

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

第五

电脑编程技巧与维护

COMPUTER PROGRAMMING SKILLS & MAINTENANCE

1994 年 11 月



Microsoft
WIN32

SOFTWARE DEVELOPMENT KIT

Tools for Writing 32-Bit Applications for Windows.

Microsoft®



HOPECAD

希望公司为美国DC公司指定国内总代理

脱颖而出

- 高质量绘图
HOPECAD绘图机采用独一无二Look Forward技术, 确保绘图, 绘弧线连续, 平滑, 无顿笔现象。每支笔都可设不同之压力, 笔速则连续可调, 从而保证绘出之图纸达到最佳质量效果。
- 高精度绘图
机械定位精度严格控制, 在0.1mm之内。可寻址分辨率, 为0.025mm, 硬件分辨率为0.0125mm。A3幅面内之真实精度超过市售的任何一种同类产品。
- 高速度绘图
最高绘图速度为560mm/s, 绘制弧线不停顿, 特别是大尺寸的圆或弧, 更显示其高速度优势。

- 超级辅助功能
8支笔, 每支笔均可单独设置笔速和笔压, 笔速, 笔压值有数码显示器进行显示; 笔所在位置坐标亦由显示器窗口实行显示。3秒钟之后, 笔不移动会自动抬起。60秒钟无动作, 笔会自动返回笔库重套上笔帽。HOPECAD绘图机采用先进笔软着落功能, 能维护笔尖的不受损坏并降低噪声。
- 兼容性及其可靠性
绘图命令DP-GL与HP-GL完全兼容, 并提供标准Centronics并行接口和RS-232C串行接口。串行接口支持目前所用到的全部握手协议。并提供软件查询功能, 便于用户自定义软件握手规定。适应大范围内幕, 速度变化的静电吸附方式, 使放纸十分方便。



DP-1801绘图仪
北京希望(集团)公司

北京希望电脑公司
北京前门大街82号 邮编: 100080
电话: 2561421 2564163 2567818
广州天河体育西路有第三层2号 邮编: 510620
电话: 7505151 7505152

上海希望电脑公司
上海南京路418号 邮编: 200052 电话: 2529929 2120632
南京希望电脑技术公司
南京市中山南路105号 邮编: 210005 电话: 4403812 4512474
成都希望电脑公司
成都新南路西村6-10号 邮编: 610015
电话: 5556333 5589787

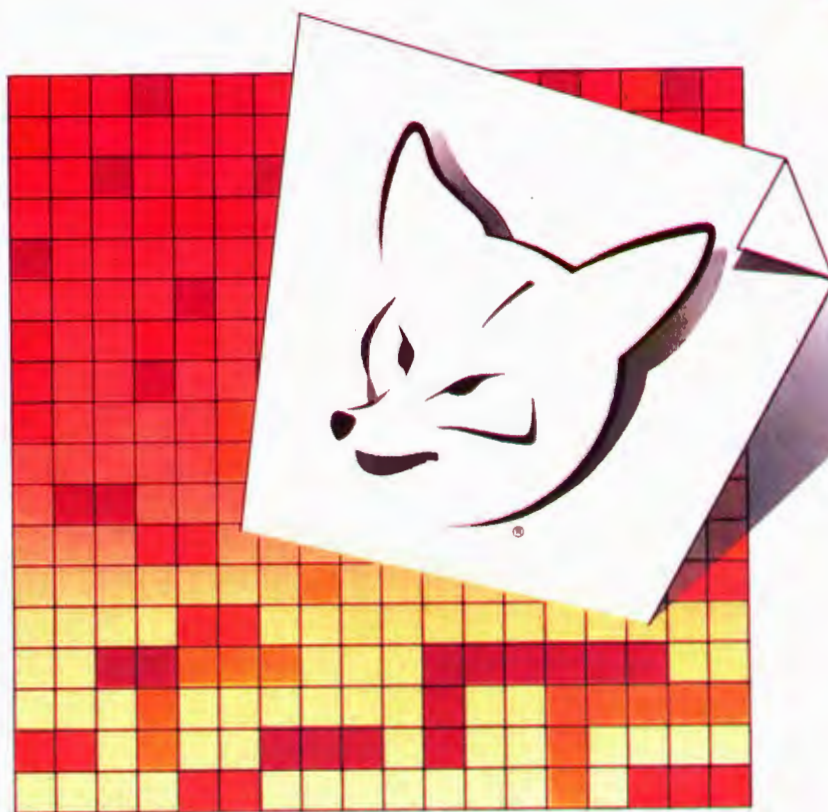


您的希望

新品
采用一台绘图仪
打印机的价格



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



Microsoft FOXPRO

LIBRARY CONSTRUCTION KIT

For MS-DOS and Windows™

日本最大的绘图仪生产厂商

MUTOH XP-500系列绘图仪

- 纸张 A120-720只 • 水平裁纸及
 存纸柜(4XP-510R有) • 自动定
 归位和上帽 • 重绘功能 • 搜糊
 覆板高质量终图 • 多重挂口设
 置 • 比例尺 • 笔排序 • 自检 •
 原点 • 校淮 • 补偿 • 镜象

MUTOH iP-220绘图仪

- 窗口 ● 偏移 ● 数字化 ● 绘图区
度高达1273mm/秒
● 绘图：支持多种类型彩色笔及多种笔触绘图。
● 彩绘：允许四种不同的彩笔绘制。
● 刻图：具有灵活的数字刻图功能。

希望电脑(集团)公司授权武藤产品总代理

诚征代理



北京希望电脑(集团)公司

地址: 北京海淀区路82号 电话: 2564163 2567387 2561421 2567818 邮编: 100080

● 上海希望公司：(021)2521656 ● 成都希望公司：(028)5589787 ● 广州希望公司：(020)7505151 ● 南京希望公司：(025)4403812

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,您立刻会成百倍地提高编程效率

《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,会成百倍地提高编程效率。《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,是全部的所见即所得。

二、开发速度,取决于打字速度:

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

三、界面友好,菜单华丽:

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

四、图文并管,功能超凡:

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

五、报表设计,复杂多样:

“专用报表”、“汇总表”、“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,高度精炼出近 30 个功能模块,几乎使所有的,信息管理系统开发所要求的,各种各样的复杂需求包容其中。

七、“智能化”技术,独树一帜:

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

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

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

全国各地代理经销商名单(同时办理当地代理赠送业务)

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

单位名称	电话
湖北省武汉市皇龙软件销售中心	7804039
江苏淮阴吴通经济技术发展公司	341264
江苏淮阴市清浦电子技术公司	335053
江苏南京电子信息产业集团公司	3342145
江西南昌亚特南达电子电脑有限公司	211926
辽宁本溪市通用计算机发展公司	246277
辽宁朝阳市比塔区银丰科技公司	219720
辽宁锦州华侨电脑科技发展有限公司	236693
内蒙古包头市理想电脑公司	556655-188
山东省泰安市云海科技实业公司	335040
山西大同计算机有限公司	242411
山西太原钢联新技术产业公司	6042720
陕西省汉中市电脑软件公司	219324
陕西西安市新航电子技术公司	4253439
上海海贝船舶技术研究所	4860037
上海两英家文经营部	2583576
上海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
新疆乌鲁木齐电子技术开发公司	263078
云南昆明工学院科技开发部	5146647
浙江杭州方欣公司软件天地	8085512
浙江杭州华东电子技术经营部	5160198
浙江杭州星邦信息处理电脑有限公司	5177185
浙江金华联想计算机通信工程公司	327475
浙江省杭州计算技术研究所	5151941

特别注意
免费赠送

无与伦比的超凡功能
令世人瞩目!!!

网络版/单用户版,统一单价:1280元(V3.0) 980元(V2.8)

特 别 注 意

全国各地(包括港、澳、台地区)经销单位,各大院校及科研机构,可以凭介绍信办理免费赠送《雅奇 MIS》手续,每单位限领一套,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地就近获得全套《雅奇 MIS》赠送软件,望各地具备条件的经销单位,速与北京公司联系,申请在当地代办《雅奇 MIS》赠送业务。

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. 信誓卡一张
7. 精美包装盒一只

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

实用第一,智慧密集

电脑编程技巧与维护月刊

1994 年第 5 期

(总第 5 期)

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 易言 艾蒙 田艾

肖贻 苏古 阮薇

温达 管逸群 业成

公关部主任 吕莉 伊梦

出版部主任 祁连顺

发行部主任 孙茹萍

编辑出版 《电脑编程技巧与维护》杂志社

地址 北京市劳动人民文化宫内

邮编 100006

编辑部电话 5123823

公关部电话 5232502

照排 《电脑编程技巧与维护》杂志社
电脑排版部

印刷 北京市地质矿产局印刷厂

订阅处 全国各地邮电局

国内总发行 山东省潍坊市邮电局

邮发代号 24-106

刊号 ISSN 1006-4052
CN24-3411/TP

广告许可证 京东工商广字 0384

每期定价 3.80 元

全年定价 45.6 元

出版日期 1994 年 11 月 8 日

新技术追踪

Microsoft 推出 Windows NT3.5 等 23 篇..... (4)

软平台

多文档界面菜单加速棒的实现方法与技巧 (9)

给 DOS 增加两个外部命令——PEEK 与 POKE (14)

优化配置文件,提高微机利用率 (15)

图形图象处理

如何用 C 语言开发 TVGA256 色图形..... (18)

给程序加个动画封面 (22)

汉字处理

基于图形环境下图形及汉字立体投影通用菜单的实现
..... (23)

直接在西文 DOS 下输入汉字的方法 (26)

西文环境下汉字放大显示技术的实现 (28)

西文状态下汉字的多种彩色立体显示 (31)

一个能打印出“字中字”的程序 (36)

编程语言

利用 VB 特别功能构造动态界面 (37)

利用 QueryDef 提高资料的快速存取 (41)

Turbo Pascal 与汇编语言混合编程技术 (47)

scanf()函数的缺陷及解决措施..... (52)

数据库

用 _Config 变量方便地跟踪每个用户的配置情况 ... (53)

搜寻复杂表达式 (56)

COMPUTER PROGRAMMING SKILLS & MAINTENANCE

开发 FoxPro 数据仓库	(58)
FoxPro 关系数据库的应用点滴	(61)
FoxPro 2.5 的弹出菜单中多选项的选择	(64)

网络与通讯

PC 机 RS-232 通信连线汇集 (66)

软件维护

微机软故障的分析与处理 (68)

Xenix 系统故障检修几例 (70)

实用软件

利用 AutoCAD 的形体文件在 Turbo C 中写字符 (71)

利用 INTL Toolit 使你的应用国际化 (73)

硬件维护

多频显示器的特点及维修(二) (75)

没有电路图时修复微机的几点经验 (78)

TH3070 打印机常见故障的维修 (79)

电脑博士信箱

DOS 与 Windows (81)

反病毒信息公告

反病毒升级信息表 (84)

编读往来







本刊第二届“热心读者”获奖名单 (86)

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都很强的技术杂志,它的主要栏目有:

软平台,该栏目主要介绍操作系统的新功能、新特性以及利用它们进行开发应用的技术技巧。**图形图象处理**,该栏目主要介绍图形图象软件的编制技术、技巧和编程的具体方法。**汉字处理**,该栏目主要介绍汉字处理的编程实用技术。**编程语言**,该栏目主要介绍编程语言的编程技术技巧及它们所提供的函数及其灵活应用技术。**数据库**,该栏目主要介绍数据库系统的新功能、新用法,以及数据库的编程技术、技巧。**计算机安全**,该栏目主要介绍加密技术、反跟踪技术及反病毒技术。**网络与通讯**,该栏目主要介绍网络操作系统和在网络环境下应用开发的技术、技巧。**软件维护**,该栏目主要介绍软件功能的用户扩充及软件故障的合理恢复方法。**实用软件**,该栏目主要介绍各种工具软件的应用技术与技巧。**硬件维护**,该栏目主要介绍计算机硬件的维护与维修经验、心得体会。







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

系统的升级换代，请立即下载
 帝霸计算机反病毒服务网

欢迎加入
 帝霸计算机反病毒服务网

热线咨询电话：(01)6422194

新技术追踪

Microsoft 推出 Windows NT 3.5

Microsoft 为了加强公司在企业领域计算 (Enterprisewide Computing) 内的地位,最近推出了 Windows NT 3.5,并将随之推出相关的服务器软件。Microsoft Windows NT 3.5 具有两种版本,即基于 Intel、MIPS 和 DEC Alpha 平台的工作站版本和服务器版本。基于 PowerPC 的版本将在晚些时候推出。随着代号为 Daytona 产品的推出,产品将提供 32 位到 16 位 OLE 2.0 的互操作性 (Interoperability)、长文件名的功能,支持 Silicon Graphics OpenGL 三维库、快速 IPX 栈和新的管理工具。但许多用户等待 Microsoft Windows NT 3.5 并不是由于它提供的新功能,而是该产品的稳定性。

Microsoft 力主开发者向 32 位开发应用转化

随着 Microsoft 公司推出 Windows NT 3.5 和 Visual C++ 的 32 位开发工具,Microsoft 公司力争开发者向 32 位应用开发转化,但开发者对此的反应并不象 Microsoft 公司预计的那样迅速。新版本的 Visual C++ 2.0 运行于基于 Intel 平台的 Windows NT 3.5 和即将推出的 Windows 95 环境中,并在这些软平台上开发应用。允许开发者为 Macintosh 和基于 MIPS 以及基于 DEC 公司 Alpha 平台的 Windows NT 开发应用的附加产品将于今年年底推出。Visual C++ 包括两种版本即 16 位版本和 32 位版本。但 16 位版本是老版本它不包括 Visual C++ 2.0 的新功能,同时 Microsoft 也不再打算改进 Visual C++ 的 16 位版本。不过,由于开发者在 16 位平台上开发应用的惯性,从 16 位的应用开发过渡到 32 位的应用开发还需要一定的时间。

NexGen 向 Intel 挑战

随着 NexGen 公司的 Nx586 芯片的面市,Intel Pentium 芯片独霸微处理器市场的局面即将结束。将面市的 P100、P90、P80 和 P75 Nx586 芯片和 Pentium 的性能不相上下,甚至更强。94MHz 的 P90、75

MHz 的 P80 和 70MHz 和 P75 分别对应于 Intel 的 100MHz、90MHz、66MHz 和 60MHz 的 Pentium 芯片。

NexGen 芯片的独特之处主要表现在 Nx586 内置协处理器、Level2 的高速缓存可使芯片性能得以提高；Nx586 芯片使用仿 RISC 微结构 (Microarchitecture) 采用标准的 X86 指令，并可将其解码成 NexGen 的 RISC86 指令；Nx586 提供可选的浮点单元；Nx586 在较低的时钟频率上能提供较高的性能。该芯片符合 RISC 原理如定长指令 (fixed instruction Length)，大容量存储器组 (Large register sets)，并实现了顺序外执行 (Out-of-order execution) 等高新技术。NexGen 芯片已得到一些厂商的支持，预计采用 Nx586 芯片的系统将于 1995 年第一季度上市。NexGen 芯片的售价将在 404 美元 (P75) 到 777 美元 (P100) 之间。

AMD 推出 DX4 级处理器

AMD (Advanced Micro Devices Inc.) 将推出一款 33/100MHz DX4 级处理器，这种处理器具有 8KB 的 WT (Write-through) 的高速缓冲存储器，Intel 公司的 DX4 具有 16KB 的同类高速缓冲存储器。但 AMD 的 DX4 产品的价格要比 Intel 的 DX4 产品便宜。AMD 公司正计划推出具有 16KB 高速缓冲存储器的 33/100MHz 的处理器产品。

IBM 将推迟 PowerPC 的宣布

IBM 公司宣布他将于 1995 年上半年推出 PowerPC，并且该系列产品将称为 PowerSeries。据称这一系列的产品包括：PowerSeries 400 台式系统、PowerSeries 600 人体工程学台式系统、PowerSeries 800 ThinkPads 系统。

SUN 宣布 UltraSparc 芯片

SUN 公司的 Sparc Technology Business 宣布了 64 位 UltraSparc 芯片，该芯片具有 200 到 400 SPECint92 的整数性能 (integer performance) 和 250 到 500 SPECfp92 的浮点性能 (floating-point performance)。UltraSparc 芯片具有内置的多媒体功能，

包括：支持二维和三维图形、图象处理，以及每秒 30 帧的 MPEG-2 视频压缩和解压缩功能。该产品的样品将于 1995 年第一季度推出。

Oracle 推出 NT 和 NetWare 环境下的 Workgroup Server

据悉 Oracle 公司将推出 Microsoft Windows NT、OS/2 和 Novell NetWare 环境下的 Oracle Workgroup Server，并将推出 Windows 环境下的 Oracle Objects 和 Oracle7。Oracle Workgroup Server 作为 Oracle7 的简版 (downsized) 于今年三月份宣布，并作为一个软件开发者工具箱提供。Windows 环境下的 Oracle7 是 Oracle 数据库开发者的台式版本。Oracle Objects 是基于 OLE 的连接软件。

反病毒产品将由硬向软转化

本刊记者从第九届计算机安全会议上获悉，计算机反病毒产品的总体市场正逐渐走下坡路。计算机反病毒产品的市场由两大产品所包揽：反或防病毒卡和反病毒软件，目前我国反病毒产品的市场基本上被反或防病毒卡所占领，但由于计算机用户安全意识的增强、反或防病毒卡本身存在的问题、计算机应用软件的大力发展以及计算机病毒自身技术的发展，使得计算机反或防病毒卡所暴露的弱点越来越多，如升级困难、和用户应用软件冲突等等。这样随着计算机用户对于计算机病毒了解的逐渐增多，反或防病毒卡的宣传性词汇已不再为用户衡量该卡的可信标准，同时，由于用户安全意识的提高购买卡的意识减弱。但另一方面被冷落了数年的计算机反病毒软件产品的市场正在抬头。有趋势表明，未来的反病毒市场将是反病毒软件的天下。

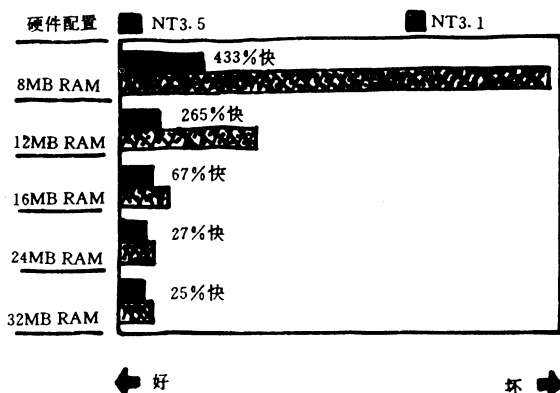
Microsoft 宣布服务器产品的价格策略

据悉，为了迎合市场的需要 Microsoft 宣布了对其 Windows NT Server，基于 Windows NT 的 BackOffice，SNA Server 2.1，Systems Management

Server 1.0(代号为 Hermes),Mail Server 3.2 和 SQL Server 4.21a 等服务器软件的价格计划。在每个产品初始价格的基础上每增加一个节点要有一定的附加费用。今年 10 月份 Microsoft 销售其 Windows NT 3.5,其价格为每个服务器 699 美元,每个 NT 工作站 39.95 美元。SNA Server 2.1 软件的价格为每个服务器 409 美元,每个客户机 64.95 美元。SQL Server 4.21a 软件的价格为每个服务器 1499 美元,每个客户机 189 美元。Systems Management Server 1.0 将于本月宣布,价格为每个服务器 649 美元,每个客户机 54.95 美元。Microsoft 宣布上述产品的价格策略是针对其竞争对手 Novell 的,Novell 的 NetWare 3.12 五个节点的价格为 1095 美元。

Microsoft Windows NT 工作站的速度比较

根据 Microsoft 提供的报告,Microsoft Windows NT Workstation 3.5 和 Microsoft NT 3.1 由于系统配置的不同,其运行速度有很大的不同,尤其是系统硬件档次越低这种表现就越明显,如图所示:



(资料来源 Microsoft)

Microsoft Windows NT 升级的目的在于高档用户

Microsoft 称 Windows NT Workstation 的主要目标市场包括两大部分:需要特殊类型应用的用户和需要高级特性如安全和不间断运行(non-stop operation)的固定用户。随着 Microsoft Windows

NT 3.5 的推出 Microsoft 将推出 Word 和 Excel 的适应于 Windows NT 3.5 的新版本。同时第三方厂商也推出了一些 Windows NT 3.5 环境下的应用系统,如 Intergraph 公司推出了一个三维 CAD 应用软件 Design Review, Seybold System 推出了销售信息系统,Autodesk 公司推出了 AutoCAD Release13。

QEMM 内存管理工具升级

最近推出的 QEMM7.5 内存管理工具包括适用于 Windows 和 DOS 的代码,并且包括 Windows 的安装和配置模块。QEMM7.5 可以检测用户运行的环境,并根据这些环境装入 Windows 或 DOS 模块。功能增强后的 QEMM 可将设备驱动程序装入内存高端,也能将基于 ROM 的代码装入高端内存,从而能为传统内存节省 200KB。QEMM7.5 引入了 QuickBoot 功能,该功能可使系统快速启动,不需要整个系统的初始化操作。

即插即用得到了硬件生产厂商的大力支持

即插即用(Plug and Play)在得到附加卡(Add-in Card)生产厂商的广泛支持的同时,也得到了板(board)级 PC 生产厂商的支持。Intel 公司的即插即用规范为用户在系统中增加各种板卡提供了极大的方便。典型的即插即用系统有即插即用 BIOS、Intel 公司的结构管理系统(Configuration Manager)和 ISA 结构实用程序(ISA Configuration Utility)。增加 Intel 的即插即用软件可使系统快速地升级到完全的即插即用状态。

Computer 公司正在销售可适应于其采用 Pentium 处理器的 OptiPlex PC 的部件。Dell 不久将销售基于 Windows 的实用程序 Dell Inspector,这种软件将使得系统的设置如 IRQ(系统中断)、DMA(直接存储器读写)更加容易。Dell 也正在努力将即插即用的概念引入其 PC 机的 Dimension 系列中。Dell 公司称在过去三年中生产的 PC 机都可以升级成即插即用的系统,这只需将其 BIOS 升级成即插即用的 BIOS。

AST 公司在其 Bravo MS 系列中提供了即插即用功能。Compaq、DEC、Gateway 2000 等公司销售

的系统都具有即插即用功能。为了迎合技术的发展许多附加卡生产厂商热衷于即插即用标准。AZtech 公司是一个大型声音卡的生产厂商,其现在提供的 Explorer 声音卡就有即插即用的能力。生产 Sound Blaster 声音卡的公司 Creative Labs 期望在今年秋季的 Comdex 展览会上推出具有即插即用功能的产品。Future Domain 推出的新的 SCSI-2 卡 18PNP300 也提供了即插即用功能。在网络产品方面,Intel 和 SMC 推出了具有即插即用功能的 ISA Ethernet 适配器。

能否升级成即插即用部件的一个问题是传统的 ISA 板,它没有附加的电路使得它可以成为即插即用的部件,而 PCI 产品则不存在这些问题。

Cisco 瞄准远程办公通信市场

Cisco System 公司最近推出了一系列远程网络通信产品。这些产品可以帮助网络管理员把他们的通讯装置连接到企业内部的网络上去。在此之前,Cisco 已有部分产品面市,但新的产品体积更小巧、更轻便。其中的 LAN Extenders 可以让用户把 Ethernet 和 WAN 网上的装置连接到低档和高档 Cisco 路由器上去。LAN Extenders 支持 LAN 网络协议,其中包括 Novell 的 IPX。对于那些不需要全天候联机的用户,Cisco 还提供了 Access Server,它可以支持 8 或 16 个 288.8 kbps 的异步通讯口以对网络数据进行存取。Cisco 的 AccessPro PC 卡是可以插入具有 ISA 或 EISA 总线 PC 机的远程路由器,该产品可以支持一个 Ethernet(或 Token Ring)和 2 个 WAN 网的连接。Cisco 目前正致力于开发能支持 8 或 16 个 Token Ring 的集线器(hub)。

增强 PC 机三维图象处理能力的产品性能有所提高

Kubta 和 Matrox 公司分别完善了自己的 3D 图象卡,增强了原有图象卡的处理能力。Kubta 公司推出的 Action Graphics 300 售价 2195 美元,但其性能和售价 20000 美元的图象工作站一样,这在加速机械 CAD 应用等方面具有重要意义。Action Graphics 支持 OpenGL、3DR 及 Hoops 3D API,能提高 AutoCAD、AutoCAD Designer、AutoSurf 和 Microsta-

tion 的速度。Action Graphics 适应于具有 VGA、Pentium 芯片、PIC 总线的系统。而 Matrox 则将其 3D SuperPack CD-ROM 和 MGA Impression Plus 3-D 图象加速卡集成,并提供了新的驱动程序。

Microsoft 公司对 Windows 95 寄予厚望

根据 Microsoft 公司的预计,Windows 95(原名 Chicago)在第一年的销售中将能卖出 2 千万到 3 千万套,甚至达到 4 千万套的销售量。Microsoft 认为,在第一年将有 7 百万到 1 千万套 Windows 95 用于原有 Windows 产品的升级,其余的主要以 OEM 方式销售。预计明年 PC 机的世界销售量将达到 4 百万台/套,而其中的 3 百万套将预装 Windows 95。

微机价格战对用户影响甚微

由 Intel 公司率先宣布的微机 CPU 芯片大幅度降价引发了新一轮微机价格战,据报道,这种价格战对用户的影响甚微,并不会因而影响用户的购买。

世界知名的计算机厂商 Compaq、IBM、Dell 及 AST 等都将大幅度削减其产品的价格。用户认为频繁的削价并不会使他们在购买时间上拖后,美国一家公司的技术顾问表示:降价并不会影响他们购机的类型,因为购买微机是工作的需要。用户不会等待这种降价,因为用户的工作不能等待。事实上,由于各种外设及插卡的出现,整体来说微机价格在上涨。

Microsoft 寻求使 OLE 开发更容易的途径

Microsoft 公司和 Symantec 公司联合宣布了一种新的转换工具,并加入到了 Microsoft 基础类(MFC)之中。这个转换工具可以使 16 位的 C 语言程序更好、更便利地向 16 位和 32 位的 C++ 语言程序转换。MFC 的这个应用也可以在多种平台上使用,如 UNIX 和 Macintosh 及第三方厂商的支持平台。

许多开发人员表示 MFC 的这个转换工具很有用途,能使 OLE 开发非常容易。Microsoft 希望对所

有层次的开发人员都能容易地使用 OLE,在不久将发表的 Visual BASIC 4.0 和 Visual C++2.0 中会增强这方面的功能。

Acer 推出第二代 Pentium 系统

Acer 美国公司最近推出了第二代 Pentium 系统,这种系统价格低、硬盘容量大并具有集成多媒体子系统。该公司正销售六种台式机和立式机,售价在 1199 美元到 2500 美元之间。Acer 公司推出的 Acer Power 小型立式机具有 90MHz Pentium 芯片、8MB 内存、810MB 硬盘、32 位 Cirrus GUI 加速芯片、1MB DRAM 和 PCI 总线,这一系统还具有 14.4Kbps FAX/MODEM、电话应答机功能、两速 CD-ROM 驱动器、扬声器和 16 位 Sound Blaster 兼容声音卡。

HP 和 Cisco 描画集线器和路由器发展蓝图

HP 和 Cisco 公司最近宣称他们将推出具有巨大功能的新产品和管理工具,分析人士认为用户可以在大批产品中灵活地挑选以适应自己的网络系统。这两家公司主要发展 100Mbps 的 100VG—AngLAN 网卡,它将适用于 Ethernet 和 Token Ring 网络。HP 公司在 100VG—AngLAN 方面具有权威的标准。

一位 HP 发言人称,合作开发计划将于 1995 年年中开始,计划重点在适用于 Cisco 路由器的 100VG—AngLAN 网卡上,它同样也适用于 HP 公司先进的集线器。

国内三大企业联盟共同发展教育电子事业

据悉,北京太极计算机公司、中国长城计算机集团公司和联想计算机集团公司共同发起、组建“教育软件联盟”,并即将成立专门从事教育软件开发、生产、销售一条龙的公司。这是我国教育电子事业发展

中的一件大事。这个专业公司的主要发展方向是:计算机辅助教育软件、家庭学习辅导软件、职业培训软件、多媒体教育类板卡、校园网络的构建。同时,在国家有关部门支持下,参与研究和制定我国教育软件的各项技术规范和标准,并将为促进和推动我国软件产业的发展作出贡献。

农行将在全国范围内建成计算机网络系统

本刊记者获悉,不久前召开的中国农业银行全国大、中城市支行计算机网络建设工作会议提出了大、中城市支行计算机网络化建设的目标和任务。力争用八年(1993~2000 年)的时间,抓好直辖市、计划单列市、省会城市和一些经济发达地区的 40 个重点网络建设工程,努力形成全国性的计算机联网,使农业银行电子化建设逐步实现网络化。

总体目标将分三个阶段来完成:第一个阶段为开发试点阶段,用两年的时间;第二个阶段为网络建设的推广应用阶段,计划用三年的时间;第三个阶段为全面推进网络建设阶段,计划再用三年时间。

龙马公司喜获大笔订单

本刊记者获悉,日前,中国人民保险公司同北京龙马公司签订合同,认购龙马公司自主知识产权的 UNIX 操作系统下字处理软件 XWORD 2000 套。

龙马公司是北京海淀新技术开发区内专门从事 UNIX 及网络系统集成解决方案的供货商,拥有雄厚的技术开发力量。公司成立至今,已成功地承接并完成了一系列大型软件工程,龙马公司为适应国内广大 UNIX 用户的需求,积极参与三金工程及金融办公自动化的建设自主开发了 UNIX/Xenix 编辑排版打印系统 XWORD,软汉字中文包 LMCP,远程无人值守通信系统 XYcomm 等系列软件产品,取得了良好的社会效益和经济效益。其中 XWORD 可以帮助 UNIX 用户象 DOS 用户使用 WPS 那样方便地编辑、排版汉字文件,深受广大国内用户的青睐。此次中国人民保险公司本着尊重知识产权、扶植民族软件产业的宗旨认购龙马公司产品,此项举措必将极大地鼓舞国内的软件开发商,并对国内软件市场起到正确的引导作用。

SKILL

多文档界面菜单加速棒的实现方法与技巧

□赵艳利

W

indows 操作系统的多文档界面 (MDI) 是一个标准的用户界面,它可以在一个 Windows 应用程序中显示和加工多个文本。每个多文档界面应用程序有一个主窗口,在这里用户可以打开并处理多个文本,每个文本显示在主窗口的一个子窗口中。由于每个子窗口都有一个边框、系统菜单、最大化和最小化按钮及一个图标,所以,用户可以象普通独立窗口那样使用它们。

传统的 Windows 应用程序使用 C 语言编写,这导致了典型的 Windows 程序在其主窗口回调函数中一段庞大的开关语句,使程序的可读性降低,程序难以维护和调试。Borland C++ 采用了一种新的机制称为对象窗口库 (Object Windows Library) —— 简称 OWL,使 C++ 程序设计人员摆脱了困境。OWL 为用户编写大型复杂 Windows 应用软件提供了极大的方便。众所周知,OWL 对应用程序代码隐藏了许多底层细节,而 OWL 本身所封装的功能对要求较高的用户来说,已经不能满足需要。为了扩充功能,用户必须花费很大精力去熟悉 OWL 机制,这也给用户带来了一定困难。本文将介绍一些有关这方面的经验与技巧。

大凡接触或使用过 Borland C++ 3.1 IDE for Windows 的用户,都会被 Borland 风格界面控制所吸引。Borland 风格控制依靠 BWCC (Borland Windows Custom Controls) 来实现,三维造型的控制框、雕刻文字的控制标签、雕刻式的窗口背景以及定制的字体的,使 Windows 程序的界面效果大为改

观。本文将详细介绍用 OWL 开发多文档界面文件菜单加速棒的实现方法与技巧。

1. 菜单加速棒

在 Borland C++ 3.1 IDE for Windows 界面的菜单下面提供了菜单加速棒 (SpeedBar),该菜单加速棒是一些小的按钮控制,用户可以使用按钮替代相应菜单命令,从而方便用户操作。同时,用户可以改变按钮的布置方式或关闭加速棒。目前,众多的商业软件都提供了此种手段。

2. 实现策略

(1) 加速棒的布置

菜单加速棒在确定布置方式后,始终在主窗口菜单下面的固定位置,主窗口在移动或改变大小时加速棒在主窗口的位置应保持不变。同时,加速棒的位置不能影响子窗口,即不能占据用户窗口空间,这就要在框架窗口构造函数 SetupWindow() 中创建用户窗口时为加速棒留出空间,同时,应用程序要响应 WM_SIZE 消息,在主框架窗口改变大小或移动后改变加速棒的位置,使其在主窗口的位置保持不变。

SetupWindow() 函数如下:

```
{
HMENU FrameMenu;
RECT R;
InitClientWindow();
RemoveClient();
FrameMenu = GetMenu(HWindow);
ClientWnd->ClientAttr->hWindowMenu = GetSubMenu(
    FrameMenu, ChildMenuPos);
GetClientRect(HWindow, &R);
if (ClientWnd->Attr.X == CW_USEDEFAULT)
```

```
{ ClientWnd->Attr. X=R. left;  
ClientWnd->Attr. Y=R. top+MenuBarHeight;//为加速棒留空间  
}  
if(ClientWnd->Attr. W==CW_USEDEFAULT)  
{ClientWnd->Attr. W=R. right-R. left;  
ClientWnd->Attr. H=R. bottom-R. top-MenuBarHeight;  
//为加速棒留出空间  
}  
... ..  
}
```

通过消息响应函数响应 WM_SIZE 消息来调整用户窗口:

```
void TMainWindow::WMSize(TMessage& Msg)  
{int ClientWidth;  
int ClientHeight;  
int xClient;  
int yClient;  
int FrameWidth;  
int FrameHeight;  
FrameWidth=SIZEMSGW(Msg);  
FrameHeight=SIZEMSGH(Msg);  
xClient=0;  
yClient=MenuBarHeight;  
ClientWidth=FrameWidth;  
ClientHeight=Frame-MenuBarHeight;  
MoveWindow(ClientWnd->HWindow,  
xClient,  
yClient,  
ClientWidth,  
ClientHeight,  
TRUE); //移动用户窗口为加速棒留出空间  
InvalidateRect(HWindow,&menuBarRectangle,0);  
}
```

(2) 实现方法

菜单加速棒采用无模式对话框实现。无模式对话框与模式对话框不同,模式对话框暂时使父窗口无效,要求用户在控制权返回给父窗口之前完成所请求的动作。而无模式对话框不使父窗口无效,所以在无模式对话框显示的同时,用户可继续在父窗口工作。由于加速棒一直处在显示状态,为不影响编辑窗口的工作,因此,加速棒只能采用无模式对话框来实现。

(3) 消息传递

多文档界面编程困难之处就在于窗口之间的消息传递。不同窗口控制不同的消息,对于标准文件菜单加速棒,消息的传递至关重要。Open 消息 CM_FILEOPEN 应传给框架窗口, Cut 消息 CM_EDITCUT 应传给文件子窗口。由于多文档界面同时操作几个文档窗口,当用户选择菜单加速棒时,文档窗口失去输入焦点,当加速棒所传递的消息传到当前活动窗口时,必须先使该窗口获得输入焦点。

(4) 自画式控制

菜单加速棒采用 Borland 风格的自画式控制, Borland 风格的资源在传统 Windows 资源上所做的最直观的改进之一就是自画式控制的应用。Resource Workshop 中使用的所有控制都属于自画式类型,在 BWCC 库的协助下创建自画式控制是很容易的。加速棒控制采用从类 TControl 中派生的自画式控制类 TOwnerDrawButton 作为用户控制的基类,该类的构造函数将取得三个指向串的指针,这些串分别代表以常态、按下和聚焦状态显示控制的位图资源名称。加速棒按钮没有聚焦状态,聚焦位图用常态位图代替。

```
class TOwnerDrawButton : public TControl  
{  
    HBITMAP hNormal,hPressed ,hFocused;  
    void DrawBitmap(DRAWITEMSTRUCT far&,HBITMAP);  
    public :  
    TOwnerDrawButton(PTWindowsObject,int,  
        char *,char *,char *);  
    ~TOwnerDrawButton();  
    ... ..  
    virtual void DrawNormal(DRAWITEMSTRUCT far&); //绘常态位图  
    virtual void DrawPressed(DRAWITEMSTRUCT far&); //绘按下位图  
    virtual void DrawFocused(DRAWITEMSTRUCT far&); //绘聚焦位图  
};  
TOwnerDrawButton:: TOwnerDrawButton ( PTWindowsObject  
    AParent,  
    int ResourceId,char * bm1,char * bm2,char * bm3)  
: TControl(AParent,ResourceId)  
{  
    hNormal=LoadBitmap(GetApplication()->hInstance,bm1);  
    hPressed=LoadBitmap(GetApplication()->hInstance,bm2);  
    hFocused=LoadBitmap(GetApplication()->hInstance,bm3);  
}  
... ..
```

(5) 资源的创建

加速棒按钮由两幅位图构成,常态位图和聚焦位图,所有资源在 Workshop 中创建,加速棒对话框以子窗口、无边框、相对坐标模式创建。

(6) 例子程序

该程序在 Borland C++ 3.1 环境下,采用大模式,动态连接方式编译通过,在工程文件中应包含 BWCC.LIB 库文件,资源文件只给出菜单部分,略去位图和对话框部分。

