

现代计算机

'95.9

总第 44 期

原名 《电脑与微电子技术》

现代计算机杂志社出版

MODERN COMPUTER

66-121



40M²LED 全彩色电视广告屏 矗立在中国出口商品交易会门前

广州中鸣公司隆重推出

公司地址：广州中山大学东北区 368 号
董 事 长：张苑岳 传真：4185464

电话：4184847
邮编：510275

现代计算机

'95.9

总第44期

原名《电脑与微电子技术》

现代计算机杂志社出版

MODERN COMPUTER

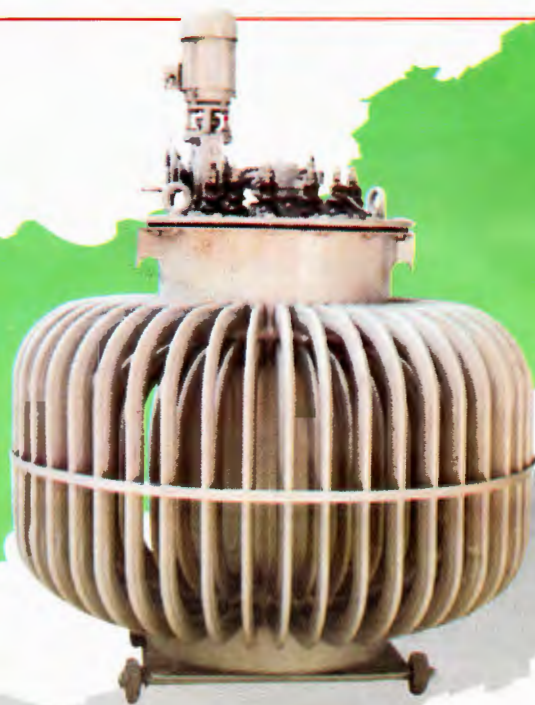


单、三相自动调压

精密交流稳压器

PRECISE AUTOMATIC AC VOLTAGE STABILIZERS

单相 TND-T 型系列
三相 TNS 型系列



中山大学电器设备厂是专业生产稳压电源的厂家，
是国家教委审定的合格企业，
自 80 年代初，
工厂一直依靠大学的先进科学技术，
秉承“信誉为本、
质量取胜”的宗旨，
精益求精地生产高质量的各种型号单、
三相精密交流自动调压稳压器，
其中 TND-T 型系列精密交流自动调压稳压器，
90 年度荣获国家教委科学技术进步奖二等奖。
产品广泛应用于各类高科技电子仪器
以及三资企业等用电设备领域。
在全国 20 多个省市设有经销维修网点。
十多年来，
真诚地服务於全社会，
受到用户的欢迎。

厂址：广州市新港西路 135 号中山大学西南区
电话：4186882
电挂：8775 电器设备厂
电码：510275

中山大学电器设备厂

广州中鸣 一鸣惊人

——广州中鸣显示技术工程有限公司进展

中鸣显示技术工程公司是中山大学与香港得时利公司合作的结晶，专业研制、生产户内外的各种显示屏。由于拥有一批出色的科研人员和高效率的生产基地，公司在短短的几年里取得了很大的发展，产品遍布省内外。

OFC 系列户外全彩色 LED 显示屏是中鸣显示技术工程公司的最新产品，集最新发光材料、视频处理技术、多媒体技术、耐候处理技术于一身，被列为 1994 年国家级重点新产品。

全彩色 LED 显示屏是以红、绿、蓝三种颜色的发光二极管作为发光材料，它比其它的显示元件更适合于户外的工作环境，所以 LED 显示屏成为近年户外广告媒体的新宠。但是，由于技术上的限制，一直不能生产出适合实际使用的蓝色 LED，国内外的显示屏全都是红黄的色调，极大地限制了广告的制作，也影响了 LED 显示屏的发展。

随着半导体技术的发展，94 年下半年，中鸣公司与美国合作公司紧密配合，采用最新出品的蓝色 LED，研制成功了全彩色 LED 显示系统，填补了国内的空白，制作了国内第一块户外彩色 LED 显示屏。经测试，它具有亮度高、层次丰富、色彩平衡、性能稳定的特点，使用寿命长更是其它代用品所无法比拟的。它的出现，给电子显示行业带来了震动，同时也为之展示了更为广阔、更为美好的前景。

