的终端打印 C 程序

源程序清单如下:

69

```
# nhead gets the value of BANNERS or 1 by default
nhead='sed -n's/^ BANNERS=//p' /etc/default/lpd'
[" $nhead" -ge 0 -a "$nhead" -le 5] || nhead=1
# print the banner $nhead times
while [" $nhead" -gt 0]
do
    echo "$x\n"
    banner "$name"
    echo "$x\n"
    [" $user"] && echo "User: $user\n"
    echo "Request id: $request\n"
    echo "Printer: $printer\n"
    date
    echo "\nMachine: $sysid\n"
    [" $title"] && banner $title
    echo "\f\c"
    nhead='expr $nhead - 1'
done
# send the file(s) to the standard out $copies times
while [" $copies" -gt 0]
do
    for file
    do
        cat "$file" 2>&1
        echo "\f\c"
    done
    copies='expr $copies - 1'
done
echo "\033ch"
echo "\033}"
exit 0

# 程序 2:
# Dumb serial printer
# Options: lp -ob no banner
shift;shift;shift;shift;shift;
# If it is necessary to change the baud rate or other stty settings for
# your serial printer modify the following line:
stty ixon ixoff 0<&1
echo "\033{"
echo "\033ch"
# send the file(s) to the standard out $copies times
for file
do
    cat "$file" 2>&1
    echo "\f\c"
done
echo "\033ch"
echo "\033}"
exit 0
```

//程序 3:

```
# include<stdio.h>
char fname[30];
main(ac,av)
int ac;
char *av[];
{FILE *f1, *f2, *fopen();
int n;
if (ac < 7) error();
```

```
for (n=6;n<ac;n++)
{ sprintf(fname,"%s",av[n]);
if ((f1 = fopen(fname,"r")) == NULL)
error(); filecopy(f1,stdout); fclose(f1);
}exit(0);
}
filecopy(f1,f2)
FILE *f1, *f2;
{ char c; char pon[10],poff[10];
strcpy(pon, "\0333\n");
strcpy(poff, "\0334\n"); fputs(pon,f2);
while ((c = getc(f1)) != EOF)
{if (c == '\n') putc('\r',f2); putc(c,f2);
} fputs("\f",f2); fputs(poff,f2);return(0);
}
error()
{ fprintf(stderr, "Usage: lp -dprinter __ name filename\n");
exit(1);}
```

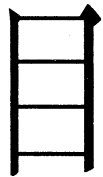
//程序 4:

```
# include<stdio.h>
char ttyname[5], fname[30];
main(ac,av)
int ac;
char *av[];
{ FILE *f1, *f2, *fopen();
int n;
strcpy(ttyname,getenv("TERM"));
if (strcmp(ttyname,"ansi") == 0){
error1(); return(1); }
if (ttyname[0] != 'c'&&ttyname[0] != 'g'
&& ttyname[0] != 'a')
{error1();}
if (ac<1) error();
for (n=1;n<ac;n++)
{ sprintf(fname,"%S",av[n]);
if ((f1 = fopen(fname,"r")) == NULL) error();
filecopy(f1,stdout); fclose(f1);
}exit(0);
}
filecopy(f1,f2)
FILE *f1, *f2;
{int c; char pon[10],poff[10];
switch (ttyname[0])
{ case 'c':strcpy(pon, "\033{\n");strcpy(poff, "\033}\n");break;
case 'a':strcpy(pon, "\0333\n");strcpy(poff, "\0334\n");break;
case 'g':strcpy(pon, "\033\n");strcpy(poff, "\033a\n");break;
} fputs(pon,f2);
while ((c = getc(f1)) != EOF)
{if (c == '\n') putc('\r',f2); putc(c,f2);
} fputs("\f",f2); fputs(poff,f2);return(0);
}
error()
{ fprintf(stderr, "Usage: lpt filename\n");
exit(1);
}
error1()
{ printf("\n\007This lpt program use only ct - 100, tvi925, adds
terminals\n\n");exit(1);
}
```

SKILL

Novell 网服务器安装技巧

□王东辉



前,随着 Novell 公司网络技术的发展和完善,安装 Novell 网的单位越来越多,而安装最普遍的是 NetWare 386 V3.1 版。

NetWare 386 V3.1 是从 NetWare 386 3.0 升级而来。它不仅提供了更强的功能和完全开放的体系,在安装上实现了菜单化和结构化,更加方便、可靠。关于如何安装,手册已给出具体步骤,在此,笔者仅结合实践谈谈在安装中的一些体会。

1. 如果您想首先使用 DOS 启动您的文件服务器以保证它可作为独立的计算机使用,那么在产生 DOS 分区后若是第一次安装 NetWare 系统,一定要保留一部分硬盘为非 DOS 使用,用以产生 NetWare 分区。