源程序清单如下:

```
#include <owl.h>  
#include <filewnd.h>  
#include <string.h>  
#include <bchkbx.h>  
#include <listbox.h>  
#include <edit.h>  
#include <commdlg.h>  
#include <button.h>
```



```
#include <bstatic.h>
#include <filewnd.h>
#include <strstrea.h>
#include <mdi.h>
#include <bwcc.h>
#define ID__HELPBUTTON 100
#define ID__OPENBUTTON 101
#define ID__SAVEBUTTON 102
#define ID__FINDBUTTON 103
#define ID__SEARCHBUTTON 104
#define ID__CUTBUTTON 105
#define ID__COPYBUTTON 106
#define ID__PASTEBUTTON 107
#define ID__UNDOBUTTON 108
#define ID__EXITBUTTON 109
#define IDM__TOPICSEARCH 110
#define IDM__FIND 24326
#define IDM__SEARCH 24327
#define IDM__EXIT 24340
#define SIZEMSGW(arg) (LOWORD(arg.LParam))
#define SIZEMSGH(arg) (HIWORD(arg.LParam))
HWND ChildWnd; PTWindowsObject ActChildWnd;
LPSTR APPLICATION__NAME="MDIDemo";
BOOL FirstTime;
__CLASSDEF(TMainWindow)
class TMainWindow: public TMDIFrame
{public: RECT menuBarRectangle; TDialog * MenuBar;
TMainWindow(LPSTR, LPSTR); ~TMainWindow(void);
virtual void GetWindowClass(WNDCLASS&);
LPSTR GetClassName() {return "MDIDEMO";}
virtual void FileNew(TMessage&) =
[CM__FIRST + CM__MDIFILENEW];
virtual void FileOpen(TMessage&) =
[CM__FIRST + CM__MDIFILEOPEN];
virtual void Paint(HDC.PAINTSTRUCT&);
virtual void WMSize(RTMessage Msg) =
[WM__FIRST+WM__SIZE];
virtual void SetupWindow();
};
class TOwnerDrawButton: public TControl {
HBITMAP hNormal, hPressed, hFocused;
void DrawBitmap(DRAWITEMSTRUCT far&, HBITMAP);
public:
TOwnerDrawButton(PTWindowsObject, int,
char *, char *, char *);
~TOwnerDrawButton();
virtual void WMLeftButtonDoubleClick(TMessage&)
=[WM__FIRST + WM__LBUTTONDBLCLK];
virtual void ODADrawEntire(DRAWITEMSTRUCT far&);
virtual void ODASelect(DRAWITEMSTRUCT far&);
virtual void ODAFocus(DRAWITEMSTRUCT far&);
virtual void DrawNormal(DRAWITEMSTRUCT far&);
virtual void DrawPressed(DRAWITEMSTRUCT far&);
virtual void DrawFocused(DRAWITEMSTRUCT far&);
};
TOwnerDrawButton:: TOwnerDrawButton ( PTWindowsObject
AParent,
```

```
int ResourceId, char * bm1, char * bm2, char * bm3)
: TControl ( AParent, ResourceId ) { hNormal = LoadBitmap
(GetApplication()->hInstance, bm1);
hPressed = LoadBitmap(GetApplication()->hInstance, bm2);
hFocused = LoadBitmap(GetApplication()->hInstance, bm3);}
TOwnerDrawButton::~~TOwnerDrawButton()
{DeleteObject(hNormal); DeleteObject(hPressed);
DeleteObject(hFocused);}
void TOwnerDrawButton:: WMLeftButtonDoubleClick ( TMessage&
Msg)
{SendMessage(HWND, WM__LBUTTONDOWN,
Msg.WParam, Msg.LParam);
}
void TOwnerDrawButton:: ODADrawEntire ( DRAWITEMSTRUCT
far&. DrawInfo)
{DrawNormal(DrawInfo);
}
void TOwnerDrawButton:: ODASelect (DRAWITEMSTRUCT far&.
DrawInfo)
{if (DrawInfo.itemState & ODS__SELECTED)
DrawPressed(DrawInfo);
else DrawFocused(DrawInfo);}
void TOwnerDrawButton:: ODAFocus (DRAWITEMSTRUCT far&.
DrawInfo)
{if (DrawInfo.itemState & ODS__FOCUS)
DrawFocused(DrawInfo);
else DrawNormal(DrawInfo);}
void TOwnerDrawButton:: DrawNormal (DRAWITEMSTRUCT far&.
DrawInfo)
{DrawBitmap(DrawInfo, hNormal);}
void TOwnerDrawButton:: DrawPressed (DRAWITEMSTRUCT far&.
DrawInfo)
{DrawBitmap(DrawInfo, hPressed);}
void TOwnerDrawButton:: DrawFocused (DRAWITEMSTRUCT far&.
DrawInfo)
{DrawBitmap(DrawInfo, hFocused);}
void TOwnerDrawButton:: DrawBitmap (DRAWITEMSTRUCT far&.
DrawInfo, HBITMAP hBitmap)
{ int x = DrawInfo.rcItem.left; int y = DrawInfo.rcItem.top;
HDC hdcMem = CreateCompatibleDC(DrawInfo.hDC);
SelectObject(hdcMem, hBitmap);
SetMapMode(hdcMem, GetMapMode(DrawInfo.hDC));
BITMAP bm;
GetObject(hBitmap, sizeof(BITMAP), (LPSTR)&bm);
POINT ptSize, ptOrg; ptSize.x = bm.bmWidth;
ptSize.y = bm.bmHeight; DPtoLP(DrawInfo.hDC, &ptSize, 1);
ptOrg.x = ptOrg.y = 0; DPtoLP(hdcMem, &ptOrg, 1);
BitBlt(DrawInfo.hDC, x, y, ptSize.x, ptSize.y,
hdcMem, ptOrg.x, ptOrg.y, SRCCOPY);
DeleteDC(hdcMem);}
class THelpButton: public TOwnerDrawButton
{ public:
THelpButton(PTWindowsObject AParent, int id)
: TOwnerDrawButton(AParent.id, "HELP1__BMP",
"HELP2__BMP", "HELP1__BMP"){} };
class TOpenButton: public TOwnerDrawButton
{public:
```

```

TOpenButton(PTWindowsObject AParent, int id)
: TOwnerDrawButton(AParent, id, "OPEN1__BMP",
    "OPEN2__BMP", "OPEN1__BMP"){};

class TSaveButton: public TOwnerDrawButton
{ public:
TSaveButton(PTWindowsObject AParent, int id)
: TOwnerDrawButton(AParent, id, "SAVE1__BMP", "SAVE2__
    BMP", "SAVE1__BMP"){};

class TFindButton: public TOwnerDrawButton
{ public:
TFindButton(PTWindowsObject AParent, int id)
: TOwnerDrawButton(AParent, id, "FIND1__BMP", "FIND2__BMP",
    "FIND1__BMP"){};

class TSearchButton: public TOwnerDrawButton
{ public:
TSearchButton(PTWindowsObject AParent, int id)
: TOwnerDrawButton(AParent, id, "SEARCH1__BMP", "SEARCH2
    __BMP", "SEARCH1__BMP"){};

class TCutButton: public TOwnerDrawButton
{ public:
TCutButton(PTWindowsObject AParent, int id)
: TOwnerDrawButton(AParent, id, "CUT1__BMP", "CUT2__BMP",
    "CUT1__BMP"){};

class TCopyButton: public TOwnerDrawButton
{ public:
TCopyButton(PTWindowsObject AParent, int id)
: TOwnerDrawButton(AParent, id, "COPY1__BMP", "COPY2__
    BMP", "COPY1__BMP"){};

class TPasteButton: public TOwnerDrawButton
{ public:
TPasteButton(PTWindowsObject AParent, int id)
: TOwnerDrawButton(AParent, id, "PASTE1__BMP", "PASTE2__
    BMP", "PASTE1__BMP"){};

class TUndoButton: public TOwnerDrawButton
{ public:
TUndoButton(PTWindowsObject AParent, int id)
: TOwnerDrawButton(AParent, id, "UNDO1__BMP", "UNDO2__
    BMP", "UNDO1__BMP"){};

class TExitButton: public TOwnerDrawButton
{ public:
TExitButton(PTWindowsObject AParent, int id)
: TOwnerDrawButton(AParent, id, "EXIT1__BMP", "EXIT2__BMP",
    "EXIT1__BMP"){};

class TMenuBar: public TDialog { THelpButton * helpButton;
TOpenButton * openButton;
TSaveButton * saveButton; TFindButton * findButton;
TSearchButton * searchButton; TCutButton * cutButton;
TCopyButton * copyButton; TPasteButton * pasteButton;
TUndoButton * undoButton; TExitButton * exitButton;
public:
TMenuBar(PTWindowsObject);
virtual void Help(TMessage&)=
    [ID __ FIRST+ID __ HELPBUTTON];
virtual void Open(TMessage&)=
    [ID __ FIRST+ID __ OPENBUTTON];
virtual void Save(TMessage&)=
    [ID __ FIRST+ID __ SAVEBUTTON];

```

```

virtual void Find(TMessage&)=
    [ID __ FIRST+ID __ FINDBUTTON];
virtual void Search(TMessage&)=
    [ID __ FIRST+ID __ SEARCHBUTTON];
virtual void Cut(TMessage&)=
    [ID __ FIRST+ID __ CUTBUTTON];
virtual void Copy(TMessage&)=
    [ID __ FIRST+ID __ COPYBUTTON];
virtual void Paste(TMessage&)=
    [ID __ FIRST+ID __ PASTEBUTTON];
virtual void Undo(TMessage&)=
    [ID __ FIRST+ID __ UNDOBUTTON];
virtual void Exit(TMessage&)=
    [ID __ FIRST+ID __ EXITBUTTON];
};

TMenuBar: :TMenuBar(PTWindowsObject AParent)
: TDialog(AParent, "MENUBAR")
{ helpButton=new THelpButton(this, ID __ HELPBUTTON);
openButton=new TOpenButton(this, ID __ OPENBUTTON);
saveButton=new TSaveButton(this, ID __ SAVEBUTTON);
findButton=new TFindButton(this, ID __ FINDBUTTON);
searchButton=new TSearchButton(this, ID __ SEARCHBUTTON);
cutButton=new TCutButton(this, ID __ CUTBUTTON);
copyButton=new TCopyButton(this, ID __ COPYBUTTON);
pasteButton=new TPasteButton(this, ID __ PASTEBUTTON);
undoButton=new TUndoButton(this, ID __ UNDOBUTTON);
exitButton=new TExitButton(this, ID __ EXITBUTTON); }
void TMenuBar: :Help(TMessage& Msg)
{ HWND hwndParent; hwndParent=GetParent(Msg.Receiver);
SendMessage ( hwndParent, WM __ COMMAND, IDM __
    TOPICSEARCH, 0L); }
void TMenuBar: :Open(TMessage& Msg)
{ HWND hwndParent; hwndParent=GetParent(Msg.Receiver);
SendMessage ( hwndParent, WM __ COMMAND, CM __
    MDIFILEOPEN, 0L); }
void TMenuBar: :Save(TMessage& Msg)
{ SendMessage(ChildWnd, WM __ SETFOCUS, 0, 0L);
HWND hwndParent;
hwndParent=GetParent(Msg.Receiver);
SendMessage (hwndParent, WM __ COMMAND, CM __ FILESAVE,
    0L); }
void TMenuBar: :Find(TMessage&)
{ SendMessage(ChildWnd, WM __ SETFOCUS, 0, 0L);
SendMessage (ActChildWnd -> HWindow, WM __ COMMAND, IDM
    __ FIND, 0L); }
void TMenuBar: :Search(TMessage&)
{ SendMessage(ChildWnd, WM __ SETFOCUS, 0, 0L);
SendMessage (ActChildWnd -> HWindow, WM __ COMMAND, IDM
    __ SEARCH, 0L); }
void TMenuBar: :Cut(TMessage&)
{ SendMessage(ChildWnd, WM __ SETFOCUS, 0, 0L);
SendMessage (GetFocus(), WM __ CUT, 0, 0L);
}
void TMenuBar: :Copy(TMessage&)
{ SendMessage(ChildWnd, WM __ SETFOCUS, 0, 0L);
SendMessage (GetFocus(), WM __ COPY, 0, 0L); }
void TMenuBar: :Paste(TMessage&)

```

```
{ SendMessage(ChildWnd, WM__SETFOCUS, 0, 0L);
SendMessage(GetFocus(), WM__PASTE, 0, 0L); }
void TMenuBar::Undo(TMessage& Msg)
{ SendMessage(ChildWnd, WM__SETFOCUS, 0, 0L);
SendMessage(GetFocus(), WM__UNDO, 0, 0L); }
void TMenuBar::Exit(TMessage& Msg)
{ HWND hwndParent; hwndParent = GetParent(Msg.Receiver);
SendMessage(hwndParent, WM__SYSCOMMAND, SC__CLOSE,
0L); }
class TFileEditWindow: public TFileWindow {
public:
TFileEditWindow(PTWindowsObject AParent,
LPSTR Title, LPSTR Filename)
: TFileWindow(AParent, Title, Filename)
{ }
virtual void WMKillFocus(RTMessage Msg) =
[WM__FIRST+WM__KILLFOCUS];
LPSTR GetClassName() {return "FileEditor"; }
void GetWindowClass(WNDCLASS&);
TMainWindow::TMainWindow(LPSTR Title, LPSTR Menu)
: TMDIFrame(Title, Menu)
{ ChildMenuPos = 0; FirstTime = TRUE; }
void TMainWindow::FileNew(TMessage& Msg)
{ ActChildWnd = GetApplication() -> MakeWindow(
new TFileEditWindow(this, APPLICATION__NAME, NULL)); }
void TFileEditWindow::WMKillFocus(TMessage& Msg)
{ ChildWnd = Msg.Receiver; }
void TMainWindow::FileOpen(TMessage& Msg)
{ char name[80]; ostrstream title(name, sizeof(name));
title << " * . * " << ends;
if (GetApplication() -> ExecDialog(
new TFileDialog(this, SD__FILEOPEN, name)) == IDOK )
ActChildWnd = GetApplication() -> MakeWindow(
new TFileEditWindow(this, APPLICATION__NAME, name));
//记录当前活动窗口
}
class TMDIDemo: public TApplication {
public:
TMDIDemo(LPSTR AName, HINSTANCE AnInstance,
HINSTANCE APrevInstance, LPSTR ACmdLine, int ACmdShow)
: TApplication(AName, AnInstance, APrevInstance,
ACmdLine, ACmdShow) {}
virtual void InitMainWindow(); void InitInstance();
};
void TMDIDemo::InitMainWindow()
{ MainWindow =
new TMainWindow(APPLICATION__NAME, "FileMenu"); }
void TMDIDemo::InitInstance()
{ TApplication::InitInstance();
HAccTable = LoadAccelerators(hInstance, "FILECOMMANDS");
BWCCGetVersion(); }
int PASCAL WinMain (HINSTANCE AnInstance, HINSTANCE
APrevInstance, LPSTR ACmdLine, int ACmdShow)
{ TMDIDemo MDIDemo(APPLICATION__NAME, AnInstance,
APrevInstance, ACmdLine, ACmdShow);
MDIDemo.Run(); return MDIDemo.Status; }
#define BACKGROUND RGB(192, 192, 192)
```

```
void TMainWindow::Paint(HDC hdc, PAINTSTRUCT& s)
{ RECT r; GetClientRect(HWindow, &r);
menuBarRectangle.left = r.left - 1;
menuBarRectangle.top = r.top - 1;
menuBarRectangle.right = r.right + 1;
menuBarRectangle.bottom = r.top + 21;
SelectObject(hdc, CreatePen(PS__SOLID, 1, RGB(0, 0, 0)));
SelectObject(hdc, CreateSolidBrush(BACKGROUND));
Rectangle(hdc, menuBarRectangle.left, menuBarRectangle.top,
menuBarRectangle.right, menuBarRectangle.bottom);
if (FirstTime)
{ MenuBar = new TMenuBar(this);
GetApplication() -> MakeWindow(MenuBar);
MenuBar -> Show(SW__SHOW); }
FirstTime = FALSE; }
void TMainWindow::WMSize(TMessage& Msg)
{ int ClientWidth; int ClientHeight; int FrameWidth; int FrameHeight;
int xClient; int yClient; FrameWidth = SIZEMSGW(Msg);
FrameHeight = SIZEMSGH(Msg);
menuBarRectangle.left = -1;
menuBarRectangle.top = -1;
menuBarRectangle.right = FrameWidth + 1;
menuBarRectangle.bottom = FrameWidth - 22;
ClientWidth = FrameWidth; ClientHeight = FrameHeight - 22;
xClient = 0; yClient = 21;
MoveWindow(ClientWnd -> HWindow, xClient, yClient,
ClientWidth, ClientHeight, TRUE);
InvalidateRect(HWindow, &menuBarRectangle, 0); }
void TMainWindow::SetupWindow()
{ HMENU FrameMenu; RECT R; InitClientWindow();
RemoveClient(); // remove ClientWnd from OWL child list
FrameMenu = GetMenu(HWindow);
ClientWnd -> ClientAttr -> hWindowMenu = GetSubMenu
(FrameMenu, ChildMenuPos);
GetClientRect(HWindow, &R);
if (ClientWnd -> Attr.X == CW__USEDEFAULT )
{ ClientWnd -> Attr.X = R.left;
ClientWnd -> Attr.Y = R.top + 21; }
if (ClientWnd -> Attr.W == CW__USEDEFAULT )
{ ClientWnd -> Attr.W = R.right - R.left;
ClientWnd -> Attr.H = R.bottom - R.top - 22; }
ClientWnd -> Attr.Style |= WS__VSCROLL + WS__HSCROLL;
if (ClientWnd -> Create() )
TWindow::SetupWindow();
else Status = EM__INVALIDCLIENT; }
void TFileEditWindow::GetWindowClass(WNDCLASS& wndClass)
{ TFileWindow::GetWindowClass(wndClass);
wndClass.hIcon = LoadIcon(GetApplication() -> hInstance, "Child
__Icon");
}
```

资源文件菜单部分:

```
FileMenu MENU
BEGIN
POPUP "&File"
BEGIN
```

(下转第17页)

给 DOS 增加两个外部命令 ——PEEK 与 POKE

□陈凤国

许

多高级语言均提供显示及修改内存单元的命令或函数 PEEK 与 POKE, 而 DOS 环境缺少这样的命令, 本文利用 Turbo C2.0 语言编写了两个小程序 PEEK.C 与 POKE.C。

命令使用格式与功能:

PEEK 命令格式:

C) PEEK(/W 或/B) 偏段地址 偏移量

功能: 显示指定内存单元的内容, 参数 /W 与/B 分别表示字还是字节。

POKE 命令格式:

C) POKE/W(或/B) 段地址 偏移量 要写入的内容

功能: 把给定的字(/W)或字节(/B)写到指定内存单元中去。

值得注意的是不管是地址还是内容均要用16进制表示。

应用实例:

C) PEEK/W 40 13

显示40:13与40:14两个字节单元中的内容, 它表示基本内存的数量(单位 KB)。

C) POKE /W 40 17 40

把字节40H 写到40:17单元, 即强制大写状态; 如果40H 改为0H, 则变为强制小写态。

注意本文提供的程序可在 TINY 模式下编译, 连接成的两个 .EXE 文件, 可用 EXE2BIN 转换为 .COM 文件。

PEEK.C 源程序清单如下:

```
#include "stdlib.h"
#include "stdio.h"
#include "string.h"
int value(char s[]);
main (int argc, char * argv[])
```

```
{if(argc!=4){
    Printf("Usage: PEEK/W 或/B 段址 偏移量\n");
    Printf("[用16进制形式, /W:字, /B:字节]\n"); exit(1);}
if(!strcmp(argv[1], "/W"))
    printf(" %04XH", peek (value (argv [2], value (argv
    [3])))); else
    printf(" %02XH", peekb (value (argv [2], value (argv
    [3]))));}
```

```
int value(char s[])
{int val=0; while(*s)
{if(*s>='0'&&*s<='9') val=val*16+*s-'0'
else
if(*s>='a'&&*s<='f') val=val*16+*s-'a'+10;
else if(*s>='A'&&*s<='F')
val=val*16+*s-'A'+10;
else {printf("\nerror parameter\n"); exit(1);}
*s++;} return(val);}
```

POKE.C 源程序清单如下:

```
#include "stdlib.h"
#include "stdio.h"
#include "string.h"
int value(char s[]);
main(int argc, char * argv[]){if(argc!=5){
    printf("用法: POKE /W 或/B 段址 偏移 写入内容\n");
    printf("[用16进制形式, /W 表示字, /B 表示字节]\n");
    exit(1);}if(!strcmp(argv[1], "/W"))
    poke (value (argv [2], value (argv [3]), value (argv
    [4]));
    else poke (value (argv [2], value (argv [3]), value (argv
    [4]));}
int value(char s[]) {int val=0; while(*s)
{if(*s>='0'&&*s<='9') val=val*16+*s-'0';
else if(*s)='a'&&*s<='f') val=val*16+*s-'a'+
10;
else if(*s)='A'&&*s<='F') val=val*16+*s-'A'+
10;
else {printf("\nerror parameter\n"); exit(1);}
*s++;} return(val);}
```

SKILL

优化配置文件,提高微机利用率

□潘泰林

对

配置文件及相应的自动批处理文件做一定的调整,使常用的任务能够高效、合理和简便地使用微机系统资源。这对于我国微机用户是很有必要的。本文以四川省邮电统计信息自动化系统(以下简称 TJGL)、全国第三产业普查数据录入软件(以下简称 SC)为例,讨论配置文件优化的重要性。

1. 在 DOS V3.30 的环境下,机型为 IBM PC/AT 286 及其兼容机

(1) 建立 TJGL 的 CONFIG.SYS 和 AUTOEXEC.BAT

① 将 TJGL 所规定的 CONFIG.SYS 更名为 CONFIG.TJ,其内容如下:

```
DEVICE=C:\DOS\VDISK.SYS 300 512 30/E
DEVICE=C:\213\ANSI.SYS
FILES=40
BUFFERS=30
LASTDRIVE=Z
```

② 将 TJGL 所规定 AUTOEXEC.BAT 更名为 AUTOEXEC.TJ,其内容如下:

```
@ECHO OFF
CLS
PATH C:\;C:\DOS\C;C:\213\C;\FOX\C;\TJGL
APPEND/E
APPEND C:\;C:\213\C;\FOX\C;\TJGL
SUBST S: D:\DBSR
SUBST T: D:\JCSJ
SUBST K: C:\DHTX\SJHC
SUBST L: C:\DHTX\SJBF
SUBST M: C:\DHTX\SBSJ
FASTOPEN C:=40 > NUL
SET XGA=V
SET VDISK=E:
SET SBJM=CQD
SET PRIN=3
```

```
PROMPT $P$G
CD\213
IFNOT EXIST E:HZK16 COPY HZK16 E:
FILE3 E2
CCCC
CV26
INT10H
CD\TJGL
FOXPLUS ZKCX
```

(2) 建立 SC 的 CONFIG.SYS 和 AUTOEXEC.BAT

① 将 SC 所规定 CONFIG.SYS 更名为 CONFIG.SC,其内容如下:

```
FILES=40
BUFFERS=40
```

② 将 SC 所规定 AUTOEXEC.BAT 更名为 AUTOEXEC.SC,其内容如下:

```
@ECHO OFF
PATH C:\;C:\DOS\C;C:\CCDOS\C;\JYM
APPEND/E
APPEND C:\;C:\DOS\C;C:\CCDOS\C;\JYM
CD\CCDOS
CCCC
KEYWB
CD\SC
CALL SC
```

(3) 建立两个应用软件间自动切换的批处理文件

① 自动切换到 TJGL 的批处理文件 TOTJ.BAT

```
@ECHO OFF
CD\
COPY CONFIG.TJ CONFIG.SYS>NUL
COPY AUTOEXEC.TJ AUTOEXEC.BAT>NUL
REBOOT
```

②自动切换到 SC 的批处理文件 TOSC.BAT

```
@ECHO OFF
CD\
COPY CONFIG.SYS CONFIG.SYS>NUL
COPY AUTOEXEC.SYS AUTOEXEC.SYS>NUL
REBOOT
```

其中,REBOOT.COM 为仿真微机系统的热启动文件,建立方法如下:

```
C:\>DEBUG
-A100
xxxx:0100 MOV AX,40
xxxx:xxxx MOV DS,AX
xxxx:xxxx MOV AX,1234
xxxx:xxxx MOV [0072],AX
xxxx:xxxx JMP FFFF:0000
xxxx:xxxx
-NREBOOT.COM
-RCX
CX 0000
:10
-W
-Q
```

执行完以上各步后,我们要运行 TJGL,只需键入 TOTJ;同样只要键入 TOSC,即可进入 SC,而不需要任何人工干预。

2. 在 DOS V5.0 的环境下,机型为 IBM PC/AT 386、486 及其兼容机。

(1)优化 TJGL 的配置文件 CONFIG.TJ

```
DEVICE=C:\DOS\HIMEM.SYS
DEVICE=C:\DOS\RAMDRIVE.SYS 300 512 30/E
DEVICE=C:\DOS\EMM386.EXE NOEMS I=C800-F000
DOS=HIGH,UMB
DEVICE=C:\213\ANSI.SYS
BUFFERS=30
FILES=40
LASTDRIVE=Z
```

其中:① RAMDRIVE.SYS 一定要用在 EMM386.EXE 之前,否则,213H 中的 FILE3.COM 将无法正确读取 RAMDRIVE.SYS 中的虚盘字库 HZK16。②如微机扩展槽中插有标准配置以外的其他功能卡(如防病毒卡),且地址在 C800-F000 之间,则应在 EMM386.EXE 命令中,加上参数 X=mmm-m-nnnn,有效值为其占用的地址范围,以免引起地址冲突。

(2)根据优化后的 CONFIG.TJ,改写相应的自动批处理文件 AUTOEXEC.TJ

```
@ECHO OFF
CLS
PATH C:\;C:\DOS;C:\213;C:\FOX;C:\TJGL
LH APPEND/E
APPEND C:\;C:\213;C:\FOX;C:\TJGL
SUBST S: D:\DBSR
```

```
SUBST T: D:\JCSJ
SUBST K: C:\DHTX\SJHC
SUBST L: C:\DHTX\SJBF
SUBST M: C:\DHTX\SBSJ
LH FASTOPEN C:=40>NUL
SET XGA=V
SET SBJM=CQD
SET PRIN=3
PROMPT $P$G
CD\213
IF NOT EXIST E:\HZK16 COPY HZK16 E:
LH FILE3 E2
LH CCCC
LH CV26
LH INT10H
CD\TJGL
FOXPLUS ZKCX
```

本例中的 213H 为省局信息中心修改版,适用于高版本 DOS。否则应照下列方法修改 FILE3.COM:

```
C:\213>DEBUG FILE3.COM
-E 027A
xxxx:027A 10.11
-W
-Q
```

经过上述优化,常规内存的剩余空间将达到 614K,使得 TJGL 的显示和运行速度明显加快。

而 SC 在规定的配置文件下,由于调用了 KEY-WB.COM(五笔字型驱动模块),占据了大量的常规内存,有时出现内存不够的现象,根据 DOS 5.0 的内存管理技术,对配置文件优化如下:

(1)优化 SC 的配置文件 CONFIG.SC

```
DEVICE=C:\DOS\HIMEM.SYS
DOS=HIGH,UMB
DEVICE=C:\DOS\EMM386.EXE NOEMS I=C800-F000
FILES=40
BUFFERS=40
```

(2)根据优化的 CONFIG.SC,改写相应的 AUTOEXEC.SC

```
@ECHO OFF
PATH C:\;C:\DOS;C:\CCDOS;C:\JYM
LH APPEND/E
APPEND C:\;C:\DOS;C:\CCDOS;C:\JYM
CD\CCDOS
LH CCCC
LH KEYWB
CD\SC
CALL SC
```

3. 在 DOS V6.0 的环境下,机型为 IBM PC/AT 386、486 及其兼容机

MS-DOS V6.0 最突出的一点,就是我们可以将不同的应用程序所要求的配置命令组合在一个 CONFIG.SYS 中,在微机启动时,用菜单方式选择

不同的系统配置。还可以在一个 AUTOEXEC. BAT 中用 GOTO %CONFIG% 写出对应的自动批处理命令。根据这一特性,我们可以优化成这样一个多重的系统配置文件:

```
[MENU]
MENUITEM=TJGL
MENUITEM=SC
MENUITEM=PUBLIC
MENUDEFAULT=TJGL,10
MENUCOLOR=15,1
[TJGL]
DEVICE=C:\DOS\HIMEM.SYS
DEVICE=C:\DOS\RAMDRIVE.SYS 300 512 30/E
DEVICE=C:\DOS\EMM386.EXE NOEMS I=C800-F000
DOS=HIGH,UMB
DEVICE=C:\213\ANSI.SYS
BUFFERS=30
FILES=40
LASTDRIVE=Z
[SC]
DEVICE=C:\DOS\HIMEM.SYS
DOS=HIGH,UMB
DEVICE=C:\DOS\EMM386.EXE NOEMS I=C800-F000
FILES=40
BUFFERS=40
[PUBLIC]
DEVICE=C:\DOS\HIMEM.SYS
DEVICE=C:\DOS\EMM386.EXE NOEMS I=C800-F000
DOS=HIGH,UMB
FILES=30
```

然后,将 AOTOEXEC. TJ 更名为 TJ. BAT, 新建一个 AUTOEXEC. BAT:

(上接第13页)

```
MENUITEM "&New", 24331
MENUITEM "&Open...", 24332
MENUITEM "&Save", 24333
MENUITEM "Save &As...", 24334
MENUITEM SEPARATOR
MENUITEM "E&xit", 24340
END
POPUP "&Edit"
BEGIN
MENUITEM "&Undo\aAlt+BkSp", 24325
MENUITEM SEPARATOR
MENUITEM "&Cu&t\aShift+Del", 24320
MENUITEM "&Copy\aCtrl+Ins", 24321
MENUITEM "&Paste\aShift+Ins", 24322
MENUITEM "&Delete\aDel", 24323
MENUITEM "C&lear All\aCtrl+Del", 24324
END
POPUP "&Search"
BEGIN
MENUITEM "&Find...", 24326
MENUITEM "&Replace...", 24327
```

```
@ECHO OFF
CLS
GOTO %CONFIG%
:TJGL
CD\
CALL TJ
GOTO END
:SC
PATH C:\;C:\DOS\;C:\CCDOS\;C:\JYM
LH APPEND/E
APPEND C:\;C:\DOS\;C:\CCDOS\;C:\JYM
CD\CCDOS
LH CCCC
LH KEYWB
CD\SC
CALL SC
GOTO END
:PUBLIC
C:\DOS\SMARTDRV.EXE
PATH C:\;C:\DOS
SET TEMP=C:\DOS
GOTO END
:END
LH DOSKEY
SET DIRCMD=/P/O
```

综上所述,经过配置文件和自动批处理文件的优化处理,使得在同一台微机上运行不同配置的应用程序,变得高效、合理和简便。尤其是随着 DOS 版本的升级(如 DOS 6.0),简便的程度更为明显,并且还可以最大限度地节省常规内存,以求应用程序的高速运行。笔者期盼本文能对读者起到抛砖引玉的作用。

SKILL

```
MENUITEM "&Next\&F3", 24328
END
POPUP "&Window"
BEGIN
MENUITEM "&Tile", 24336
MENUITEM "&Cascade", 24337
MENUITEM "&Arrange &Icons", 24335
MENUITEM SEPARATOR
MENUITEM "C&lose All", 24338
END
END
```

SKILL

★	独特的升级技术,最适合中国国情	★
★	欢迎加入	★
★	帝霸计算机反病毒服务网	★
★	热线咨询电话:(01)8123896	★

如何用 C 语言开发 TVGA256 色图形

□张战军

T

VGA 卡是 Trident 微系统公司推出的一种通用的低成本、全功能、单芯片超级 VGA 图形适配器,它与标准 VGA、EGA、CGA、Hercules 和 MDA 卡完全兼容。鉴于国内所使用的 SuperVGA 适配器大部分是 8900 系列 TVGA 卡,本文将详细介绍在其上如何用 C 语言开发高分辨率 256 色图形的技术。

TVGA8900(C)卡所支持的五种 256 色显示模式如表 1 所示。由于大多数 C 编译器的库函数提供的图形函数并不全部支持这五种图形模式,用汇编语言编程要涉及很多控制寄存器参数设置,晦涩难懂,难以编写调试,很多用户不习惯使用。为此,笔者认真研究了 TVGA 图形适配器的内部结构,结合自己开发图形、图象软件的实践经验,运用 C 语言和 BIOS 功能调用编制了一套兼容 TVGA 图形卡各种 256 色模式的基本图形函数。本文以这些函数为例,讲述 TVGA 卡的存储器组织,调色结构以及用 C 语言开发图形软件的方法,以期能起到抛砖引玉的作用。

表 1 TVGA8900(C)卡支持的五种 256 色显示模式

模式	分辨率	要求 VRAM 大小	VRAM 基地址	颜色数
13H	320 * 200	64000B, 256K	0A00000H	从 256K 种颜色中选出 256 种颜色
5cH	640 * 400	64K * 3 + 59392B, 256K		
5dH	640 * 480	64K * 4 + 45056B, 512K		
5eH	800 * 600	64K * 7 + 21248B, 512K		
62H	1024 * 768	64K * 12B, 1024K		

1. 调色机制

TVGA256 色模式与其 16 色模式不同,图象色彩信息采用存取 DAC 寄存器的方式。DAC(数据模拟转换器)方框图如图 1 所示。

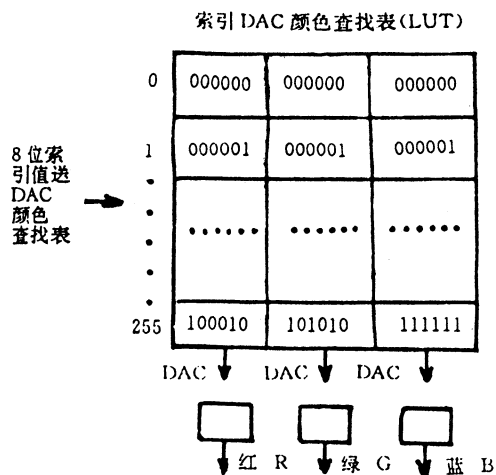


图 1 DAC 方框图

从图 1 中可看出 DAC 由两部分组成:第一部分是 256 个 18 位的寄存器,它起着颜色对照表的作用,每个寄存器读出内容中红、绿、蓝

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

三原色各占6位,因而最多可以定义262144(256K)种颜色。视频存储器 VRAM 中存放的屏幕像素值仅为这256个颜色寄存器的索引值,所以同屏仅能显示256种颜色,一个像素颜色值需要8位。颜色对照表的输出分别送到三个数模转换电路,它们把数字信号转换成模拟信号送到专用的模拟监视器上去。用户可以根据显示需要设定这256颜色寄存器中红、绿、蓝(取值范围为0~63)的值。那么怎样读写颜色寄存器呢?建议不要直接使用颜色寄存器读写,这涉及到几个寄存器的配合使用,容易损伤显示器,况且在图形开发时调色板设置和读出操作一般并不频繁,因此无需考虑它会影响作图速度。BIOS INT10H 为此提供了强有力的支持,直接调用即可(详见源程序)。

2. 存储器组织

在256色模式下,TVGA 存储图象信息时采用压缩像素法,即每个屏幕像素点信息占用显示缓冲区一个字节,且像素点坐标与其存放位置两者之间呈线性关系。微机为 VRAM 提供的地址空间仅为0A0000:0到0A0000:0FFFFH的一个段(64KB),而256色各模式所需占用的地址空间(见表1)除模式13H外都大于64KB。因此要对 VRAM 进行操作,必须把它们按段分别映射到0A0000:0H到0A0000:0FFFFH的地址空间中,然后就可以直接对 VRAM 进行读写操作。TVGA 采用分页显示机制,即把1MB的VRAM分成16段(也就是16页,实际上最多使用12页,每页64KB),把段号(0~11)的bit1取反后送I/O端口3C4H的第14号寄存器,然后就可以在地地址0A0000:0H到0A0000:0FFFFH之间操作VRAM的这一段。

3. TVGA256色编程常用寄存器读写

虽然对于TVGA高分辨率256色图形模式允许以64KB和128KB页模式分割VRAM,但由于单色卡和CGA卡使用内存空间0B0000:0H到0B0000:0FFFFH,每页128KB就不能与其共存,因此推荐使用64KB页模式,避免地址冲突。

64KB页模式下选择页号通过方式控制寄存器1,它是TVGA为支持增强显示模式而增加的标准VGA卡上没有的寄存器,作为操作定时序寄存器的一员。由于TVGA的每个寄存器都没有自己独特的输入、输出地址,所以对它们的存取需分两步(以存取方式控制寄存器1为例):首先必须把方式控制寄存器的索引0EH放到操作定时序地址寄存器(输出地址为03C4H),然后再通过操作定时序数据寄存器(输入/输出地址为3C5H)对方式控制寄存器进行读写操作,也就是把所选页号输入/输出到方式控制寄存器。但页号的bit1在写入时必须按反值写入,而读

出时则按原值(非反)读出(详见源程序)。

4. 基本图形函数源码

本文利用BIOS INT 10H所提供的功能,结合TVGA寄存器操作,用Turbo C 2.0编制了设置显示模式、设置调色板、读取调色板、清屏、写点、读点、画线、画圆、多边形填充等基本作图函数,对TVGA256色的五种模式都作了兼容考虑。关于BIOS INT10H所提供的功能以及计算机图形学原理,限于篇幅,这里不作详细介绍,请读者参考有关资料。有了这些基本图形函数,读者可根据实际需要很容易地写出功能更加强大的图形函数,这也就达到了本文的目的。另外值得说明的是:考虑到图形的生成速度,源程序采用了直接读写VRAM的技术,并把屏幕行始偏移作了预算,有效地提高了作图速度。

源程序清单如下:

```
#include "stdio.h"
#include "dos.h"
#include "math.h"
long Addr[768]; /* 行始偏移缓冲区 */
int Mode; /* 设置工作模式,并预算行始偏移 */
Set __mode(int mode)
{
    long i; union REGS r; r.h.ah=0; r.h.al=mode;
    int86(0x10,&r,&r);
    switch(mode) {
        case 0x13: for(i=0; i<200; i++) Addr[i]=320*i; Mode=0x13;
                    break;
        case 0x5c: for(i=0; i<400; i++) Addr[i]=640*i; Mode=0x5c;
                    break;
        case 0x5d: for(i=0; i<480; i++) Addr[i]=640*i; Mode=0x5d;
                    break;
        case 0x5e: for(i=0; i<600; i++) Addr[i]=800*i; Mode=0x5e;
                    break;
        case 0x62: for(i=0; i<768; i++) Addr[i]=1024*i; Mode=0x62;
                    break;
        case 0x03: break;
        default:
            printf("Not support this mode. \n"); exit(0);
    }
}
/* 设置调色板 */
Set __pattern(unsigned char pat[256][3])
{
    struct SREGS inreg;
    union REGS reg;
```

★ 帝霸计算机反病毒服务网 ★
★ 专用软件(普及版) ★

★ 特价优惠:20元/套(含邮费) ★

★ 热线咨询电话:(01)8123896 ★

```

reg. x. ax=0x1012;
reg. x. bx=0;
reg. x. cx=256;
reg. x. dx=FP __ OFF(pat);
inreg. es=FP __ SEG(pat);
int86x(0x10,&reg.&reg.&inreg);
}
/* 获取调色板 */
Set __ pattern(unsigned char pat[256][3])
{struct SREGS inreg;
 union REGS reg;
 reg. x. ax=0x1017;
 reg. x. bx=0;
 reg. x. cx=256;
 reg. x. dx=FP __ OFF(pat);
 inreg. es=FP __ SEG(pat);
 int86x(0x10,&reg.&reg.&inreg);
}
/* 清屏成所设的颜色 */
Clr __ scr(unsigned char color)
{
 long i;
 unsigned int L;
 unsigned char Page,p;
 unsigned char far * scr=(unsigned char far *)0xA0000000,
 far * scr1;
 scr1=scr;
 switch(Mode){
 case 0x13:for(i=0;i<64000;i++) * scr++=color;
 break;
 case 0x5c:
 case 0x5d:
 case 0x5e:
 case 0x62:switch(Mode){
 case 0x5c:Page=3; L=59392; break;
 case 0x5d:Page=4; L=45056; break;
 case 0x5e:Page=7; L=21248; break;
 default: Page=12; L=0;
 }
 }
for(p=0;p<Page;p++){/* 设置显示页 */
 outportb(0x3c4,0x0e);
 outportb(0x3c5,p^ 0x02);
 scr=scr1;
 for(i=0;i<65536;i++) * scr++=color;
 }
 outportb(0x3c4,0x0e);
 outportb(0x3c5,p^ 0x02);
 scr=scr1;
 for(i=0;i<L;i++) * scr++=color;
 break;
 default: break;
}
}
/* 写点 */
Put __ pixel(int x,int y,unsigned char color)
{
 long Offset;