本着精益求精的原则，中鸣人潜心研究，对整个显示系统进行了科学论证，不断完善，特别是在结构和耐候方面，更是设计出适合不同纬度地区气候特点的系统，对耐热、耐寒、防紫外、抗老化等方面的性能有很大提高，使 OFC 系列户外全彩色 LED 显示屏更趋完善。

1994 年 11 月，中鸣公司参加了 94' 北京国际广告“四新”展示交易会。会上，作为唯一的 LED 全彩色显示屏，OFC—28 全彩色 LED 显示屏（5 平方米局部）吸引了众多的行家，并以丰富的色彩、生动的图象、逼人的光焰和良好的工艺、结构获得一致的好评。（王光英同志参观时特意驻足中鸣公司展位，留心听取了介绍，并给予肯定）。

1995 年 4 月，中鸣公司制作的全彩色 LED 显示屏矗立在广州外贸中心（交易会）的正门，并在春交会的开幕式上正式开播，以清晰、稳定的图像直播了春交会开幕式的盛况，博得众多来宾的赞誉。该屏开播以来，经历了风雨、雷电和酷暑的考验，每天播出十多小时，一直正常运行。

1995 年 5 月，中鸣公司通过美国代理公司参加了在多伦多举行的 NESA（Northern American Electronic Sign Association）展示会，在有美国、加拿大、日本、韩国、台湾等国家、地区专业公司参加的情况下，中鸣送展的全彩屏仍引起全场瞩目，为开拓海外市场迈出了成功的一步。

中山大学 电脑照排科技中心

博采众长 自成一格
技术先进 服务周到

经营：高档精密照排系统

普及型轻印刷排版系统

电脑多媒体网络系统

书报刊排版印刷

承接：印刷厂技术更新改造

新老排版系统换代升级

信息管理系统软件开发

电脑排版技术培训

我们的客户遍布茂名、珠海、湛江、深圳、阳江、广州
和省内外，欢迎垂询，欢迎更多的朋友加入我们的行列

诚聘：电脑技术人员 2 名。条件：男性，大专文化以上，
年龄 30 岁以下。有意者请寄来自述简历及半身脱
帽相片一张，听候约见。一经录用，待遇从优。

地址：广州中山大学东北区 368 号 邮编：510275

经理：梁 勇

电话：4186300—1999

《现代计算机》订户报销凭证

| | | | | | | |
|--|--------------|---------|-----------------|---------|-----------------|---------|
| 单位全称或收件人姓名 | | | | | | |
| 单 价 分 类 | 1 册单价 3.80 元 | | 2—4 册优惠价 3.60 元 | | 5 册以上优惠价 3.40 元 | |
| | 上半年 | 全 年 | 上半年 | 全 年 | 上半年 | 全 年 |
| | 11.40 元 | 22.80 元 | 10.80 元 | 21.60 元 | 10.20 元 | 20.40 元 |
| 册数 | | | | | | |
| 金额 | | | | | | |
| 金额 (大写) 仟 佰 元 角 分 | | | | | | |



说明：此单与汇款单据一起作报销凭证

(以下请订户填写清楚寄回杂志社 邮编：510275 地址：广州中山大学 信汇银行帐号见背面)

《现代计算机》订户回执单

| | | | | | |
|------------------------------|------------------------------|-------------------|-------|-------|---|
| 单位全称及收件人姓名 | | | | | |
| 订 期 | | 册 数 | 单 价 | 金 额 | 订购单位章 |
| <input type="checkbox"/> 上半年 | <input type="checkbox"/> 全 年 | | | | |
| 汇出金额合计 (大写) | | 仟 佰 元 角 分 | | | 邮汇 <input type="checkbox"/> 信汇 <input type="checkbox"/> |

1

收

份

2

收

份

3

收

份

4

收

份

5

收

份

6

收

份

欢迎订阅《现代计算机》杂志

《现代计算机》杂志，原名《电脑与微电子技术》，系应用技术刊物，中山大学主办，创刊于1985年，1995年更名《现代计算机》。本刊以注重实用性、新颖性与知识性为特色，立足国内，面向用户、面向科研、面向生产、面向管理、面向教学。读者对象为计算机用户、科技人员、院校师生及其爱好者。