2. NetWare 386 3.1 安装时,先要运行 SERVER 命令(在 Operating System-1 盘上)。这个命令在加载安装程序之前,先要检查系统是否有 1M 字节的扩展内存(Extended Memory),并且扩展内存必须从高端内存区 UMB(介于 640K 到 1MB 之间)的结尾,即高内存区 HMA(High-Memory Area)处开始。这样,若在 CONFIG.SYS 中包含了“DOS=HIGH”命令,DOS 就会驻留在 HMA 中,这会使 SERVER 认为无法加载而退出安装,并显示“Insufficient Memory to run NetWare 386 (Requires at least 1 MegaByte of Extended memory)”。另外,当 CONFIG.SYS 中包含“DEVICE=C:\DOS\EMM386.EXE”时,由于 EMM386.EXE 需要 1MB 以上的内存用于扩充内存,故同样会使安装失败。所以,在安装前,CONFIG.SYS 中的这两条命令要去掉。

3. 如果采用“留取镜像”或“磁盘双重化”技术以保护数据,那么辅助分区要大于或等于主分区。若采用先用 DOS 启动的文件服务器,那么,最好在辅助分区也生成同样一个 DOS 分区,以便在主分区无法启动时,可以改由辅助分区启动。

4. 关于卷段的生成。如果一个卷跨越两台或更多台硬盘,那么会增加对该卷的存取速度,但是也带来一个问题,那就是一旦卷中某一硬盘故障或被关闭,那么整个卷也就被关闭。当然,若采用上面的磁盘镜像技术,就可防止此类问题。

5. 在安装 LAN 驱动程序时,最好使用系统给出的缺省值,若网上已有别的服务器,那么它使用的值,别的服务器就不能再用。比如,有一台服务器的 Interrupt Number 选中了 3,那么该值就不能再被其他服务器选用。所以这也限制了网上服务器的数量。

6. 若一台服务器中有多块网板,那么就需要多次安装 LAN 驱动程序,这时就必须给每个驱动程序以不同的网络编号,这个编号不同于服务器的 IPX 内部网络编号。它即可用于同一服务器上不同驱动程序的通讯,也可用于不同服务器的驱动程序通讯。

上面列出的有关 Novell 网络安装的几点,是安装服务器时经常遇到的问题,如何更好地发挥服务器的效能,很大程度上取决于服务器的配置。目前的 NetWare 386 V3.1 比 NetWare 286 V2.15 的安装更容易,只要弄清自己的网络设备、合理利用,一定能配置一台高效、安全的服务器。

SKILL

Novell 网络应用系统常见故障的维护技术

□朱 猛

本

文以 NetWare 386 V3.11 为例,根据笔者管理和使用医院网络管理信息系统的实践经验,谈谈一些常见网络故障的排除方法和体会。

下面就最常见的几种故障加以讨论。

一、连线故障

连线故障占有 LAN 故障的 80%,其现象是:NetWare 工作在进网时执行 NETX.COM(如 NET3.COM)后死机或出现如下错误信息:

“A file server could not be found”

这类错误由 DOS 外壳引起,NETX.COM 执行时试图同网络建立一个连接,但在给定超时间隔内没有 Server 响应此请求,因而工作站无法入网。出现这类错误的最可能原因有:电缆断裂;电缆与 BNC T 型头的连接出现松动,引起断路或短路;工作站网卡驱动程序与网卡不配套;I/O 地址冲突或中断号冲突;病毒引起 IPX.COM 和 NETX.COM 损坏等。

一般来说,在试图分离网络上通讯故障时,必须采取自底向上的策略。从电缆本身开始,然后测试网络接口板,NetWare 的 COM-CHECK 实用程序提供了一种简单的工具来同时检查网板、电缆和其他设备的完整性,如果确认连接有误,可以从以下几个方面来检查并解决:

1. 电缆是否断裂,用万用表检测(如果有条件的话,可以用网络电缆测试仪检测),方法是在任一台工作站上断开网卡与电缆的连接,从 T 型头处断开电缆连接,测量两段电

缆,正常情况下的阻值应分别为 50 欧姆左右,用这种断开测量的方法,逐次排除即可找出故障电缆,找出后,换上一根好电缆即可排除故障。

2. BNC 头与 T 型头的接触是否良好。这类故障比较常见,检查方法是将 T 型头与 BNC 头断开,观察 BNC 的插针是否松动或低于端口边沿过多,否则会造成接触不良。

3. 如果是粗缆以太网,还应仔细检查粗 BNC 头的插针与同轴心的焊接情况是否良好,收发器电缆的焊接对应序列有无错误,焊接状态正常与否。

下面介绍一种笔者经常使用的很有效的故障定位方法:首先,服务器与故障线路上最近的一个工作站相连,其余的则断开,用终端匹配电阻分别加在工作站和服务器网卡的 BNC T 型插头的一端,在该工作站上执行 IPX.COM 和 NETX.COM,如正常,则表明该工作站及其服务器的连接正常。再加入与该工作站相邻的另一工作站,采用同样方法,执行 IPX.COM 和 NETX.COM,若正常则继续加入下一个相邻的工作站,直到加入某工作站时不能进网,则故障肯定在该工作站及其到上一个相邻工作站的连接上,此时,按上述方法逐次检查即可。

二、网络接口板故障

网络接口板故障也是在连网过程中经常碰到的问题,有可能是计算机内部的其他硬件冲突,也有可能是运行 IPX.COM 出现如下错误信息:“ETHERNET HARDWARE FAILS TO RESPOND”,说明网卡设置与网络工作站