```

```

char Page;
static char Old __ Page=0;
unsigned char far * address;
switch(Mode){
 case 0x13:Offset=Addr[y]+x;
 address=(unsigned char far *) (0xA0000000L+Offset);
 * address=color;
 break;
 case 0x5c:
 case 0x5d:
 case 0x5e:Offset=Addr[y]+x;
 Page=(Offset>>16);
 if(Page!=Old __ Page){
 outportb(0x3c4,0x0e);
 outportb(0x3c5,Page&^ 0x02);
 Old __ Page=Page;
 }
 Offset=Offset&65535;
 address=(unsigned char far *) (0xA0000000L +
 Offset);
 * address=color;
 break;
 case 0x62:Offset=Addr[y]+x;
 Page=y>>6;
 if(Page!=Old __ Page){
 outportb(0x3c4,0x0e);
 outportb(0x3c5,Page^ 0x02);
 Old __ Page=Page;
 }
 Offset=Offset&65535;
 address=(unsigned char far *) (0xA0000000L+Offset);
 * address=color;
 break;
 default:break;
}
}
/* 获取点像素值 */
Get __ pixel(int x,int y)
{
 long Offset;
 char Page;
static char Old __ Page=0;
unsigned char far * address;
unsigned char color;
switch(Mode){
 case 0x13:Offset=Addr[y]+x;

```

★	帝霸计算机反病毒服务网	★
★	专用软件(普及版)	★
★	特价优惠:20元/套(含邮费)	★
★	热线咨询电话:(01)8123896	★

```

address=(unsigned char far * )(0XA0000000L+Offset);
color= * address;
break;
case 0x5c:
case 0x5d:
case 0x5e:Offset=Addr[y]+x;
    Page=(Offset>>16);
    if(Page!=Old __ Page){
        outportb(0x3c4,0xE);
        outportb(0x3c5,Page^ 0x02);
        Old __ Page=Page;
    }
    Offset=Offset&65535;
    address=(unsigned char far * )(0XA0000000L +
        Offset);
    color= * address;
    break;
case 0x62:Offset=Addr[y]+x;
    Page=y>>6;
    if(Page!=Old __ Page){
        outportb(0x3c4,0xE);
        outportb(0x3c5,Page^ 0x2);
        Old __ Page=Page;
    }
    Offset=Offset&65535;
    address=(unsigned char far * )(0XA0000000L +
        Offset);
    color= * address;
    break;
default: break;
}
return(color);
}
/* 为提高画线速度采用 Bresenham 算法 */
Draw __ line(int x1,int y1,int x2,int y2,unsigned char color)
{
    int Delta __ x,Delta __ y,temp,Cycle,Step __ x,Step __ y;
    Delta __ x=x2-x1;Delta __ y=y2-y1;
    if(Delta __ x<0){Delta __ x=-Delta __ x; Step __ x=-1;}
    else Step __ x=1;
    if(Delta __ y<0){Delta __ y=-Delta __ y;Step __ y=-1;}
    else Step __ y=1;
    if(Delta __ y<Delta __ x){
        Cycle=Delta __ x>>1;
        while(x1!=x2){
            Put __ pixel(x1,y1,color);
            Cycle+=Delta __ y;
            if(Cycle>Delta __ x){Cycle-=Delta __ x;y1+=Step __ y;}
            x1+=Step __ x;
        }
        Put __ pixel(x1,y1,color);
    }
    else{Cycle=Delta __ y>>1;
        temp=Delta __ x;Delta __ x=Delta __ y;Delta __ y=temp;
        temp=Step __ x;Step __ x=Step __ y;Step __ y=temp;
        while(y1!=y2){
            Put __ pixel(x1,y1,color);

```

```

            Cycle+=Delta __ y;
            if(Cycle>Delta __ x){Cycle-=Delta __ x;x1+=Step __ y;}
            y1+=Step __ x;
        }
        Put __ pixel(x1,y1,color);
    }
}
/* 为提高画圆速度采用 Bresenham 算法 */
Draw __ circle(int xc,int yc,int r,unsigned char color)
{
    int x,y,xf,yf,r2,e,asp,temp;
    if((Mode==0x13)|| (Mode==0x5c)) asp=120;
    else asp=100;
    x=r;y=0;e=0;r2=r*r;
    while(x>=y){
        y=(int)(sqrt(r2-x*x));
        temp=e;e=y;y=temp;
        while(y<=e){
            xf=(asp*x)/100;
            yf=(asp*y)/100;
            Put __ pixel(xc+xf,yc+y,color);Put __ pixel(xc+xf,yc-x,color);
            Put __ pixel(xc+xf,yc-y,color);Put __ pixel(xc+xf,yc+x,color);
            Put __ pixel(xc-xf,yc+y,color);Put __ pixel(xc-xf,yc-x,color);
            Put __ pixel(xc-xf,yc-y,color);Put __ pixel(xc-xf,yc+x,color);
            ++y;
        }
        --x;
    }
}
/* 采用扫描线种子多边形填充算法 */
Flood __ fill(int x,int y,unsigned char bc,unsigned char color)
{
    int xl,xr,i;
    i=x;
    while(Get __ pixel(i++,y)!=bc);
    xr=i-2;i=x;
    while(Get __ pixel(i--,y)!=bc);
    xl=i+2;
    for(i=xr;i>=xl;i--)Put __ pixel(i,y,color);
    for(i=xr;i>xl;i--){
        if((Get __ pixel(i,y+1)!=color)&&(Get __ pixel(i,y+1)!=bc))
            Flood __ fill(i,y+1,bc,color);
        if((Get __ pixel(i,y-1)!=color)&&(Get __ pixel(i,y-1)!=bc))
            Flood __ fill(i,y-1,bc,color);
    }
}

```

SKILL

最少的投资
最佳的方案
最多的组合

挂接式打印机共享器

北京向宇计算机公司

地址:北京复兴路甲20号27分号312,314

邮编:100036 电话:8263441,2063366呼1511

☐任劲松

程

本文中 Title1p 就是一个动画封面的例子,该程序会在屏幕上画出一个大型图案“R”,并会从头到尾层层变色。

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

void title(char * sProgramName, char * sVersion, char
          * sYear)
{ char chr[2], lights[38 * 2], color = 10;
  int cordx, cordy, lightpos = 39; textmode(C80);
  textbackground(MAGENTA); clrscr();
  __setcursortype(__NOCURSOR);
  textcolor(WHITE); gotoxy(43, 5);
  printf(sProgramName); gotoxy(43, 7);
  printf("Version "); printf(sVersion);
  textcolor(LIGHTRED); gotoxy(43, 9);
  printf("Written by Comi Ren"); gotoxy(43, 11);
  printf("(C) Copyright Softgenius "); printf(sYear);
  gotoxy(39, 18); textcolor(YELLOW);
  printf(" Welcome to the world of Softgenius");
  textcolor(LIGHTRED); //print lights; gotoxy(39, 17);
  printf(" * * * * * "); gotoxy(39, 19);
  printf(" * * * * * ");
  gettext(39, 17, 76, 17, lights); gotoxy(39, 18);
  printf(" * "); gotoxy(76, 18); printf(" * ");
  gotoxy(25, 22); textcolor(LIGHTGREEN + BLINK);
  printf("Press any key to continue...");
  gotoxy(1, 2); textcolor(MAGENTA);
  printf("          \\\\\\\\\\\\\\\\\\n\\r"); //char(220).
```

```

cprintf("           \\n\\r");
cprintf("         \\n\\r");
cprintf("       \\n\\r");
cprintf("     \\n\\r");
cprintf("   \\n\\r");
cprintf(" \\n\\r");
cprintf(" \\n\\r");
cprintf(" \\n\\r");
cprintf(" \\n\\r");
cprintf(" \\n\\r");
cprintf(" \\n\\r");
cprintf(" \\n\\r");
cprintf(" \\n\\r");
cprintf(" \\n\\r");
cprintf(" \\n\\r");
cprintf(" \\n\\r");
cprintf(" \\n\\r");
cprintf(" \\n\\r");
cprintf(" \\n\\r");
cprintf(" \\n\\r");
for (;) { if ((color += 4) > 15) color -= 15;
for (cordy = 2; cordy < 22; cordy++) {
for (cordx = 1; cordx < 80; cordx++) {
gettext(cordx, cordy, cordx, cordy, chr);
if (chr[0] == '\\' ) {chr[1] = chr[1] & 0xf0 | color;
puttext(cordx, cordy, cordx, cordy, chr);}
} if (++lightpos > 40) lightpos = 38;
puttext(lightpos, 17, lightpos + 37, 17, lights);
puttext(78 - lightpos, 19, 115 - lightpos, 19, lights);
delay(200); if (bioskey(1)) //quit.
{bioskey(0); textbackground(BLACK);
textcolor(LIGHTGRAY);
for (color = 0; color < 5; color++){
for (cordy = 1; cordy <= 25; cordy+=5){
gotoxy(1, cordy + color); clrscr(); } delay(50); }
gotoxy(1, 1);
textmode ( LASTMODE ); __ setcursortype ( __
NORMALCURSOR); return; } } } void main()
{ title("title1", "1.0", "1994.8"); printf("Test end!\\n");
getch(); }
```

SKILL

基于图形环境下图形及汉字立体投影通用菜单的实现

□ 李晓华

最

近,以图形用户界面为主导的人机交互技术正迅猛发展,计算机的使用、操作对用户来说更容易、更友好。新的思想、新的概念、新的系统正不断涌现。这些新的界面对用户来说是友好的,而对编程来说,要设计出具有图标或图形的用户界面,需要掌握新的知识,要对图形学有一个更深刻的认识,只有这样才能设计出完美的系统。

当前,对 GUI 还没有一个精确的定义,不过一般 GUI 应具有这样一些特点:一是,必须具备位映象图形技术,用户能在屏幕上实现有关图形(图象)的操纵。通过图形取代传统的字符显示方式,使用户易学、易用,操作直观、方便;二是,通过其他点设备(如鼠标、光笔等)来完成对系统的操作(通常称为点\按式图形用户界面);三是,允许通过用户改变系统某些状态,如移动/窗口的大小等;四是,具有良好的在线式联屏帮助系统,以便用户能随时了解当前操作系统的功能。

在图形用户界面中,其图形菜单是最重要的。因为它是计算机与用户之间的综合操作环境。因而设计出友好的图形菜单是图形用户界面最基本的也是最关键的。本文就是依据这样的思想,用 Turbo C 在中文(Super CCDOS)环境下实现图形汉字立体投影通用菜单的。

1. 图形环境下汉字的显示

利用 Turbo C 中的图形函数实现汉字菜单,传统的方法是编写汉字显示程序,其方法是:读取汉字字库(如16X16点阵的 CLIB 或24X24点阵 CLIB24字库)中的字模点阵,

并通过 putpixel(x,y,color)函数来实现汉字显示。这种显示方式有一个缺陷,就是必须依赖具体的汉字字库,而不是中文环境,通用性差。由于它要从字库中读取点阵,因而显示速度较慢。为了克服这一不足,本文用 BIOS 的 INT 09H 号功能实现汉字的显示。实践证明,该方法在图形环境下实现汉字的显示是可取的。

2. 中文环境下图形与汉字共存

为了能在中文环境下实现位映象图形与汉字字符共存,需要对 Turbo C 中的 outtext(char * textstring) 和 outtext(int x,int y, char far * textstring) 函数进行改造,编写新的显示函数以实现汉字的输出。本文采用以下方法实现图文显示:增加图形环境下显示汉字的函数 printa(char * textstring[],int att,int x,int y),该函数调用 INT 10H 09号功能。其中 textstring 为输出的汉字字符的地址,att 为字符属性(包括字符前景色和底色),x,y 是输出字符的位置(x=0-24,y=0-79)。有关图形的操作,以像素为基本单位(x=0-639,y=0-479)。

3. 图形环境下汉字菜单的实现

本程序采用图文并茂的形式实现菜单以增加用户的直观感。用户通过观察图形符号便知道所选择的项。通常的菜单有三种形式:光条菜单、下拉式菜单、弹出式菜单。

在这三种菜单中,可以加入图形功能,以便实现图形环境下保持原有菜单的功能。传统的窗口菜单在画方框(窗口边沿)时,采用的是全中文字符,占用了一行汉字的位置,而在图

图形环境下的图文并茂菜单,采用象素点实现画方框,不需要占用汉字字符的位置。在本方法中,除画点是在图形环境下实现的外,菜单中的图形也是以象素为单位实现的,如图1所示:

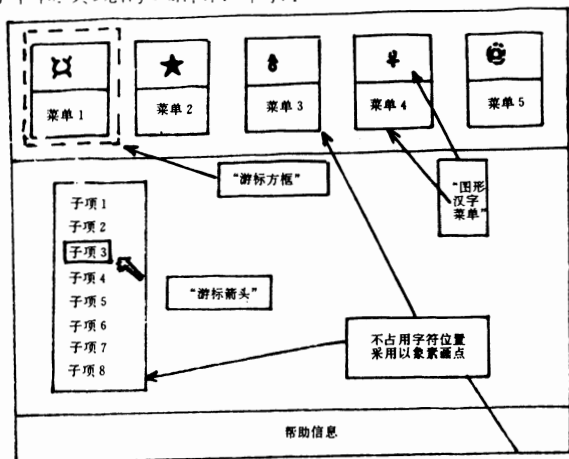


图1 图形环境下菜单实现的例子

该菜单的指示器是通过一个“游标方框”取代传统的光标或光条。当选中一项时(按 Alt+X),便弹出一个子菜单。在子菜单中通过一个“游标箭头”指示选择子项。其中画框是用图象功能,即不占用字符位置。

在程序中 int sub __menu(int x,int y,char * m[],int vide __x,int vide __y) 函数实现通用立体投影汉字子菜单的功能。其中,x、y 是汉字字符显示的起始位置,* m[] 是子菜单的起始地址,vide __x、vide __y 以象素为单位画框的起始位置。为实现函数的通用性,本文采用了设置窗口函数、保存图象函数等。

(1)用以下函数实现弹出之前屏幕内容的保存: 进入时保存:

```
setviewport(vide __x,vide __y,vide __x+70,vide __y+155,1);
b0=malloc(imagesize(0,0,70,155));
getimage(0,0,70,155,b0);
```

退出时恢复:

```
setviewport(vide __x,vide __y,vide __x+70,vide __y+155,1);
putimage(0,0,b0,COPY_PUT);
```

(2)用以下函数实现通用性和立体投影:

```
setviewport(vide __x+10,vide __y+10,vide __x+70,vide __y+155,1);
clearviewport();
setfillstyle(SOLID_FILL,YELLOW);
setviewport(vide __x,vide __y,vide __x+60,vide __y+145,1);
bar3d(0,0,60,145,0,0);
```

源程序清单如下:

/* 程序名:grapmenu.c

功能:基于图形环境下图标及汉字立体投影通用菜单的实现

编程语言:Turbo C2.0

运行环境:Super CCDOS 汉字系统

函数:

main()——主程序指定一个主菜单 * f 和5个子菜单: * m1 到 m5
 arrow()——画游标箭头
 box()——画游标方框
 graph __ f()——画图形1
 graph __ f()——画图形2
 printa()——汉字显示
 sub __ menu()——通用立体投影汉字菜单 *

```
#include "stdlib.h"
#include "stdio.h"
#include "graphics.h"
#include "fcntl.h"
#include "conio.h"
#include "dos.h"
#include "math.h"
FILE *fp;
void printa(char *,int,int,int);
void graph __ f();
void graph __ f1();
void star();
int sub __ menu(int,int,char * m[],int,int);
void * Rar, * Rarl, * Rar2;
void * b;
main()
{int v,i;long length;unsigned char by[33];int gmode,gdriver;
unsigned char * f[]={"菜单一","菜单二","菜单三","菜单一"
"菜单一"};
unsigned char * m1[]={"装载","保存","写入","目录","变目",
"专退","退出"};
unsigned char * m2[]={"装载","保存","写入","目录","变目",
"专退","退出"};
unsigned char * m3[]={"装载","保存","写入","目录","变目",
"专退","退出"};
unsigned char * m4[]={"装载","保存","写入","目录","变目",
"专退","退出"};
unsigned char * m5[]={"装载","保存","写入","目录","变目",
"专退","退出"};
detectgraph(&gdriver,&gmode);
if(gdriver<0){
printf("there is not graphics displayer \n");
exit(1);}
/* printf("deteet graphics driver is %d,mode is %d\n",
gdriver,gmode);
getch(); */
initgraph(&gdriver,&gmode,"c:\\tc\\bgi");
setwritemode(1);arrow();box();
graph __ f();graph __ f1();cleardevice();
setcolor(15);setfillstyle(SOLID_FILL,37);
bar3d(0,0,639,465,0,0);setcolor(WHITE);
line(1,46,639,46);line(1,80,639,80);setbkcolor(3);
putimage(60,8,Rar1,XOR_PUT);
putimage(180,8,Rar2,XOR_PUT);
for(i=0;i<=5;i++){
printa(f[i],0x34,3,i*15+7);
```

```

i=0;
putimage(50,4,b,XOR_PUT);
while((v=specialkey())!=45)
switch(v){
case 80:
    switch(i){
    case 0:
        sub__menu(3,i*15+7,m1,50+i*120,47);
        break;
    case 1:
        sub__menu(3,i*15+7,m2,50+i*120,47);
        break;
    case 2:
        sub__menu(3,i*15+7,m3,50+i*120,47);
        break;
    case 3:
        sub__menu(3,i*15+7,m1,50+i*120,47);
        break;
    case 4:
        sub__menu(3,i*15+7,m1,50+i*120,47);
        break;
    }
setviewport(0,0,630,479,1);
break;
case 75:
    putimage(50+i*120,4,n,XOR_PUT);
    i=(i<=0)?4:(i-1);
    putimage(50+i*120,4,b,XOR_PUT);
    break;
case 77:
    putimage(50+i*120,4,b,XOR_PUT);
    i=(i>=4)?0:(i+1);
    putimage(50+i*120,4,b,XOR_PUT);
    break;
    }
clesograph();
}
int specialkey(void)
{
int key;
while(bioskey(1)==0);
key=bioskey(0);
key=key&0xff?0:key>>8;
return(key);
}
int getbit(unsigned char c,int n)
{
return((c>>n)&1);
}
int box()
{
setcolor(5);
b=malloc(imagesize(1,1,59,72));
setfillstyle(XHATCH_FILL,33);
bar3d(1,1,59,72,0,0);
getimage(1,1,59,72,b);
}

```

```

int arrow()
{
int size;
int raw[]={0,0,0,4,2,4,10,12,12,10,4,2,4,0,0,0};
setfillstyle(SOLID_FILL,34);
fillpoly(8,raw);
size=imagesize(0,0,12,12);
Rar=malloc(size);
getimage(0,0,12,12,Rar);
return;
}
void printa(s,a,r,c)
char *s;
int a,r,c;
{
union REGS our__regs,write__regs;
while(*s)
{
our__regs.h.ah=2;
our__regs.h.dh=r;
our__regs.h.bh=0;
our__regs.h.dl=c++;
int86(0x10,&our__regs,&our__regs);
write__regs.h.ah=9;
write__regs.h.bh=0;
write__regs.h.cx=1;
write__regs.h.bl=a;
write__regs.h.al=*s++;
int86(0x10,&write__regs,&write__regs);
}
}
int sub__menu(int x,int y,char *m[],int vide__x,int vide__y)
{
int v;
int i=0;
void *b0;
setviewport(vide__x,Vide__y,Vide__x+70,Vide__y+155,1);
b0=malloc(imagesize(0,0,70,155));
getimage(0,0,70,155,b0);
setviewport(vide__x+10,vide__y+10,vide__x+70,vide__y+155,1);
clearviewport();
setfillstyle(SOLID_FILL,YELLOW);
setviewport(vide__x,vide__y,vide__x+60,vide__y+145,1);
bar3d(0,0,60,145,0,0);
for(i=0;i<7;i++)
    printa(m[i],110,i+x,y);
i=0;
outtextxy(600,200,'');
putimage(45,12,Rar,XOR_PUT);
while(v!=45){
v=specialkey();
switch(v){
case 72:
    putimage(45,12+i*18,Rar,XOR_PUT);
}
}
}

```

(下转第51页)

直接在西文 DOS 下输入汉字的方法

□李山林 杨朝霞

对

一般的程序员来说,不借助中文环境,在西文 DOS 环境下显示或打印汉字已并非难事,只要给他一个标准的16点阵或24点阵汉字库,他就可以随心所欲了。关于这方面的文章,很多杂志都有详细的说明,然而,翻了很多计算机方面的刊物,都找不到关于西文 DOS 环境下汉字输入的介绍,即使是经验丰富的软件开发人员,当他开发的软件在运行过程中必须要输入一些中文信息时都不得不借助于并受制于 DOS 中文系统。

受制于中文 DOS 系统的原因有三:第一,它不是占用你本来并不富裕的常规内存就是占用你的扩展内存,这大大削减你可利用的空间。第二,中文系统还存在一个兼容性的问题,或者不能与你已有的软件完全兼容,造成死机,或者破坏你界面的显示。第三,也是所有中文系统不可避免的一条,显示速度变慢,以 DOS 中 TYPE 命令显示一个文件所需时间为例,它比在西文 DOS 环境下执行同样的命令慢得多。汉字的输入比汉字的输出(即显示、打印)复杂得多,一般人对汉字输入的原理不甚了解,然而了解其原理的人又缺少一种“输入字典”。这个字典的作用就是将键盘上的二十六个英文字母与六千七百多个汉字对应起来。因为对应的方式很多,所以输入的方法也有上百种,大致可分为三类:1. 利用拼音对应的输入法,如:拼音输入法;2. 利用字型笔划对应的输入法如:五笔字型法、五十字元法等;3. 利用音形结合对应的输入法。建立“输入字典”是一项非常繁重的劳动,它令很多优秀的软件开发人员望而却步。

目前,汉字输入的方式多是采用“查字典匹配”的方法,很类似查新华字典,新华字典中汉字的顺序是根据其读音从小到大排列的,读音相同的汉字,放在一起,根据以上规则,可以很容易查到所需要的汉字。“输入字典”也是按照一定的顺序将汉字排列起来,以拼音输入法所使用的“拼音字典”为例:汉语拼音共有402个,首先将这402个拼音从小到大排列成一个拼音表,然后将读音相同的汉字放在一起形成一组(一般将常用字放在前面,输入时击键次数最少),于是每个拼音对应一组读音相同的汉字,共有402组汉字,最后将每组汉字也按拼音顺序作相应的排列,在拼音与汉字组之间建立一个索引,通过索引就可以找到它所对应的汉字组。

举一个实例,还是以拼音输入法为例,如果要输入汉字“爱”,首先找出“ai”在拼音表中的排列序号,然后,根据序号找出它所对应的索引,最后通过索引将读音为“ai”的汉字组挑出来,共有22个汉字:“爱”,“埃”,“碍”,“矮”,……,“霭”,这一部分的工作由程序员完成,然后程序将用户挑选的汉字记录下来形成汉字串,这就是汉字输入的原理和方法。

方法有了,那么“字典”又从何而来呢?如果自己建“输入字典”,工作量太大,既费时又耗力。笔者经过摸索,利用 UC DOS 3.0 的“输入字典”,开发出不依赖任何中文系统,直接在西文 DOS 环境下输入汉字的软件。

现在,比较流行的中文系统 UC DOS 3.0 有三种基本输入法:1. 拼音输入法,2. 简拼输入法,3. 双拼输入法。这三种输入法的“字典”

名分别为:py.ovr、jp.ovr 和 sp.ovr。只要分析出“字典”文件的格式,就能为我所用了。现以拼音字典文件 py.ovr 为例,在字典文件 py.ovr 中,共有402个拼音,每个拼音占用6个字节(汉语拼音最多只有6个字母,如“zhuang”),不足6个的以空格补全,汉字由内码表示(有关汉字内码、区位码以及顺序码方面的知识请参见有关文章的介绍,这里不再赘述),每个内码占用2个字节,文件 py.ovr 由四个部分组成:

第一部分为文件头,占32个字节,从0到0x20(0x表示为十六进制)。

第二部分为由402个拼音构成的拼音表,从0x20到0x98C 共占2412=402*6个字节,每个拼音占用6个字节,用 DEBUG 命令就可以一目了然:

```
C:\>debug py.ovr
-d 100
0100 55 43 44 4F 53 20 4F 56-52 20 46 49 4C 45 1A 00 UCDOS OVR FILE..
0110 92 01 00 00 6c 09 90 0C-12 41 36 44 00 00 00 00 ....1....A6D....
0120 61 20 20 20 20 20 61 69-20 20 20 20 61 6E 20 20 a ai an
0130 20 20 20 61 6E 67 20 20 20-61 6F 20 20 20 20 62 61 ang ao ba
0140 20 20 20 20 20 62 61 69 20-20 20 20 20 62 61 6E 20 20 bai ban
```

第三部分为索引地址,从0x98C 到0xCB0共402*2=804个字节:

```
C:\>debug py.ovr
-d A80
0A80 7A 75 6E 20 20 20 7A 75-6F 20 20 20 9C 0C C8 0C zun zuo ....
0A90 EC 0C F2 0C 20 0D 54 0D-6A 0D 94 0D B0 0D E0 0D ...T.j...
```

每个拼音对应一个索引地址,每个索引地址占2个字节,索引地址+0x20为汉字组在文件中的截止地址。具体的算法如下:

$\text{LastOffsetAddress}[i] = \text{IndexAddress}[i] + 0x20$

第四部分为汉字内码,从0xCB0到0x4132。

由第三部分所述,不难看出第*i*个拼音所对应的汉字组在文件中的起始地址为:

$\text{LastoffsetAddress}[i-1]$ 。这组汉字的字节数为(如果*i*=0, $\text{LastOffsetAddress}[-1] = 0xCB0$):

$\text{LastOffsetAddress}[i] - \text{LastOffsetAddress}[i-1]$

仍然以拼音“ai”为例,起始地址 $\text{LastOffsetAddress}[0] = 0xCB0$ 汉字组的字节数为:

$\text{LastOffsetAddress}[1] - \text{LastOffsetAddress}[0] = 0x0CE8 - 0xCB0 = 44 = 22 * 2$ 这22个内码就是上面所述22个汉字的内码,如下所示:

```
C:\>debug py.ovr
-d 0DB0
:0DB0 B0 AE B0 A3 .....
:0DC0 B0 AD B0 AB B0 A4 B0 A6-B0 AC B0 A5 B0 AF B0 A7 .....
:0DD0 B0 A9 B0 A8 B0 AA DE DF-E0 C8 E0 C9 E6 C8 E8 .....
:0DE0 EA D3 ED C1 EF CD F6 B0-B0 B2 B0 B4 B0 B5 B0 B6 .....
:0DF0 B0 B8 B0 B3 B0 B1 B0 B7-B0 B0 DA CF DB FB DE EE .....
```

其他两个字典 sp.ovr, jp.ovr 的结构与 py.ovr 类似,有了输入字典,直接在西文 DOS 下输入汉字比你想象的要容易得多,笔者提供的 C 语言源程序主要完成如下的工作:输入拼音,程序输出它所对应的汉字组内码,后面的功能由读者自己实现。源程序在 Borland C++ 3.1 环境下调试通过,在 UC-DOS3.0 下运行。

源程序清单如下:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define No -1
unsigned int FirstAddress=0xcb0;
void main(void)
{
    unsigned int LastOffsetAddress[402],IndexAddress[402];
    char Table[402][6],InStr[100],sTable[10];
    int size,Find,cLen,compare,i,j;
    unsigned char *cStr;
    long address;
    FILE *fp;
    fp=fopen("py.ovr","rb");
    if (!fp)
    {
        printf("\nCannot open the file \"py.ovr\"");
        exit(1);
    }
    fseek(fp,0x20l,SEEK__SET);
    fread(Table,sizeof(char),402*6,fp);

    /* 读出402个拼音放入拼音表 Table[] 中 */
    fread(IndexAddress,sizeof(int),402,fp);

    /* 将402个索引地址放入 IndexAddress[i]+0x20 */
    for (i=0;i<402;i++)
        LastOffsetAddress[i]=IndexAddress[i]+0x20;

    /* 由索引 IndexAddress[]算出偏移地址 LastOffsetAddress[] */
    printf("\nPlease Input pinyin string:\n");
    scanf("%s",InStr);
    size=strlen(InStr);
    if (size>6)
    {
        putchar(7);
        printf("\nError about pinyin string");
        exit(0);
    }
    for (i=size;i<6;i++)
        InStr[i]=0x20;
    Find=No;
    InStr[6]='\0';
    for (i=0;i<402;i++)
```

(下转第51页)

西文环境下汉字放大显示技术的实现

□ 李晓华

本

文分析了西文环境下汉字放大显示的方法。讨论了如何从字库中提取汉字点阵字模,并提出实现汉字显示的两种方法,以达到汉字放大的目的。

一、引言

汉字系统作为我国使用微机的支持环境,正越来越受到重视。目前除长城公司开发的 CCDOS(CCLIB CCLib24等)外,还有比较流行的“金山汉字”、UCDOS、“明星中文”、“王码汉字”系统等。这些中文系统,为我国使用计算机带来了极大的方便。但由于汉字环境占用内存空间较大,对于档次较低的计算机来说,由于内存空间有限,因此使用这些系统有一定的困难。而且有些应用系统基本上不具备开发汉字显示的条件。如在工控系统中,只要汉字菜单显示,不需要中文环境支持,汉字的显示独立于中文操作系统,应用系统可以在西文 DOS 下工作,又如制作电视广告词时需要使用较美观的字体,包括字体大、小的控制,字体颜色可选等。为此本文设计出能满足上述要求的程序。

二、如何从字库中读取汉字点阵字模

汉字字符的显示实质上是将字符点阵映射到 CRT 上,为此可以从标准或非标准的汉字字库中提取点阵字模,以实现汉字的显示。在点阵字库中,每一字节为8点。根据点阵的不同,每个汉字占用的字节数是不相同的,如16*16点阵汉字,占用32字节。24*24点阵的汉字占用72字节。为计算出汉字在字库中

的位置,采用下列方法:一个汉字由双字节组成,其基本结构为第一字节由区码加0AH,第二字节由位码加0AH组成。汉字字库的存放,若某一汉字的第一字节为C1,第2字节为C2,那么该区汉字的区码为QM,位码为WM,则汉字在字库中的记录号为REC:

$$QM = C1 - 0AH$$

$$WM = C2 - 0AH$$

$$REC = (QM - 1) * 94 + WM - 1$$

由于汉字系统的不同,记录号是不同的。下面是几种较流行的汉字系统:

1. 金山汉字系统

从金山汉字系统中的压缩字库 XSDOS.LPH 读取模点阵:

16*16点阵字模在字库中的记录:

$$REC = (QM - 18) * 94 + WM - 1 + 188L + 27072L$$

24*24点阵字模在字库中的记录:

$$REC = (QM - 18) * 94 + WM - 1 + 188L + 3045690L$$

其中:REC 是汉字在字库中的记录号, QM 是汉字的区码,WM 是汉字的位码

2. 长城汉字系统 CCDOS

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

从长城汉字系统的 CCLIB 字库中读取字模点阵:

CCLIB 16 * 16点阵字模在字库中的记录:

$REC = (QM - 18) * 94 + WM - 659L$

CCLIB 24 * 24点阵字模在字库中的记录:

$REC = (QM - 18) * 94 + WM - 847L$

其中:REC 是汉字在字库中的记录号,QM 是汉字的区码,WM 是汉字的位码。

3. 213汉字系统

从213汉字系统的 HZK16中读取16 * 16点阵字模:

$REC = (QM - 18) * 94 + WM - 1$

24 * 24点阵字模(HZK24H/HZK24S/HZK24F/HZK24K)字库中的记录:

$REC = (QM - 18) * 94 + WM - 1 + 188L$

其中:REC 是汉字在字库中的记录号,QM 是汉字的区码,WM 是汉字的位码。

知道了点阵字模的读取方法后,就可以根据点阵字模显示汉字了。

三、西文环境下汉字放大显示

1. 实现汉字放大显示的方法

在西文环境下实现汉字放大的基本思想是:在应用程序中先按汉字的区位码计算该汉字在字库中的记录号,根据记录号从标准字库或非标准字库中提取汉字点阵字模。最后把汉字点阵的字模送到显示器上。根据需要可在送点阵字模时,实现汉字的放大、缩小、转置、前景色和背景色的调整。其在CRT上实现显示字模点阵信息的处理,可采用以下两种方法来实现:

(1)直接向显示缓冲区填写汉字点阵

大家知道,EGA 显示卡的显示缓冲区的起始地址为 A000H,四个位面分别是红、蓝、绿和亮度,CRT 上的每一个实际位点对应于四个位面的相应位,它可从64种颜色中同时选16种颜色,它的16种颜色的组合方式是由4个位面决定一个点。因而可以从汉字库中提取点阵来放入显示缓冲区,以达到汉字显示的目的。为实现汉字的放大和缩小,可以对汉字点阵的每一字节进行处理。通常16 * 16点阵的汉字为32字节,每一行由2字节组成($2 \times 8 = 16$ 点)。24 * 24汉字点阵为72字节,每一列由3字节组成($3 \times 8 = 24$ 点)。下面是用 Turbo C2.0编写的24 * 24点阵汉字的显示程序(金山汉字库 XSDOS.LPH):

```
..
outport(0x3c4,2);
outport(0x3c5,port);          //port 面位
ort __p=(char far *)0xa000000;
```

```
fp=fopen("c:\xsdos.lph","r");    //打开字库文件
REC=(QM-18)*94+WM-1+188L+304560L;
//根据区位码计算记录号
REC=REC*72;
FSEEK(fp,REC,0);//调整指针到该汉字的位置
fread(fp,buffer,72); //取点阵字模到数组中
for(i1=0;i1<3;i1+1){
for(i2=0;i2<8;i2+1){
for(i3=0;i3<24;i3+1){
if(getbit(buffer[i3*3+i],7-12))
*crt __p=buffer[i2*2+i2];
//把点阵放入 EAG 显示缓冲区
ort __p+1;    //调整下一点的位置
}
}
}
..
```

若要对汉字进行放大处理,则只需对每字节按横向或纵向处理就行了。

(2)利用画点、画线、画园等函数显示汉字点阵

在 Turbo C 2.0中提供了较丰富的图形函数,如画点函数 putpixel(int x,int y,int pixelcolor);画填充椭圆 fillellipse(int x,int y,int xradius,int yradius),以(x,y)为中心,用当前填充色和填充方式画椭圆,边线为当前色。利用这些图形函数,就可根据汉字点阵实现在西文环境下显示汉字了。

2. 具体实例

本程序实现了在西文环境下放大显示汉字,该程序的设计思想是:使用者根据自己提供的文件,并指定放大(纵向、横向)倍数、背景色和字符色在CRT上显示汉字。否则程序按默认值设定。

程序说明:ROWFD、COLFD 分别为字间距和行间距。fdx、fdy 分别为横向和纵向放大倍数。

源程序清单如下:

```
/* (1)typcfd.e
(2)功能:西文状态下放大显示汉字
(3)使用格式
typedef info1 info2 info3 info4
info1=filename 被显示文件名
info2=FDX      X 放大倍数
info3=FDY      Y 放大倍数
info4=COLOR __back 背景色
```

最少投资

最佳方案

最多组合

挂接式
打印机共享器

北京向宇计算机公司
地址:北京复兴路甲20号27分号312,314
邮编:100036 电话:8263441,2063366呼1511

```

info5=COLOR __cahr 字符色 */
#include "stdio.h"
#include "graphics.h"
#include "math.h"
#include "string.h"
#include "stdlib.h"
#define ROWFD 27
#define COLFD 27
main(int argc,char *argv[])
{
FILE *fp1;
FILE *fp;
unsigned int c,c1,c2,f 0;
int i1,i2,i3;
float x,y;
int x1,y1,fdx,fdy,h;
long longh;
unsigned char by[73];
int gmode,gdriver;
unsigned char filename[11];
int color __back,color __char;

detectgraph(&gdriver,&gmode);
initgraph(&gdriver,&gmode,"c:\\tc\\bgi");

fp1=fopen("c:\\XSDOS.LPH","rb");
fp=fopen(argv[1],"r");
if(fp1==NULL||fp==NULL){
printf("Can't open %s or c:\\hzk24\\n",argv[1]);
exit(1);
}

if(argc>1) strcpy(filename,argv[1]);
else{
printf("format:TYPEE filename info2 info3 info4 info5");
exit(1);
}
h=4;
if(argc>3){
fdx=atoi(argv[2]);
fdy=atoi(argv[3]);color __back=atoi(argv[4]);
color __char=atoi(argv[5]);
}
else{
fdx=3;
fdy=3;
color __back=12;
color __char=14;
}
x1=0;
y1=0;
setfillstyle(SOLID __FILL,color __back);
bar3d(0.0,639,479,0.0);
setcolor(color __char);
setfillstyle(SOLID __FILL,color __char);

while(c __getc(fp))!=EOF {

```

```

if(c=='\n') {x1=0;y1+=1;}
if(c==0x20) {x1+=1;}
if(c!=0xa0)
if(f==0) {
c1=c-0xa0;
f++;}
else{
c2=c-0xa0;
length=(c1-18)*94+c2-1+188;
f=0;
length=length*72+304560L;
fseek(fp1,length,SEEK __SET);
fread(by,1,72,fp1);

if((x1*ROWFD)*fdx>getmaxx()){
x1=0;
y1+=1;}
if((8*h+y1*COLFD)*fdy>getmaxy()){
getch();
y1=0;
clearviewport();
setfillstyle(SOLID __FILL,color __back);
bar3d(0.0,639,479,0.0);
setcolor(color __char);
setfillstyle(SOLID __FILL,color __char);}
x=x1*ROWFD;
y=y1*COLFD;
for(i2=0;i2<3;i2++){
for(i3=0;i3<8;i3++){
for(i1=0;i1<24;i1++){
if(getbit(by[i]*3+i2,7-i3))
fillellipse((x+i1)*fdx,(y+i2*8+i3)*fdy,2,2);
/* putpixel((x+i1)*fdx,(y+i2*8+i3)*fdy,14); */
}
x1+=1;
}
}
getch();
fclose(fp1);
fclose(fp);
}
int getbit(unsigned char c,int n)
{
return((c>>n)&1);
}

```

SKILL

<p>最少的投资</p> <p>最佳的方案</p> <p>最多的组合</p>	<p>挂接式</p> <p>打印机共享器</p>
<p>北京向宇计算机公司</p> <p>地址:北京复兴路甲20号27分号312,314</p> <p>邮编:100036 电话:8263441,2063366呼1511</p>	

西文状态下汉字的多字体彩色立体显示

□李 辉

在

西文状态下显示汉字最简单、最常用的方法是：用汉字字处理软件等工具（如 Word Star、CCED 或 WPS 等）将要显示的汉字串作为普通字符串写入源程序中，再编译成可执行文件。使用这种方法有以下几个缺陷：