主要栏目：研制·开发·应用、实用技术、网络与通信、软件纵横、编程技巧、经验与交流、善工与利器、智慧与趣味、初学者之友。

本刊1996年为双月刊，大16开本，将扩大篇幅，增加栏目。每期实用文章12万字以上。双月20日出版。每期定价3.80元，全年价22.80元。特别欢迎师生集体订阅。优惠条件请看订单。

出版发行：现代计算机杂志社

联系地址：广州市中山大学 刊号：CN44-1415/TP

邮政编码：510275

电话：(020) 4186300-6540

开户银行：广州市工商行新港西路分理处

账号：中山大学 003-0264-0004878

《现代计算机》订户调查问卷

答卷人情况：

①姓名：

②性别：

③年龄：

④专业：

⑤邮编：

⑥电话：

⑦传真：

⑧职业：

☐ 教师

☐ 学生

☐ 研究人员

☐ 工程技术人员

☐ 政府职员

☐ 管理人员

☐ 公司经理、厂长

☐ 其他

⑨单位的性质：

☐ 高等院校

☐ 中专中技

☐ 普通中学

☐ 国家机关

☐ 事业单位

☐ 国有企业

☐ 三资企业

☐ 全民研究所

☐ 民办研究所

☐ 电脑公司

☐ 其他

⑩单位的行业：

☐ 计算机

☐ 通信

☐ 电子电器

☐ 科技

☐ 教育

☐ 设计部门

☐ 行政社团

☐ 工业

☐ 交通运输

☐ 商业

☐ 农林牧

☐ 地矿、测量、气象

☐ 工程、水利

☐ 军工

☐ 医疗卫生

☐ 金融财会

☐ 其他

⑪单位或个人使用的主要计算机类型：

☐ 286

☐ 386

☐ 486

☐ 586 (奔腾)

☐ 苹果

☐ 其他

⑫单位的计算机网络情况：

☐ 已有的网络类型

☐ 拟建网络类型

⑬您喜欢看本刊那些栏目：

☐ 研制·开发·应用

☐ 实用技术

☐ 网络与通信

☐ 软件纵横

☐ 编程技巧

☐ 经验与交流

☐ 善工与利器

☐ 智慧与趣味

☐ 初学者之友

⑭您还订阅那些计算机报刊：

☐ 中国计算机用户

☐ 微电脑世界

☐ 中国计算机报

☐ 计算机世界报

☐ _____

☐ _____

☐ _____

☐ _____

⑮您对本刊的建议：

主 办 中山大学
编辑出版 《现代计算机》杂志社
主 编 张纬铮
联系地址 广州新港西路 135 号中山大学
邮政编码 510275
电 话 4186300—6540
刊 号 CN44—1415/TP
邮发代号 46—121
广告许可 粤 010329 号
广告总代理 广东省广告公司
直通电话:7752254 传真:7778225
印 刷 中山大学印刷厂
总 发 行 广东省报刊发行局
订 阅 处 全国各地邮局
每 期 订 价 2.00 元 零售价 3.00 元

公开发行 1995 年 9 月 20 日出版

本刊图文版权所有
未经允许不得转载

注重实用性、知识性、趣味性

面向科研、面向生产、面向管理、面向用户、面向教学

明年本刊自办发行

- 不用到邮局去查找邮发代号
- 欢迎新订户来函索取明年的订单
- 老订户的订单将随刊奉送

目 录

☆ 研制·开发·应用 ☆

通用数字图象处理软件在 PC 机上的研制
..... 叶 屹 蒋大宗②
BP 算法的模拟程序 朱诗兵 李迎春⑥

☆ 实用技术 ☆

绘图软件中动态仿真调色板的设计 齐元华⑨
FOXBASE 伪编译文件的还原 卓华宾⑪
Drafting 实现机械制图辅助标注 肖鹏东⑮

☆ 软件纵横 ☆

多媒体视频与音频同步的探讨 贾建平⑮
高分辨屏幕图形的存贮和恢复方法示例
..... 樊启柏⑱

☆ 网络与通信 ☆

NetWare 386 内存管理优化方法
..... 周 斌 翟 坦⑳
PC 机与 8031 的并行数据通讯及自校正
标识符算法 张 乐 孙 华㉑
如何检测串行口 高铁红㉓