系统设置参数发生冲突。遇到这类问题可采用如下解决方法:

1. 查看工作站网卡驱动程序是否与网卡配套。如果在不清楚网卡类型而随便从其他工作站拷来 IPX.COM 的情况下便可能出现这类错误。解决方法是弄清楚网卡类型、IRQ 和 I/O 值,运行 WSGEN.EXE,重新生成正确的 IPX.COM,然后再运行 IPX.COM 看故障是否排除。

2. 察看网卡与计算机内部的其他硬件,如显示卡、网卡等,有无硬件中断冲突,如存在冲突,可改变网卡的 IRQ 值及 I/O 地址,然后运行 WSGEN.EXE,重新生成 IPX.COM,再运行 IPX.COM,若故障现象仍然存在,可拔下网卡,插入到其他在网上运行的且工作正常的工作站中,以便确认是否网卡有故障,若有故障,换上好的网卡,从而将故障排除。

3. 对于病毒引起的 IPX.COM 和 NETX.COM 文件损坏,解决方法是杀毒并重新生成 IPX.COM 和 NETX.COM。

另外,在网络维护中 NetWare 还提供了一些很有用的实用程序,例如 VREPAIR 实用程序,它可用来改正由于文件分配表和目录表项不匹配所引起的硬盘故障,还可消除错误的镜像文件。VREPAIR 是一个非破坏性的实用程序,用它修改别的文件时,不会删除或覆盖原来的数据;COMCHECK 实用程序可以在网络运行的任何时间用它把有故障的电缆和网络接口板分离出来。还有其他一些实用程序,此处不再一一列举。

三、应用软件或其他软件冲突

这类由应用软件的配置或使用不当而产生的故障,在实际应用中经常遇到,下面介绍一些笔者在应用中碰到的实际问题及解决方法。

1. 运行一应用程序后,执行网络命令发现所有网络搜索驱动器丢失。

一般用户在网络中建立了登录批文件,其中用 MAP 命令设置了必要的可搜索逻辑驱动器,如果应用程序又使用了 DOS 的 PATH 命令来指明路径,则 PATH 命令删除并重写了 NetWare 的搜索驱动器,在使用上势必造成混乱。解决的方法是去掉应用程序中的 PATH 命令或用 NetWare 的 MAP 命令代替 PATH 即可。

2. 有硬盘工作站在入网后进入汉字系统或退出 MFOXBASE 时死机,而无盘工作站则一切正常。

经检查发现该工作站 DOS 系统版本与文件服务器硬盘 DOS 系统版本不一致,从而使文件服务器

硬盘没有本地 DOS 设备驱动程序,所以当软件运行在映像到本地服务器卷上时,程序返回错误而死机。解决的方法是用文件服务器硬盘 DOS 系统启动工作站,执行 SYS C:传输系统即可。

3. 用户在运行 FoxBASE 时出现“Cannot create program workspace.”的错误信息,进不到点状态。

这是由于 FoxBASE 在运行中往往要产生一些临时文件,在启动系统或使用过程中要求当前目录下是可写的,即用户在当前目录下应具有 Write(可写)权。出现上述错误就是因为用户在当前目录下无权往服务器上写入文件,解决方法是:如是因为网络管理员权限设置过严,没有为用户目录设置可写权,则运行 SYSCON 正确设置权限即可;如是因为用户没有进到自己具有可写权的目录下,则进入具有可写权的用户目录下重新启动 FoxBASE 即可。

4. 多个用户同时对某个共享文件进行读写操作时,产生冲突。

Novell 网络支持下工作的 MFOXBASE 是多用户数据库管理系统,其最大特点是允许多个用户同时使用一个库文件。如果应用软件编程人员没有采取一些避免冲突的措施,那么用户在使用中会经常因为打不开文件、文件或记录加锁失败以及某种错误导致程序执行中断。因此,如何在网络环境下编写符合任务要求的应用系统就变得非常重要,以下几点经验供参考:

1. 如果条件允许,对使用共享数据库的用户在入网时间上进行限制或分配,以减少冲突机会。但这限制了工作站的工作灵活性和网络运行效率。

2. 一般情况下应以共享方式打开库文件(SET EXCLUSIVE OFF)。

3. 对共享数据库,可先建立一个临时数据库,任何程序打开共享库文件读出需要的记录到临时文件后,关闭原文件,对临时文件进行操作,当操作完成后再一次更新到共享数据库中去。

4. 在保证数据安全的前提下,能用文件加锁方式,就不要采用独占方式,能采用记录加锁方式,就不用文件加锁方式,且应尽量减少加锁时间。

5. 在程序中设置冲突陷阱,对要求打开数据库的时刻,都设置 ON ERROR 语句,一旦发生冲突,冲突处理程序开始执行,给予用户以相应提示信息,并由程序测试冲突是否结束,等到冲突结束后立刻继续原来的程序。

四、打印机及打印队列故障

1. 当应用程序要求打印机动作,而因为打印机

(下转第 58 页)

DOS 与 Windows

贵刊创刊号“DOS 与 Windows”栏目中给出了在安装 EMM386. EXE 以后屏幕出现“EMM386 exception”错误信息后死锁的解决方法,那么这是什么原因引起的?