1. 程序的运行离不开汉字环境，影响程序的运行速度；
2. 显示方式单一，无法进行旋转、放大等操作，字型显示单一，无法同时显示多种字体，也不能繁简混合显示，所以远远不能满足程序的需要；
3. 画面固定，不能产生动画的效果；
4. 无法将所作的汉字画面绘制到打印机上。

本文旨在介绍一种不需要汉字系统支持，在西文状态下能以多种变换方式显示多种字体的汉字，繁简能同时显示，并可对汉字进行旋转、放大等操作。而且还可进行汉字动态移动，产生动画的效果，亦可随时将作出的屏幕在打印机上绘制出来。

本文提供的程序在 IBM 及其兼容机、四通 286、Olivetti、浪潮 0540 等机器上运行通过。

一、实现方法

利用 2.13H 提供的各种字库，其具体内容如下表所示。

1. 在 16 点阵字库 (HZK16) 中，存有 1 区至 87 区的所有汉字及图形字模，每个汉字及图形字模占 32 个字节，其点阵信息按行存放。其具体排列如图 1 所示：

HZK16	16点阵字库	01—87区，每区94汉字，每字32字节
HZK24S	24点阵宋体字库	16—87区，每区94汉字，每字72字节
HZK24F	24点阵仿宋体字库	16—87区，每区94汉字，每字72字节
HZK24H	24点阵黑体字库	16—87区，每区94汉字，每字72字节
HZK24K	24点阵楷体字库	16—87区，每区94汉字，每字72字节
HZK24E	24点阵繁体字库	16—87区，每区94汉字，每字72字节
HZK24T	24点阵图形字库	16—87区，每区94汉字，每字72字节

16列	
字节1	字节2
b7..b0	b7..b0
...	...
16行	...
...	...
b7..b0	b7..b0
字节31	字节32

图1 16点阵信息排列图

因此，16点阵字库中每个汉字或图形符号字模的首字节可通过下式定位：

$$\text{首地址} = ((\text{区号} - 1) * 94 + \text{位号} - 1) * 32$$

其中：

区号 = 汉字或图形符号国际码的高字节值 - 0xa0

位号 = 汉字或图形符号国际码的

低字节值-0xa0

2. 在24点阵字库(HZK24S、HZK24F、HZK24H、HZK24K及HZK24E)中,存有从第16区到87区的所有汉字的字模,而01区到15区的图形符号的字模则存放于图形字库(HZK24T)中,每个汉字及图形符号的字模占72个字节,其点阵信息按列存放。其具体排列如图2所示:

		24列	
	字	b7.....b7	字
	节	节
	1	b0.....b7	70
	字	b7.....b7	字
	节	节
24行	2	b0.....b0	71
	字	b7.....b7	字
	节	节
	3	b0.....b0	72

图2 24点阵信息排列图

因此,24点阵字库中每个汉字或图形符号字模的首字节可通过下式定位:

当区号大于或等于16时,首地址=((区号-16)*94+位号-1)*72

当区号小于16时,首地址=((区号-1)*94+位号-1)*72

其中:

区号=汉字或图形符号国际码的高字节值-0xa0

位号=汉字或图形符号国际码的低字节值-0xa0

3. 对点阵信息的处理

(1)对16点阵信息的处理

根据构成16点阵汉字字模的32个字节,采用两重循环和一条判断语句,外层循环控制列,内层循环控制所取的字节,而判断语句则用于按该字节的每位是“1”(或“0”)来决定是否在屏幕上的对应位置写点(或不写点),这样就能在屏幕上的指定位置形成一个16×16点阵的汉字。同时我们可以利用一些技巧来显示出变化多样的汉字。

(2)对24点阵信息的处理

根据构成24点阵汉字字模的72个字节,采用两重循环和一条判断语句,外层循环控制行,内层循环控制所取的字节,而判断语句则用于按该字节的每位是“1”(或“0”)来决定是否在屏幕上的对应位置写点(或不写点),这样就能在屏幕上的指定位置形成一个24×24点阵的汉字。并可根据情况打开不同的字

库来显示不同字型的汉字。

4. 对形成图形的打印

由于C语言没有提供打印屏幕图形的命令,为此利用它提供的getpixel()函数,根据屏幕上的某点是否有图象点,来决定是否在打印机打印出该对应点来,从而打印出屏幕上某指定位置的图形来。或者通过修改DOS的INT 05H中断并使之常驻内存,从而随时可以通过调用INT 05H中断来打印出图形来。

5. 汉字显示多样化

汉字在放大和缩小、旋转、倾斜、动画技术、立体汉字及美术字屏幕的实现,分别见后面的函数说明。利用这些基本函数我们可以设计出非常漂亮的汉字图形,并且可以随时打印出来。

二、程序使用说明

由于16点阵与24点阵的各个函数基本相同,故主要以24点阵为例来说明。

本文提供了几个基本函数,利用这些基本函数,你可以设计出非常好看的屏幕来。程序DEMO.C是为了说明其用法,提供的一个例子。

基本函数的具体用法如下:

disp16_0(int x0,int y0,int color,char *w):

将汉字W(16点阵)用color所指定的颜色在屏幕的(x0,y0)处显示出来;

disp24_0(int x0,int y0,int color,char *w):

将汉字W(24点阵)用color所指定的颜色在屏幕的(x0,y0)处显示出来;

disp24_1(int x0,int y0,int k,int kx,char *w):将汉字W倾斜显示,K为放大倍数,kx为显示顺序;

disp24_2(int x0,int y0,int kx,int ky,char *w):将汉字W放大或缩小显示,kx,ky为水平和垂直方向的放大倍数。

disp24_3(int x0,int y0,float k,char *w):将汉字W左右错切,k为错切系数;

disp24_4(int x0,int y0,float k,char *w):将汉字w上下错切,k为错切系数;

disp24_5(int x0,int y0,float a,char *w):将汉字w以a角度进行旋转;

w24read(char aa[],char n,char zt):将数组aa中的每个汉字按指定字体的字模读到w24[72]中。

演示用函数:

disp_s1():在屏幕上方显示环状字符串“西文状态下汉字的多字体”;

用“宝合”UPS 电源, 再不怕电网突然掉电



PH-500S



PH-1000

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



PH-1KL



PH-3KL



PH-600

“宝合”UPS 电源获奖一览表

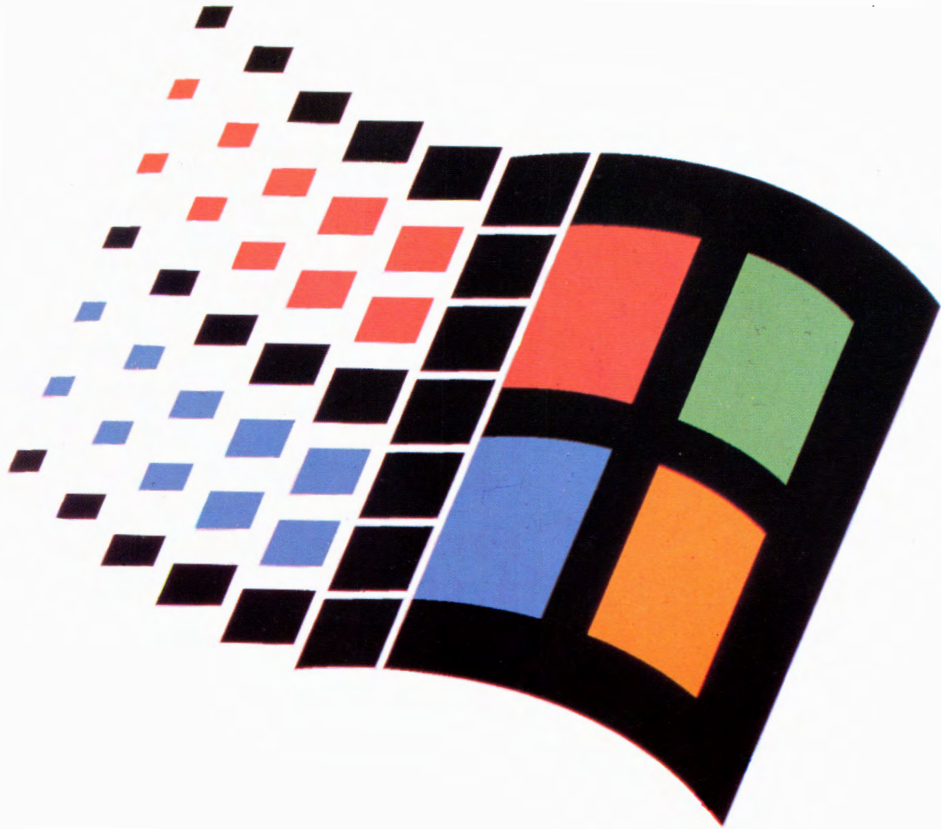
年 度	获 奖 名 称
1989 1991, 1992	北京市新技术产业开发试验区优秀拳头产品
1991	91 年度国家级新产品
1992	被国家科委成果管理办公室推荐为国内优秀产品 北京市科学技术进步三等奖
1993	被北京市技术监督局评为北京市企业标准成果奖 全国第七届运动会指定产品



北京希望电脑公司电源部
 地址: 北京海淀路 82 号
 邮编: 100080
 信箱: 北京 8721 信箱
 电话: 2567819, 2561058
 传真: 2561057

上海希望电脑公司
 地址: (200052) 上海新华路 416 号
 电话(传真): (021) 2521656, 2569929
 南京希望电脑技术公司
 地址: (210005) 南京市中山南路 105 号
 电话: (025) 4410794, 4410549
 传真: (025) 4410548

成都希望电脑公司
 地址: (610015) 成都市新南路四维村街 6 号
 电话(传真): (028) 5589787, 5556333
 广州希望电脑技术公司
 地址: (510620) 广州天河体育西路育菁三街二
 电话: (020) 7505151, 7505152, 7505153
 传真: (020) 7500275



Microsoft®
WINDOWS NT™



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

北京希望电脑公司资料事业部

地址：北京海淀路 82 号

信箱：(100080) 北京 8721 信箱

电话：2562329、2541992

传真：2561057





MICROSOFT
WINDOWS NT[™]
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[™]

disp __s2(): 显示立体的彩色汉字“彩色”、“立体”。

disp __s3(): 在屏幕中央向左右两侧放大显示“显示系统”;

disp __s4(): 在屏幕中央依次放大显示“无级变倍”;

disp __s5(): 在屏幕下方呈左右错切显示字符串“环宇电子集团公司”、石家庄电视机厂;

disp __s6(): 在屏幕左右两则呈上下错切显示各种字体的汉字。

源程序清单如下:

DEMO. C:

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
#define pi 3.1415926535
void disp16_0();
void disp24_0();
void disp24_1();
void disp24_2();
void disp24_3();
void disp24_4();
void disp24_5();
void w24read();
void disp __s1();
void disp __s2();
void disp __s3();
void disp __s4();
void disp __s5();
void disp __s6();
```

```
FILE *fp16, *fp24_s, *fp24_f, fp24_h, *fp24_k, fp24_e;
unsigned long index;
int no, len, color;
char w16[72], w24[72]
```

main ()

```
{
    int gd, gm;
    fp16=fopen("c:\\213\\hzk16", "rb");
    fp24_s=fopen("c:\\213\\hzk24s", "rb");
    fp24_f=fopen("c:\\213\\hzk24f", "rb");
    fp24_h=fopen("c:\\213\\hzk24h", "rb");
    fp24_k=fopen("c:\\213\\hzk24k", "rb");
    fp24_e=fopen("c:\\213\\hzk24e", "rb");
    detectgraph (&gd, &gm);
    initgraph (&gd, &gm, "");
    rectangle (0, 0, 639, getmaxy());
    disp __s1();
    disp __s2();
    disp __s3();
    disp __s4();
    disp __s5();
```

```
disp __s6();
getch();
fclose (fp16);
fclose (fp24_s);
fclose (fp24_f);
fclose (fp24_h);
fclose (fp24_k);
fclose (fp24_e);
closegraph();
}
```

void disp16_0(int x0, int y0, int color, char *w)

```
{
    int i, j;
    for (i=0; i<16; i++)
        for (j=0; j<16; j++)
            if (w[i*2+j/8]&(1<=(7-j%8))) putpixel (x0+j, y0+i, color);
}
```

void disp24_0 (int x0, int y0, int color, char *w)

```
{
    int i, j;
    for (i=0; i<24; i++)
        for (j=0; j<24; j++)
            if (w[i*3+j/8]&(1<=(7-j%8))) putpixel (x0+i, y0+j, color);
}
```

void disp24_1(int x0, int y0, int k, int kx, char *w)

```
{
    int i, j, k0, k1, ky;
    for (i=0; i<24; i++)
        for (j=0; j<24; j++)
            if (w[i*3+j/8]&(1<=(7-j%8)))
            {
                k0=(36-i)*(j-12)*k/24;
                k1=(36-i)*(j-12+1)*k/24;
                for (ky=k0; ky<=k1; ky++)
                    putpixel(x0+i*kx, y0+ky, color);
            }
}
```

void disp24_2(int x0, int y0, int kx, int ky, char *w)

```
{
    int i, j, x, y, x1, y1;
    x=x0; y=y0;
    for (i=0; i<24; i++)
    {
        for (j=0; j<24; j++)
        {
            if (w[i*3+j/8]&(1<=(7-j%8)))
                for (x1=x; x1<=x+kx; x1++)
                    for (y1=y; y1<=y+ky; y1++)
                        putpixel (x1, y1, color);
        }
        y+=ky;
    }
    y=y0
```

```

x+=kx;
}
}
void disp24__3 (int x0,int y0,(float k,char * w)
{
int i,j;
for (i=0;i<24;i++)
for (j=0;j<24;j++)
if (w[i*3+j/8]&(1<=<7-1&8)) putpixel (x0+i+(int) (j*k),y0
+j,color);
}
void disp24__4 (int x0,int y0,float k,char * w)
{
int i,j;
for (i=0;i<24;i++)
for (j=0;j<24;j++)
if (w[i*3+j/8]&(1<=<7-j&8)) putpixel (x0+i,y0+j+(int) (i*
k),color);
}
void disp24__5(int x0,int y0,float a,char * w)
{
int i,j,x,y;
float olda,r;
for (i=0;i<24;i++)
for (j=0;j<24;j++)
if (w[i*3+j/8]&(1<=<7-j%8))
{
if (j==0) olda=0;
else olda=atan (1.0*i/j);
r=sqrt (1.0*i*i+j*j);
x=(int)(r*sin (olda+a));
y=(int)(r*cos(olda+a));
circle (x0+x,y0+y,color);
}
}
void w24read (unsigned char aa[],unsigned char n,unsigned char
zt)
{
no=((aa[2*n]-0xb0)*94++aa[2*n+1]-0xa1);
index=(long)no8721;
switch (zt)
{
case 1:
fseek (fp24__s,index,SEEK __SET);
fread (w24,1,72,fp24__s);
break;
case 2:
fseek (fp24__f,index,SEEK __SET);
fread (w24,1,72,fp24__f);
break;
case 3:
fseek (fp24__h,index,SEEK __SET);
fread (w24,1,72,fp24__h);
break;
case 4:
fseek (fp24__k,index,SEEK __SET);
fread (w24,1,72,fp24__k);

```

```

break;
case 5:
faeek(fp24__e,index,SEEK __SET);
fread (w24,1,72,fp24__e);
break;
}
}
void disp __s1()
{
float a,alpa;
int i,b,x,y;
unsigned char aa[]="西文状态下汉字的多字体";
len=strlen(aa)/2;
b=(640-24*len)/len
x=b/4;
for (i=0;i<len;i++)
{
w24read (aa,i,1);
y=(200.0-sqrt(200.00*200.0-(320-x)/3.0*(320-x)/3.0))
*2;
a=pi/4.0;
alpa=a-(a*2*i)/(len-1);
disp24__5(x+10,y,alpa,w24);
x+=24+b
}
}
void disp __s2()
{
int i,x0,y,x1;
unsigned char aa[]="立体",bb[]="彩色";
len=strlen(bb)/2;
x0=320-54;
x1=320+30;
y=40;
for (i=0;i<len/2;i++)
{
w24read(bb,i,1);
color=1
disp24__0(x0,y,1,w24);
color=2
disp24__0(x0+2,y-2,1,w24);
w24read (bb,len-1-i,1);
color=4;
disp24__0(x1,y,1,w24);
color=5
disp24__0(x1-2,y-2,1,w24);
}
}

```

<p>最少的投资</p> <p>最佳的方案</p> <p>最多的组合</p>	<p>挂接式</p> <p>打印机共享器</p>
<p>北京向宇计算机公司</p> <p>地址:北京复兴路甲20号27分号312,314</p> <p>邮编:100036 电话:8263441,2063366呼1511</p>	

```

x0+=28;
x1+=28;
}
len=strlen(aa)/2;
x0=320-80;
x1=320+40;
y=getmaxy()-80;
for(i=0;i<len/2;i++){
    w24read(aa,i,1);
    color=4;
    disp24_2(x0,y,2,2,w24);
    color=5;
    disp24_2(x0+2,y-2,2,2,w24);
    w24read(aa,len-1-i,1);
    color=1;
    disp24_2(x1,y,2,2,w24);
    color=2;
    disp24_2(x1-2,y-2,2,2,w24);
    x0+=28;
    x1+=28;
}
}
void disp__s3()
{
    int i,ox=320,oy=100;
    unsigned char aa[]="显示系统";
    len=strlen(aa)/2;
    for(i=0;i<len;i++){
        {
            w24read(aa,i,5);
            color=i+1;
            disp24_1(ox-100-30*i,oy,i+1,1,w24);
            disp24_1(ox+100+30*i,oy,i+1,-1,w24);
        }
        color=1;
    }
}
void disp__s4()
{
    unsigned char aa[]="无级变倍"
    int i,x,y;
    len=strlen(aa)/2;
    x=320-12;
    y=40;
    for(i=0;i<len;i++){
        {
            w24read(aa,i,5);
            color=(8-i)%8;
            disp24_2(x,y,i+11,w24);
            x-=12;
            y+=24;
        }
    }
}
void disp__s5()
{
    unsigned char aa[]="环宇电子集团公司",bb[]="石家庄电视机厂"
    int i,x,y;

```

```

len=strlen(aa)/2;
x=(320-len*24)/2;
y=getmaxy()-30;
for(i=0;i<len;i++){
    {
        w24read(aa,i,1);
        disp24_3(x,y,-0.5,w24);
        Jx+=24;
    }
}
len=strlen(bb)/2;
x=320+(320-len*24)/2;
y=getmaxy()-30;
for(i=0;i<len;i++){
    {
        w24read(bb,i,1);
        disp24_3(x,y,0.5,w24);
        x+=24;
    }
}
}
void disp__s6()
{
    unsigned char aa[]="宋体仿宋黑体",bb[]="楷体图形繁体";
    int i,x,y;
    len=strlen(aa)/2;
    x=0;
    y=(getmaxy()-24*len)/2;
    for(i=0;i<len;i++){
        {
            if(i<2){w24read(aa,i,1);color=1;}
            else if(i<4){w24read(aa,i,2);color=2;}
            else {w24read(aa,i,3);color=3;}
            disp24_4(x,y,0.5,w24);
            y+=24;
        }
    }
    len=strlen(bb)/2;
    x=639-24;
    y=(getmaxy()-24*len)/2;
    for(i=0;i<len;i++){
        {
            if(i<2){w24read(bb,i,4);color=4;}
            else if(i<4){w24read(bb,i,1);color=5;}
            else {w24read(bb,i,5);color=6;}
            disp24_4(x,y,-0.5,w24);
            y+=24;
        }
    }
}

```

SKILL

<p>最少的投资</p> <p>最佳的方案</p> <p>最多的组合</p>	<p>挂接式</p> <p>打印机共享器</p>
<p>北京向宇计算机公司</p> <p>地址:北京复兴路甲20号27分号312,314</p> <p>邮编:100036 电话:8263441,2063366呼1511</p>	

一个能打印出“字中字”的程序

□周顺生

本

文介绍一个能打印出“字中字”的程序,程序执行的结果是打印出一个大字,而这个大字的每个点是由该字的小字组成的。当用户用放大镜观察打印机打印出的汉字时,会发现每个汉字是由一定的点组成的,字不同,点的位置也不相同。这些点的位置一般来自于主机的点阵字库,汉字的每个点对应字库的一位二进制数。本程序先从字库中读字的点阵数据作为打印的位置信息,用其对应的字代替原打印汉字的每一个点,这就形成了“字中字”。本程序使用的字库是2.13 H 24点阵宋体字库,库名文件为“HZK24S”。

本程序采用 Turbo C 2.0 所编,输入并编译后就能运行了,可以产生名为“P”的文件,用 WPS 等字处理程序调用该文件即可打印,并能修改小字的大小,也可以在挂接汉字打印驱动程序后执行本程序,打印出“字中字”。

源程序清单如下:

```
/* 本程序为字中字的程序 */
#include<stdio.h>
#include<dos.h>
#include<fcntl.h>
int openhzk(void);
int puthzk24(char *);
int getbit(unsigned char,int);
int handle,fp;
main()
{ char *p="国";/*要打印的字*/
  openhzk(); puthzk24(p); close(handle); exit(0);
}
int openhzk()
{ handle=open("hzk24s",O_RDONLY|O_BINARY);
  if(handle== -1){
```

```
puts("Error on open clib"); getch(); exit(1); }
}
int puthzk24(char *p)
{ FILE *fp; unsigned int i,c1,c2;
  int f=0,rec,i1,i2,i3,i4,i5,u,m=1,n=1;
  /*大字的放大倍,m为横向放大倍数,n为纵向放大倍数*/
  long l=72; unsigned char by[72],*s,*t,t1,t2;
  fp=fopen("p","wb");
  while((i=*p++)!=0){
    if(i>0xa1)
      if(f==0){ t1=i; c1=(i-0x0b0)&0x7f; f=1; }
      else {
        c2=(i-0xa1)&0x07f;t2=i;
        f=0; rec=c1*94+c2; l=rec*72L;
        biosprint(0,0x0d,0); biosprint(0,0x0a,0);
        biosprint(0,0x0d,0); biosprint(0,0x0a,0);
        fputc('\x0d',fp);fputc('\x0a',fp);
        fputc('\x0d',fp);fputc('\x0a',fp);
        sleep(1);system("cls");
        lseek(handle,l,SEEK_SET); read(handle,by,72);
        for(i2=0;i2<=2;i2++)
          for(i3=0;i3<8;i3++)
            for(i5=0;i5<n;i5++){fputc('\x0d',fp);fputc('\x0a',
              fp);
              biosprint(0,0x0d,0); biosprint(0,0x0a,0);
              for(i1=0;i1<24*m;i1=i1+m)
                for(i4=0;i4<m;i4++)
                  if(getbit(by[i1/m*3+i2],7-i3))
                    { fputc(t1,fp);fputc(t2,fp); biosprint(0,t1,0); biosprint
                      (0,t2,0); }
                  else {
                    biosprint(0,0x20,0); biosprint(0,0x20,0);
                    fputc(' ',fp);fputc(' ',fp);}
                }
              }
            } return(0);}
  int getbit(unsigned char c,int n)
  { return((c>>n)&1);}
```

SKILL

利用 VB 特别功能构造动态界面

□杨 林

有

许多关于 GUI 设计问题的好书,但是作为程序设计的一个非常重要的方面的“实时管理”,在某种程度上被一带而过甚至忽略了。屏幕实时管理分为两个方面:静态和动态。静态管理就是我们通常认为的用户界面设计过程那一部分的所有东西:选择、集合和图象元素的输出,例如显示形式、编辑控制和菜单。

VB 界面设计环境的优势是允许你可视的仔细构造并调整界面。这个过程也是静态实时管理的结果。首先,选择表达正确含义的控制,如按钮和检查盒传递选择,命令按钮可使工作更容易,图标是“一幅图象价值千言”的最佳 GUI 的实现。然后,放置和集中你想在运行时显示的控制内容——这样,你就完成了对应用程序屏幕实时运行的静态管理。

尽管图形界面设计的功能提供了在很大程度上帮助了可视化编程的实施,但是,这些功能同时也会限制编程人员使用这些工具。可视化工具会给编程人员一种错误的工作感觉,使程序员相信静态设计工作就是整个界面制作的工作。而实际上,在某些场合却需要程序员考虑到屏幕实时的显示控制,下面具体描述一下。

现有的技术提供了列表盒的水平卷动,但是对于一个复合盒中的文字盒部分怎么办呢?如果动态生成的柱状图标识阅读不了又怎么办?如果用户希望单独测量 MDI 文件的长度而又要不破坏所有显示信息又怎么办?试图对于每一种可能的情况都设计和编程,这不是一个好办法,较好的解决方式是在更一般的方式上构造界面。本文谈到的动态屏

幕界面设计就可以实现这种可能性。

动态界面

当你考虑动态屏幕界面时,可能在你脑海中出现的第一件事是隐蔽和非隐蔽控制、加载和卸载的控制数组元素、显示和隐藏子对话框等。其实这些都是静态界面管理,因为在设计和预先决定何时怎样显示这些元素中已经安排好了,管理这些显示大部分是通过程序控制,而不是用户控制。

动态实时管理能使用户修改屏幕控制来达到适合用户的最佳状态。这样,使用者不会被局限在设计者认为最好方法的思路中,被用户授权实时改变屏幕来适应他分析观察显示的信息的需要。

这些听起来可能象一个难以完成的任务。但是利用 VB 特别功能的帮助,你可以看到一个动态屏幕界面能很容易的构造好,不论显示区域怎样改变大小,它都能按比例缩放控制。达到这个目标的步骤是在开始时记住控制的相对位置和大小,在缩放时按比例进行调节位置,为了证明这点,我们选用一个中等大小的屏幕区域为例,这个区域会频繁被改变,其中有一个标识文字盒、列表盒、复合盒、图象、图形和命令按钮。

一眼看来,这种显示的方式看起来足够了:使用者可以通过文字盒的帮助在列表盒中寻找感兴趣的特殊的条目。通过在复合盒中的选择可以改进搜索,相应的图象可以在图形图象控制下显示。

站在使用者角度仔细观察,你会发现在列表盒和复合盒中的文字显示并非很好:重要的

识别信息可能被截掉了。图象看起来不错,一些重要的细节看不清楚。柱状图,它是在运行中生成的,但是难以分辨文本标识的单词。这些问题通过一些非常简单的代码就可以解决。

补救方法的第一步是在代码中定义一个特别记录结构,你可以利用它来获得控制的相对大小和位置的信息。

定义可以象如下这样:

```
Type SizeStruct
  Top As Wingle
  Left As Single
  Width As Single
  Height As single
End Type
```

然后,不论是在显示方式级上作说明还是作一个全局定义,在相同代码模式下产生这个结构的一个实例作为一个动态数组。

在显示方式级上注明这些如下所示:

```
DimCtlDelta()As SizeStruct
```

在加载情况下,需要在控制中循环来获得最初的相对位置信息。例如,存储一个给定控制的上界位置,通过显示的标尺高度(ScaleHeight)的百分比来确定,左界位置通过标尺宽度(ScaleWidth)的百分比来确定。

VB 通过控制集合来适合这种需求,控制集合是在显示上所有控制的内部数组,具有单独的 Count 属性。

使用控制集合,过程如下:

```
ReDimPreserve CtlDelta _
  (0 To Controls.Count - 1)
For 1% = 0 To Controls.Count - 1
  CtlDelta(i%). _
    Top = Controls(i%).Top / _
    Controls(i%).Parent. _
    ScaleHeight
  CtlDelta(i%).Left = Controls _
  (i%).Left / Controls(i%). _
  Parent. Scalewidth
  CtlDelta(i%).Width = Controls _
  (i%).Width / Controls(i%). _
  Parent. Scalewidth
  CtlDelta(i%).Height = Controls _
  (i%).Height / Controls(i%). _
  Parent. Scalewidth
Next i%
```

最后,不论何时改变大小,都可以通过增加类似的过程给“改变大小”事件来传递这些百分比形成新的显示范围。完整的程序清单列在文后。

改变大小的设计

这种技术仅提供了按比例地放大,这点请特别注意,这意味着,不仅控制的缩小和放大是按比例的,而且他们之间的距离也是按比例地。在无关的控制例如一个列表盒和一个图象之间这种影响是可以接受的,但对于相关的控制例如一个标识和一个文字盒之间则就不可接受。有几个简单的规则可以在这方面提供一个参考。

- 将标识放在相关控制的上面而不是旁边。
- 主要的东西放在垂直线上而不是水平线上。
- 垂直方向使控制宽度成为一线。
- 将目标控制放在一个范围内,以此大小为标准约束一个区域的大小比例。

对于许多好的静态实时管理的应用,只要增加这样一点代码就可以提高界面的效果。对于某些显示方式来说,按比例缩放还不够,在上例显示中,例如,列表可能非常的长,用户可能想很快的浏览它,而不想等待图象显示。也许用户对这些图象并不关心,或者也许用户不关心图象而只是想看看柱状图的细节部分。

除去这些限制,对于最终用户这种按比例改变大小的屏幕具有许多好处,用户可以把精力集中在应该注意的地方,调整视距得到细节部分,也许对于程序来说最大的好处是在开发中节省了时间。

VB 中完成比例缩放的程序清单如下:

```
VERSION 2.00
Begin Form Form1
Caption "Auto-Sizing Demo"
ClientHeight = 4860
ClientLeft = 1770
ClientTop = 930
ClientWidth = 5040
Height = 5265
Left = 1710
LinkTopic = "Form1"
ScaleHeight = 4860
ScaleWidth = 5040
Top = 585
Width = 5160
Begin GRAPH Graph1
AsciiLegen = "Cat # 1~Cat # 2~Cat # 3~Cat # 4~Cat # 5"
Background = 7' Lighr Gray
BottomTitle = "Tree Categories"
GraphTitle = "TreePopulationByCategory"
Height = 1815
Left = 2700
TabIndex = 8
Top = 2160
Width = 2055
End
Begin ComboBox Combo1
```

```
Height = 300
Left = 300
TabIndex = 7
Text = "Combo1"
Top = 3660
Width = 2235
End
Begin CommandButton Command3
Caption = "Quit"
Height = 375
Left = 3420
TabIndex = 5
Top = 4260
Width = 1335
End
Begin PictureBox Picture1
Height = 1755
Left = 2700
Picture = RESIZE.FRX:0000
ScaleHeight = 1725
ScaleWidth = 2025
TabIndex = 6
Top = 300
Width = 2055
End
Begin CommandButton Command2
Caption = "Restore"
Height = 375
Left = 1740
TabIndex = 4
Top = 4260
Width = 1335
End
Begin CommandButton Command1
Caption = "Maximize"
Height = 375
Left = 300
TabIndex = 3
Top = 4260
Width = 1335
End
Begin ListBox List1
Height = 2370
Left = 300
TabIndex = 2
Top = 1020
Width = 2235
End
Begin TextBox Text1
Height = 315
Left = 300
TabIndex = 1
Top = 600
Width = 2235
End
Begin Label Label1
Caption = "Search For Type:"
```

```
Height = 315
Left = 300
TabIndex = 0
Top = 300
Width = 2235
End
End
DefInt A-Z
'Declare module level dynamic array
Dim CtlDelta() As SizeStruct
Sub Command1 __ Click()
WindowState = MAXIMIZED
End Sub
Sub Command2 __ Click()
WindowState = NORMAL
End Sub
Sub Command3 __ Click()
Unload Form1
End Sub
Sub Form __ Activate()
list1.AddItem "this is the First tree category type"
list1.AddItem "this is the Second tree category type"
list1.AddItem "this is the Third tree category type"
list1.AddItem "this is the Fourth tree category type"
list1.AddItem "this is the Fifth tree category type"
list1.AddItem "this is the Sixth tree category type"
list1.AddItem "this is the Seventh tree category type"
list1.AddItem "this is the Eighth tree category type"
list1.AddItem "this is the Ninth tree category type"
list1.AddItem "this is the Tenth tree category type"
list1.AddItem "this is the Eleventh tree category type"
list1.AddItem "this is the Twelfth tree category type"
list1.AddItem "this is the Thirteenth tree category type"
list1.AddItem "this is the Fourteenth tree category type"
list1.AddItem "this is the Fifteenth tree category type"
Combo1.AddItem "this is the First treeSub __ category type"
Combo1.AddItem "this is the Second tree category type"
Combo1.AddItem "this is the Third treeSub __ category type"
Combo1.AddItem "this is the Fourth treeSub __ category type"
Combo1.AddItem "this is the Fifth treeSub __ category type"
Combo1.AddItem "this is the Sixth treeSub __ category type"
Combo1.AddItem "this is the Seventh treeSub __ category type"
Combo1.AddItem "this is the Eighth treeSub __ category type"
Combo1.AddItem "this is the Ninth treeSub __ category type"
Combo1.AddItem "this is the Tenth treeSub __ category type"
'select an item in the Combe box
Combo1.ListIndex = 7
End Sub
Sub Form __ Load()
'NOTE: For Screen, Form, and Printer objects the Width
'And Height properties are always measured in TWIPS.
'For the (non-MDI) Form and Printer objects the internal
'client area coordinate system can be changed to other
'scalings. For simplicity you may want to leave the
'coordinate system set to the VB default of TWIPS.
'However, some API routines require pixels so you'll need
```



```

'to do conversion
'Begin UIInitialization
'Perform any desired startup sizing of the
'form here,before processing coordinates.
WindowState=NORMAL
'Set desired scale mode here. Use the SCALEMODE property
'for pre-defined scalings. use the
'SCALETop,Left,Width,Height properties directly or the
'SCALE method to define a custom coordinate system.
'ScaleMode=TWIPS
'Size the control dimension proportion array
ReDim Preserve CtlDelta(0 To Controls.Count-1)
'Loop through each control on the form and capture its
'dimensions as a percentage of the scalable dimensions of
'the form. (won't work for the line contro)
For i%=0 To Controls.Count-1
    CtlDelta(i%). Top=Controls(i%). Top/Controls(i%). __ Parent.
        ScaleHeight
    CtlDelta(i%). Left=Controls(i%). Left/Controls(i%). __ Parent.
        ScaleHeight
    CtlDelta(i%). Width = Controls(i%). Width/Controls(i%). __
        Parent. ScaleHeight
    CtlDelta(i%). Hight = Controls(i%). Hight/Controls(i%). __
        Parent. ScaleHeight
Next i%
End Sub
Sub Form __ QueryUnload(Cancel As Integer. __
    UnloadMode As Integer)
    Set Form1=Nothing
End Sub
Sub Form __ Resize()
    'Loop through each controlon the form and
    'reset their dimensions as a percentage of
    'the scalable dimensions of the form.
    For i%=0 To Controls.Count-1
        If Typeof Controls(i%)Is TextBox The.
            'Set all but the height ,unless it's multiline
            Controls(i%). top=Controls(i%). Parent. __
                ScaleHeight * CtlDelta(i%). Top
            Controls(i%). Left=Controls(i%). Parent. __
                ScaleWidth * CtlDelta(i%). Left
            Controls(i%). Width=Controls(i%). Parent. __
                ScaleWidth * CtlDelta(i%)Width
            If Controls(i%). MultiLine=True Then
                Controls(i%). Height=Controls(i%). __
                    Parent. ScaleHeight *
                    CtlDelta(i%). Height
            End If
        Else If Typeof Controls(i%) Is Label Then
            'Set only the position ,don't
            'change dimensions
            Controls(i%). top=Controls(i%). Parent. __
                ScaleHeight * CtlDelta(i%). Top
            Controls(i%). Left=Controls(i%). Parent. __
                ScaleWidth * CtlDelta(i%). Left
        Else If Typeof Controls(i%) Is

```

```

        CommandButton Then
            'Set only the position ,don't
            'change dimensions
            Controls(i%). Top=Controls(i%). Parent. __
                ScaleHeight * CtlDelta(i%). Top
            Controls(i%). Left=
            Controls(i%). Parent. ScaleWidth * CtlDelta(i%). Left
        Else If Typeof Controls(i%) Is CombiBox Then
            'Set all but the height ,unless
            'it's style 1
            Controls(i%). Top=Controls(i%). Parent. __
                ScaleHeight * CtlDelta(i%). Top
            Controls(i%). Left=Controls(i%). Parent. __
                ScaleWidth * CtlDelta(i%). Left
            Controls(i%). Width=Controls(i%). Parent. __
                ScaleWidth * CtlDelta(i%)Width
            If Controls(i%). Style=1 Then
                Controls(i%). Height=Controls(i%). __
                    Parent. ScaleHeight __
                    * CtlDelta(i%). Height
            End If
        Else If Typeof Controls(i%) Is HScrollBar Then
            'Set all but the height
            Controls(i%). Top=Controls(i%). Parent. __
                ScaleHeight * CtlDelta(i%). Top
            Controls(i%). Left=Controls(i%). Parent. __
                ScaleWidth * CtlDelta(i%). Left
            Controls(i%). Width=Controls(i%). Parent. __
                ScaleWidth * CtlDelta(i%). Width
        Else If typeof Controls(i%) Is VScrollBar Then
            'Set all but the Width
            Controls(i%). Top=Controls(i%). Parent. __
                ScaleHeight * CtlDelta(i%). Top
            Controls(i%). Left=Controls(i%). Parent. __
                ScaleWidth * CtlDelta(i%). Left
            Controls(i%). Height=Controls(i%). Parent. __
                ScaleHight * CtlDelta(i%). Height
        Else If Typeof Controls(i%) Is Menu Then
            'Skip it Else If Typeof Controls(i%) Is Time Then
            'Skip it Else
            Controls(i%). Top=Controls(i%). Parent. __
                ScaleHeight * CtlDelta(i%). Top
            Controls(i%). Left=Controls(i%). Parent. __
                ScaleWidth * CtlDelta(i%). Left
            Controls(i%). Width=Controls(i%). Parent. __
                ScaleWidth * CtlDelta(i%). Width
            Controls(i%). Height=Controls(i%). Parent. __
                ScaleHeight * CtlDelta(i%). Height
        End If
    Next i%
    'Perform any desired customsizing/updates
    'of the controls here.
    'return focus to textbox
    Text1.SetFocus
End Sub

```

(本文英文资料由 Microsoft 提供)

SKILL

利用 QueryDef 提高资料的快速存取

□ 杨 兰

Q

QueryDef 是 Visual Basic 3.0 非常强大的功能之一,对于 VB 开发人员同样也是最易于使用的功能之一,QueryDef 是 SQL 方式,它不是在你的程序代码里,而是存储在数据库中。和任何 SQL 方式一样,他们采用 CreateDynaset 方法来建造 Dynaset 对象,在数据控制记录源(Data ControlRecordSource)属性中产生记录串,或者利用执行(Execute)方法来更新或删除一串记录,但是不象标准 SQL 方式,这个 QueryDef 不是在运行期间,而是在写入数据库时,由 Access 分析和优化完成一个 QueryDef 运算的方法。

当你在 VB 程序中利用 Access 的 Jet engine(喷气发动机)执行一个 SQL 方式写操作时,第一件要做的事情是从语法上分析询问——把它分解成必要的步骤,第二,Access 试图产生一个能实施的策略,它使用各种能影响操作的索引,以及其他的内部方法来优化对询问的处理。最后,它执行这个询问,一般,询问的语法分析和优化要花费较长时间。采用 QueryDef,Access engine 可节省很多时间。使用 QueryDef,你的应用程序能从 Access engine 中得到极佳的效果,与其他方法比较,可节省百分之七十五的时间。这里我们讨论的是如何提高相对缓慢的,硬盘密集的数据库运算,因为运行结果的时间上的差异是用秒,而不是毫秒来衡量,因此程序的设计观念对用户来说很有关系。

为什么开发人员不更多地使用 QueryDef 呢?不幸的是,对 QueryDef 的支持并不包含在 Visual Basic 的数据管理器(Data

Manager)之中。VB3.0 专业版在使用数据库对象时能够生成和使用 QueryDef,这些都需要支持 QueryDef 的 Access 或第三方数据库管理软件或者数据字典。如果你有 VB3.0 的专业编辑器,你也可以编一个 VB 程序利用数据资料存取对象来写 QueryDef。另外,除去上述优点,QueryDef 并不总是最好的方法,有时 QueryDef 比其他方法运行速度要慢,或者使用不方便,为了表明怎样才能使 QueryDef 运行得最好,我们将比较完成的一个运算的代码,这个运算利用 ISAM 方法处理表格寻找(Table Seeks),生成 VBench,这个程序可用来测试五种 VB 数据存取的方法。这个 VBench 应用略做修改可以直接在你的数据库中工作,这样读者便有了一个在任何情况下分析什么方法是最好的工具。一旦明白了如何生成和评价 QueryDef 的过程,读者就能知道在 VB 中什么时候用它们,什么时候不用它们。

QueryDef 的使用

让我们来检查增加、修改、删除和报废 QueryDef 的方法。在例子中我们假设已说明和打开了你的数据库 MyDb,将 MyQd 指定为 QueryDef 对象。注意 VB 专业编辑器需要这些操作,因为从 VB 中,这些操作是直接的,所以读者要记住要获得预先优化好处就需要将 QueryDef 写入你的数据库,这可使用或编写其他实用程序来完成,但它并不是运行程序的一部分。

第一步,一般而言,是利用数据库对象的 CreateQueryDef 方法来生成 QueryDef。读者

必须详细说明 QueryDef 的名称和 SQL 格式:

```
Set MyQd = MyDb. CreateQueryDef ( "
TestQd", _
"SELECT * FROM TestTable WHERE _
TestField1=5000 ORDER BY TestField2;" )
```

你可以通过打开、重置它的 SQL 属性来修改 QueryDef:

```
Set MyQd=MyDb. OpenQueryDef("TestQd")
MyQd. Sql="SELECT * FROM TestTable _
WHERE TestField1=1 _
ORDER BY TestField2;"
```

利用 Delete QueryDef 功能可以从一个数据库中移走一个已存在的 QueryDef。你可以不要一个 QueryDef 对象,但是你应有一个数据库对象,并详细说明存在数据库中的 QueryDef 的名称(不是 QueryDef 的逻辑名称)。

```
MyDb. DeleteQueryDef("Test Qd")
```

如果你的代码不知道数据库中的 QueryDef 名称,你就必须检查数据库的结构。获得数据库中的 QueryDef 的列表并不象大多数 QueryDef 操作那么直接,Accessx 认为 QueryDef 是表格类型,但是 QueryDef 不出现在 TableDef 集合中,也没有 QueryDef 的集合。

VB 只能通过 ListTable 方法来获得 QueryDef 的列表,ListTable 生成一个数据库中 QueryDef 和表格的“快照”(Snapshot),你可以检查每一个表项的属性来判定真正的 QueryDef,一旦发现 QueryDef,你必须打开 QueryDef 获取它的 SQL 格式,在例子中,我们定义“快照”为 MyList。

下面是代码清单:

```
Set MyList=MyDb. ListTables()
MyList. MoveFirst
Do Until MyList. EOF
    'Check if table is type 5->QueryDef
    If (MyList ! TableType And 5)=5 Then
        Set MyQd=_
        MyDb. OpenQueryDef(MyList ! Name)
        'Do anything here—we'll just print the info
        Debug. Print MyList ! Name,MyQd. SQL
        MyQd. Close
    End If
    Mylist. MoveNext
Loop
```

现在你已经明白了如何增加、删除和列表 QueryDef,剩下的事情是让它们工作起来,依据 QueryDef 的类型和数据库存取方式,有几种方法可以执行 QueryDef。选择(SELECT)QueryDef 常用来

生成数据控制记录串、dynaset 或者快照。

如果你正在使用数据控制,将数据控制的记录源(RecordSource)属性置为 QueryDef 的名称。

```
Data1. RecordSource="TestQd"
```

```
Data1. Refresh
```

为了生成一个 dynaset 或者快照,可使用 CreateDynaset 和 CreateSnapshot 方法。你仅需要附注数据库对象和 QueryDef 的名称:

```
Set MyDyna = MyDb. CreateDynaset ( "
TestQd")
```

```
Set MySnap=CreateSnapShoot("TestQd")
```

执行一个 QueryDef 就是如此简单,你可以选择编写方法:

```
MyDb. Execute("TestQd")
```

或

```
Set MyQd=MyDb. OpenQueryDef("TestQd")
```

```
MyQd. Execute()
```

```
MyQd. Close
```

不要将执行方法(Execute Method)和 ExecuteSQL 弄混淆了。ExecuteSQL 仅在合并 ODBC 表格时使用,同时,不要在 SELECT 询问时使用执行(Execute),因为执行将不返回记录单,而以生成一个运行时间错误来代替。

有价值的灵活性

直到现在,所有的例子都是标准的或静态的 QueryDef。这些 QueryDef 的困难是你仅能通过替换整个 SQL 方式来改变操作。Access 不得不去重新优化他们,降低了预编译询问的优势。然而,有另外一种类型的 QueryDef,称为带参数 QueryDef,它允许使用者定义参数值。带参数 QueryDef 用程序中或者使用者定义的值替换 Set 和 Where 中的值。完成一个参数化 QueryDef,你需要增加一个 PARAMETERS 子句,改变你的 WHERE 或 SET 子句来反映参数名称,而不是字面上的值。

典型的选择 SELECT 和更新 UPDATE 格式如下:

```
PAPAMETERS UserLowRange Long,UserHighRange Long;—
SELECT * FROM TestTable WHERE MyField1> _
UserLowRange And MyField2<UserHighRange _
ORDER BY MyField1,MyField2
```

```
PAPAMETERS UserValue1 Long,UserLowRange Long, _
UserHighRange Long;UPDATE TestTable SET _
MyField1=MyField1 * UserValue1 WHERE—
MyField1>UserLowRange And MyField2<UserHighRange
```

注意 PARAMETERS 子句必须以分号结尾。子

句中各项间用逗号隔开,PARAMETERS 数据类型说明是不标准的,表 1 列出了有效的 PARAMETERS 类型和具有相等作用的类型。

当谈到在 WHERE 和 SET 子句中参数变化时,不象你提供字面意义,你不必提供界线如单引号或英镑符号。

带参数 QueryDef 的生成,列表和删除与静态 QueryDef 非常相似。但是执行一个带参数 QueryDef 需要不同的构成方法。因为你必须为参数提供数值,如下所示,简单地打开 QueryDef 然后将数值置于参数序列中,然后使用 QueryDef 对象执行,不是数据库对象。

```
Set MyQd=OpenQueryDef("TestQd")
MyQd ! UserLowRange=5000
MyQd ! UserHighRange=5500
MyQd ! UserValue=.9
MyQd.Execute
MyQd.Close
```

你必须使用 QueryDef 对象构造方式执行参数化选择询问,如同:

```
Set MyQd=MyDb.OpenQueryDef("TestQd")
Set MyDyna=MyQd.CreateDynaset()
```

接着,你不必详细指明例如英镑符号的界线,因为 Access 已经知道了数据类型。如果你试图将错误类型的值置入,将会产生一个错误。在使用带参数 QueryDef 时最大的不同是你不能象数据控制的记录源那样使用带参数 QueryDef。

上面是 QueryDef 的简单总结,现在你知道了怎样去使用 QueryDef,问题是,什么时候使用他们。做为一个 VB 数据库开发人员你面对的最大问题是在一个特殊情况下如何决定一个最好的数据存取方式。你必须决定是用 ISAM 还是 SQL 方法。如果你采用 SQL,你必须决定是采用数据控制,dynaset 还是快照方式。决定了这些以后,你需要决定是采用线 VB SQL 代码还是一个 QueryDef,这篇文章的目的就是检查影响决定使用 QueryDef 的 SQL 的各个因素。

SQL 与 ISAM 方法相比最基本的优点就是 SQL 可以用很少的代码来有效地完成一个复杂的运算,数据库发动机决定了要完成请求必须的步骤,而且可很快完成。SQL 能用一条语句就可以完成同样运算而且经常执行得更快,显然还编制一个 ISAM 循环来合计一张表或修改一串记录是时间的浪费。不同于 ISAM 寻找方法,SQL 方式不需要他们的目标区域被索引,这提供了更大的灵活性。如果这个区域是索引的一部分,那么 SQL 会有效地利用索引,

使用寻找(seek)方法的 ISAM 存取是一个有效的但有局限性的工具。而 SQL 是一个灵活的,全功能的数据语言。

当然,QueryDef 不能解决所有数据库的问题,在有些时候、有些地方不必或不能使用它。在有的情况下使用 ISAM TABLE Seek 会更快些,例如通过索引辨别记录情况。当存取较少记录的表格时,QueryDef 表现不出速度的优势,甚至比在线代码 SQL 方式更慢。QueryDef 不能在非 Access 数据库中使用,例如 dBase,除非你合并这些表格到一个 Access 数据库,对于一个数据控制,带参数 QueryDef 不能用做记录源。QueryDef 也不能象在线 SQL 代码那样容易地修改,所以他们更难生成、调试和维护。最后,还有一个代码安全问题。QueryDef 因为存储在数据库中,对于任何用户都是开放的,只要他有 Access 的复制版本和进入数据库的权利。

清楚 QueryDef 的缺点,如何在提高速度方面平衡它们是一个大问题。VBench 程序有助于解决这个问题。为了运行这个程序,点击 Populate 产生 1 万条记录试验串,然后,点击更新(Update)来检测一个 Action 操作,或者选择测试一个 Select 操作。注意,GetTickCount() API 调用返回毫秒值,但是 PC 时钟每 18 毫秒才更新一次,所以对于非常快的运算如搜索,必须重复执行多次求一平均值,表 2 是对于样本数据的测试结果。使用这个程序的关键是重新修改来适合的要求,可以通过改变数据库或者通过修改算法来实现以反映你的应用程序中的行为。

在优化 Access 数据操作中没有一个十全十美的规则。当面对一个数据读取优化问题时,应用测试程序去检测不同的结果。同时应注意你的检测策略,要在数量上和真实数据上相差不多,不应是小型的测试数据库。最后要做出创造性的设计多种不同的方法和搜集尽可能多的真实条件,以做出最好的判断。

这个 VBench 程序建立了一个可修改的结构来测试更新和存取数据库的不同方法。为了使用这个程序,必须生成一个 Access 数据库 IOTEST.MDB,建立一个表 TestTable,增加两个域:ID(数据类型 Long)和 Cost(Double),增加一个索引 Idkey,用 ID 作为它的唯一域。将 Primary 和 Unique 标志设置为真。然后,用一个单独的表建一个新的工程文件,在表中安置一个命令按钮数组,四个按钮(Command(0)到 Command(3)),按钮命名为"Update","Select","Populate"和"Exit",表中安置一个五个文本盒数组,其名称为 Text1(0)到 Text1(4),生成他们的标识"ISAM Table Seek","Dynaset Find"

First”, “Inline VBSQL”, “Statre query Def” 和 “Parameter Query Def”.

表 1

Parameter Type	AccessType	Visual Basic Type
IEEESingle	Single	Single
IEEEDouble	Double	Double
Bit	Boolean	Integer
Byte	Byte	Integer
Short	Integer	Integer
Long	Long	Long
Currency	Currency	Double
Date	DateTime	Date Variant
Binary	LongBinary	NO Equivalent
Text	Text	Fixed Length String
LongText	Memo	Variable Length String

表 2

	Update	Select
ISAM Table Seeks	6.15	2.20
Dynaset FindFirst	17.85	12.08
VB Code SQL	4.23	5.33
Static QueryDef	3.13	5.05
Parameter QueryDef	3.19	5.88

*Declarations

Option Explicit

Dim db As database

Dim tbl As Table

Dim dyna As dynaset

Dim qd As querydef

Dim foo As Integer

Declare Function GetTickCount Lib "User" () As Long

Sub CloseDb()

On Error Resume Next

dyna. Close

tbl. close

qd. Close

db. Close

DoEvents

End Sub

Sub Command1 __ Click(Index As Integer)

Dim TimeCount As Long, TimeStart As Long

Dim foobar As Integer, foo As Integer

ReDim cost(5) As Double

TimeCount=0

*Set the hourglass

Screen. Mousepointer=11

Select Case Index

*Update Operations

*The objective here is to examine 1000 out of 10,000

*records, and modify those that match a given

*criteria—about 200 writes

Case 0

Frame3d1. Caption="Update Operation Benchmarks"

FileCopy"c:\vb\IoTest. Mdb",c:\vb\IoTest. bak"

*Seeks and indexed sequential updates.....

*Open the database logicals

OpenDb

*Start the clock

TimeStart=GetTickCount()

*Use transactions for speed

BeginTrans

*Jump to first ID using a fast seek

tbl. seek">=",5000

*Continue searching till eof or brdak

Do Until tbl. EOF

*Test if reached top of range

If tbl ! id<6000 Then

*Test if record meets change criteria

If tbl ! cost >30 And tbl ! cost<40 Then

tbl. Edit

tbl ! cost=tbl ! cost *.9

tbl. Update

tbl. Update

End If

Else

*Exit if past top of range

Exit Do

End If

*Get the next record

tbl. MoveNext

Loop

CommitTrans

*Stop the clock and report

TimeCount=GetTickCount()-TimeStart

Text1(0)=Format\$(TimeCount/1000,"# #0.00")

CloseDb

FileCopy"C:\vb\IoTest. bak",c:\vb\IoTest. mdb"

OpenDb

*Start The Clock

TimeStart=GetTickCount()

*Use transactions for speed

BeginTrans

*Jump to first record

dyna. FindFirst"Id">=5000"

*Continue searching till eof or braeak

Do Until dyna. EOF

*Test if reached top of range

If dyna ! id<6000 Then

*Test if record meets change criteria

If dyna ! cost >30 And dyna ! cost <40 Then

dyna. Edit

dyna ! cost=dyna ! cost *.9

dyna. Update

End If

Else

*Exit loop if past top of range

Exit DO

End If

```
'Get the next record
dyna.MoveNext
loop
CommitTrans
'Stop the clock and report
TimeCount=GetTickCount()-TimeStart
Text1(1)=Format$(TimeCount/1000,"##0.00")
CloseDb
'Test SQL processing with inline VBcode....
'Start each iteration with same file
FileCopy "c:\vb\joTest.bak","c:\vb\joTest.mdb"
'Open the database logicals
OpenDb
'Start The Clock
TimeStart=GetTickCount()
'Execute the statement on database DB
db.Execute"Update testtable Set Cost=Cost*.9—
    where cost>30 and cost<40 and id>=5000 __
    and id<6000 and id/2=id\2"
'Stop the clock and report
TimeCount=GetTickCount()-TimeStart
Text1(2)=Format$(TimeCount/1000,"##0.00")
CloseDb
'Start each iteration with same file
FileCopy "c:\vb\joTest.bak","C:\vb\joTest.mdb"
'Open the database logcals
OpenDb
'Store the procedure before startiong the clock
Set qd=db.CreateQueryDef("TestQd","Update testtable __
    Set Cost=Cost*.9 where cost>30 and __
    Cost<40 and id >=5000 and id <6000")
qd.Close
'Start The Clock
TimeStart=GetTickCount()
'Use Transactions for speed
TimeCount=GetTickCount()-TimeStart
'Use Transactions for speed
BeginTrans
'Execute the querydef on database DB
db.Execute "TestQd"
CommitTrans
'Stop the clock and report
Text1(3)=Format$(TimeCount/1000,"##0.00")
'Remove the querydef—normally we wouldn't do this
db.DeleteQueryDef"TestQd"
CloseDb
'ParameterQueryDef.....
'Start each iteration with same file
FileCopy "C:\vb\joTest.bak","C:\vb\joTest.mdb"
'Open the database logicals
OpenDb
'Store the procedure before startiong the clock
'Set qd=db.CreateQueryDef("TestQd","Parameters __
    IdSearch1 Long, IdSearch2 Long, CostSearch1 __
    IEEEDouble, CostSearch2 IEEEDouble; Update __
    testtable Set Cost=Cost*.9 where cost>__
    CostSearch1 and cost<CostSearch2 and __
```

```
    id>=IdSearch1 and id<IdSearch2"")
qd.Close
'Start the clock
TimeStartx=GetTickCount()
BeginTrans
'Open the querydef
Set qd=db.OpenQueryDef("TestQd")
'Set the paramter values
qd ! IdSearch1=5000
qd ! IdSearch2=6000
qd ! CostSearch1=30
qd ! CostSearch2=40
'Execute the querydef
qd.Execute
CommitTrans
'Stop the clock and report
TimeCount=GetTickCount()-TimeStart
Text1(4)=Format$(TimeCount/1000,"##0.00")
'Remove the querydef—normally we wouldn't do this
qd.Close
db.DeleteQueryDef"TestQd"
CloseDb
'Select Operations =====
'The objective of this benchmark is to read selected
'records from X to Y. For seeks and findfirst we need to
'get to X, then move through the records testing the
'Criteria.
'For QueryDefs we need to select the desired records
'and then scan them all to achieve a similar result.
'Our Criteria is all records with an even numbered ID
'between 5000 and 5100, for example 5000, 5002, 5004, etc.
Case 1
Frame3d1.Caption="Select Operation Benchmarks"
'Initialize and open the database logicals
OpenDb
'Table Seek.....
'Start the clock
TimeStart=GetTickCount()
'Repeat the operation to get a meaningful time
For foo=1 To 10
    'Position to the first record with a seek
    tb1.Seek="",5001
    'Look at each record
    For foobar=1 To 98
        If tb1 ! id/2=tb1 ! id\2 Then
            'do something if it is even
        End If
    tb1.MoveNext
Next foobar
'Reset for next Seek
tb1.Seek="",1
Next foo
'Stop the clock and report
TimeCount=GetTickCount()-TimeStart
Text1(0)=Format$(TimeCount/1000,"##0.00")
dyna.MoveFirst
```

```
TimeStart=GetTickCount()  
dyna.FindFirst"Id=5000"  
For foo=1 To 99  
    If dyna ! id /2=dyna ! id\2 Then  
        'do something if it is even  
    End If  
dyna.MoveNext  
next foo  
TimeCount=GetTickCount()-TimeStart  
Text1(1)=Format$(TimeCount/100,"##0.00")  
'Inline VB SQL Code.....  
'Start The Clock  
TimeStart=GetTickCount()  
For foo=1 To 10  
    'Execute the statement on database DB  
    dyna.Close  
Set dyna=db.CreateDynaset("Select * From __  
    Testtable where Id>=5000 and Id<5100 __  
    and Id\2=Id/2")  
Do Until dyna.EOF  
    dyna.MoveNext  
Loop  
Next foo  
'TimeCount=GetTickCount()-TimeStart  
Text1(2)=Format$(TimeCount/1000,"##0.00")  
Set qd=db.CreateQueryDef("TestQd", __  
    "Select * From TestTable where Id>=5000 __  
    and Id<5100 and Id\2=Id/2")  
qd.Close  
'Start The Clock  
TimeStart=GetTickCount()  
For foo=1 To 10  
    'Execute the querydef on database DB  
    dyna.Close  
Set dyna=db.CreateDynaset("TestQd")  
Do Until dyna.EOF  
    dyna.MoveNext  
Loop  
Next foo  
'Stop the clock and report  
TimeCount=GetTickCount()-TimeStart  
Text1(3)=Format$(TimeCount/1000,"##0.00")  
'Remove the QueryDef  
db.DeleteQueryDef("TestQd")  
CloseDb  
OpenDb  
Set qd=db.CreateQueryDef("TestQd","Parameters __  
    IDSearch1 Long,IDSearch2 Long;Select * __  
    From Testtable where id>=IDSearch1 __  
    and Id<IDSearch2 and Id\2=Id/2")  
qd.Close  
TimeStart=GetTickCount()  
For foo= 1 To 10  
    'Open the querydef  
Set qd=db.OpenQueryDef("TestQd")  
    'Set the paramter values  
qd ! IdSearch1=5000
```

```
qd ! IdSearch2=5100  
    'Build the dynaset from the querydef  
dyna.Close  
'Note the required alternate syntax!  
Set dyna=qd.CreateDynaset()  
Do Until dyna.EOF  
    dyna.MoveNext  
Loop  
'Close the querydef  
qd.Close  
Next foo  
'Stop the clock and report  
TimeCount=GetTickCount()-TimeStart  
Text1(4)=Format$(TimeCount/1000,"##0.00")  
'Remove the QueryDef  
db.DeleteQueryDef("TestQd")  
CloseDb  
'Populate the DB Test Set  
Case 2  
    Frame3d1.Caption=""  
'Initialize costs so we'll have a nice variety  
cost(1)=19.95  
cost(2)=29.95  
cost(3)=39.95  
cost(4)=49.95  
cost(5)=9.95  
'Use transactions for speed  
BeginTrans  
OpenDb  
db.Execute"Delete * from TestTable"  
For foo=1 To 10000  
    foobar=foobar+1  
    if foobar>5 Then foobar=1  
    tbl.AddNew  
    tbl ! cost=cost(foobar)  
    tbl ! id=foo  
    tbl.Update  
Next foo  
CommitTrans  
CloseDb  
'Exit the program  
Case 3  
    Soreen.MousePointer=0  
End  
End Select  
Soreen.MousePointer=0  
End Sub  
Sub OpenDb()  
On Error GoTo 0  
    'Open the logicals  
Set db=OpenDatabase("C:\vb\JoTest.mdb")  
Set tbl=db.OpenTable("TestTable")  
tbl.Index="Idkey"  
Set dyna=db.CreateDynaset("TestTable")  
End Sub
```

(本文英文资料由 Microsoft 提供)

SKILL

Turbo Pascal 与汇编语言混合编程技术

□董占山

Turbo Pascal 是 80 年代开始流行的优秀的 Pascal 编译系统,被广大编程人员所使用,它以编译速度快、生成的目标代码高速和紧凑而著称。在大多数情况下,只使用 Turbo Pascal 即可以完成各种各样的程序编制工作,但是,在硬件接口程序、实时控制程序及大规模浮点运算时,都需要用汇编语言来编程。虽然 Turbo Pascal 提供了 INLINE 语句和命令,以及内在汇编程序(Turbo Pascal 6.00),但这是远远不够的。本文详细讨论了 Turbo Pascal 与汇编语言混合编程的技术,并列举了大量的实例。

一、Turbo Pascal 的调用协定

Turbo Pascal 程序在调用汇编子程序时,要涉及到子程序的调用方式、函数或过程的参数传递方法和函数如何返回值的问题,现分述如下。

1. 调用子程序的方式和子程序的返回方式

Turbo Pascal 程序在调用汇编子程序时,可以是近调用也可以是远调用,因此, Turbo Pascal 程序在对汇编子程序进行调用时,根据调用方式的不同,有两种不同的保存返回地址的方法:①近调用时,因是段内调用,仅将偏移地址 IP 入栈,占 2 字节;②远调用时,因是段间调用,要将代码段值 CS 和偏移地址 IP 入栈,占 4 字节。

在主程序中直接调用汇编子程序时,一般采用近调用,汇编子程序采用近返回方式,用 RET 指令;在 Turbo Pascal 的单元中使

用汇编子程序时分两种情况:①在单元接口部分说明的子程序,在汇编子程序中要用远返回,用 RETF 指令;②在单元解释部分说明的子程序,汇编子程序要用近返回方式,用 RET 指令。

汇编子程序在运行结束后,为了能正确地返回到调用程序,栈顶指针必须指向正确的返回地址,它通过在返回指令 RETF(或 RET)中给出参数入栈时所占的字节数的方法实现的。

2. 参数传递的方法

Turbo Pascal 是利用堆栈向过程和函数传递参数的,参数按从左到右的顺序被压入堆栈中,例如调用过程 PROC(A,B,C:INTEGER;VAR D)时,其堆栈情况见图 1。



图 1 Turbo Pascal 远调用汇编程序 PROC 的堆栈情况

Turbo Pascal 在调用子程序时,有两种传递参数的方法,即传值和传地址。下面分别说

明这两种参数传递方法。各种类型参数入栈的方法见表 1。

(1) 传值方式

在 Turbo Pascal 的过程或函数的形式参数表中,以值参形式定义的参数,且类型是记录、数组、字符串、指针等复合类型以外的各种类型,如字节型 (BYTE)、短整型 (SHORTINT)、整型 (INTEGER)、字型 (WORD)、长整型 (LONGINT)、字符型 (CHAR)、布尔型 (BOOLEAN)、实数型 (REAL) 等, Turbo Pascal 在调用子程序时,直接将实参值依次从左到右顺序压入堆栈中,汇编子程序可以直接从堆栈中取得实参的值。

(2) 传地址方式

在 Turbo Pascal 的过程或函数的形式参数表中,以变量形式定义的参数及以记录、字符串、数组、指针等复合类型定义的值参, Turbo Pascal 在调用子程序时,是将调用程序的实参地址依次按从左到右的顺序压入堆栈的。汇编子程序从堆栈中取得实参的地址,即可得到参数的值。同样,汇编子程序可以把运算结果存放到对应的变量中,以便传回调用程序。

表 1 各种类型参数入栈的方法

形参类型	传递方式	栈中字节数
char, boolean byte, shortint, integer, word	传值	2
longint, single	传值	4
real	传值	6
double	传值	8
string, pointer 变量	传地址	4

3. 函数返回值的传递

Turbo Pascal 函数返回值的传递方式根据函数返回值类型的不同而异,有采用传地址的方式进行的,也有采用寄存器方式进行的,如采用传地址的方式,其地址(4 字节)首先入栈,然后才压入函数参数,最后压入函数的返回地址。各种函数返回类型的传递方式见表 2。

表 2 各种函数返回类型的传递方式

函数返回类型	返回方式	所占字节数
boolean, byte, char, shortint	在寄存器 AL 中	1
word, integer	在寄存器 AX 中	2
longint, real, string	在 DS:SI 指的地址中	4

二、汇编子程序的编写格式

根据 Turbo Pascal 的调用协定,外部汇编子程序的通用编写格式如下:

```
TITLE 程序名
DOSSEG
LOCAL @@
.MODEL TPascal
.CODE
ASSUME CS:@CODE
PUBLIC 过程或函数名
```

过程或函数名:

```
PUSH BP
MOV BP, SP
...
POP BP
RETF 参数占堆栈字节
END
```

上述汇编子程序是 Turbo Assembler 的格式,本文汇编子程序均采用这种格式,对此汇编子程序格式说明如下:

1. 汇编模块要采用 TPASCAL 模式;
2. 在汇编模块中,必须把 Turbo Pascal 调用的过程或函数说明为 PUBLIC 属性;
3. 子程序返回指令视具体情况而定,近调用用 RET,远调用用 RETF;
4. 返回指令后的参数是指该子程序形式参数表中所有参数入栈后所占堆栈的字节数;
5. 汇编模块结束要写 END。

三、Turbo Pascal 程序的编写格式

在 Turbo Pascal 中,声明外部子程序的格式如下:

```
procedure prc(a, b:integer; var c: real);external;
function func(a, b:integer):real; external;
```

即在通常的 Turbo Pascal 过程或函数的声明后加上 external 关键字。在声明了外部过程或函数的主程序或程序单元中,要用编译指令 {\$L},把汇编好的目标模块加载进来。

在 Turbo Pascal 程序中使用外部汇编过程或函数时,方法和一般的 Turbo Pascal 过程和函数没有两样。

四、主程序中使用外部汇编子程序的典型例子分析

在 Turbo Pascal 主程序中直接使用外部汇编子程序时,一般采用近调用方式,所以汇编子程序返回

1. 无参数传递的过程

```
program prog1;
{ $L prog1.obj}
procedure DisplayOk; external;
begin
指令为 RET, 在特别指明采用远调用方式时, 要用 RETF 返回指令。
    DisplayOk;
end.
Title PROG1
LOCAL @@
DOSSEG
.MODEL TPascal
.CODE
ASSUME CS:@CODE
OkMsg db 'OK 1', odh, 0ah, '$'
; Procedure DisplayOk
PUBLIC DisplayOk
DisplayOk:
push ds ; 保存数据段
push cs ; 代码段入栈
pop ds ; 弹出数据段
mov ah, 09 ; 显示字符串
mov dx, offset OkMsg ; 字符串地址
int 21h ; DOS 功能调用
pop ds ; 恢复数据段
ret ; 近返回
end ; 汇编子模块结束
```

2. 传递字符型值参的过程

```
program prog2;
{ $L prog2.obj}
procedure Displayint(ch : char); external;
begin
    Displayint('a');
end.
Title PROG2
LOCAL @@
DOSSEG
.MODEL TPascal
.CODE
ASSUME CS:@CODE
; Procedure Displayint
PUBLIC Displayint
Displayint:
    Push bp
    mov bp, sp
    mov ah, 02 ; 显示字符
    mov dl, [bp+4] ; 从栈中取参数
    mov 21h ; DOS 功能调用
    pop bp
    ret 2 ; 形式参数在栈中占 2 字节
end
```

3. 传递字符型值参和整型变参的过程

```
program prog3;
```

```
{ $L prog3.obj}
procedure ProcArg(ch : char; var i : integer); external;
var i : integer;
begin
    ProcArg('d', i);
    writein(i);
end.
Title PROG3
LOCAL @@
DOSSEG
.MODEL TPascal
.CODE
ASSUME CS:@CODE
; Procedure ProcArg
PUBLIC ProcArg
ProcArg:
    push bp
    mov bp, sp
    xor ax, ax
    mov al, byte ptr [bp+8] ; 取字符参数
    les si, [bp+4] ; 取整数变量的地址
    pop bp
    ret 6 ; 形式参数在栈中占 6 字节
end
```

4. 传递字符值参返回整型的函数

```
Program prog4;
{ $L prog4.obj}
function func(ch : char) : integer; external;
begin
    writeln(func('a'));
end.
Title PROG4
LOCAL @@
DOSSEG
.MODEL TPascal
.CODE
ASSUME CS:@CODE
; Procedure func
PUBLIC func
func:
    push bp
    mov bp, sp
    xor ax, ax
    mov al, byte ptr [bp+4] ; 取字符参数值
    pop bp
    ret 2 ; 形式参数在栈中占 2 字节
end ; 子程序返回值在寄存器 AX 中
```

5. 传递字符串型值参和返回字符串型的函数

```
Program prog5;
{ $L prog5.obj}
function func(s : string) : string; external;
begin
    writeln( func('abcd') );
```

```

end.
Title PROG5
LOCAL @@
DOSSEG
.MODEL TPascal
.CODE
ASSUME CS:@CODE
;Procedure func
PUBLIC func
func:  push bp
      mov bp,sp
      push ds
      xor ch,ch
      lds si,[bp+4] ;取字符串 S 的地址
      les di,[bp+8] ;取返回值地址
      mov cl,[si]
      inc cl
      cld
@@@1:  lodsb
      stosb
      loop @@@1
      pop ds
      pop bp
      ret 4; 字符串 S 的地址在栈中占 4 字节
end

```

6. 传递长整型值参和返回长整型的函数

```

program prog6;
{$L prog6.obj}
function func(li :Longint) :Longint; external;
var i :longint;
begin
  i:=func(111111110); writeln(i);
end.
Title PRLG6
LOCAL @@
DOSSEG
.MODEL TPascal
.CODE
ASSUME CS:@CODE
; Procedure func
PUBLIC func
func:  push bp
      mov bp,sp
      mov ax,[bp+4] ;取长整型数高位
      mov dx,[bp+6] ;取长整型数低位
      les si,[bp+8] ;以函数返回地址
      mov es:[si],dx
      mov es:[si+2],ax
      pop bp
      ret 4 ;长整型数 LI 在栈中占 4 字节
end

```

7. 传递实型数值参和返回实型数的函数

```

program prog7;
{$L prog7.obj}

```

```

function func(r :real) :real;external;
var r: real;
begin
  r :=func(11111.1110);
  writeln(r);
end.
Title PROG7
LOCAL @@
DOSSEG
.MODEL TPascal
.CODE
ASSUME CS:@CODE
;Procedure func
PUBLIC func
func:  push bp
      mov bp,sp
      mov ax,[bp+4] ;取实数 R 的值
      mov bx,[bp+6]
      mov dx,[bp+8]
      les si,[bp+0ah] ;取函数的返回地址
      mov es:[si],dx
      mov es:[si+2],bx
      mov es:[si+4],ax
      pop bp
      ret 6;实数 R 在栈中占 6 字节
end

```

五、单元中使用汇编模块的情况

在下面的演示单元 DEMOU 中声明了两个外部汇编函数, p1 是在单元接口部分定义的, 在汇编模块中采用远返回方式, p2 是在单元的解释部分声明的, 在汇编模块中采用近返回方式。在单元 DEMOU 的过程 p3 中又调用了函数 p1 和 p2, 调用 p2 采用近调用, 没有问题; 当调用 p1 时, 因其是在接口部分定义的, 必须采用远调用方式, 这可以用编译指令 { \$F } 来完成, 在调用 p1 之前, 加上 { \$F+ }, 在调用之后, 加上 { \$F- } 即可。

```

program prog8;
uses demou;
begin
  if p1(1) then Writeln('Far call complete 1');
  P3;
end.
unit demou;
interface
function p1( a: integer) :boolean;
procedure p3;
implementation
{$L demou.obj}
function p1( a: integer) :boolean; external;
function p2( a: char) :boolean; external;
Procedure p3;
begin

```

```

        if p2('a') then writeln('Near call complete 1');
        { $F+ }; 打开远调用编译指令
        if p1(1) then Writeln('Far call again 1');
        { $F- }; 关闭远调用编译指令
end;end.