☆ 经验与交流 ☆

利用 CCED 进行彩色打印的尝试
..... 代永壮 李继红㉔
微机磁盘 CMOS 设置出错常见故障
排除 5 例 花 成㉖
DOS 命令巧用好处大 ... 贾建平 于冬梅㉗
汉化 AutoCAD 10.0 版配用 Q387 的
经验 刘 兵㉙
口令动态输入的实现 贾建平 陈剑萍㉚
整理软磁盘经验谈 王杰民㉛

☆ 编程技巧 ☆

含原字符的阴影处理技巧 李红胜㉜
用@〈行 1,列 1〉CLEAR TO 〈行 2,列 2〉
语句产生动画效果 闰 锦㉞
“参数库”菜单实现方法及应用
..... 黄焕如 王 玲㉟
长度受限情况下包含汉字字符串的
打印方法 金 海 谢 卫㊱

☆ 动态与信息 ☆

国家信息中心隆重推出《中国计算机应用进展专辑》⑤
全球智能芯片市场前景⑤ 本刊明年改双月刊、自办
发行㉒ 广州年底将举办华南电子盛会③① 欢迎订阅
《电子产品世界》③⑥ 欢迎订阅《适用技术市场》月刊③③
欢迎订阅《市场经济时报》④⑧

通用数字图象处理软件在PC机上的研制

西安交通大学生物医学工程研究所[710049] 叶 屹 蒋大宗

内容摘要:我们在PC机上用C语言实现了一个通用数字图象处理软件,该软件集成了数字图象处理领域中的几十个常用算法,可对 $512 \times 512 \times 8\text{bits}$ 的灰阶图象进行单一或复合处理及作伪彩色显示。该软件采用中文菜单工作方式,界面简洁,操作简单,易学易用。该软件的实现为在微机上普及数字图象处理技术提供了一个实用工具。

关键词:数字图象 软件 通用

一、前言

图像处理技术的应用已越来越广泛,但是数字图象处理软件受硬件的限制和影响,过去为满足图象显示的需要,常常采用专用的图象处理和显示设

备,而专用的设备价格昂贵,这不仅增加了图象处理用户的负担,而且数字图象处理软件的编制要针对此类专用图象处理设备硬件的约定进行开发,通用性较差。微型计算机显示技术的进步为实现价格低廉、通用的数字图象处理软件提供了硬件支持。我们分析了微型机上常用的TVGA8900C显示卡的结构和功能,直接对其显示缓冲区进行操作。用C语言实现了一个基于该卡的通用数字图象处理软件,该软件集成了数字图象处理领域中的几十个常用算法,可对 $512 \times 512 \times 8\text{bits}$ 的灰阶图象进行单一或复合处理以及作伪彩色显示。该软件采用中文菜单工作方式,界面简洁,操作简单,易学易用。通过使用表明,该软件具有良好的适用性,不仅能够满足处理生物医学图象、工业图象、遥感图象的需要,而且还可作为教学演示软件使用。该软件的实现为微机上普及数字图象处理技术提供了一个实用工具。本文将介绍该软件的实现及其功能。

二、TVGA 显示卡的硬件支持

TVGA卡是trident公司生产的PC机显示卡,可在 1024×786 , 800×600 , 640×480 和 300×200 等分辨率下显示256种的色彩,在该卡中有一个可将数字颜色信号转换为模拟信号的数模转换功能块DAC(见图一),DAC中有一组256个颜色寄存器,每个寄存器可根据需要赋值,由于在一个颜色寄存器中R、G、B成份各占6位,因而最多可配制 $2^6 = 262144$ 种色彩。而DAC中只有256个颜色寄存器,显然显示屏上一次最多可出现256种色彩。修改这些颜色寄存器就可得到我