引起 EMM386 exception 错误的最一般的原因是执行了一个坏的应用程序。通常,这类问题的引起是因为一个程序试图使用它没有通过正确的渠道向 DOS 请求使用的内存。也有时, EMM386 试图使用 PC 机的另外一些硬件,如一个 CD-ROM Player 或者视频卡已经使用的那部分内存空间。

本刊创刊号对此问题的解答中的 $X = \text{xxxx} - \text{xxxx}$ 这个开关 X 为控制上位内存使用的区域。这个区域是从 EMM386. EXE 已确认可用的地址空间减去指定的段地址范围。有效值从 A000 到 FFFF。

近来当我用 CHKDSK 命令时,出现一条错误信息提示有两个文件交叉链接。什么是交叉链接文件?怎样才能修复它们?以后怎样避免它们再次出现?

所谓交叉连接文件是指当两个独立的分配链(每个链对应一个文件)中出现同一个簇

时,就认为文件交叉链接了。正常情况下,磁盘上每一个分配过的簇都属于且只属于一个文件。因为交叉链接的文件有共享的簇,所以在许多方面存在危险。如果删除一个文件,另外一个文件也受影响,因为删除文件会将所占用的簇在文件分配表中对应的条目写为 0,以后 CHKDSK 会报告发现了“非法簇”。此外,如果一个文件的数据有了改变,另一个文件的内容也可能随之变化,无论从哪一方面看,交叉链接的文件都会带来灾难。

交叉链接现象的发生有不同的原因:硬盘数据区破坏,或硬盘控制器硬件故障都会导致错误的发生;当加电启动失败的情况发生时也会出现文件交叉链接现象。

为了修复这一错误,可用新的文件名拷贝这两个文件,因为 CHKDSK 命令会提示你所发生的交叉链接文件。然后删除原来的文件。这时如果愿意可以将文件拷贝改回原来的文件名,然后从建立这两个文件的应用程序对它们进行调用。一般来讲,由于有交叉链接现象出现,其中一个文件会保证正确,而另一个会受到破坏。用这样对文件进行调用的方法可以识别哪一个文件是好的,哪一个是不好的。在分配链中交叉链接的簇出现

越靠后,被损坏的文件中保持正确的数据量就越大。

如果执行 CHKDSK 时带有选项/F,则 CHKDSK 会顺着一个文件的分配链进行跟踪,如果在文件分配表 FAT 中遇到 0 或者簇号高于磁盘上簇的高编号,CHKDSK 就会报告发现了非法的分配单位。所谓“非法的分配单位”意思是在 FAT 中遇到的簇号值不可能是对的,而且还说明在非法簇号之后的各簇中所存放的那部分文件的内容丢失了。这时 CHKDSK 会将非法簇号改为分配链结尾符,并修改存放于文件的目录项中文件长度的值,用这种办法修复错误。如果非法簇号正好是记录在文件目录项中的起始簇号,则文件长度被修改为 0。文件的其余部分内容还以丢失的簇的形式存放于磁盘上。CHKDSK 负责将它们转化为根目录下的独立文件,文件名的格式为 FILFnnnn. CHK, nnnn 是 0000 - 9999 之间的数字。这时可检查 CHKDSK 建立的 .CHK 文件,如果你能辨认出这个文件,就可以将它加到被截取的文件上,这样恢复原来文件的工作就算完成了。

在 DOS 5 下装载设备驱动程序到高内存有什么要求?

! 如果你的机器系统是 80386 或更高的 CPU, 就可以利用 DOS 5 的扩展内存管理程序装载绝大部分设备驱动程序和内存驻留程序到高位内存, 但是同时必须符合下面几条要求:

(1) 系统必须具有至少 350K 的可利用的扩展内存。

(2) CONFIG. SYS 文件中必须包含有一个设置 HIMEM. SYS 的 DEVICE 命令, 而且这个命令必须比任何其他的 DEVICE 命令或 DEVICEHIGH 命令先出现。

(3) CONFIG. SYS 文件必须包含一条 DOS=UMB 或 DOS=HIGH, UMB 命令。

(4) CONFIG. SYS 文件必须包含一条关于 EMM386. EXE 的 DEVICE 命令, 且这个命令必须包含 NOEMS 或 RAM 开关, 必须出现在关于 HIMEM. SYS 的 DEVICE 命令之后, 但是又在任何 DEVICEHIGH 命令之前。

(5) CONFIG. SYS 文件必须包含一个关于装载到上位内存的每一个设备驱动程序的 DEVICEHIGH 命令。

(6) AUTOEXEC. BAT 文件应该包括一个在上位内存区域运行的每一个内存驻留程序的 LOADHIGH 命令。

! 在 DOS 5.0 下我有一个应用程序运行需要 DOS 4 版本, 是否需要制作一个旧的 DOS 版本的复制盘?