Title DEMOU
LOCAL @@
DOSSEG
.MODEL TPascal
.CODE
ASSUME CS:@CODE
;function p1
PUBLIC p1
p1:      push bp
        mov bp,sp
        xor ax,ax
        cmp ax,[bp+4]
        jnz @@1
        mov al,0
        jmp @@2

```

```

@@1:      mov al,1
@@2:      pop bp
        retf 2 ;此函数在单元接口部分定义
;function p2
PUBLIC    p2
p2:      push bp
        mov bp,sp
        mov ax,'a'
        cmp ax,[bp+4]
        jnz @@3
        mov al,0
        jmp @@4
@@3:      mov al,1
@@4:      pop bp
        ret 2;此函数在单元解释部分定义
end

```

本文详细介绍了 Turbo Pascal 与汇编语言混合编程的技术,并给出了许多典型实例,读者可参照实例的格式进行混合语言编程。

SKILL

(上接第 25 页)

```

        i=(i<0)? 6:(i=1);
        putimage(45,12+i*18,Rar,XOR_PUT);
        break;
case 80:
        putimage(45,12+i*18,Rar,XOR_PUT);
        i=(i>=6)? 0:(i+1);
        putimage(45,12+i*18,Rar,XOR_PUT);
        break;
}

setviewport(vide __ x,vide __ y,vide __ x+70,vide __ y+155,1);
putimage(0,0,b0,COPY_PUT);
free(b0)
}

void graph __ f()
{
    setcolor(4);
    setfillstyle(SOLID_FILL,4);
    circle(150,130,20);
    floodfill(150,130,4);
    Rar1=malloc(imagesize(130,120,170,160));
    getimage(130,110,170,160,Rar1);
}

void graph __ f1()
{
    struct fillsettingstype save;
    char savepattern[8];
    char gray50[]={0xf0,0xf0,0xf0,0xf0,0xf0,0xf0,0xf0,0xf0};
        setbkcolor(3);
        setfillpattern(gray50,11);
        picslice(100,100,0,270,18);
    Rar2=malloc(imagesize(80,80,120,120));
    getimage(80,80,120,120,Rar2);
}

```

SKILL

(上接第 27 页)

```

{
    for (j=0;j<6;j++)
        sTable[j]=Table[i][j];
    sTable[6]='\0';
    compare=strcmp(lnStr,sTable);
    if (compare==0)
    { Find=i;
      break; }
    }
    if (Find==No)
    { printf("\n No Find this pinyin!");
      exit(0);
    }
    if (Find==0)
    { address=FirstAddress;
      cLen=LastOffsetAddress[0]-FirstAddress;
    }
    else
    { address=LastOffsetAddress[Find-1];
      cLen=LastOffsetAddress[Find]-LastOffsetAddress[Find-1];
    }

    /* 得到拼音为 lnStr 的汉字组在文件中的起始偏移地址 address 以及长度 cLen */
    cStr=(unsigned char *)malloc(sizeof(char)*(cLen+1));
    fseek(fp,address,SEEK_SET);
    fread(cStr,sizeof(char),cLen,fp);
    cStr[cLen]='\0';

    /* 提到汉字组 cStr[] */
    printf("The Chinese is %s",cStr);
    free(cStr);
    fclose(fp);
}

```

SKILL

scanf()函数的缺陷及解决措施

□夏大松

用

C 语言编制程序,当需要按照某种格式从标准输入中读取数据时,常调用标准库函数 scanf()。将表示数字量的字符转换成数字量。但在实际使用过程中。当程序编制欠妥或输入不严格时,就会出现错误。请看下面二例:

例一中,虽然①,②是两条完全一样的 scanf()函数语句,但执行结果却不同,这是因为 scanf()函数对输入流的要求很严格,如输入流中的字符与控制字符串的定义冲突(不一致),将出现该字符不能被正确转换并赋给参数变量的情况,但它们仍被保留在 stdin 中。由于第一次调用 scanf()时键入的回车换行符不能赋给变量 b(类型不一致),因此 '\n' 仍留在 stdin 中,当第二次调用 scanf()输入时,'\n' 则作为第一个输入字符赋给变量 a,而本次输入的“t 567”中的字符 't' 又与变量 b 的类型冲突,未能正确转换,此时,a 值为 '\n',用 printf()输出时,造成回车换行,b 的值仍是原来的值 345,因此得到不正确的结果,解决办法是将②改成:scanf("%c%c%d",&a,&b);或在语句②的前面加上 getchar(),利用此函数从 stdin 中吸收掉 '\n'。

而例二中,循环语句中反复调用 scanf()接收单个字符时,把接收单个字符定义为接收 2 个元素以上的字符型数组(留出存放 '\0' 或 '\n' 的位置),然后使用格式控制“%1s”从 stdin 中提取一个字符,这样也可有效地解决回车换行符 '\n' 无处存放的问题。但在输入数据时仍必须严格按控制字符串的要

求输入字符,否则将导致不正确的结果。

例一:

```
#include <stdio.h>
main()
{char a;int b;
scanf("%c%d",&a,&b);/* ①语句 */
printf("%c%d\n",a,b);
scanf("%c%d",&a,&b);/* ②语句 */
printf("%c%d\n",a,b);
}
```

〈运行结果〉

a 345	✓	①第一次输入
a 345		结果正确
t 567	✓	②第二次输入造成换行
345		结果不正确

例二:

```
#include <stdio.h>
main()
{char a[3],b[3];int k;
for (k=0;k<3;k++){
scanf("%1s%1s",&a,&b);
printf("a=%s b=%s\n",a,b);
}
}
```

〈运行结果 1〉

```
12 ✓ 正确的输入
a=1 b=2
ac ✓
a=a a=c
89
a=8 a=9
```

〈运行结果 2〉

```
248 ✓ 错误的输入
a=2 b=4
abc ✓
a=8 b=a 结果不正确
a=b b=c 结果不正确
```

SKILL

用__ Config 变量方便地跟踪每个用户的配置情况

□刘耀东

几乎所有商业应用都允许用户保留诸如颜色、声音及打印驱动程序的程序设置情况。实际要保留的信息在不同的程序之间有所不同，尽管管理的任务基本相同。这里，我们编写了一个叫做__ Config 的子程序，在保留选择何种应用信息将要存储时，该子程序可以压缩存储和检索信息。这是一个我们从面向对象设计(OOP)中借鉴的一些思想的极好范例，同时也是备注字段的有趣应用。

__ Config 的使用

__ Config 描述如下：

```
= __ Config("PUT", <User>, <Setting>, <Info>)
```

用于保存用户名为<User>的值<Setting>的设置<Info>；以及

```
= __ Config("GET", <User>, <Setting>, <Info>)
```

用于检索用户<User>的设置<Setting>和<Info>中保存的值。下面是我们设置屏幕的一些摘录，以便说明我们如何用__ Config 来阅读和保存声音设置。

```
PRIVATE PcUserName
PcUserName=cWhoAml()      &&Or whatever the
                             && application uses

* Get Sound setting.
PRIVATE PcSound
PcSound="ON" &&"Factory" default
PnStatCode=;
    __ Config ( " Get ", PcUserName," Sound ",
                @PcSound)
* * Get configured sound setting.
IF PnStatCode <. 0          &&If not set yet.
    = __ Config (" Get ", "~SYSTEM~", " Sound",
```

```
@PcSound)
* * Get system default,remains "ON"
* * if system not set .
    = __ Config ( " Put ", PcUserName," Sound ",
                PcSound)
* * Save it for current user
ENDIF
SET Bell      &PcSound
?"Old sound setting is "+set("bell")
*
* ...edit PcSound...
*
* Save Sound setting.
= __ Config ( " Put ", PcUserName," Sound ", ALLTRIM
                (PcSound))
SET Bell PcSound
?"New sound setting is "+ set ("bell")
```

我们已发现超出最初设想的__ Config 的种种用途，其中的一个应用是我们需要跟踪最后使用系统的用户。利用__ Config 则变得非常简单：

```
* LastUsed. Prg
* Print date user last used program.
* Use only once at start of program.
PRIVATE PcUserName
PcUserName=cWhoAml()      &&Or whatever
                             && application uses

PRIVATE PdLastUsed
PdLastUsed=(/ /)
PnStatCode=__ Config("GET",PcUserName,;
                    "LASTUSED",@PdLastUsed)
* * Get date last used.
IF PnStatCode<0
    ?"First Time User."
ELSE
    ?"Last Login:" +DtcC (PdLastUsed)
```


ENDIF

```
=_ Config("PUT", PcUserName, "LASTUSED", Date())
```

实现方法

创建一个简单(唯一)的 __ Config.dbf 表以保存程序设置。设置用户和用户识别保存的信息字段 FcCfgID 和 FcUser。由于 __ Config 并不知道数据类型,故实际数据的储存有些技巧,通过使用 "Save To Memo" 和 "Restore From Memo" 命令来克服这一困难,以便数据存储在备注字段内而不管其原来的类型如何。若欲储存和检索的数据为一数组,则一定要用 ACOPY() 来代替简单的赋值,故这种情形一定要分别处理。利用 TYPE() 函数来确定存储或检索的数据是数组还是标量。要处理的代码在子程序 PCOPYX 中。

附带的源程序代码说明了将 GET 和 PUT 任务结合在单一的函数里。在这两种任务里,在读或写数据之前,需要查找 __ Config 中合适的记录。由于两种任务放在同一函数里只须把公用代码放在开头部分。

附加特征

基本的子程序可以在几个方面加以扩展。可以:

- 将文件名做为参数传递来代替对 __ Config 编码。
- 使用 Save Macros to Memo 和 Restore Macros to Memo 来追加 PUT MACRO 和 GET MACRO 命令允许用户定义热键。
- 加入 CHECK 命令,以便确定设置是否已被保存,但不对其值进行检索。
- 加入 CLOSE 命令,以关闭 __ CONFIG.DBF。
- 使用错误处理。

追加一条新命令只不过是主 Case 语句中追加另一个 Case。

__ Config 服从如下语法:

```
<Return Code> = Function(<Command>, <Parameters>)
```

这里 <Command> 是和某一活动有关的有限的一组任务中的一个。除了将公用代码结合起来,这样做是有好处的。一个是允许你在完成某一活动(功能)的程序代码和程序的其余部分之间定义一个清晰的接口,以减少公用信息的数量。对 __ Config 而言,活动是管理程序设置中的信息。任务有:

- 保存某一用户程序设置的值;
- 检索某一用户程序设置的值。

这确定了 __ Config 中使用的 GET 和 PUT 命令。程序的其余部分并不知道 __ Config 如何存储或检索信息或任何有关其内部工作的其他事情。

另一个好处是所有相关的代码存储在一个自我包含的源文件中,而不要使用 Set Procedure to 命令。这样,当必须进行改变时,所有相关的代码都尽展眼前。

80%的面象对象程序设计

对那些熟悉面向对象程序设计的人,在一个单一函数内可以把使用命令看成和附加的抽象数据类型的方法相类似。这离真正面向对象语法还有很长的路要走,但或许正是这 20% 的 OOP 将产生 80% 的好处。

```
* -----
* FUNCTION __ Config
*
* SYNTAX
* =_ Config(McCmd, McUser, McPar, @MxData)
*
* PARAMETERS
*   McCmd—Confirmation command="PUT" or "GET"
*   that put and get data in the second parameter.
*
*   McUser — User to whom the information applies,
*   "—SYSTEM—"for system data
*   MxData — Data to put ,or data retrieved—
*   LEFT ALONE IF NOT FOUND Pass by reference.
*
* RETURN
*   0=OK, -1=Data not found,
*   -99=Invalid code
*
* EXAMPLE
```

```
* =_ Config("GET", WhoAml(), "Printer", __ Curptr)
* -----
```

```
Function __ Config
Parameters McCmd, McUser ,McPar, MxData
* Save work area to restore on exit.
PRIVATE PcOldSel
PcOldSel = Select()
* Save Exclusive setting to restore on exit
PRIVATE PcOldExcl
PcOldExcl = Set("Exclusive")
SET Exclusive OFF      &&. Will open __ Config
                        &&. nonexclusively.

PRIVATE PnRtnCode
PnRtnCode=0  &&. default
```

香港金山公司
北京金山软件公司
经营: DEC、Hp 代理, 方正系列汉卡
地址: 100088 北京海淀知春路 22 号北京金山软件公司
电话: 2049624, 2040009—3035, 2061869
传真: 2061869

```

* Open the file
IF ! Used(' __ Config')
    * CREATE THE FILE IF IT IS NOT THERE
    IF ! File(' __ Config.DBF')
        CREATE Table __ Config;
        * (FcUser C(10),FcCfId C(20),FmData M)
        * * Insert an index command here if you anticipate
        * * a large number of records. this will enable
        * * optimization in the locate command used below.
        USE in __ Config
    ENDIF
    * Once open, the file will remain open until
    * Program exit, this speeds execution.
    SELECT (SETEX( 1))
    USE __ Config again
ELSE
    SELECT __ Config
ENDIF

* Parameter conditioning
McCmd=IIF (TYPE ('McCmd')='c';
    UPPER(TRIM(McCmd)), '')
McUser=PadR(IIF (TYPE ('McUser')='C';
    ,UPPER(McUser), ''), LEN(FcUser))
McPar=PadR(IIF (TYPE ('McPar')='C';
    ,UPPER(McPar), ''), LEN(FcCfId))

* Find the record in __ Config.
LOCATE FOR __ Config. FcCfId=McPar and;
__ Config. FcUser=McUser
DO CASE
CASE McCmd=="PUT"
    * Make separate variable to store data.
    PRIVATE __ STORE __
    __ STORE __=.F.
    DO pCopyX with MxData, __ STORE __

* If no such Config id , then add it.
IF ! Found (' __ CONFIG')
    INSERT Into __ config (FcCfId, FcUser);
    Values (McPar, McUser)
ENDIF

* Now on the record, save the data,
IFRLOCK()
SAVE ALL Like __ STORE __ to Memo __ Config. FmData
ELSE
    PnRtnCode=-2    && Unable to lock record
ENDIF

* Clean up.
RELEASE __ STORE __
UNLOCK
CASE McCmd=="GET"
    * See if the data is there.
    IF Found(' __ Config')
        * Retrieve the data from the memo field. the data

```

```

* was previously saved as __ STORE __ the next
* line will create a variable __ STORE __, which
* contains the saved value.
RESTORE FROM Memo __ Config. FmData Additive
DO pCopyX with __ STORE __, MxData
    * Clean up.
    RELEASE __ STORE __
    * if data isn't there then set error code.
ELSE
    PnRtnCode=-1
ENDIF
OTHERWISE
    PnRtnCode=-99
ENDCASE

* Restore environment and exit.
IF PcOldExc1="ON"
    SET Exclusive ON
ELSE
    SET Exclusive OFF
ENDIF
SELECT (PcOldSel)
RETURN PnRtnCode
* -----
PROCEDURE pCopyX
Parameters MxSource, MxTarget
* Copy any type of data, including arrays, from
* MxSource to MxTarget. pass parameters by reference.
PRIVATE MxI    && index
DO CASE
CASE TYPE ( * MxSource) ="U"
CASE TYPE ("MxSource(1)") ="U"
    MxTarget=MxSource
OTHERWISE    && An array.
    * * Get an error if destination isn't the same
    * * Length as the source array.
    IF ALLEN(MxSource, 2)=0    && If 1-dimensional
        Dimension MxTarget(ALLEN(MxSource))
    Else && 2-dimensional
        Dimension MxTarget(
            ALLEN (MxSource, 1), ALLEN(MxSource, 2))
    ENDIF
ACOPY(MxSource, MxTarget)    && Copy the array
ENDCASE
RETURN

```

(本文英文资料由 Microsoft 提供)

SKILL

香港金山公司
北京金山软件公司
经营: DEC、Hp 代理, 方正系列汉卡
地址: 100088 北京海淀知春路 22 号北京金山软件公司
电话: 2049624, 2040009-3035, 2061869
传真: 2061869

搜寻复杂表达式

□刘耀东 译

你

曾经试图使用一个似乎是你正在寻找的 FoxPro 函数,结果却发现令你大失所望吗?在笔者需要 Lookup() 函数返回第一个和最后一个名字时笔者对此感触颇深。不幸的是,Lookup() 函数只返回单一的字段,因此笔者编写了可返回包括用户自定义函数的任意有效表达式的 elookup() 函数。对 elookup() 的简单调用如下:

```
m.lcFullname=elookup("ALLTRIM(Last-  
Name)+", "+FirstName", m.lcCustID)
```

这就是该函数的最简单用法,在此假定当前选定的别名是雇员表。上述调用的语法要求使用两个参数,但有四个可供选择的参数。

如果你试图传递一个表达式给 FoxPro 固有的 Lookup() 函数,则会弹出 "Invalid function argument value, type, or count" 这一错误信息。你可以搜寻顾客标识号 ID,然后把最后一个和第一个顾客的名字链接在一起。在不断反复执行类似的代码之后,最终形成了这一子程序。

elookup() 函数参数次序和作用类似于 Lookup() 函数的有关参数,但两个新参数可使你能够指定搜索的别名和存储记录指针的标志。除了要返回的字段和搜索值外, FoxPro 的 Lookup() 须命名欲进行搜索的第二个参数。第四个可选的参数指明了索引标志,但强制进行搜索而不进行定位。

elookup() 函数的六个参数为 Lookup() 函数的超集,但即便是类似的参数,作用也稍有不同,笔者让第三个参数成为可选的参数。如果该参数未进行传递,则只要激活索

引,函数就会完成一次搜索。传递欲搜索的字段名来产生一次查找。即便有活动索引,笔者也不用,以帮助 Rushmore 优化查询。传递第四个参数(欲使用的索引标志)通常意味着第三个参数变得毫无意义,这样,第三个参数可做为空串。

FoxPro 的 Lookup() 的一个限制是对当前工作区以外的操作无能为力,无法告诉它对哪个表进行搜索。将其与 seek() 函数进行对比, seek() 接受指向表的可选择参数。笔者通过使第五个参数可接受工作区别名而采用了这种技术。如果该工作区被打开,则进行选择。若未打开,则 elookup() 打开之,但遵循 FoxPro 的 SQL SELECT 语句的方式,以保留打开。无论那种方式,当前工作区予以保存,并在 elookup() 结束时予以恢复。最后, elookup() 的最后一个参数使得你返回搜索表中相对于搜索前初始位置的记录指针。FoxPro 无条件地将其定位于找到的记录,若搜索失败,则定位于库尾。若想让附加字段离开原记录,这样做是有意义的。

尽管 Lookup() 和 elookup() 函数中各个参数在本质上是相同的,还是存在一个关键的

北京市威特电子技术公司

威特家用电脑系列

教育系列软件

电话:2558868, 2577999

语法差异,你必须将返回的表达式(参数 1),搜索字段(参数 3)和标志名(参数 4)用引号括起来。否则, FoxPro 试图对它们进行求值,并将值传递给 elookup()——这通常会导致不可预见的结果例如:产生“variable not found”之类的错误信息。

elookup()的表达式计算的最为有用的副产品是支持用户的定义函数 UDF。很多开发者利用 UDF 来整理姓名、城市、州、邮政编码等很多其他公共的相互联系的事物。你可以告诉 elookup()执行一个 UDF,并得到其返回值。需要掌握的关键一点是在 elookup()到达所希望的记录(或文件尾)之前,UDF 进行计算。当调用 UDF 时,作为参数传递给 UDF 的字段名进行计算,而不管何时该 UDF 都做为参数传递给 elookup()。当然,整 UDF 调用(函数名和参数)在传递给 elookup()时,一定要用引号括起来。

若传递的是字符串常量,则必须使用不同的引号字符串(双引号,单引号):

```
m.lcCitystate=elookup("ALLTRIM(city)+'',';  
+ "state",m.lcZipCode,"zipCode")
```

可用 UDF 来代替:

```
m.lcCityState=elookup("fmtctyst(City,State)",;  
m.lcZipCode,"zipCode")
```

FoxPro 的 Lookup()函数在很多情况下是非常有用的,但笔者发现它在几个重要方面失效。当需求增加时,你会发现 elookup()函数是一个值得信赖的替代品。

```
* elookup—expression lookup  
* Maurice Frand,72167,736  
* 12/12/93 Version 1.0  
* Copyright Maurice Frank 1993
```

```
PARAMETERS m.tcRetExpr,m.tcSrchExpr,m.tcSrchFld,;
```

```
m.tcTagName,m.tcAlias,m.tlSaveRecno
```

```
PRIVATE lnSelect,lcOrder,lnRecno,lnPeriod,;
```

```
lnLeftparen,lnStart,lcAlias,lxRetVal
```

```
IF EMPTY(m.tcRetExpr) OR EMPTY(m.tcSrchExpr);
```

```
OR (EMPTY(m.tcAlias) AND EMPTY(ALIAS()))
```

```
* required parameters missing
```

```
* or no alias specified and no table selected
```

```
* we can't do anything,so bail out early
```

```
RETURN "" &&char type in most cases
```

```
ENDIF &&missing required parameters
```

```
m.lnSelect=SELECT()
```

```
IF NOT EMPTY(m.tcAlias)
```

```
IF NOT USE 'D'(m.tcAlias)
```

```
USE (m.tcAlias) IN 0
```

```
ENDIF
```

```
SELECT(m.tcAlias)
```

```
ENDIF
```

```
m.lnRecno=MIN(RECNO(),RECCOUNT())
```

```
m.lcOrder=ORDER()
```

```
IF NOT EMPTY(m.tcTagName)
```

```
SET ORDER TO (m.tcTagName)
```

```
SEEK m.tcSrchExpr
```

```
ELSE &&.no index tag specified
```

```
IF NOT EMPTY(m.tcSrchFld)
```

```
* a tag may be in effect,but is it the
```

```
* right one?
```

```
* turn off tag to rushmore optimize the locate
```

```
SET ORDER TO
```

```
LOCATE FOR EVAL(M.tcSrchFld)=m.tcSrchExpr
```

```
ELSE
```

```
&&.no search field Specified
```

```
* do a seek using the controlling index
```

```
* make sure there is a controlling index
```

```
IF NOT EMPTY(ORDER())
```

```
SEEK M.tcSrchExpr
```

```
ELSE &&.no controlling index
```

```
* the function was called incorrectly
```

```
* go to the phantom record at the
```

```
* end of file to get correct data
```

```
* type of return expression
```

```
LOCATE FOR .F.
```

```
ENDIF &&.not empty(order())
```

```
ENDIF &&.not empty(m.tcSrchFld)
```

```
ENDIF &&.not empty(m.tcTagName)
```

```
* lx means this memvar can hold data of any type
```

```
m.lxRetVal=EVAL(m.tcRetExpr)
```

```
* restore the record pointer if desired
```

```
IF m.tlSaveRecno AND m.lnRecno>0
```

```
GOTO m.lnRecno
```

```
ENDIF
```

```
SET ORDER TO (m.lcOrder)
```

```
SELECT (m.lnSelect)
```

```
RETURN m.lxRetVal
```

调用 elookup()函数的正常语法:

```
<retval>=elookup(<cRetExpr>,; (expression to return)  
<cSearchExpr>[,;(expression to search for)  
<cSearchField>[,;(field to search in)  
<clndexTag>[,;(index tag to use)  
<cAlias>[,;(work area alias to search in)  
<ISaveRec>]]] (true to restore record  
pointer)
```

(本文英文资料由 Microsoft 提供) SKILL

300 元省一台打印机

SXD 系列

全自动、半自动、多功能打印机共享器

地址:北京清华大学科学馆 302 邮编:100084

电话:2594866 联系人:魏宝英,张罗平

开发 FoxPro 数据仓库

□刘耀东 译

本

文要描述的不是一个程序或应用,而是构造 FoxPro 信息库的一个框架,或称数据仓库。它基本上是一个从面向处理(transaction-oriented)的捕捉新数据的系统分离出的只读数据库。它可以提取摘要和详细数据,数据分析人员据此能够查询和报告指定时间的事物状态。由于它在结构上同面向处理的系统相分离,为了进行高速检索,用户可以优化数据仓库而不减慢其他处理。

就主框架而言,在支持信息驱动事物中的决策时,数据仓库提供了管理大量数据的基础结构。大量信息量依赖于大量历史数据。在大型机构已经使用了这类决策支持(DS)数据库时,PC 的革命已经使得各种大小的事务能够捕捉大量可利用的数据,以确保竞争优势。由于其良好的文件查询速度,尤其是 FoxPro,使得在便宜的台式计算机上管理大量的数据存储成为可能。

仓库对象

有几种构造数据仓库的理由:首先,用户可利用象 FoxPro 的报告编写器和第三方的电子表格和分析软件包那样通过交互式工具来读写数据。由于数据仓库是只读的管理者并不需要担心用户会搞乱公司的数据。若授权的用户需要改变数据,可以给授权用户提供数据集合的拷贝而不必改变原始数据。

数据仓库可使用户能够合并多种应用、数据源及场合的数据。FoxPro 的可编程性允许用户编写顾客摘要及由其他的数据库应用、不同的数据文件格式、电子表格文件和关系数据库资源集成的数据的输入子程序(利

用可选择的 FoxPro 连接工具)。甚至能够利用 FoxPro 连接工具由主框架中提取数据,来同如 Trinzic 的 InfoHub(支持 ODBC)界面的“Middleware”进行对话。当然要做到这一点,应当从主框架系统管理员那里获得帮助。

数据仓库的最终目标是为数据分析人员做出正确的决策提供所需要的信息。访问大量的历史数据,可以跟踪趋势并进行预测,分析竞争者的行为,生成详尽的财务报告,监视重要的动态。

为什么不是实时的

在实时处理中从多场合、不匹配的文件格式中集成数据是不切实际的。维护远程连接、管理分布式处理、最低限度地减少网络信息传输量、保证可用性所遇到的问题,会逐步升级直至最终失去控制。甚至在一个简单的数据库中,单一的网络环境下,决策事务处理系统和决策支持系统(DS)混和起来会降低数据检索和更新的速度。例如,在 FoxPro 中完成的查询很快,并且在有很多个索引标志的情况下是前后一致的。当索引标志很多时,数据更新是缓慢的。

此外,还要考虑网络性能的问题。当处理系统要求可预见的、快速响应时,引进用户自行设计单元来查询系统是不明智的。最后,实时系统可能提供即时信息,但分析人员经常需要对机构数据的打印结果深思熟虑几小时、数天甚至几周。而数据仓库可提供与某一指定时刻相一致的数据。

在构造数据仓库之前

通常,修改现有的应用是不明智的,而利

用数据仓库计划新的应用是可能的。下面是为确保数据质量应遵循的步骤：

- 采用数据中心法进行开发。也就是说，将数据和应用分别设计。通常 FoxPro 数据库具有特定应用的结构或包含特定应用的代码。

- 考虑采用诸如规范化或实体关系模拟法等形式设计方法，以便数据结构超脱特定应用。

- 至少要使数据库文件化。在全部组织机构的范围内最大限度地创立开发者能够重新利用的标准数据结构。

构造数据仓库

构造数据仓库的过程包含下述四个步骤：

1. 建立逻辑数据模型；
2. 设计仓库数据库；
3. 编制由源数据文件转换至仓库数据库的“转换”子程序；
4. 装载和维护数据仓库。

建立数据模型包括显示已存在的数据，然后评价用户汇总需求概要。例如，假如系统每天捕捉了 5000 个处理，用户将只想看按周或月的汇总结果。在这种情形下，你可以概述原始数据，并且只输入汇总数据（每周或每月一次）。如果关系数据库服务器内已存在可操作的数据，可以利用简单的 SQL SELECT 命令来汇总数据。

用户的访问是建立数据模型的关键。与用户进行对话以估计原始数据的位置（在多个地点、在不同的应用中以及在不同的服务器上），识别出数据累积的关键事务区域。最重要的是，向用户询问原始数据的质量。

设计数据库包括识别数据仓库要捕捉的不同来源的文件和字段。此外，还要识别必须转换或重定格式的字段。例如数据源采用代码列表（部门代码），用户可能需要对其进行转换。这就提供了将部门代码 01、02、03 变成“销售部”、“市场部”和“管理部”的可能

在这一步骤，可以转换不兼容的数据类型，例如，部门代码在一个数据库内是按数值型存储的，在另一个数据库内则按字符型存储。在某些情况下，你可能必须用 FoxPro 的低级文件函数或外部的 C 语言子程序来将数据转换为可接受的数据类型。此外，第三方提供的产品可能会有助于转换工作，例如 Trinzic 的 InfoPump 可在不同的数据来源之间移动数据并同时将其转换为不同的格式。

用户也可以转换不一致的数据定义。例如，一个数据库可能将月份以 1 到 12 的数据来表示，另一个数据库可能用诸如 JAN, FEB, MAR 之类的月份缩

写等。在数据仓库内，可以建立月份的一致表示，而不管数据源。

在数据库设计过程中，用户一定要识别潜在的文件大小，注意 FoxPro 的记录极限为 1 亿或者表的大小为 2GB。

在这一阶段，一定还要确认数据转换的硬件要求。如果全部系统是由网络联在一起的则不必追加任何设备。对于远程用户而言，可使用经由标准电话线路的调制解调器或者是专用的高速线。这取决于如何对数据进行更新，公认的“SneakerNet”（由软盘或磁带加载数据）可提供合理的解决。在多用户服务的场合，只要向数据仓库的所在位置发送邮件即可。

更新周期

为确定仓库数据的更新区间，向用户确认如下问题：数据分析人员每分钟、每周或者每月需要新的数据吗？他们需要对数据仓库进行人工更新吗？在某些情况下，需要一份整个组织的打印结果。在其他的情形之下，你可能只想利用新的处理结果来更新数据仓库，或者利用来自特定应用的处理来更新数据仓库。最后，你一定要考虑网络性能及事物处理传输量对网络信息的影响。理想的形式是在网络活动最小时更新数据。更新区间和范围将部分地确定你能够有效利用的通信媒体。

转换数据

FoxPro 数据仓库系统的优点是用户可以利用 FoxPro 编程语言的灵活性来建立由不同的数据源移进数据仓库的转换程序。数据仓库框架要求的子程序很象任何其他 FoxPro 应用。

为了转换数据，调查是否存在能将数据转换为 FoxPro 格式的或者 FoxPro 容易输入的实用程序是必要的。如果市场上有这样的产品，重新编写是无意义的。例如，诸如 Q+E Database Library (Windows 下的) 的软件包能够检索很多数据源的数据并且以 FoxPro 能够理解的格式存放。

为了做到自力更生，FoxPro 提供了自身的 IMPORT 和 APPEND FROM 命令，提供了十六种源文件的格式。如果 FoxPro 不能直接输入一个外部文件，有可能将原始数据写成 FoxPro 能够以 APPENDED DELIMITED 形式追加的 ASCII 码文件。如果数据驻留在某一关系数据库服务器内，用户可以使用 FoxPro 可选择的连接工具库来检索数据、或者利用其他的库和驱动程序来检索数据。

在转换数据时，可以重新命名字段。如果你遇到列（字段）名为 A1、A2、A3 等等的原始数据，想要在数据仓库内赋予它们更多的描述名字。可以建立数据仓库词典以尽可能详尽地在 10 个字符字段名的

范围内来描述每个字段。不幸的是, FoxPro 最多只能接收 10 个字符以内的字段名。

下一步, 要使多个字段象前面在月份举例中描述的那样一致。应选择一个来代替月份定义的三种表示。

再下一步, 转换派生数据进各字段。例如, 如果数据分析人员需要仅看每周的平均销售额, 建立一个 AVGSALES 字段来保留汇总而不是在数据仓库内重新计算之, 预先计算派生字段有助于提高效率并节省磁盘空间。这是为检索诸如数据仓库的远程数据库而预留的“非正常”格式。

最后, 在完成了转换计划后, 再次确认。然后就可以编写子程序并加载数据仓库。

进一步的转换

在进行转换期间, 我们有机会进行几项其他的操作:

- 净化——可去掉备份并使字段更为有效。例如, 可以检测丢失和损坏的数值; 可以检索无效的州或划分美国邮区的五位号码, 或者能检测做好的有关的完整性。

- 时间标记版本——可以在数据仓库内提供记录更新日期和时间的版本字段。在一累积更新的表里, 进入数据仓库的新记录的时间/日期标志也是新的。用户然后能够检索指定更新的记录组。

- 跨越地点的数据复制——可以传播跨越多个地点的更新以便机构内任意地点的分析人员都有同样的数据。

决策支持优化

为了进行检索和查询, 数据更新不一定进行的很快。例如, 当用户在 FoxPro 中追加(APPEND)或

替换(REPLACE)一条记录时, 更新的速度取决于索引文件标志(.CDX)文件的数目。标志越多, 更新的越慢。另一方面, 标志越多, 查询的响应越快。

为了优化决策支持用户, 使用如下技术:

- 对每个字段建立索引以支持特定设计的查询。这甚至将提供最一致的和可预见的性能, 因为用户几乎不可能通过使用非优化表达式来打败 Rushmore 优化技术。

- 预连接(Prejoin)关系以简化用户的数据结构并改进执行情况。例如, 典型地用户保存了标题和细节, 然后利用 SET RELATION 或 SQL 的 SELECT 语句, 通过采用发票号来做好主关键字。在数据仓库内, 可以将这些表格合并起来以简化随后的查询并改进执行情况。这样的非规范化产生一张重复标题的每个细节的表, 这对想要做为“只读”文件的表来讲并不会成为一个问题。

- 利用 NOUPDATE 选择或 SQL SELECT 来检索数据。NOUPDATE 可改进性能, 因为 FoxPro 并不需要处理代码或涉及内存管理等系统内部的问题。

事务开放

从事务处理的立场来看, 数据仓库概念增强了事务处理的数据中心的观点, 并且使得能够得到为了进行更好地决策所要求的数据成为可行的, 它将数据库的概念模型和物理数据库设计分离开来, 并且将与应用相关的数据减少到最低限度。它在不同数据源和不同的地点的信息源之间架起了桥梁。简言之, FoxPro 数据仓库并不只是一个应用, 它是信息驱动应用开发的蓝图, 而无论是否完成了数据仓库。 (本文英文资料由 Microsoft 提供) **SKILL**

(上接第 78 页)

请求信号(CPU HRQ)是否正确。如果 CPU 的各个工作条件都满足, 但 CPU 却未工作(可通过测试 I/O 插槽中的各地址总线 and 数据总线信号来判断它是否工作), 说明 CPU 已损坏。

测量集成电路引脚的直流电阻

正常的集成电路芯片, 它的各个输入和输出脚对正 5 伏引脚(电源脚)或对地引脚的电阻都有一定的数值, 例如 74LS245 芯片, 将万用表置电阻档高档位(20K 欧档), 测得它所有的输入脚对电源脚的电阻应为 19K 欧左右, 而反向电阻应呈截止状态(电阻无穷大); 它的输出脚以电源脚的电阻为 10 至 20K 欧, 反向电阻为无穷大。如果发现某输入或输出脚与电源脚的电阻为零, 或与正常值不符, 则说明此芯片

已被击穿或者已丧失功能; 如果发现两个类似的输入脚或输出脚对电源脚的电阻值相差太大, 一般情况下也说明此芯片有故障。

升、降温法

遇到因机内某个芯片热稳定性差使机器不能正常工作的情况, 便可用升、降温法。比如, 某微机工作时间稍长一些, 由于机内某元件升温或者环境温度较高, 机器出了故障。这时可等到机器出现故障后, 用沾了酒精的棉球放在那些被怀疑是热不稳定的芯片上, 人为降低它的温度(降温法), 观察机器能否正常工作, 从而找出是哪些芯片热不稳定; 也可以关机待机器冷却后, 再开机, 用电吹风对机内一些芯片加热, 提高它的温升(升温法), 当加热某芯片时, 故障再现, 就说明此芯片已坏, 应更换。 **SKILL**

FoxPro 关系数据库的应用点滴

□于 光 马竹青

FoxPro 运行速度快、数据处理功能强、编程灵活,且与 dBASE III、dBASE IV 和 FoxBASE+ 兼容,深受广大用户的青睐。它不仅是较为理想的关系数据库管理系统,也是从事数据管理、事务管理、办公自动化的强有力工具。FoxPro 可以在 DOS 和 Windows 两种平台上使用,目前最高版本为 FoxPro for DOS 2.5 和 FoxPro for Windows 2.5。

下面,笔者结合长时间学习和使用 FoxPro 的经验从几个方面探讨 FoxPro 的设计和使用技巧。

一、实现用户帮助

FoxPro 有一个优点,就是可以方便地使用它的帮助系统,在 FoxPro 应用程序中实现用户帮助。FoxPro 的扩展帮助系统提供了一组索引的屏幕,当按 F1 键时就会出现。这些帮助屏幕实际上是数据库 FOXHELP.DBF 的备注型字段。然而,这些复杂的帮助屏幕对运行应用程序本身一般没有直接的益处,因为这些标准 FoxPro HELP 屏幕并不能告诉用户关于应用程序的任何信息。我们可以用 SET HELP TO 命令来改变 FoxPro 的帮助数据库使之成为应用程序所选择的帮助数据库。在此数据库备注字段中存入应用程序的帮助文本,这样在应用程序运行时按 F1 键就可以显示这些帮助文本。

如果应用程序的帮助数据库名不是 FOXHELP.DBF 而是其他名时,那就必须注意下面两点:

1. 此数据库的第一个字段必须是字符型

字段。该字段应将含有应用程序的帮助标题,而且可以用任何名字来命名这个字段。

2. 第二个字段应是备注型字段。此字段应含有应用程序运行时需要显示在帮助屏幕上的帮助信息。

在帮助数据库中,还可以有其他的字段。但是除了第二个(备注型)字段外,用户不能看见任何其他字段的内容。

为实现应用程序的帮助功能,可在此应用程序前面使用 SET HELP TO Helpfilename 命令, FoxPro 就可知道使用应用程序的帮助文件而不是用缺省的标准帮助文件 FOXHELP.DBF。同时,应使用 ON KEY 语句,以便当按下 F1 键时, FoxPro 实施对帮助系统的调用。我们可以用以下语句完成这些任务:

```
SET HELP TO MYHELP.DBF  
ON KEY = 315 HELP
```

这里的 315 是 F1 键的 ASCII 码。为了随时得到所希望的帮助内容,可在程序中适当位置使用 SET TOPIC TO Helptopicname 语句。这样就可使 FoxPro 跟踪程序运行并及时准确地提供屏幕帮助信息。

下面结合实例说明如何在应用程序中实现用户帮助。首先,建立用户帮助数据库

北京市威特电子技术公司

威特家用电脑系列

教育系列软件

电话:2558868,2577999

USERHELP.DBF。它含有两个字段：一个字符型字段 TOPIC，另一个是备注型字段 HelpText。字段的内容如下所示：

Topic	HelpText
Infor	输入名字、身份证号、工资、工作日期。当结束输入时，按 Ctrl+W 来保存此新雇员。
Editor	输入雇员的身份证号且编辑。当出现表格时，用 PgUp 或 PgDn 键来查看其他雇员。结束后，按 Ctrl+W 保存改变并且返回到菜单。
Eraser	输入雇员身份证号以删除此记录、在删除之前必须确认。
Reporter	确认打印机已打开，且按一个键打印报表。

然后，使用 FoxPro 帮助系统和 SET HELP TO、SET TOPIC TO 和 ON KEY 命令来实现帮助。程序示例如下：

```
SET HELP TO USERHELP
SET TALK OFF
ON KEY=315 HELP
DO WHILE .T.
CLEAR
USE STAFF INDEX ID _NO
STORE 0 TO Mychoice..
DIMENSION Choices(5,1)
STORE"1. 追加记录" TO Choices[1]
STORE"2. 编辑记录" TO Choices[2]
STORE"3. 删除记录" TO Choices[3]
STORE"4. 打印记录" TO Choices[4]
STORE"5. 退出系统" TO Choices[5]
@4,4 MENU Choices,5 TITLE "职工系统"
READ MENU TO Mychoice
DO CASE
CASE Mychoice=1
SET FORMAT TO STAFF
SET TOPIC TO INFOR
APPEND
CASE Mychoice=2
SET NEAR ON
SET TOPIC TO EDITOR
CLEAR
FINDIT="9999999999999999"
@5,5 SAY"按身份证号编辑?"GET FINDIT
READ
SEEK FINDIT
SET FORMAT TO STAFF
EDIT
SET NEAK OFF
CASE Mychoice=3
CLEAR
SET TOPIC TO EARSER
FINDIT="9999999999999999"
THEANS="N"
@5,5 SAY"按身份证号删除?"GET FINDIT
READ
```

```
SEEK FINDIT
IF FOUND()
@6,5 SAY Lastname
@6,30 SAY Firstname
@8,5 SAY "删除此记录? Y/N:"GET THEANS
IF UPPER(TheAns)="Y"
DELETE
ENDIF
ELSE
@6,5 SAY"不能找到这样的记录!"
WAIT WINDOW
ENDIF
CLEAR
CASE Mychoice=4
SET TOPIC TO REPORTER
WAIT WINDOW"按一键打印报表..."
REPORT FORM STAFF TO PRINT
CASE Mychoice=5
QUIT
ENDCASE
ENDDO
```

其中，STAFF 是全体雇员相关信息的数据库。

二、应用程序中的出错处理

多数情况下一个复杂的应用程序在经过反复测试之后，交予用户使用时会仍然会出现一些错误，使得用户不知所措，有时不当的操作也会损坏应用程序。为了防止发生这类错误，当错误终止程序的执行时，应向用户提供一个友好的提示界面，即使用错误收集例程。错误收集例程中使用了 ON ERROR 命令，此命令放置在应用程序的开始部分，可使 FoxPro 在发生错误时采取相应动作。它的格式为：

ON ERROR command

当检测到错误时，FoxPro 将执行指定的错误处理命令。通常，此指定命令是 DO programname，这里的 Programname 是错误收集例程的文件名。它可以是一个通知用户错误已发生的简短程序，并且返回到主菜单或 DOS 提示符；也可以包含一系列 CASE 语句来分析错误并且纠正它。下面给出一个实例：

* 显示主菜单程序 MENU.PRG

INTER—SACE V2.0
AutoCAD 12 版中文环境软件

北京浩辰技术开发公司

电话:2576505

```
SET TALK OFF
ON ERROR DO PROBLEMS
DO WHILE .T.
CLEAR
USE STAFF INDEX ID __ NO
...显示菜单并且选择等其他命令...
* 出错处理程序 PROBLEMS.PRG
CLEAR
? "程序错误发生,记录信息."
? MESSAGE()
?
WAIT WINDOW
QUIT
```

在以上的简单出错处理例子中,主程序 MENU.PRG 的第二个语句是 ON ERROR 语句。如果发生了错误,FoxPro 将调用子程序 PROBLEMS.PRG。在 PROBLEMS.PRG 中,向用户显示出错信息,并且使用 MESSAGE() 函数显示出错类型,一旦用户按了一个键,即返回到 DOS 状态。

三、事件驱动的编程

实现事件驱动用户界面的关键在于设计从一个单一点能提供广泛选择项的菜单。FoxPro 的菜单条与命令窗口的结合就是事件驱动用户界面的优秀例子。通过选择给定的菜单选择项,就可以打印数据、浏览文件、编辑记录、执行查询,或使用各种桌面工具。即通过单个菜单条能访问所有的菜单选择项。在 FoxPro 中,事件驱动命令可以使用事件驱动类型的菜单。它们不同于其他命令,是因为它不会引起某些动作立即发生,而是在某个事件发生时才执行指定动作。这类命令均用关键字 ON 开头,例如,ON BAR, ON ERROR, ON KEY, ON SELECTION BAR, ON KEY LABEL, 详细说明可参见 FoxPro 2.5 的命令手册。

一旦在你的程序中执行到这类命令,FoxPro 继续监视环境,查找由你程序指定的事件。当 FoxPro 检测到这种事件的发生时,就会执行此语句指定的命令。在多数情况下,此命令是调用另一个程序的 DO 命令。而这个程序将负责处理所希望的事件,把控制交给被调用程序。例如,一个菜单条的指定栏将调用 REPORTER.PRG 程序,以提供更多的打印报表选项。因此,在程序中应该含有语句:

```
ON SELECTION PAD __ option 4 OF __ MSYSMENU DO
REPORTER
```

当选中此菜单栏(PAD)时,则执行 REPORTER.PRG 程序。

四、访问计算机通讯口

应用程序中可以使用 FoxPro 提供的低级文件

函数访问计算机通讯口。最初,这些低级文件函数是用来管理文件的。如果用 DOS 设备名 COM1 或 COM2 去替换文件名,就可以从这些口上读数据或写数据。但是必须记住,在读或写通讯口的数据之前,要初始化它们。一个简单的办法是用 DOS 的 MODE 命令,在 FoxPro 中可用 RUN 语句来调用它。例如:RUN MODE COM1 1200,N,8,1,初始化 COM1 口为 1200 波特率、无奇偶检验、8 位数据位和一个停止位。一旦完成 COM1 的初始化,就可以用低级文件函数把此通讯口和文件一样来读或写。

例如,从用户那里接受到一个电话号码,然后使用与 Hayes 兼容的连接在 COM1 上的调制解调器来拨这个电话,可以用下面程序来实现。

```
SET TALK OFF
RUN MODE COM1 1200,N,8,1
&& 初始化 COM1 口
STORE SPACE(12) TO Phonenum
CLEAR
@5,5 SAY "电话号码?" GET Phonenum PICTURE "9999999-9999"

READ
MODEM=FOPEN('COM1',2)
&& 打开 COM1 口
IF (MODEM<0)
WAIT WINDOW;"在 COM1 发生错误!"
RETURN
ENDIF
PREFIX='ATDT'
DIALIT=FPUTS(modem,prefix+phonenum)
WAIT WINDOW"等待响铃,拿起电话,按一个键。"
DIALIT=FPUTS(MODEM,'ATH')
&& 挂上电话,复位调制解调器
DIALIT=FPUTS(MODEM,'ATZ')
=FCLOSE(MODEM)
RETURN
```

FoxPro 中还提供了一套软件开发工具:屏幕构造器、报表生成器、标签设计器、菜单构造器和项目管理器,且还可以建立独立于 FoxPro 而只在 DOS 下运行的执行文件 *.EXE。总而言之,FoxPro 编程功能很强,用它可以很容易地设计一个灵活、多功能的关系数据库应用管理程序。

SKILL

UNIX 找龍馬

为您提供 TCP/IP、LAN Manager、X.25 等网络工程

地址邮编:北京海淀西操场一号天府饭店 428-433 室

电话号码:(01)2545980 2545981 2569499 2532515

FoxPro 2.5 的弹出菜单中多项的选择

□ 罗 辉

在

一般的菜单系统中,只提供菜单选项的单选功能,即一次只能选择一个选项,然后转相应的功能模块执行相应的动作。但有时候,需要提供一次同时选择多个菜单选项,然后再一次性转入一个功能模块执行的能力。当然,你可以利用系统的复选框控制钮(对 FoxBASE + 可为每个菜单选项提供多个 GET 语句和相应的变量),再根据变量的值来决定哪些选项被选择了,以便执行相应的动作。但是,其屏幕布局似嫌复杂,一般需要在同时选择多个菜单选项的情况下,这些菜单选项具有一定的相似性,如果用复选框控制钮的设计方法实现,似乎打破了它们之间的联系,人机界面显示不严谨。

在 FoxPro2.5 环境下的菜单设计中,考虑到了这一需要。它提供的 DEFINE POPUP 命令中有一个 MULTISELECT 子句。利用这一子句,可方便地实现菜单选项的多重选择能力。

MULTISELECT 子句允许在弹出菜单中一次选择多个菜单选项。被选择的选项提示符左边将出现一个选中的标记字符。因此如果在 DEFINE POPUP 命令中使用 MULTISELECT 子句以提供多选项功能,必须同时还要使用 MARGIN 子句为每个选项前保留一个字符的位置,以便在菜单选项被选择时放置选择标记。

在 FoxPro for DOS 下,通过按住 Shift 键不放,用空格键或回车键选择一个选项,并用光标移动键移动光条选择连续的其他选项,然后按回车确认;或按住 Shift 键,使用

鼠标点按或拖拽,可完成连续或离散的菜单选项的选择。

在 FoxPro for Windows 下,将由 SET KEYCOMP 的设置来决定多选择的操作方式:

1. 如果 SET KEYCOMP 是 DOS 时,其选择方法与 FoxPro for DOS 是一样的。

2. 当 SET KEYCOMP 是 WINDOWS 时,如果用键盘选择,方法也与 FoxPro for DOS 下的操作是一样的;如果是用鼠标选择,首先点按第一个选项,然后拖拽到最后一个选项以选择一组连续的选项;或按住 Ctrl 键不放,点按单个选项,以选择离散的多个选项。

利用 MRKBAR() 函数可拾取被选择的选项,再决定相应的执行动作。它实质是检测该选项是否被置了选择标记,如果某选项被置了标记,则该函数将返回逻辑真值,否则将返回逻辑假值。但这种方式不能在带有 PROMPT FIELDS、PROMPT STRUCTURE、PROMT FILES 等子句的 DEFINE POPUP 命令中使用。

下面设计一个选择工资项目进行数据编辑的例子。利用弹出菜单的多选择功能,提供一个工资项目选择的菜单。用户可从中选择多个工资项目,按回车确认后,用 ESC 或 TAB 键退出菜单的选择。程序接着将激活 BROWSE 窗口对选择的工资项目进行编辑。注意:如果没选择任何一个工资项目,将默认在 BROWSE 窗口中编辑所有工资项目。

其中工资数据库为 GZK.DBF。为使字段名用中文显示,另外建立一个工资字段汉字库

GZZDHZK.DBF,其中存放了 GZK.DBF 中字段名相对应的汉字字段名。

GZK.DBF 和 GZZDHZK.DBF 的结构和内容如下:

GZZDHZK.DBF 结构:

Field	Name	Type	Length	Dec
1	FIELD__NAME	C	10	
2	CODE	C	17	

GZK.DBF 库结构:

Field	Name	Type	Length	Dec
1	BH	C	8	0
2	XM	C	10	0
3	BM	C	16	0
4	JB	N	9	2
5	LC	N	9	2
6	WB	N	9	2
7	GW	N	9	2
8	IB	N	9	2
9	BF	N	9	2
10	YF	N	9	2
11	LZ	N	9	2
12	KK	N	9	2
13	IF	N	9	2

GZZDHZK.DBF 库内容:

FIELD__NAME	CODE
BH	编号
XM	姓名
BM	部分
JB	基本工资
LC	粮差
WB	物补
GW	岗位津贴
IB	书报费
BF	补发工资
YF	应发工资
LZ	零存整取
KK	扣款
IF	实发工资

相应的任意工资项目数据编辑程序如下:

*** 任意编辑工资项目的数据程序 ***

```
SET TALK OFF
IF USED("gzdhzk")
    SELECT gzdhzk
    SET ORDER TO 0
    GO TOP
ELSE
    SELECT 0
```

```
USE (LOCFILE (".\gzdhzk.dbf","DBF","工资字段汉字库在哪里?"));
AGAIN ALIAS gzdhzk;
ORDER 0
ENDIF
CLEAR
EDITFIELD=""
@ 4,5 SAY "请你从列中选择要编辑的域:"
DEFINE POPUP fieldslt FROM 5,5 TO 10,20;
MULTISELECT MARGIN &&. 定义多重选择的弹出菜单
FOR I=1 TO RECCOUNT()
    DEFINE BAR I OF fieldslt;
    PROMPT code MARK CHR(3)
    SKIP
ENDFOR
ON SELECTION POPUP fieldslt DO yourchoice
ACTIVATE POPUP fieldslt
IF USED("gzk")
    SELECT gzk
    SET ORDER TO 0
GO TOP
ELSE
    SELECT 0
    USE (LOCFILE (".\gzk.dbf","DBF","工资库在哪里?"));
    AGAIN ALIAS gzk;
    ORDER 0
ENDIF
IF ! EMPTY(ALLTRIM(EDITFIELD))
    BROWSE FIELDS &EDITFIELD
ELSE
    BROWSE
ENDIF
USE
RETURN
PROCEDURE yourchoice &&. 对被选择项执行相应操作
fldtable=""
FOR count=1 TO CNTBAR('fieldslt') &&. 拾取被选项
    IF MRKBAR('fieldslt',count)=.T. &&. 对被选项执行操作
        GO COUNT
        fld=ALLTRIM(field__name)
        cod=ALLTRIM(code)
        fldtable=fldtable+fld+" :H="+cod+" ", "
    ENDIF
NEXT
fldtable=SUBSTR(fldtable,1,LEN(fldtable)-1)
editfield=fldtable
RETURN
```

SKILL

UNIX 找龍馬

为您提供远程无人值守通信系统 XYcomm

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

PC 机 RS-232 通信连线汇集

□ 管逸群

随

随着电子技术的发展,硬盘容量越来越大,软件规模也越来越大,在工作中我们经常需要在微机之间进行文件的传输。最通常的方法是利用 PC 机的 RS-232C 口来实现,若是近程的(30 米),RS-232C 口之间可以通过电缆线直接连接;若是远程的,则通过 MODEM(调制解调器)电话线连接。RS-232 口有 25 芯和 9 芯之分,它们之间连接的电缆线接法有所不同。现提供给大家参考。

一、PC 机间的连接线接法

1. DOS 与 DOS 通信的一种接法

(1)PC 机的 RS-232C 口 9 芯与 25 芯

9 芯 RS-232C 口	25 芯 RS-232C 口
1	8
2	2
3	3
4	5
5	7
6	6
7	4
8	20

(2)PC 机的 RS-232C 口 9 芯与 9 芯

9 芯 RS-232C 口	9 芯 RS-232C 口
1	1
2	3
3	2
4	8
5	5
6	6
7	7
8	4

(3)PC 机的 RS-232C 口 25 芯与 25 芯

25 芯 RS-232C 口	25 芯 RS-232C 口
2	3
3	2
4	4
5	20
6	6
7	7
8	8
20	5

2. DOS 与 UNIX

(1)PC 机的 RS-232C 口 9 芯与 9 芯

9 芯 RS-232C 口	25 芯 RS-232C 口
1	6
2	2
3	3
4	8
5	7
6	20
7	4
8	5

(2)PC 机的 RS-232C 口 9 芯与 9 芯

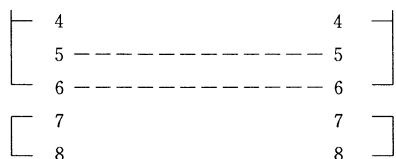
9 芯 RS-232C 口	9 芯 RS-232C 口
1	1
2	3
3	2

FBASE DOS 下的最佳
多媒体数据库平台

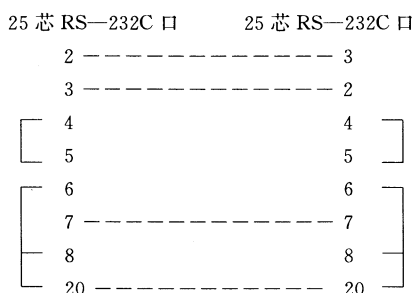
新未来电子技术公司

电话:2573355-271,272

地址:清华东门清华园宾馆一层

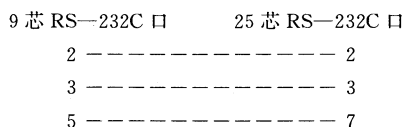


(3) PC 机的 RS-232C 口 25 芯与 25 芯

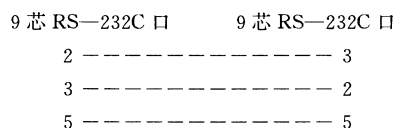


3. DOS 与 SCO UNIX (SCO Xenix)

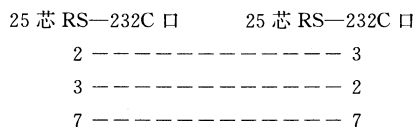
(1) PC 机的 RS-232C 口 9 芯与 25 芯 (简易)



(2) PC 机的 RS-232C 口 9 芯与 9 芯 (简易)



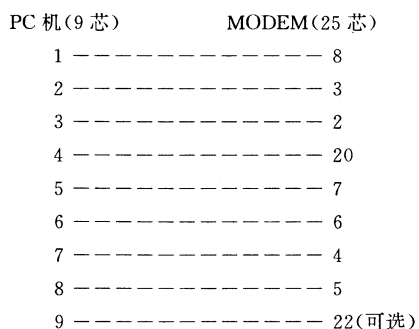
(3) PC 机的 RS-232C 口 25 芯与 25 芯 (简易)



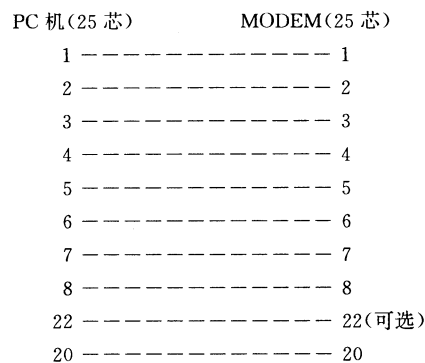
二、PC 机与 MODEM 间的连接法

一般而言, MODEM 都为 25 芯, 而 PC 机的 RS-232 口有 9 芯和 25 芯之分。

1. PC 机 RS-232C 口为 9 芯, MODEM 为 25 芯



2. PC 机的 RS-232C 口为 25 芯, MODEM 为 25 芯



三、PC 机间通过并行口通信的连接线接法



SKILL

《FoxPro 编程经验与技巧》

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

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

UNIX 找龍馬

为您提供龙马 PC 终端卡 XTEC

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

微机软故障的分析与处理

□杜长勇

在

微机运行过程中,软件故障率要比硬件故障率高得多。常见的软件故障可概括为以下四种情况:(1)由系统软件出错引起的故障。(2)由应用软件出错引起的故障。(3)由用户对微机系统各参数的不正确设置造成的。(4)由计算机病毒对软件的侵害而引起的。下面通过一些实例来分析上述四种软故障的发生原因及排除方法。

1. 故障现象:系统加电后,硬件自检通过,由硬盘引导操作系统时,系统进入 ROM BASIC。

故障分析与处理:引导操作系统失败而进入 ROM BASIC,一般故障发生在硬盘主引导记录上。系统硬件自检通过后,由 ROM BIOS 中的引导程序读取硬盘的主引导记录。主引导记录含有硬盘的分区信息和检查硬盘数据的程序代码等。当引导记录中没有自举分区标志(80H)或者最后两个字节不是 55AA 时,即主引导记录被破坏,显示器上显示 MISSING OPERATION SYSTEM,系统无法引导 DOS 而进入 ROM BASIC。此外,主引导记录被破坏还表现在以下几点:

(1)四个分区中任一个引导指示字节都标为无效,或者四个分区中多于一个分区含有可引导标志。此时,显示器上显示“INVALID PARTITION TABLE”,不能正确引导 DOS。

(2)读取活动分区中的引导扇区,若不能成功地读出,则显示“ERROR LOADING OPERATION SYSTEM”,不能正确引导 DOS。

主引导记录损坏的处理方法是:

(1)使用无病毒的 DOS 系统,从软磁盘驱动器 A 上引导 DOS 系统。

(2)将硬盘 C 上的有用文件备份。

(3)用低级格式化程序格式化硬盘。

(4)对硬盘重新进行 DOS 分区。并将其激活。

(5)运用 A 盘上的“FORMAT.COM”命令,将 C 盘格式化,完成驱动器 C 上的软件安装操作。

2. 故障现象:系统加电后,无法正确引导 DOS,显示器上显示:

NON - SYSTEM DISK OR DISK ERROR

REPLACE AND STRIKE ANY KEY WHEN READY

故障分析与处理:这类故障是由于 DOS 引导扇区出了问题而引起的,当引导 DOS 时,DOS 引导扇区的程序部分从根目录的开始扇区读取目录的前两个目录项(文件名),与数据区保留的两个 DOS 系统文件名相比较,看是否相同。若其中有一个文件名不相同,则出现上述故障现象。DOS 引导出错的另一种表现是当引导程序将第一个隐含的系统文件读入内存时,若发生读盘错误,则显示器上显示:DISK BOOT FAILURE,而后进入死循环。造成引导扇区出问题的原因很多,有可能是在格式化磁盘时,没有给出传递操作系统文件的参数“/S”,或者是格式化盘时的操作系统与现存盘上的操作系统版本不一样;也有可能是磁盘文件目录中前两个文件名与 DOS 引导扇

区的数据部分保留的文件名不同;或者是 DOS 引导扇区被计算机病毒所破坏。这种故障的解决办法是:首先用正确的系统软盘引导系统,然后用 PC-TOOLS 或者 DEBUG.COM 软件将两个隐含文件 IBMBIO.COM 和 IBMDOS.COM 装入磁盘的引导扇区;或者用 SYS.COM 命令将两个隐含文件传入磁盘目录区中,并确保引导记录中系统文件名与目录区登记的文件名一致。

3. 故障现象:启动 DOS 时,显示器显示:BAD OR MISSING COMMAND INTERPRETER”。

故障分析与处理:若正确引导 DOS 操作系统,系统必须读取两个隐含文件和 COMMAND.COM 文件(DOS 的命令处理程序)。当 COMMAND.COM 文件被改写、覆盖或者删除时,就会出现此类故障。这种故障的另一种表现形式是显示器上显示:INCORRECT VERSION。解决办法就是用备用的 DOS 软盘启动,然后再将相同版本的 COMMAND.COM 拷贝到启动盘上。

4. 故障现象:给系统加电,硬盘自检一会儿之后,不能正确引导 DOS,光标在屏幕左上角闪烁,不能从键盘上输入任何信息,系统“死机”。

故障分析与处理:初步推断为硬盘引导程序被破坏,但从软盘上也不能引导操作系统,至此便想到是否系统设置的参数已丢失,使系统在启动时检测不到这些参数而“死机”。于是着手设置系统参数:重新冷启动机器,冷启动的同时,按下 Ctrl+Alt+ESC 三个功能键,系统进入 SETUP 设置界面,按照机器原来的工作参数设置完后按 F10 键,屏幕上即出现:“ARE YOU SURE (Y/N)?”的选择信息。按下 Y 键后,机器即自动热启动,稍后机器便能正常工作。

(上接第 70 页)

信息。此时可肯定是/etc/init 文件损坏。处理方法:用引导盘启动系统,将硬盘根文件系统安装到软盘上,再把软盘上的/etc/init 文件拷贝到硬盘上即可。/etc/init 的主要作用是:打开主控台/dev/console 准备读写;检查文件系统的一致性;执行/etc/rc 进行必要的设置和初始化;读/etc/ttys,为每个打开的终端和虚拟屏幕创建一个进行读写的环境。上述四个过程中,哪一个文件损坏,都会影响系统的正常进入,可用前面类似方法进行相应的恢复。

8. 系统引导时出现“Init:/dev/ttyxx: getty keeps dying-there may be a problem”错误信息,此时一定是/etc/getty 文件被破坏。处理方法:先进入单用户的系统维护方式,把 Xenix 系统 N1 盘上的

5. **故障现象:**运行 dBASE III 时,系统提示打开文件太多。

故障分析与处理:DOS 装载的标准设备包括标准输入、标准输出、标准打印机、软盘以及硬盘设备。DOS 支持这些设备不需用户任何命令指定。但要扩充输入设备、输出设备的控制特性,就需要一个设备驱动程序 CONFIG.SYS。而系统在装入时,只定义 8 个文件描述字,其中 DOS 系统占用 5 个,留下 3 个给用户使用,也就是说允许用户程序打开的文件数只有 3 个。运行 dBASE III 时,系统提示打文件太多,是因为当 dBASE III 需要打开多个程序文件或数据库文件时,由于留给用户使用的文件数只有 3 个,系统会遇到无文件描述字分配的问题,所以提示打开文件太多。排除这一故障的方法是:在设备驱动程序 CONFIG.SYS 中定义 FILES=20,保证用户有足够的文件数使用。

6. **故障现象:**开机硬盘系统正确启动后,主机和打印机联接不上,即用 Ctrl+P 或 TYPE XXX> PRN,PRINT SCREEN 等命令时,主机死机或打印机无响应。

故障分析与处理:造成这种故障的原因很多,有可能是硬件故障,也有可能是软故障。首先要考虑是否是软故障,或者是否是由计算机病毒引起的。

用一无病毒的系统软盘从 A 驱动器中启动系统后,检查一下打印机和主机能否联上。试验的结果是能联上,证明主机与打印机联不上是由于病毒引起的。这种计算机病毒把打印机口地址 378H 改为 300H,造成打印机不能打印。排除方法是用消毒软件将病毒清除,故障即自然消失。

SKILL

getty 文件拷贝到硬盘的/etc/目录上,即可解除故障。/etc/getty 文件的主要作用是根据/etc/gettydefs 文件调整数据传送速率和注册信息。若/etc/gettydefs 文件损坏时,也可用上述方法恢复。

SKILL

UNIX 找龍馬

为您提供 UNIX XENIX 编辑排版打印系统 XWORD

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

Xenix 系统故障检修几例

□宋玉长 李 雷

X

enix 是国内外比较流行的多用户分时操作系统,目前其应用范围日益广泛。下面笔者根据自己的体会,列出几例 Xenix 系统常见故障检修方法,作为贵刊 94 年第 2 期《Xenix 引导时故障及其维护》一文的补充。

1. 主控键盘锁住,但不影响终端、打印机的运行,系统仍可以使用。处理此故障首先保证你未意外地键入 Ctrl+S,然后检查键盘的插头是否插好、键盘的锁键是否在锁定状态下,若故障不能排除,可按 Ctrl+Q 几次,然后检查能否在屏幕上敲入字母,按回车键或 Del 键几次,如仍不能使用主控键盘,试试拔下主控键盘插头,然后再插上,一般可恢复。

2. 恢复没有回应或锁住的终端。没有回应的终端指不显示在键盘上敲入的字符的终端,而终端锁住是不响应任何输入的。

(1)终端锁住处理方法:首先找一正常工作的终端,敲入 W 并回车,查出锁住终端的名字,再输入 \$pst 锁住终端名,找出与锁住终端有关的进程识数。然后依次输入: \$kill -9 进程标识数。\$kill -15(软件中断), \$kill -0(后台终止打印),锁住终端即可恢复正常。

(2)终端无回应处理方法:按 Ctrl+J 键(注意由于终端不显示打入内容,输入必须准确)后,终端一般应恢复。

3. 终止失控进程:失控进程是由终端启动的但无法停止的程序,形成死循环。停止失控进程的方法:找一个没有锁住的终端或主控台,作为超级用户(root)登录,键入 #ps -

e,找出失控进程的终端标识号。

键入 #kill -15 失控进程标识号。若过几秒后,程序没停下,键入: #kill -9 标识号;执行上述步骤后,发生终端无回应情况,再按 2 中方法处理,直到显示“login:”为止。

4. 假脱机打印系统不正常:运行 lpint 命令提示打印机忙,无法重新安装假脱机打印系统。处理方法:首先用 usr/lib/lpshut 命令关闭打印队列后执行/usr/lib/lpmove 命令移打印队列,然后执行命令 lpinit,完成打印机的重安装,再执行/usr/lib/lpmove,移回打印队列,最后执行/usr/lib/lpshut 命令即可使打印机正常使用。

5. 第一次开机,系统检测到“E”,硬盘灯一直亮,不能进入系统;重开机当系统检测到“E”时,系统死机。处理办法:先用 N1 盘从软驱启动系统,并把硬盘挂入软盘系统,检查/dev/console 文件,发现/dev/console 变为普通文件,用命令 mknod 将此文件改为设备文件,然后卸掉硬盘,重新启动系统即可。

6. 屏幕上循环显示“login:”信息,不能进入系统。此时可判定是/etc/login 文件受到损坏。处理方法:用“N1”盘启动系统,进入硬盘根文件系统,将卷标为“B2”的软盘,插入/dev/fdo96 软盘驱动器。在根目录下,键入“tar xv2. /etc/login”恢复/etc/login 程序。另外/etc/login 程序的运行,要用到/etc/passwd 文件,若该文件损坏,同样可用上述方法将“N1”盘上 Passwd 文件拷到硬盘的/etc 目录下。

7. 屏幕上出现“Can't exec/etc/init”错误

(下转第 69 页)

利用 AutoCAD 形体文件在 Turbo C 中写字符

□ 蒋先刚

在

在 Turbo C 中可以利用函数 `outtext()` 和 `settextstyle()` 写出一定大小和方向的西文字符,但是不能写任意大小和方向的字符,有时不能满足一些工程软件的设计需要。笔者利用 AutoCAD 已经建立的字体形体文件成功地在 Turbo C 中写出了任意大小和方向的字符,下面简要介绍其实现方法。

AutoCAD 提供各种字体的形体文件,我们选择比较好看的字体的形体文件 `COMPLEX` 和 `ITALIC.SHP`,它们是按照一定的规则制定的矢量图形码文件。其写字过程实际是一矢量图形码的解释输出过程。考虑到读写文件的方便性,首先通过程序 `CHANGESH.C` 将 `COMPLEX.SHP` 和 `ITALIC.SHP` 转换成 `COMPLEX.SHC` 和 `ITALIC.SHC`。程序 `CHANGESH.C` 可将 `COMPLEX.SHP` 和 `ITALIC.SHP` 中的逗号和空格去掉,并使得 `COMPLEX.SHC` 和 `ITALIC.SHC` 更具可读性和格式化。

程序 `WRITESHP.C` 完成读取 `COMPLEX.SHC` 或 `ITALIC.SHC` 文件并按要求写出了任意大小和方向的字符。程序依据给定条件,通过读取形体文件中的矢量值,将各形体矢量在子坐标系下进行投影计算,并将其换算成总体坐标系下的 `X,Y` 坐标,用画线函数显示它们。用户也可用此方法编制和读取形体文件来书写任意大小和方向的矢量汉字。该程序在 IBM/AT286 上运行通过,使用 AutoCAD2.18 的字体形体文件。

程序中应用的基本数学计算公式是:

$$x_9 = k * \cos(\text{ang})$$

$$\begin{aligned} y_9 &= k * \sin(\text{ang}) \\ x_8 &= \text{startx} + \text{scalex}(x_9 * \cos(\text{wangle}) - y_9 * \sin(\text{wangle})) \\ y_8 &= \text{scmaxy} - \text{starty} - \text{scaley}(x_9 * \sin(\text{wangle}) + y_9 * \cos(\text{wangle})) \end{aligned}$$

程序中各变量说明如下:

`style` 是所要写的字符类型;
`words` 是所要写的字符串;
`startx, starty` 是字符串的起始坐标;
`scalex, scaley` 是字符串在 `X,Y` 方向的放大系数;
`wangle` 是字符串的角度;
`wcolor` 是字符串的颜色;
`k` 是形体各矢量的长度;
`ang` 是形体各矢量对其子坐标系 `X` 轴的夹角;
`x9, y9` 是形体各矢量在子坐标系下的 `x, y` 分量;
`x8, y8` 是在总体坐标系下各矢量将在屏幕上显示的 `x, y` 分量;
`scmaxy` 为调整用户坐标和屏幕坐标的差别而设。

源程序清单如下:

程序 `CHANGESH.C` 清单:

```
/* Changesh c:change * shp to *.shc */
#include <string.h>
#include <stdio.h>
#include <conio.h>
void writewords(void);
void main(void)
{ writewords(); getch(); }
/* Change AutoCAD's *.shp to *.shpl */
void writewords(void)
{ FILE *fp, *fr; char s;
  if ((fp=fopen("complex.shc", "w"))==NULL) {
    printf("Could not open the file\n");
```

```
return; }
if((fr=fopen("complex.shc","w"))==NULL) {
printf("Could not open the file\n");
return; }
while (feof(fp)==0) {
s=getc(fp);
if (s==(char)32) continue; /* delete the space */
if (s==(char)44) s=(char)32; /* change ", " to space */
printf("%c",s); putc(s,fr); }
fclose(fp); fclose(fr); }
```

程序 WRITESHP.C 清单:

```
/* Writeshp.c : Write words by using AutoCAD's *.shp */
#include <math.h>
#include <io.h>
#include <ctype.h>
#include <string.h>
#include <graphics.h>
#include <stdio.h>
#include <stdlib.h>
#define PI 3.14159/180.0
int scmaxy;
void writewords(char *,char *,int,int,float,float,int,int);
void main(void)
{ int gd=DETECT,gm; initgraph(&gd,&gm,"");
scmaxy=getmaxy(); cleardevice();
writewords("COMPLEX","COMPUTER WORLD",100,100,1.5,1.5,20,RED);
getch(); closegraph();
} /* Program of write words at ang angle and size */
void writewords(char * style,char * words,int startx,int starty,
float scalex,float scaley,int wangle,int wcolor)
{ FILE *fp;
char * filename=" ",wordi,s1[5],name[20],ch[5],*ch0=" ";
char *ch1=" ",*ch2=" ",*ch3=" ",*ch4=" ",*charascii=" ";
int n=0,ascii,chlen,wordlen,i,j,color=0,number,chn1,chn2,chn3;
float x9=0.0,y9=0.0,x8,y8,s,k,dx=0.0,dy=0.0;
float ang; strcpy(filename,"c:");
strcat(filename,style); strcat(filename,".SHC");
if ((fp=fopen(filename,"r"))==NULL) {
printf("Could not open the file\n");
return; } chlen=strlen(words);
for(j=0;j<chlen;j++)
{ fseek(fp,0,SEEK-SET); wordi=(words+j);
ascii=toascii(wordi); itoa(ascii,ch0,10);
ch4=strcpy(ch4," "); charascii=strcat(ch4,ch0);
wordlen=strlen(charascii);
a: if (feof(fp)!=0)
goto b3; else
fscanf(fp,"%s%d%s",s1,&n,name);
if (strcmp(s1,charascii,wordlen)==0)
{ i=0; a1: while (i<=n)
{ fscanf(fp,"%s",ch); i++;
if (atoi(ch)==atoi(ch+1)) {
ch3=ch+2; chn3=atoi(ch3);
if (isalpha(*ch3)) chn3=toascii(*ch3)-55;
```

```
if (chn3==1||chn3==3||chn3==5||chn3==7||chn3==9||
chn3==11||chn3==13||chn3==15)
s=1.0/cos(22.5*PI);
if (chn3==2||chn3==6||chn3==10||chn3==14)
s=1.0/cos(45.0*PI);
if (chn3==0||chn3==4||chn3==8||chn3==12)
s=1.0; * (ch+2)='\0';
ch2=ch+1; ch2=atoi(ch2);
if (isalpha(*ch2))
chn2=toascii(*ch2)-55; k=(float)chn2*s;
ang=(float)chn3*22.5; dx=k*cos(ang*PI);
dy=k*sin(ang*PI); x9=x9+dx;y9=y9+dy;
x8=startx+scalex*(x9*cos(wangle*PI)-y9*sin(wangle*PI));
y8=scmaxy-starty-scaley*(x9*sin(wangle*PI)+y9*cos(wangle*PI));
if (color==0) moveto(x8,y8);
goto a1; }
number=atoi(ch);
switch(number) {
case 0: break;
case 1: color=wcolor;
break;
case 2:
color=0;
break;
case 9:
case 8:
a3:fscanf(fp,"%s",ch);
i++;
ch1=ch+1; chn1=atoi(ch1);
fscanf(fp,"%s",ch2);
chn2=atoi(ch2); i++;
x9=x9+(float)chn1;
y9=y9+(float)chn2;
x8=startx+scalex*(x9*cos(wangle*PI)-y9*sin(wangle*PI));
y8=scmaxy-starty-scaley*(x9*sin(wangle*PI)+y9*cos(wangle*PI));
if (color==0)
moveto(x8,y9);
if (number==8) break;
if (chn1==0&&chn2==0)
break; goto a3;
}
goto a1;
} }
else
{ for(i=1;i<=n;i++)
fscanf(fp,"%s",ch); }
goto a;
b3: ;
}
fclose(fp);
}
```

SKILL

利用 INTL Toolkit 使你的应用国际化

□刘耀东译

许

多用户都在为使应用国际化而寻求帮助。FoxPro 是至今为止利用广泛支持代码页和排序顺序来管理数据的最好工具。INTL Toolkit 现在提供了将用户的菜单和屏幕的全部正文串的透明翻译。它允许用户选择运行时操作的语言,或者为应用提供某种语言版本。

INTL 使用一张包含系统里全部正文串的表。这些正文串包括菜单笺,强制编码正文串,WAIT WINDOW 提示,诸如无线按钮和检查框之类的对象提示,窗口标题等等。该表的多个列包含每种语言的每个字符的翻译转换。当你运行应用时,将以全局变量的方式从合适的列中取出各个变量。

你可以为用户提供改变全局变量值的菜单选择,立即驱动每个正文串的即时翻译,一个简单应用可用多种语言来使用而不必提供编译后应用的多种版本。

INTL 包含什么

INTL 由一个主程序,几个支持程序和一张缺省串翻译表组成。你从来也不使用主程序 INTL.PRG 和 INMENU.PRG,GEN-SCRNX 和 GEWMENUX 做为其过程的一部分来调用它们以修改你的菜单和屏幕。修改则调用五个支持程序。

I.PRG 是你的应用所使用的“翻译”函数。例如,一个窗口定义通常会看起来象这样:

```
IF NOT WEXIST(" _qlulljeb") .
DEFINE WINDOW _qlulljeb;
FROM INT((SRCW()-25)/2,INT((SCOL()-77)/2);
```

```
TO INT((SROW()-25)/2)+24;
INT((SCOL()-77)/2)+76;
TITLE "The Ultimate FoxPro Reference"
ENDIF
```

则变成了这样:

```
IF NOT WEXIST(" _qlulljeb")
DEFINE WINDOW _qlulljeb;
FROM INT((SROW()-25)/2,INT((SCOL()-77)/2);
TO INT((SROW()-25)/2)+24,;
INT((SCOL()-77)/2)+76;
TITLE I("The Ultimate FoxPro Reference")
ENDIF
```

标题串已经通过对 I() 函数的调用而加以压缩了,当运行包含这一窗口定义的屏幕时,Foxpro 将在翻译表中查找串“The Ultimate FoxPro Reference”,然后返回所选择语言的翻译版本。NOHOT.PRG 为一个从串中去掉分配的热键的用户自定义函数。NOTHOT() 在下推按钮提示菜单笺或者其他利用热键指定需要的字符串时使用,以便去掉这些指定后提供参考。NOHOT() 广泛用于产生屏幕代码。WPADR.PRG,WPADC.PRG 和 WPADL.PRG 是 PADR(),PADC() 和 PADL() 函数的 Windows 下的相应函数。生成的屏幕程序利用这些函数来恰当地调整运行时以翻译串长度为基础的正文串位置。STRINGS.DBF 包括英语、法语和德语正文串字段(你必须提供翻译后的字符串)。你可以通过向语言表里追加字段来追加某种语言定义。

如何使用 INTL

假定你有一个用 Power Tools 构造的应

用(项目管理器,菜单生成器和屏幕生成器)。屏幕包含标题、正文串、无线按钮、检查框、下推按钮和弹出式菜单。

1. 修改 STRINGS.DBF 以便它包含应用所能采用的每种语言。输入每个原始字符串翻译后的字符串(INTL 并不完成英语单词向另一种语言的实际翻译)。然后 STRINGS 将象类似这样查找:

cOriginal	cFrench	cGerman
One	Un	Eins
Two	Deux	Zwei
Three	Trois	Drei

2. 将 GENSCRNX/GENMENUX 放在你的 FoxPro 主目录下。

3. 向你的 CONFIG.FP/W 文件中追加如下几行(调整好路径,重新启动 Foxpro):

```
__ GENMENU="C:\FPD\GENMENUX.FXP"
__ GENSCRN="C:\FPD\GENSCRNX.FXP"
__ MNXDRV2="INTLMENU.PRG"
__ SCXDRV5="INTL.PRG"
__ INTLTIMING="RUN"
```

4. 将 INTL.PRG、INTLMENU.PRG、I.PRG、NOHOT.PRG 和 WPAD?.PRG 放在项目目录里(笔者将其放在笔者的 COMMON 目录里,并将 FoxPro 路径指向 COMMON)。

5. 将 STRINGS.DBF/FPT/CDX 放在你的源程序代码目录里。

6. 利用 Rebuild All 重建项目文件。注意 CONFIG.FP, GENMENUX 和 GENSCRNX 等新语句被以调用,并且执行了 INTL 驱动程序。

7. 查看所产生的.SPR 文件。你会看到每个正文串已被函数"I"括起来,原来的正文串则做为一个参数。

8. 经过一次类似的函数调用自动翻译你的代码里的正文串。检查你的.PRG 文件和屏幕片断来查找 WAIT 和类似的命令:

```
WAIT WINDOW "Customer not found" NOWAIT
```

将变为:

```
WAIT WINDOW I("Customer not found") NOWAIT
```

更好的是你可以使用 MSGSVC.PRG, 该文件是由 INTL 提供的,可用来处理你的全部信息显示需求。在本文稍后的部分将更详尽地讨论 MSGSVC()函数。

9. 进入 STRINGS.DBF 表,该表现在包括应用菜单、屏幕的每个字符串记录,并提供了每个字符串的翻译版本。

10. 运行应用。全部字符串将按英语方式出现(或者当你设置 INTL 时在系统配置里指定了你的

本国语言)。

11. 在应用以外或在主程序里初始化变量 __INTLANG 等同于欲翻译成的语言:

```
__INTLANG="German"
```

12. 再次运行应用以显示所有翻译后的正文串。

INTL 如何工作

考虑一下没有 INTL Toolkit,你将如何设计多种语言的应用。每个屏幕和菜单必须进行复制并按每种语言进行修改——无论你怎么做它都是维护上的恶梦。

解决的办法依赖于你已经了解了的新的工具: GENSCRNX,包括 INTL Toolkit。正如你在过去所看到的 GENSCRNX 是 FoxPro 自身的 GENSCRNX 的前期或后期处理器。它通过使用 GENSCRNX 提取你的原来的.SCX/.SCT 文件,修改它们的拷贝,为新文件提供材料,然后(可选择地)修改最终的.SPR 文件。

GENSCRNX 的优点在于它包含了其他开发者扩展 GENSCRNX 函数的异常分支。INTL 就是对 GENSCRNX 的这样一个扩展——它找出你的菜单和屏幕中的每个正文串,然后将其用 INTL 翻译函数 I()括起来。

菜单生成器前置处理器 GENMENUX,对菜单所做的一切就象 GENSCRNX 对屏幕所做的一样,利用 INTL Toolkit 也可将其包括起来。GENMENUX 最初是由 Stevell Black 为 INTL 设计的,后来被他的一个加拿大同事 Andrew Ross Macndil 改进得功能更强和更加通用。象 GENSCRNX 一样,它位于公共定义域内。

利用 GENSCRNX 和 GENMENUX 驱动程序以使用一个局部函数来压缩字符串,是一个简单的解决办法。然而,在 INTL 实现解决的大宽度问题时还有很多细小的弯路要走。集中或分别显示的字符串,调整在不同语言里大小的改变之类的对象关系是项令人振奋的任务;INTL 产生所需要的代码并且工作得很好。

(本文资料由 Microsoft 提供) **SKILL**

300 元省一台打印机

SXD 系列

全自动、半自动、多功能打印机共享器

地址:北京清华大学科学馆 302 邮编:100084
电话:2594866 联系人:魏宝英,张罗平

多频显示器的特点及维护(二)

□柳永林

本

期我们继续以实例的方式讨论多频显示器的维修(例 2)。

机型: NEW 多频 VGA 彩色显示器

故障现象:

1. 无显示, 屏幕没有高压静电反映, 但指示灯、显象管灯丝亮。2. 行不同步, 屏幕没有高压静电反映, 但指示灯、显象管灯丝亮。

故障分析与维修: 现在市场上销售的显示器以及用户所有使用的显示器, 面板上贴上的标签名目繁多, 往往与里边的线路板不对应。比如某公司买到一批 AST 主机, 而买不到或不想买 AST 显示器, 于是就将 CASPER 显示器标签拿下贴上 AST 显示器标签。而很多用户对显示器的型号并不了解, 因此市场上流行的型号很多、很乱, 本文所述故障机器就是一例。该显示器的电源属于并联型开关电源一直流串联稳压电源。电路原理框图如图 6 所示。

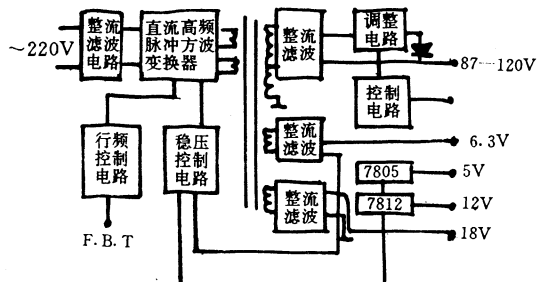


图 6 电源原理框图

指示灯电压由 7805 输出端提供, 灯丝电

压由开关电源 6.3V 供电。两个灯都亮说明开关电源是正常的。根据该机故障现象, 判断行输出或行输出电源有问题。

检查行输出, 单机加电测量行输出有关电压, 加速极电压 VG2 为 104V (正常值为 300V), 行输出集电极电压为 1.7V (正常值为 87-120V), 视放输出电源电压为 2.4V (正常值为 90V), 以上实测电压数值说明行输出工作极不正常。用数字表二极档“在线”测量行输出 PN 结参数是正常的, 说明行输出管未坏。行输出与行电源之间接有保护电阻(或叫限流电阻), 经目视检查电阻 R825 (0.47/1W) 电表测量其阻值几乎变成无穷大, 表面涂的漆完好, 因此, 造成行电源电压绝大部分都降落在电阻上了, 而集电极只有 1.7V 电压。用一只 0.68/1W 电阻替换, 为了行管安全, 将电阻一端与行电源相连, 另一端接上假负载 300-400/50W 可调电阻, 示意图如图 7 所示。

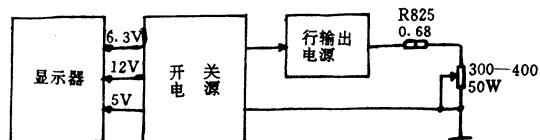


图 7 行输出电源试验原理框图

加电适当调整假负载电阻阻值, 负载两端电压可在 68-92V 之间变化, 当调整开关电源中电位器 VR803 时可以改变行电压输出,

并可调到正常值。这时可去掉假负载,将电阻 R825 焊在电路板上。显示器单机加电只有 40V,显象管阳极高压为 12KV(正常值为 25KV—26KV),灯丝电压正常为 6.3V,12V。行输出电源为什么这么低呢?这个问题留在后面讲。下面联机看看是否有显示。该机是多频 VGA 显示器,它既可以和标准 VGA 显示卡联机,也可以和 SVGA、8514/A、TVGA 卡联机,现将 VGA 各种显示卡的行场同步信号的频率、极性以及分辨率列表如下:

表 1 VGA 卡的行场同步信息
频率、极性、分辨率

显示方法	分辨率	行频	极性	场频	极性
VGA	640 * 480 点阵	31.47KHz	H—	59.94	V—
SVGA	800 * 600 点阵	35.16KHz	H+	56.07HZ	V+
8514/A	1024 * 768 点阵	35.52KNz	H+	86.96HZ	V+
TVGA	1024 * 768 点阵	35.52KHz	H+	83.96HZ	V+

该机与任一显示卡联机都有显示,但都不能同步。有关的电路图如图 8 所示。

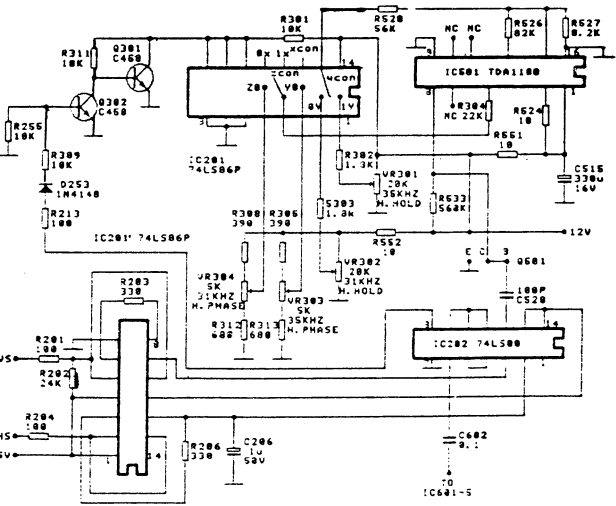


表 3 同步脉冲极性

显示方法	行同步信号极性	场同步信号极性
640×350	+	-
640×400	-	+
640×480	-	-
800×600	+	+

对于同步信号的检查及其同步问题的故障维修,一般用示波器法进行检测为好。若没有示波器,就用排除法或替换法,将 86P 芯片焊下来进行测试,没有仪器就更换一个。如果你有示波器可观测 86P 芯片输入输出波形是否正常。首先看输入脉冲波形,结果是正常的。观测 11 脚输出波形也是正常的,IC202 输入(1,2 脚),输出 3 脚的波形还是正常,但经过耦合电容 C528 后就没有波形了,为什么呢?首先怀疑电容器开路,因此更换一个电容,结果不变。继续往下检查,发现电路板有一个三极管没接上(该管不需要焊上),但连线接错,电容器一端应该与基极、集电极相连,然后与 IC501(TDA1180)8 脚接通。但电容器却与地线短接了,结果将同步信号短路掉,所以 8 脚没有同步信号,造成不同步。当把基极与地断开而与集电极相连后,同步信号便加到了 TDA1180 的 8 脚,这时适当调整行频电位器就能同步了。但调哪个电位器就得弄清楚行频的高低与 RC 时间常数的关系。下面介绍一下行频自动跟踪系统。

行频自动跟踪系统有两种方式:一种是对同步信号的频率连续跟踪,例如长城机 GW-500 就是采用频率连续跟踪,这里不作介绍;另一种是按频段进行跟踪。它的特点是:用处理后的同步信号去控制一个多路开关,通过多路开关的切换来自动改变行振荡器的 RC 时间常数,从而改变振荡频率和相位达到行频自动跟踪的目的。本文所述故障系统就采用后一种方式。具体电路如图 7 所示(自己测绘的电路可能有误,仅供参考)。从电路图可以看出该机有两组 RC 时间常数。当显示器与 VGA 卡(640×480 点阵)联机时,行、场同步信号 HS、VS 极性相同,均为负极性脉冲,经过 74LS86P 芯片(四异或门)处理后输出逻辑低电平,加到与非门 74LS00 芯片(四二输入与非门)的 12 脚、13 脚,则 11 脚恒为高电平,再经一个“与非门”和两个三极管倒相,使多路开关转换器 CD4053 芯片的 9 脚、10 脚、11 脚呈现低电平。从 CD4053 芯片真值表可看出 4 脚与 5 脚接通,VR304 通过电阻 R307 被接到 TDA1180P 芯片的相位控制端 15 脚,调整 VR304 可以改变行振荡的相位。同时 15 脚接通 2 脚,VR302 可以改变行振荡频率。

当显示器 640×400 点阵显示卡联机或与 640×350 点阵显示卡联机时,行、场同步信号 HS、HV 为

表 4 真值表

INH	控制输入			导通开关		
	选择			Z	Y	X
0	0	0	0	Z ₀	Y ₀	X ₀
0	0	0	1	Z ₀	Y ₀	X ₁
0	0	1	0	Z ₀	Y ₁	X ₀
0	0	1	1	Z ₀	Y ₁	X ₁
0	1	0	0	Z ₁	Y ₀	X ₀
0	1	0	1	Z ₁	Y ₀	X ₁
0	1	1	0	Z ₁	Y ₁	X ₀
0	1	1	1	Z ₁	Y ₁	X ₁
1	X	X	X	—		

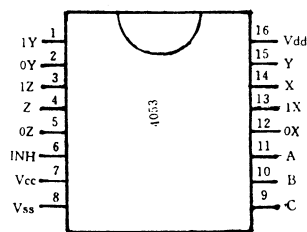


图 11 4053—三 2 通道多路转换器/多路分配器

一负一正和一正一负,极性相反。与上述的工作过程相同,74LS86P 芯片 11 脚恒为高电平,再经过一个“与非门”和两个三极管倒相 CD403 芯片的 9 脚、10 脚、11 脚恒为低电平,这时 CD4053 芯片转换开关状态不变,仍然是 VR304、VR302 起作用,所以行振荡频率和相位没变化,与 640×480 点阵的显示卡联机一样。当显示器与 800×600 点阵和 1024×768 点阵的显示卡联机时,行、场同步信号 HS、VS 的极性均为正极性,经过 74LS86P 和 74LS00 芯片处理后,在 74LS00 芯片 11 脚呈现逻辑低电平,再经一个“与非门”和两个三极管倒相,使多路开关转换器 CD4053 内部开关转换器 CD4053 芯片的 9 脚、10 脚、11 脚却变为高电平,从而使 CD4053 内部开关 Z 动作,4 脚与 3 脚接通,VR303 通过电阻 R307 与 TDA1180 芯片 5 脚接通,调 VR303 可以调整 35KHz 行振荡的相位。开关 Y 动作将 15 脚与 1 脚接通,通过电阻 R302 与电位器 VR301 接通,15 脚通过电阻 R528 与 TDA1180 芯片 15 脚相连,VR301 可改变 35KHz 振荡器的频率。从上述分析可知,显示方式不同所对应的积分常数也不同,不能乱调电位器。

显示器单机加电时,多路开关转换器 CD4053 不动作,行振荡时间常数比较大,行频比较低,但行电源调整管不工作,这时行电源直接由 D805 稳压管提供,仍为 40V,所以阳极高压和加速极电压都很低,以保护显示器安全。

SKILL

没有电路图时修复微机的几点经验

□ 杜 勇

由

于机器的类型很多,每当外出维修,经常碰到身边没有机器图纸的情况。随着维修实践的积累,笔者总结出几条在没有机器逻辑电路图时的维修方法。

观察法

碰到一台有故障的机器,首先要仔细观察线路板接插件是否有松动,接触不良、虚焊、脱焊、断线、短路、元件锈蚀、损坏等明显的故障,同时也要观察在加电时是否有火花、异常噪声、过热、烧焦等现象的出现。养成仔细观察的良好习惯,有时能使许多问题迎刃而解。笔者曾修复一台型号为 CWY-1K 的参数稳压器,根据用户反映,在使用时有白烟和烧焦气味,笔者便将电路板取出,直接用观察法看出有一电解电容外面塑料胶皮被烧焦,电容有液体流出。用同型号的好的电容将此电容替换掉,参数稳压电源正常工作。

代换法

用相同的部件代换机器中相同的部件,根据故障现象的变化,能很快缩小故障范围;也可以根据自己的维修经验用好的芯片来代替怀疑可能是不正常的芯片,观察故障现象的变化,便能确定此芯片的好坏,从而有助于迅速查到故障源。特别是一些大规模集成电路芯片,在没有任何图纸和资料的情况下,判断它的好坏的唯一办法就是代换法。当然,在代换前一定要保证电路板上没有短路的情况存在,以免将好的芯片烧坏。对于主机板,根据出现故障的概率的大小,要先替换输入输出接口芯片,后替换一些固存启动程序的芯片,最后再替代 CPU 等核心控制电路的芯

片。

对比法

即用相同逻辑,相同结构的两部件的各点波形电压进行对比来迅速查出故障源。例如,将一有故障的微机的主机板各关键点的波形与另一台正常机器的主机板的各关键点的波形对比,查出波形不同的各点,进而就能查出故障源。再比如在 I/O 插槽上同时插入相同的两块插件板,通过对逻辑功能相同的各点波形进行比较,找出不同之处等。

测量中、小规模集成块的逻辑信号

一些中、小规模集成电路芯片,其逻辑功能简单明了,可在加电的情况下,用逻辑笔对其进行逻辑测试。例如 74 系列集成块,参照 74 系列芯片手册,如果测量它的逻辑功能与手册上不符,在判定工作条件正常的情况下,便可初步判断它是坏芯片。笔者曾修复一台 BCM0530 主机中的一串并卡,它的故障现象是不能进行数据通讯。笔者在测试串并卡上 U6(MC1489)的逻辑功能时,笔者发现它的输入端 3 脚是高电平,而它的输出端 13 脚却是浮空状态(应该是低电平),是不对的,将其换掉,机器便能正常进行数据通讯。

测量集成块的工作条件

对于大规模集成电路芯片,由于其功能复杂,除对其直流电阻值进行测量外,通常采用测量其工作条件的方法来判断它的好坏。比如 CPU 芯片,先测量它的基本工作条件(开机时的复位信号 RESET,系统时钟信号 CLK,准备好信号 READY)是否满足,再看 CPU 保持

(下转第 60 页)

TH3070 打印机常见故障的维修

□杜长勇

无

论是在机械结构方面,还是在电路方面,3070 打印机都可称得上是针式打印机的代表性产品,所以只要懂得了怎样修复 3070 打印机,那么也就能修理其他型号的打印机。笔者根据自己的维修经验和手中的资料,阐述了 TH3070 打印机的常见故障的修复方法和思路。

一、故障现象:打印机打印时不走纸,打印字迹成一条黑印。

故障分析与维修:打印机不走纸,可能是由于机械方面的原因,也可能是电路方面的原因。机械方面的问题主要是看 PJ12 插头是否插好,齿轮啮合是否正常,或者电机传动轴螺钉是否松动。按步骤将打印机拆开后,检查上述机械部件,并重新拧紧各螺钉,故障并未消失,说明机械部件是正常的。应重点检查电路部分,走纸电机是四相步进电机,如果步进电机缺一相或几相,或者步进电机的驱动电路损坏,则它不能正常运转,造成走纸不正常。在排除走纸电机本身损坏的可能性后,用示波器检查它的逻辑电路。走纸电机的线圈 3、4、5、6(四相)端,通过插头 PJ12 接到 IC43(TD62064)上,再通过 IC44(7406)接到 IC15(8155)上,IC15(8155)输出的脉冲通过上述路线控制电机运行,用示波器测量上述器件的输入输出波形并未发现有器件损坏。输纸电机的高压(正 16 伏)电路是由 IC14(74LS74),IC61(74LS06)及 Q8 组成,然后接到步进电机线圈 1、2 端。用示波器测量 IC14 的 12 脚有波形输入,而输出端 9 脚却呈不高不低的浮空状态,换下 IC14 后,故障

排除。

二、故障现象:打印机加电后打印头回到左界仍然不停步,继续向左端撞去,电机发出怪叫声。

故障分析与维修:发生这种情况,一种是左界开关被左移,或者左界开关坏,或者控制电路 ROM、2764,IC27(8155)坏。得知检查通过左界开关是正常的,调整左界开关的位置,故障并未消失。测量 IC27 的 33 脚有波形变化,说明打印头已送出“回车”信号到 8155,故障可能是 8155 或 2764。换掉 8155 芯片,故障消失。

三、故障现象:打印的字形中缺点,使打印出的字形当中有细白缝。

故障分析与维修:打印漏点,可能是对应的打印针断针、短针或者是打印头上线圈断线、失效,或者是打印头驱动电路有故障。拆下打印头观察,并没有断针,短针现象,用另一台机器正常的打印头连上此台打印机,仍不能打印,说明原来打印头线圈没断,打印头是好的。主机送来的要打印的数据,经 3 片 74LS374 送到 6 片 TD62064 芯片进行信号放大,驱动打印头出针,检查 6 片 TD62064。有针数据送

**FBASE DOS 下的最佳
多媒体数据库平台**

新未来电子技术公司

电话:2573355-271,272

地址:清华东门清华园宾馆一层

来时,输入为高,经 TD62064 反相后为低。当测试到 IC56 时,发现它的输出为常高,说明 IC56 损坏,更换之,故障排除。

四、故障现象:在打印的字形中出现一条连续的横线。

故障分析与维修:有一条连续的横线,说明印字连点,即某根针连续出针所造成。检查针数据形成和针驱动电路。用示波器检查 IC62——IC64 (74LS373)的输出端,为时高时低的波形,这是正确的;检查 IC40——IC42、IC55——IC57 的输出,除了 IC41 的输出为常低的电平外,其余都是正常的波形信号,说明 IC51 (TD62064)已坏,更换掉 IC51,故障得以排除。

五、故障现象:打印机与主机联机后打印不出字来,但打印机自检正常。

故障分析与维修:打印机自检正常,说明打印机本身控制部分基本正常。在排除打印驱动程序与打印机不匹配等情况后,联机不打印的故障多出在接口电路上。打印机与主机之间的接口信号主要有以下几种:1、数据选通信号(STB),通过 IC72 加到 IC27 的第二脚。2、打印机发给主机的回答信号(ACK),它的传输路线是 IC27—5——IC52—13——IC74—5——PJ32—21。3、忙信号(BUSY),它的传输路线是 IC27—37——IC52—1——IC50—10——IC76—11——PJ32—23。用示波器检查数据选通信号和回答信号并未发现有故障,当检查忙信号时,IC(8155)的 37 脚有脉冲波形,IC52 的第一脚也有很好的波形,但 IC52 的第 2 脚却为浮空状态,由此便可推测,主机接受不到打印机发出的正确的忙信号,便不向打印机发送数据,从而出现联机不打印的现象。换掉 IC52 后,故障排除。

六、故障现象:给打印机加电后,字车不回左界。

故障分析与维修:主控电路有问题或者电机驱动有问题都会造成字车不动。可以先模拟字车的回车动作来判断主控电路有没有问题。打印机加电后,按动左界开关后再按动“联机/脱机”键,发现联机灯正常亮灭,说明 CPU(8085)及其主控电路是正常的,故障出在电机驱动电路上。检查驱动电路,测得 IC61(74LS06)第 3 脚输出有脉冲,但第 4 脚输出却为高电平,造成字车电机缺相,使字车不能走动,更换之,故障排除。

七、故障现象:给打印机加电后,只有电源灯亮,打印头无动作,也没有其他反应。

故障分析与维修:电源灯亮,说明电源部分基本正常,打印机无其他反应,说明打印机加电后,自检没有完成,所以故障出在打印机 CPU 及主控电路,

或者是 ROM 电路上。对于 ROM 电路的检查,只有用一块好的 ROM 芯片换掉原来的 ROM 芯片,故障仍未排除,说明故障不在 ROM 电路上。打印机的主控电路由 IC26(8085),IC69(74LS373),IC37(74LS139),IC27,IC15(8155)等组成,一般情况下,CPU8085 不会出故障,用示波器检查 IC69 的输入/输出脚,脉冲信号正常。再检查 IC37 的 4 脚为低电平,5 脚为浮空状态(它们应互为反相关系)是不正确的,换掉 IC37,重新开机,机器工作正常。

八、故障现象:当打印机用窄行纸打印时正常,但用宽行纸时不能打印且故障灯亮。

故障分析与维修:能用窄行纸打印,说明打印机的控制电路和驱动电路都是正常的,所以故障出在机械方面。当用宽行纸打印时,字车要走到打印机的右端,这点刚好与用窄行纸打印不一样。仔细观察,在传动皮带上装有一个限位片,当字车向右运动时,为了防止撞壁,此限位片接通越界开关,通过控制电路控制字车停止移动,所以这是由限位片的位置调整不当引起的上述故障现象,将限位片向右,重新调整,即可排除故障。

九、怎样给打印头换针

当打印针短针或断针时,就要给打印头换针。换针时,首先拆下后盖,把打印头朝下,尾部朝上,用镊子取出要换的针,然后把新针在对应位插下,打印针从尾部到探出头,共要经过四道孔槽,最上边的孔和第二道孔是很容易插下去的,上面这两个孔实际上起着打印针的定位作用。接着,针顺势而下,穿过导向块,导向块是一个尼龙片,在它上面有两个弧形孔,奇数针和偶数针分别穿过其中的一个孔,针在穿过这两个孔时,先用镊子把针捋几下,把奇数针和偶数针分别捋在两个孔内捋平,使该针基本上处于该处的位置,针进入最后一个孔后,就从打印头冒出来,若用手无法装进时,可用镊子在第三、四孔之间再捋几下,或者边捋边插,一直到针穿出。换好针后,可用锉刀将新针的比其他针长出的部分锉掉,使新针和其他针平齐,然后再盖上盖。到此,打印机换针工作就算完成了。换针时,切忌把拆盖后的打印头头朝上放置,以免其他的针掉下来。


SKILL


UNIX 找龍馬

为您提供 UNIX XENIX 软汉字中文包 LMCP

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

DOS 与 Windows

 如何使用 QBasic 帮助中的范例程序?

 初学编程的人可以通过运行 QBasic 中的 Help 一项的程序范例学到很多有用的关于 QBasic 的知识。这一程序可以使 QBasic 的 Copy 和 Paste 选择项用起来十分方便。

使用步骤如下:

(1) 从屏幕顶行最右端的 QBasic 的 Help 菜单中选择“Index”选项;

(2) 从所列的关键字中选择一个你所需要的范例程序。例如,选择 COLOR 描述入门;

(3) 突出范例程序的所有行;

(4) 从 Edit 菜单中选择 Copy 选项,或直接按下 Ctrl+Ins 键;


(5) 按下 Esc,退回到 QBasic 的主屏幕;


(6) 从 Edit 菜单中选择 Paste 选项,或直接按下 Shift+Ins 键;

(7) 按下 Shift+F5,或选择 Run 菜单中的 Start 选择项;

(8) 用列出的内容来做实验,直到懂得它如何工作为止。


当然,你也可以使用这个方法,将一段段程序联结起来,看它们是如何相互作用的。


 如何替换 DOS 编辑器中 OPEN 选择项中的“*.TXT”成为“*. *”?

 当你选择 MS-DOS 全屏幕编辑器——EDIT 的文件菜单(File)中


的 Open 选项时,显示出的文件表中只包括那些后缀为 .TXT 的文件,在 QBasic 中的文件菜单(File)的 Open 选项也有同样的限制,它只能列出以 .BAS 为后缀的文件。但是,你可以改变这些程序以便它们显示当前目录中的所有文件。


但是改变程序的操作中涉及到编辑 QBASIC.EXE 程序(EDIT.COM 是 QBASIC.EXE 的一部分),这样做有一定的危险,所以最好在对程序做任何改动之前先将 QBASIC.EXE 和 EDIT.COM 做备份,备份好后,使用一个磁盘编辑器,例如 NORTON, PCTOOLS 的磁盘编辑功能,寻找这样的字符串:*.TXT,在 QBASIC.EXE 程序中,它出现在偏移量 247,413 的地方。用星号(*)的十六进制的值 2A 来代替“TXT”中的第一个 T,以改变所指定的文件类型,使程序以后再执行时,列出所有的文件(*.*).插入“*”后,再插入一个空字符,代替“TXT”中的“X”和第二个“T”,同时代表字符串结束的标志。然后寻找“*.BAS”字符串,在 EDIT.COM 程序中,这个字符串位于偏移量 248,752。用一个星号(*)代替“BAS”中的“B”,然后插入空字符来代替“A”和“S”,并表示字符串结束。将修改后的文件存盘,退出编辑,一切就都做好了。

 非 Windows 程序如何在 Windows 环境下运行?

 使用 PIF 编辑器编辑出一个相对于该非 Windows 程序的 PIF 文

件,即可使之在 Windows 下运行。


 如何快速退出 Windows?


 方法一:(1)按 Ctrl+Tab 激活 Program Manager;


(2)双击控制菜单;


(3)确认。

方法二:(1)按 Ctrl+Tab 激活 Program Manager;(2)按 Alt+F;(3)选 Exit;(4)确认。

 如何修改 Windows 中的几个重要的系统初始化文件?


 可在 Program Manager 中,按 Alt+F,选 Run 菜单项,键入 sysedit,即可对 system.ini、Win.int、Config.sys、autoexec.bat 进行修改,其备份文件名后缀为 .syd。


 如何收缩子目录?

 方法一:定位要收缩的子目录,按“—”;

方法二:定位要收缩的子目录。选 Alt+T 中的 Collapse Branch 选项。

方法三:双击要收缩的子目录图标。

 如何选择多个连续文件?

 先定位第一个文件,再按 Shift,定位最后一个文件。

如何选择多个非连续文件?

方法一:先定位第一个文件,再按 Ctrl,定位其余各文件。

方法二:使用键盘。(1)按下 Shift+F8;(2)按光标键定位;(3)按空格选择;(4)完成后按 Shift+F8。

如何改变 Windows 的 shell?

(1)在 Program Manager 下运行 sysedit;

(2)修改 system.ini 的 shell 命令行为 Pragman.exe 或 Winfile.exe。

如何把时钟总是放在桌面上?

(1)激活 clock 程序;(2)按 Alt+空格,选 always on Tap;(3)按 Alt+S,选 No title。

在 Windows 下如何运行 DOS 命令?

运行 MS-DOS Prompt。

注意:此时若切换回其他应用,应按 Ctrl+Esc;若退出 MS-DOS,则应打入 DOS 命令 Exit。

如何使用剪贴板?

在激活窗口时,按 Alt+PrtScr 拷贝当前活动窗口。再按 Prtsc 拷贝当前屏幕。

注意:只在 386 增强模式下。

如何选择实型图形的边框颜色(画笔中)?

用右键选定。

如何修改时间片?

运行 Control Panel 中的“386 增强”程序项,修改 Minimum Timerice 数值即可。

如何在书写器中输入图象?

(1)激活活笔编辑图象,并将欲剪贴部分拷贝/剪贴且粘贴板;
(2)激活 WRITE,定位输入点;
(3)选 Alt+E,选 Paste。

如何在画笔中精确定位?

按 Alt+V,选择 CursorPosition。

如何创建自己的应用肖像(在包装中)?

(1)打开画笔或其他绘图文件;
(2)画一幅画或打开一个文件;
(3)把画拷入剪切板;
(4)打开 Object Package;
(5)选择 Appearance 窗口;
(6)从剪切板上粘贴;
注:此时没有标签。

如何创建一个包含完整的文本文件的包装?

(1)在 File 子菜单中选 Export;
(2)在对话框中选一个文本文件;
(3)然后可按 Insert Icon 编辑肖像;
(4)从 Edit 菜单中选 Copy Package;
(5)打开目标粘贴应用,选用 Edit 中 Paste 即可。

在以文件管理器作为 Windows 的外壳时,如何运作?

直接执行某一应用程序的可执行文件就可以了。

切换输入方式的 Ctrl+Space 键为什么,有时只有右边有效,有时又两边都有效?

这要由主群组—控制面板—输入方法中设定。

可不可以把一个图标从一个程序组放到另一个程序组?

可以。把要拖动的程序项直接拖到另一个程序组就可以。如果是 copy,拖动时同时按 Ctrl 就可以了。

如何实现跨磁盘操作?

首次必须平铺的打开多个窗口(用文件管理器中的窗口菜单中选新建窗口来打开多个窗口)用 Alt+拖动的方式实现跨盘的移动。

Alt+Tab 简捷键有什么作用?

在各个被打开的窗口之间切换。有时某一程序用了整个窗口时,用此键操作十分方便。

为什么有的程序项不可拖曳?

此程序项也许被设置成了只读属性,这时移动、拷贝、创建、删除此程序项都会出现错误信息。

如果 Windows 应用程序能同时操作两个以上的文档文件,如何创建能启动多个文档文件的程序项?

在程序项特性对话框的 Command Line 中首先指定执行文件名(设定关联已存在),其后紧接着是每个文档文件的名字。

使用外部图标的方法是什么?

在 Change Icon 对话框的 File Name 文字框中键入含有外部图标文件的文件名。在 Windows 目录下, Moricons.DLL 的文件中含有大量可用图标, Program.EXE 中也含有许多可用图标。

退出 Windows 时要保持窗口布局(下次启动 Windows 被恢复),应如何实现?

(1)退出之前选 option in save setting on Exit 命令。

(2)未选 save setting on Exit 的前提下,选 File 菜单中的 Exit 的同时按 Shift 键。

? Windows 的菜单项可否由用户修改?

! 可用 VB 编写宏来,根据用户需要修改 Windows 系统菜单,Windows 留有这方面接口。

? 一种数据文件可否关联多个可执行程序?

! 不可。

? 运行 Windows 时经常出现 VAE (不可恢复错误)时怎么办?

! 在程序管理器中运行 DRWATSON.EXE,出现 VAE 后,用户可详细描述当时的状况,反馈给 Microsoft。

? Windows 中关联文件定义在何处?

! 在 WIN.INI 中的 extensions 部分。

? Windows 3.1 和 Windows NT 有什么区别?

! Windows 3.1 是建立在 DOS 上的一种图形用户界面,依赖于 DOS;Windows NT 是一种大型的、独立的操作系统网络版。

? Windows 3.1 可不可以作为一种独立操作系统而不依赖于 DOS?

! 不可。但 Windows 95 (即 Chicago)可以,它是一种图形用户界面的独立的操作系统。

? system.INT 文件和 system.SYS 文件有什么作用?

! system.INT 用于设置硬件的当前状态。

system.SYS 中是一些关于系统的信息。

? 当某一非 Windows 应用程序建立在 Windows 下后,有时运行不了,是什么原因?

! 可能是此非 Windows 应用程序的 PIF 文件某些设置太小,改变其中的某些设置,即重新改写这个非 Windows 应用程序的 PIF 文件,就可以了。

? REG.DAT 是一个什么文件?

! 标记数据库(只是 Windows 3.1 中有此文件),它登记所有应用程序的信息来源。

? 当要建立某个应用程序时(非 Windows 下的到 Windows 下的),如果它有多个可执行文件怎么办?

! 主群组—Windows 设置程序—选项目菜单—安装菜单—安装应用程序,它就会自动生成某应用程序的各程序项。

? 切换前台程序的方法都有什么?

! (1)Alt+Tab;(2)Alt+Esc;(3)使用 Task list。

? 在 Windows 运行中运行 MS-DOS prompt 程序有什么注意事项?

! 运行带/F 参数的 MS-DOS 实用程序是不安全的,故运行此类程序应先退出 Windows。

? 什么样的程序不可和某应用程序相连接?

! 扩展名为.GRP 和.FON 的文件,.GRP 文件包含有关 Program Manager 组的数据,.FON 文件含有字库的有关信息。它们不能直观读取和编辑。

? 内有一注释点的书页符号代表什么?

! 它代表一个具有隐蔽性或两者兼而有之的文件,这些属性通常赋给系统中的重要文件,删除或更改这些

文件不能导致灾难性的后果。可由 view 菜单中的按文件类型取消这一缺省设置。

? Notepad 是一种什么工具?

! 它是一个简单文本编辑器,它创建的文件可被任何 Windows 或非 Windows 的字处理程序调用。其文件最大可达 50,000 个字符。

? 控制面板 control pad 上的“打印机”程序项与 Main 主群组中的“打印管理器”有什么区别?

! (1)完成的任务不同。打印管理器是管理系统上连接的所有打印机,它把打印信息送入磁盘上的一个假脱机文件(spool file),并管理一个打印队列,显示队列的各个信息,“打印机”是用来安装打印机驱动程序,连接驱动程序和端口,设置打印方式等。

(2)存在方式不同。打印机在控制面板上,而打印管理器在主群组中,或随打印命令自启动,打印完后自行消失。

? 如果屏幕保护口令忘记了怎么办?

! 可以打开 CONTROL.INT 文件,将 pwprotected=1 的行改为 pwprotected=0。

? 为什么在 file manager 中,不能启动网络共享资源的 Pwindows 安装程序?

! 因为不论 Windows 还是 Pwindows 的 Setup 程序都是 DOS 下的可执行程序,它们必须在 DOS 下启动,不能在 Windows 环境下启动 Windows。

? Windows 3.1 中有无五笔字型输入法?

! Windows 3.1 只有全拼、国标/区位输入法,若要装五笔字型输入法是可以的,但需有微软公司提供的一种接口,具体接法可与王码公司联系。

SKILL

帝霸计算机反病毒服务网 专用软件升级信息表

基本描述

库类别:	0	病毒名称:	93v
攻击对象:	2	病毒长度:	545

第(1)基址描述

0/0/0:1/82/FF00

第(1)特征描述

1/AE/3D. FE. 87. 74. 1F. BA. E8. 5. B8. 1C. 25. CD. 21. B8. 21. 35

第(1)控制描述:

1/100/E9. 1D. 1,1/229/1. C6. 29. FE. FC. F3. A4:1/230

基本描述

库类别:	0	病毒名称:	93v
攻击对象:	3	病毒长度:	545

第(1)特征描述

0/AE/3D. FE. 87. 74. 1F. BA. E8. 5. B8. 1C. 25. CD. 21. B8. 21. 35

入口描述:

0/2E1,0/4E6,0/4F9,0/434,0

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。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组建了帝霸反病毒服务网,希望该软件能更好地为广大用户服务,为广大用户建立和维护一个安全干净的计算机工作环境。

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。增强版对上网会员免费提供,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:

★(100836)北京海淀区定慧西里19号楼帝霸计算机技术研究所,联系电话:(01)8123896,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. 每月按时邮寄一份新病毒的完整升级信息表,网员可以对本网专用软件自行升级。
3. 网员发现病毒,本网可在24小时内做出快速反应,全面分析,形成升级信息表,并传送给网员,网员可以将其输入本网专用软件,将此病毒清除(传送可用电话或传真)。此项服务一年中4次以内免费,超过4次按30元/次收费。
4. 每月免费赠送一本《电脑编程技巧与维护》杂志。
5. 向网员通报特定病毒的发作及流行情况,及时提醒网员加强预防。

6. 为网员提供反病毒技术咨询、有偿修复硬盘、恢复数据等服务。

7. 不定期举办反病毒技术研讨会和学习班,学习和研讨反病毒技

术的发展,介绍计算机反病毒技术,网员可获优先和优惠。

8. 为网员推广计算机技术产品牵线搭桥。

9. 为网员提供项目咨询和相关的技术服务。

10. 为网员提供合法的常用软件,仅收成本费。

二、B类:90元/年(面向个人用户)

服务内容:

1. 每月按时邮寄一份新病毒的完整的升级信息表,用户可以对本网软件自行升级。不定期为网员提供相关的开发经验、技术、技巧。
2. 每年本网发表的软件升级版本,均免费赠送一套。
3. 网员发现新病毒,本网以尽可能快的速度给网员解决问题,并将升级信息表列入邮寄的资料中,回寄给网员,解决网员的问题。
4. 通报新病毒发现地点及泛滥区域,提醒用户预防。
5. 有偿提供修复硬盘,恢复数据等服务。
6. 本网举办各种技术研讨会和学习班时,可获优先。

三、专项服务:价格面议(面向系统用

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)北京海淀区定慧西里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。

户)

服务内容:

按网员要求提供全方位服务。

帝霸计算机反病毒服务网入网申请表(复制有效)

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

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

本刊第二届“热心读者”获奖名单:

- | | | | |
|--------|-----------------------------|--------|-----------------------------|
| 1 邵 刚 | 江苏,苏州大学工学院 | 21 刘 宁 | 陕西,西安电子科技大学 |
| 2 钟耀伟 | 福建,将乐县乐兴石英晶体有限公司 | 22 智效曾 | 山西,太原重机学院计算站 |
| 3 谢永泽 | 江苏,南通天生港发电厂 | 23 张党群 | 青海,李家峡水电四局 |
| 4 叶国芳 | 北京,环境保护局 | 24 陈代忠 | 安徽,合肥工业大学 |
| 5 张兴明 | 北京,汉声电脑有限公司 | 25 唐政科 | 湖南,9613 厂军品科研所 |
| 6 赵立平 | 北京,中科院教学所 | 26 赵德伟 | 云南,寻甸县财政局 |
| 7 丁 彻 | 北京,清华大学 | 27 冯继勇 | 江西,南昌飞机制造公司 |
| 8 靳 涛 | 北京,解放军总参通信部自动化工
作站信息服务中心 | 28 刘建国 | 天津,天津市煤气河东营业所 |
| 9 李 理 | 北京,公安部第一研究所 | 29 邸合星 | 河北,秦皇岛市糖酒公司 |
| 10 郭 涛 | 北京,中国石油天然气总公司 | 30 李惠然 | 天津,河北工学院 |
| 11 刘欣联 | 北京,国家气象中心 | 31 孙秀海 | 山东,淄博齐鲁石化检修公司 |
| 12 王洪波 | 湖南,湘南地质勘察院 | 32 郑晓军 | 山东,东营石油大学(华东)胜华炼
厂重质油研究所 |
| 13 陈 翔 | 河南,洛阳测量与跟踪技术研究所 | 33 贾庆亮 | 山东,莒信涤纶有限公司 |
| 14 宋玉长 | 河南,邓州市人民银行 | 34 王宏亮 | 上海,黄浦区中医医院 |
| 15 李 军 | 湖北,武汉新世界制冷工业有限公
司 | 35 严 峰 | 上海,国家医药管理局上海医药设
计院 |
| 16 唐有瑜 | 湖北,湖北省机电设备股份有限公
司 | 36 吴 翀 | 福建,厦门东南亚工程发展有限公
司 |
| 17 王 康 | 贵州,中国江南航天工业集团公司 | 37 黄晓伟 | 福建,福建机电学校微电子技术开
发中心 |
| 18 沈水富 | 浙江,浙江师范大学 | 38 杨亚肃 | 辽宁,解放军 201 医院 |
| 19 朱 猛 | 江苏,淮阴市第二人民医院 | 39 张善荣 | 四川,重庆碱胺实业总公司 |
| 20 钱建辉 | 辽宁,大连理工大学 | 40 毛世秋 | 内蒙,呼盟大雁矿务局 |

《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	

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。

《电脑编程技巧与维护》订阅单(复印有效)

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

订阅须知:

△本刊每月8日出版,每期订价:3.8元,全年订价:45.6元。1994年出版发行6期。邮局公开发行,邮发代号24-106。

△请将本卡寄至:(100006)北京市劳动人民文化宫内《电脑编程技巧与维护》杂志社发行部。联系电话:(01)5123823。

△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142服。银行帐号:8014054。帐户:电脑编程技巧与维护编辑部。

△款到后即按月寄刊。

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_____

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"/> 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"/> 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: <input type="checkbox"/> 寄取公司资料 | <input type="checkbox"/> 寄取产品目录 | <input type="checkbox"/> 寄取产品资料 |
| <input type="checkbox"/> 询问价格 | <input type="checkbox"/> 建立业务联系 | <input type="checkbox"/> 其他_____ |