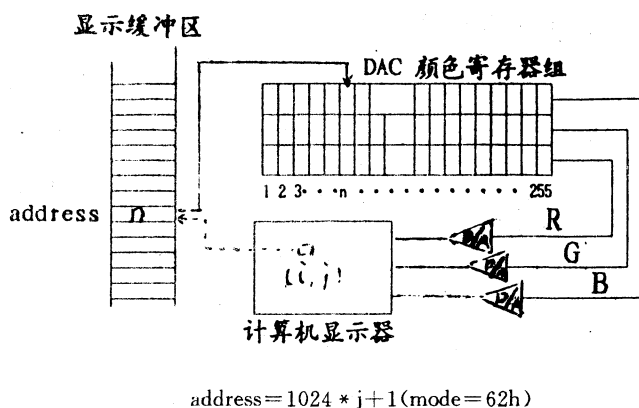
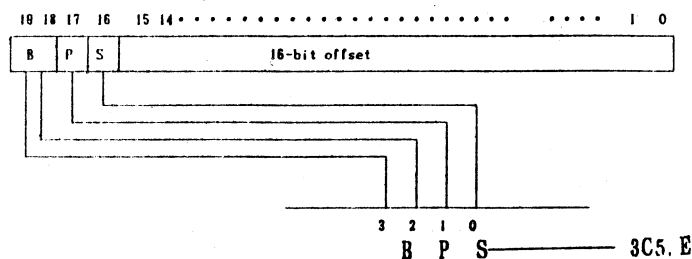


图 1



address 16-bit CPU segment

S Segment: 64K each P Page: 128K each

B Bank: 256K each Bank: 512K each

图 2 64K 模式寻址示意图

们所需的色彩。此外,从图一也可以看出显示缓冲区的单元值决定被显示的色号,而该色号的视觉特征是由颜色寄存器组的内容唯一确定的。

TVGA8900卡中有1M显示内存,在显示256种颜色模式下,显示屏上一个像素占用其内存中一个字节,在64K模式下,1M内存分成16个64K段映射到其基地址A000:0000上,故用户在访问某一内存单元时,先得执行一次选块操作,确定一个64KVRAM段。VRAM的地址寻址方式见图二,根据该图可编制读写像素的函数。

三、基于TVGA卡的底层C语言函数设计

Turbo C的标准库函数是不支持TVGA/256色显示模式的,为开发TVGA卡的显示功能,我们编制了数字图象软件所必需的底层C函数。

1. 要利用TVGA卡的256色显示模式,必须先进入该模式。我们用BIOS中断调用如下实现进入模式的函数。

```
void SetVideoMode(int mode)
{ union REGS regs;
  regs.h.ax = mode;
  /* mode=62h 1024×768; mode = 5eh 800×600 */
  int86(0x10,&regs,&regs);
  /* mode=5dh 640×480 */
}
```

2. 在图象处理软件中,读写像素是基本操作,根据图二很容易地实现写像素的函数(读像素的函数可类似实现);

```
void myvga256_putpixel(int i, int j, int color)
{
  unsigned long k;
  unsigned int block;
  unsigned char far * addr;
  k = (long) j * 1024 + i;
  block = ((k >> 16) & 0xf) ^ 2;
  outportb(0x3c4, 0x0e);
  outportb(0x3c5, block);
  addr = MK_FP(0xA000, (k & 0xffff));
  *addr = (color & 0x00ff);
}
```

3. 为使显示屏满足显示要求,必须对DAC颜色寄存器进行修改。下面用BIOS调用实现的函数就可满足要求;

```
char pal[256][3];
void MySetDAC(char pal[][3])
{ struct REGPACK rl;
  rl.r_ax = 0x1012;
  rl.r_bx = 0;
  rl.r_cx = 256;
  rl.r_es = FP_SEG(pal);
```

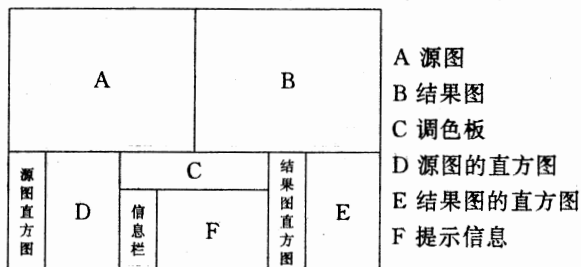
```
  rl.r_dx = FP_OFF(pal);
  intr(0x10,&rl);
}
```