! 不需要旧的 DOS 版本就可以运行你的应用软件。DOS 5 和 DOS 6 都包括 SETVER 命令, 使用这个命令, 可以让大多数应用软件误认为它们是在旧的 DOS 版本上运行, SETVER 能在这些程序与 MS-DOS 5 或 MS-DOS 6 之间架起一道桥梁, 在 SETVER 的沟通下, 那些只能在

DOS 4.0 上运行的程序能在 DOS 5 或 DOS 6 上运行。在 CONFIG. SYS 中进行如下设置:

```
DEVICE = C: \ DOS \ SETVER. EXE
```

然后, 在 SETVER 命令行中加入应用软件的名称和它运行所需的 DOS 版本。如下:

```
SETVER FILENAME 4.00
```

经过这样设定之后, SETVER 就将 FILENAME 程序列入本身的版本中, 当 FILENAME 要在 DOS 5 或 DOS 6 上运行时, SETVER 就协助 FILENAME 正确回答 DOS 版本询问而顺利闯过 DOS 5 或 DOS 6 的关口。

但要注意, 设置了 SETVER 后, 必须重新启动, 才能运行设置的程序。

! 系统升级到 DOS 5 后, 在根目录下有一个名为 WINA20.386 的文件。这个文件是干什么用的? 可以从根目录下删掉这个文件或者将它移到 DOS 子目录下, 而不会影响机器的正常工作吗?

! WINA20.386 是一个 Microsoft Windows 3.0 的支持文件。如果你使用一个 Microsoft 内存管理程序, 例如 EMM386. EXE 就不需要这个文件支持来运行 Windows 3.0, 这就意味着你可以将这个文件移到你的 DOS 子目录中或干脆简单地整个删除这个文件。

! 近来在我的 386 计算机上安装 DOS 5 和 Windows 3.1, 听说它的磁盘高速缓存不能将它的内容立即写到硬盘上, 是怎么回事?

! 磁盘高速缓存是一个内存区域, 从硬盘上读出的数据就存在这个区域里, 这样当程序再

次请求这些数据时, 就可以从高速缓存中得到, 而不用进行硬盘访问了, 这样大大提高了硬盘速度, 也是 DOS 下使用扩展内存和扩充内存最好的方法。使用 SMARTDRV. SYS 磁盘高速缓存驱动程序是实现这种操作的一种方法。无论你使用的是 DOS 5 还是 DOS 6, 都可以使用下面的命令来强制立即将高速缓存的内容写到磁盘:

```
SMARTDRV /C
```

但是对 DOS 5 来说, 不需要这样做。这个版本的 SMARTDRV. SYS 总是立即将信息写到磁盘上的。

相反, DOS 6 的磁盘高速缓存程序 SMARTDRV, 当程序执行写入操作时, 数据并不立即写入磁盘, 而是先写入磁盘高速缓存, 直到高速缓存已满时, 才能将数据由高速缓存写入磁盘中。当程序要从磁盘中读取数据时, 会先到磁盘高速缓存中寻找, 若发现有数据, 就直接从磁盘高速缓存中读取, 否则从磁盘中读取。

为避免重新启动机器时丢失信息, 在重新启动之前使用 SMARTDRV /C, 以确保将磁盘高速缓存的数据都写入磁盘中(如果按 Ctrl + Alt + Del 键启动, SMARTDRV 会自动将高速缓存内的数据写入磁盘中)。

! 我的机器里装的是 DOS 5.0, 但是我必须与机器里的 DOS 版本是 2.1 的用户交换信息, 难道那些用户必须升级到 DOS 5.0 才能读由我的 DOS 版本生成的盘吗? 如不必须, 我如何在 DOS 5.0 下格式化一张软盘来使运行 DOS 2.1 的系统读取其内容?

! DOS 2.1 和以后的 DOS 版本都有处理软盘的能力, 所以, 升级是不必须的。交换软盘所产生的问题有可能和硬件相关, 如

果是新一些的 PC 机,它可能装备一个高容量驱动器,这种驱动器适合于 1.2M, 5.25" 软盘或 1.44M, 3.5 软盘。旧型号的 PC 运行 DOS 2.1 版本的,有可能装备的是一个只能读低密 360K, 5.25" 软盘或 720K, 3.5" 软盘的磁盘驱动器。


在高密驱动器中格式化一张 360K 软盘,以使低密驱动器可以读它,用 FORMAT 命令的如下格式进行:


```
FORMAT A:/F:360
```


格式化 720K 的磁盘,键入:


```
FORMAT A:/F:720
```

即使使用上面的命令或方法解决问题,也需要小心。当使用 2.1 系统格式化的软盘时,最好是从这张软盘中读取信息但是不要往里写入信息。

 DOS 5 的 HIMEM.SYS 程序是应该放在 CONFIG.SYS 文件的开头处呢,还是放在结尾处?

 应该放在 CONFIG.SYS 文件的开头处。设备驱动程序和其他程序,例如 EMM386 使用扩展内存。Microsoft 的 DOS 5 手册建议你在激活任何其他要求使用扩展或扩充 mwdhr 设备的驱动程序之前先使用 HIMEM.SYS。

 能否提供更多的关于 DOS 的堆栈命令参数的一些信息,这些参数的使用影响哪些程序等一些诸如此类的问题?

 可以加一个 STACKS 命令到 CONFIG.SYS 文件中。“STACKS”是堆栈的意思,堆栈是低位内存的一个区域,堆栈是被用于程序中断时暂时保存数据和 CPU 登记信息的数据控制区域。

STACKS 命令的语法如下:

```
STACKS=[number],[size]
```

其中 number——堆栈数目,

由 8 到 24

size——堆栈大小,由 32 到 512

例如想保留 12 个堆栈,每个堆栈 256 个字节,就必须在 CONFIG.SYS 文件中加入如下一行:

```
STACKS=12,256
```


在 IBM PC 系列,PC/XT 系列及 IBM 兼容机上,默认的堆栈大小设置是与如下命令等效的: STACKS=0,0;在其他所有的 PC 机上,默认的值是 STACKS=9, 128。

DOS 也用堆栈来监督硬件中断。硬件中断是告诉 CPU 中断它当前的操作信号,每次一个硬件中断发生时,DOS 暂时从正在运行的程序处接管控制,转到它自己的内部堆栈,然后将控制转给专门的处理中断的软件程序。当中断被处理完毕后,DOS 恢复最初的堆栈并且返回到应用程序。


如果 STACKS 设置很小,可用堆栈被用尽。当这种情况发生时,就会出现如下的一个错误信息:


```
Intended stack overflow, system halted!! (堆栈溢出,系统挂起)
```


这时可通过增加分配堆栈数目来解决这个问题,但是 DOS 操作手册上没有说明当堆栈满了并超过它们被分配的容量时,DOS 也同样会显示这条错误信息。所以如果仅增加堆栈的数目是解决不了问题的,还要试着增加每一个堆栈的大小。

 自从我的机器安装了 DOS 5 之后,外挂式 5.25" 软驱在正常格式化盘时总有问题,但是使用早期版本(如 DOS 2.11, DOS 3.0)时就没有问题。当我试图在外挂式 5.25" 软驱中格式化一张

360K 软盘时,机器总认为它是一张 720K 的软盘。如何能让 DOS 5 知道外挂软驱和内置软驱的容量是不一样的呢?

 因为你先前使用的是旧的 DOS 版本,你的外挂式软驱可能是需要一个升级的与 DOS 5 相适应的设备驱动程序。DOS 3.3 及更高的 DOS 版本,绝大多数外挂式软驱都需要一个由销售商提供的设备驱动程序,使它们与 DOS 版本兼容。建议你和外挂软驱的厂家或销售单位联系一下,询问一下有关升级驱动程序的事。

 近来,每当我试图装入一个新的应用程序时,我的计算机总是出现一连串 DOS 的错误信息:先是“Divide overflow”,紧接着是“Memory allocation error, Cannot load COMMAND”并且无设备时试图使用应用程序也会发生同样的错误,问题何在?

 这个问题不是因为 DOS 引起的,最可能的原因是由应用软件的一个错误或缺陷而引起的。“Divide overflow”错误信息常常表明一个程序试图进行非法的数学功能,例如试图用 0 去除一个数字。如果这个错误信息没有在其他程序中出现,可以排除是一个坏的内存芯片引起这个错误信息的出现的可能性。

一般出现“Divide overflow”错误时,结束程序,返回到 DOS。“Memory allocation error”错误信息提示了应用软件在释放 DOS 分配给它使用的内存时失败了。当一个程序无法释放它先前占的内存时,DOS 通常就无法装载它的命令处理程序 COMMAND.COM 的暂驻内存的部分到内存中,以继续操作。这个致命错误会使系统停止,需重新启动系统。

SKILL

反病毒升级信息表

基本描述

库类别: 0
攻击对象: 4

病毒名称: CHQ1
病毒长度: 0

第(1)基址描述

39/82/0:37/82/0

第(1)特征描述

0/2C/BB. 00. 06. CD. 13. C6. 06. 1D. 7C. 00. EA. 4D. 06. 00. 00

病毒分析:

此病毒发现于云南地区(由本网云南网员率先提供),它被命名为CHQ1。这是一种启动型的病毒,传染软盘的BOOT引导区,硬盘的主引导区。该病毒在对磁盘进行读写以及复位等操作时实施传染。当用被病毒传染的软盘或硬盘启动时,病毒首先进入系统并抢占系统高端的2K内存驻留,然后引导正常系统启动,该病毒截取磁盘读写中断INT 13H以获取控制权。CHQ1病毒无明显的触发条件和表现症状。被感染的硬盘无明显破坏,但对软盘目录区有潜在的破坏作用。

升级操作指南

为了对帝霸计算机反病毒服务网专用软件(普及版)进行升级,请运行该软件,选择DBNET主菜单中的“病毒参数库

维护”项,然后选择“追加记录”子项,用户可以对照所发布的病毒升级信息,将其录入到软件记录中。用户按TAB键,可将红色的光标移到下一项参数录入域中。注意,在除基本描述(前四项)外的录入过程中,每一大项可能含若干子项,此时,录入域的上方有红色数字指示。用户按升级信息表键入一个子项并回车后,该域的显示内容被清空并进入下一子项的录入,如不存在下一子项,再按回车键即进入下一大项的录入编辑。升级信息录入完成后,按ESC键,回答确认追加提示Y,即完成了对帝霸反病毒软件的一次自我升级。

需要提请读者注意的是:反病毒升级信息表是经过精密剖析病毒而得来的,因此为了保证参数追加正确有效,请严格按参数表录入,并请仔细检查。

帝霸计算机反病毒服务网专用软件——DBNET

帝霸计算机反病毒软件是帝霸(DEBUG)计算机反病毒服务网自行开发的专用软件,是一套自成系统的,集查找病毒、清除病毒、快速自我升级于一体的、最具实用价值的新一代开放式反病毒软件,它的全新的开放式和自升级的反病毒概念,可以让使用者参与实际反病毒的工作,有效地提高使用者自身的反病毒能力。开放式技术是当今世界计算机技术发展的潮流和方向。帝霸计算机技术研究所的技术人员经过多年对计算机病毒与反病毒技术的研究,顺应时代潮流,将开放式技术溶于反病毒技术之中,形成了一套完整的科学理论,在学术上取得了重大突破,获得了各方面专家的一致赞誉。以此理论为基础投入大量的人力物力完成了它的商品化,形成了今天的帝霸计算机反病毒软件。

该软件采用独创的数据结构方法来描述病毒,从而可以很快地分析新出现的病毒,形成一套升级信息表,用户获得这张表即可自我升级帝霸反病毒软件。

该软件由三大功能模块组成:1.全面、快速的检测功能模块;2.安全彻底的病毒消除功能模块;3.功能强大、使用方便的反病毒库管理模块。

该软件的主要特点有:1.采用C++与汇编语言混合编程,提高了程序的运行效率;2.采用类Windows的图形界

面,极大地提高了用户使用的方便性;3.勿须汉字系统支持,直接在西文状态下显示汉字图标;4.采用开放式反病毒技术概念,用数据结构的方式描述病毒,从而使用户不需要重新编程即可完成对新病毒的检测及清除的自身升级;5.使用户自身的反病毒能力与最新的反病毒技术和反病毒信息同步。

在该软件的基础上,帝霸计算机技术研究所又进一步组建了帝霸反病毒服务网,希望该软件能更好地为广大计算机用户服务,为广大用户建立和维护一个安全干净地计算机工作环境。

帝霸计算机反病毒专用软件分普及版和增强版两个型号,两者都可以消除目前国内流行的各种病毒。增强版可以通过邮寄给网员的完整的升级信息表来检测并清除以后新出现的病毒,普及版只能通过《电脑编程技巧与维护》杂志每月所刊登的部分升级信息表来升级并检测新病毒。增强版对入网会员免费提供,普及版则对所有人员均免费赠送(收取20元/套的成本费和邮寄费)。普及版可按下述任一地址邮购:

★(100006)北京市劳动人民文化宫内《电脑编程技巧与维护》杂志社,联系电话:(01)5123823,5232502,联系人:刘伟

★(100836)北京海淀定慧西里19号楼帝霸计算机技术研究所,联系电话:(01)8123896,8263441,联系人:管逸群

帝霸计算机反病毒服务网简介

帝霸计算机反病毒服务网(Debug Computer Anti-Virus Service Net 简称 DEBUG NET),是中国第一家在计算机反病毒领域内提供快速反应的专业服务网,得到了国内外有关方面,特别是《电脑编程技巧与维护》杂志的大力支持和协作,它的运营将填补我国计算机安全方面的一项空白,必将为我国计算机正常健康地运作,建立正常干净的计算机工作环境、开发环境作出贡献。

收费标准及服务内容:

一、A类:300元/年(面向单位用户)

服务内容:

1. 每年免费提供一套本网最新版本的专用软件。
2. 每月免费赠送一本《电脑编程技巧与维护》杂志。
3. 每月按时邮寄一份新病毒的完整升级信息表,网员可以对本网专用软件自行升级。
4. 网员发现病毒,本网可在24小时内做出快速反应,全面分析,形成升级信息表,并传送给网员,网员可以将其输入本网专用软件,将此病毒清除(传送可用电话或传真)。此项服务一年中4次以内免费,超过4次按30元/次收费。
5. 向网员通报特定病毒的发作及流行情况,及时提醒网员加强预防。
6. 为网员提供反病毒技术咨询、有偿修复硬盘、恢复数据等服务。

7. 不定期举办反病毒技术研讨会和学习班,学习和研讨反病毒技术的发展,介绍计算机反病毒技术,网员可获优先和优惠。

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。

2. 网员发现新病毒,本网以尽可能快的速度给网员解决问题,并将升级信息表列入邮寄的资料中,回寄给网员,解决网员的问题。

3. 每年本网发表的软件升级版本,均免费赠送一套。

4. 通报新病毒发现地点及泛滥区域,提醒用户预防。

5. 有偿提供修复硬盘,恢复数据等服务。

6. 本网举办各种技术研讨会和学习班时,可获优先。

三、专项服务:价格面议(面向系统用户)

服务内容:

按网员要求提供全方位服务。

入网方法

1. 申请入网者先填写入网申请表,并寄往下述地址:

(100836)北京海淀定慧西里19号楼帝霸计算机技术研究所

2. 入网费在寄出申请表的同时可通过邮局汇往上述地址。

3. 帝霸计算机反病毒服务网在收到申请表及入网费后即将网员证、有关票据、帝霸反病毒服务网专用软件寄给网员。

4. 网员在收到网员证后即成为正式网员,可按相应服务类别享受服务。

帝霸计算机反病毒服务网入网申请表(复制有效)

联系人	单位名称		年 龄	
所在部门	职 称	职 务		
地 址	通讯地址			
电 话	传 真	邮 编		
从事工作				
微机型号				
网员类别	<input type="checkbox"/> A类:300元/年 <input type="checkbox"/> B类:90元/年 <input type="checkbox"/> 专项服务			
入网费	汇出人民币(大写)		<input type="checkbox"/> 邮局 <input type="checkbox"/> 银行	
	汇出时间 19 年 月 日			
单位公章	年 月 日	经办人		

★帝霸计算机反病毒服务网热线咨询电话:(01)8123896 热线汉字寻呼:(01)8298866 呼1111。

《电脑编程技巧与维护》订阅单(复印有效)

地 址				邮 编	
单 位				收 件 人	
起订日期	年 月 日			共 计 期	
订阅份数		款 数		汇款方式	

《电脑编程技巧与维护》订阅单(复印有效)

地 址				邮 编	
单 位				收 件 人	
起订日期	年 月 日			共 计 期	
订阅份数		款 数		汇款方式	

诚征全国直销代理

自《电脑编程技巧与维护》月刊征订启事和各报刊杂志的消息及广告刊出之后,不少单位的图书馆、资料室、培训部门、书刊经销商以及广大热心读者来信、来电询问本刊的出版发行情况,并要求作为本刊的直销代理,有些读者、同仁甚至不辞劳苦来本刊编辑部谈直销事宜。这一方面反应出本刊“实用第一、智慧密集”的办刊原则的正确性,另一方面又反应了读者想尽早、方便地得到本刊的愿望,为此,我社决定,在全国诚征直销代理(单位或个人),其原则是:方便读者互惠互利、大家共享、共同发展。

具体办法如下:

① 包销:杂志社给予包销客户以较大优惠。② 代销:杂志社给予较好的优惠,在结清第一次代销杂志的款项后,按代销客户的要求发放下批杂志。为保证包销与代销客户的利益,本杂志社对首次代销的客户以 50% 的优惠价提供 10 份样刊试销。代销和包销的优惠率及合同如下:

包销与代销优惠率:

包销	优惠率	代销	优惠率
10~50 份	20%	10~50 份	18%
50~100 份	25%	50~100 份	22%
200 份以上	30%	200 份以上	25%

包销与代销合同(复印有效)

甲方名称					
通信地址					
邮政编码		电话		传真	
甲方代表签字			经办人		
乙方名称	《电脑编程技巧与维护》杂志社				
通信地址	北京市劳动人民文化宫内				
邮政编码	100006	电话	5232502	传真	
乙方代表签字	孙茹萍		经办人	刘伟	

经双方协商同意×××××××(甲方)愿作乙方(《电脑编程技巧与维护》杂志社)的包销或代

者,按×××份的优惠率进行结算。本合同自双方代表签字之日起生效。

订 阅 须 知:

△本刊每月八日出版,每期订价:三·八元,全年订价:四十五元六角。一九九四年出版发行六期。邮局公开发行,邮发代号二四—一〇六。
△请将本卡寄至:(一〇〇〇〇六)北京市劳动人民文化宫内《电脑编程技巧与维护》杂志社发行部。联系电话:(〇一)五一二三八二二。
△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)

为使本刊越办越好,请读者填写此卡。我们将根据寄回的征询卡,挑选100名本刊热心读者,免费赠送本刊1995年全年杂志。

读者姓名:_____年龄:_____职务或职称:_____

工作单位:_____电话:_____

通信地址:_____邮编:_____

▲您对本刊所设栏目的看法:(好:✓,一般:○,不好:×)

- | | | | |
|--------------------------------|-------------------------------|----------------------------------|---------------------------------|
| <input type="checkbox"/> 新技术追踪 | <input type="checkbox"/> 软平台 | <input type="checkbox"/> 图形图象处理 | <input type="checkbox"/> 汉字处理 |
| <input type="checkbox"/> 编程语言 | <input type="checkbox"/> 数据库 | <input type="checkbox"/> 计算机安全 | <input type="checkbox"/> 网络与通讯 |
| <input type="checkbox"/> 软件维护 | <input type="checkbox"/> 实用软件 | <input type="checkbox"/> 硬件维护 | <input type="checkbox"/> 电脑博士信箱 |
| <input type="checkbox"/> 新产品 | <input type="checkbox"/> 明星企业 | <input type="checkbox"/> 反病毒信息公告 | |

▲您认为本期各栏目的质量如何(好:✓,一般:○,不好:×)

- | | | | |
|--------------------------------|---------------------------------|----------------------------------|--------------------------------|
| <input type="checkbox"/> 新技术追踪 | <input type="checkbox"/> 软平台 | <input type="checkbox"/> 图形图象处理 | <input type="checkbox"/> 汉字处理 |
| <input type="checkbox"/> 编程语言 | <input type="checkbox"/> 数据库 | <input type="checkbox"/> 计算机安全 | <input type="checkbox"/> 网络与通讯 |
| <input type="checkbox"/> 软件维护 | <input type="checkbox"/> 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)

凡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_____