当然,也可通过读写DAC寄存器端口来修改颜色寄存器组。

四、计算机图象显示的设计

该通用数字图象处理软件是针对512×512×8bits的灰阶图象进行处理的,图象数据在外存中是以行为记录逐行存储,呈栅格结构,其中每一个字节对应一个像素的灰度值,故有256个灰阶。由于DAC组的每一个颜色寄存器都是18位,其中RGB成分各6位,由色彩合成原理,我们给R、G、B赋以相同的值就可得到深浅不同的灰阶,灰阶个数最多有 $2^6=64$ 个;而灰阶图象有256个灰阶,为解决二者不相同的问题,我们依次将一个灰阶依次对应于4个颜色寄存器,这样不用改变灰阶图象的灰度值,又可达到显示的要求。虽然,屏幕上显示的图象只有64个灰阶,但由于人眼的灰阶分辨力只有几十个,对我们的图象处理软件而言,显示是满足要求的。

TVGA卡有多种分辨力的256色显示模式,在处理软件中,我们采用1024×768×256显示模式,基于简洁实用的目的,屏幕如下图分割:



这种安排可使用户在对源图进行处理的同时能够对比观察分析源图、结果图对应部分,从D、E两个直方图可以看出图象在处理前后直方图的变化,信息栏以24×24点阵汉字的形式给出操作提示及其他有关信息,便于使用。

五、软件体系结构及功能简介

图象处理是处理图象及图象信息的一门科学,也可以认为是一种揭示图象数据抽象性的专门的二维信号处理技术。数字图象处理的一个很重要的目的就是采用一系列数字处理技术去改善图象的视觉效果或将图象转换成一种更适合于人或机器处理的形式。

数字图象处理的方法基本上可分为频率域处理和空间域处理两大类,前者是在付里叶变换域上进行修改处理,分析处理后感兴趣的频率分量,然后将修改后的付里叶变换值作付里叶逆变换,得到

经过处理后的图象。由于频率域处理要求较长的时间和较大的运算空间,这要求计算机有较高的配置,我们这里不实现。后者是在原图上直接进行数据运算。它主要包括四种处理方式:

点处理 图象中转换像素值仅依赖于原始像素值或者也可能与它在源图象位置有关。

区处理 图象中转换像素值依赖于原始像素值和它周围像素值。

帧处理 图象中转换像素值仅依赖于一个或多个附加图象中的像素值。

几何处理 通过一些几何变换来改变一个图象中像素的值和位置。

我们实现的这个软件主要是对空间域进行处理,这样不需要PC机有较高的配置,而且每进行一个图象处理所需时间不会太长。

为了用户使用方便,该软件采用文本模式和图象模式相结合的方式。在文本模式中采用下拉式中文菜单,用户用方向键移动亮条进行选择,回车键确认,ESC放弃当前操作或回到上一级菜单。此外,还可用热键激活对应功能。一旦确认做某件工

作,界面将切入 1024×768 模式进行图象处理,处理完毕后,可再回到文本模式进行新的选择。该软件体系结构如图三。

下面简要介绍图中各部分的功能。

图象显示与I/O处理支持功能代码主要完成与TVGA卡有关的操作(也就是前文中的基于TVGA卡的C语言函数)以及下述(1)和(10)的功能。图象处理支持函数代码对内存中的图象数据进行读和写。图象处理算法代码主要完成下列(2)到(9)的功能。

(1)文件操作模块 完成数据文件输入和结果数据存盘的功能,还可选择输入调色板文件以输入用户自定义的调色板。

(2)灰阶处理模块 完成对灰阶图象进行直方图均衡、线性拉伸、对数变换、锯齿波变换、灰阶图象层析、灰阶等高线、灰阶直方图规定化、马赛克处理等处理工作。

(3)噪声消除模块 由于图象在获取和传输过程中会引入噪声,该模块提供有 $N \times N$ 均值滤波、均值容差滤波以及 $N \times N$ 十字形、水平、垂直方式的非线性滤波(中值、膨胀、腐蚀)。

(4)边缘提取模块 边缘提取的目的就是突出图象的边缘,除边缘以外的图象中其他部分经过这一处理后通常都削弱或完全去掉了,处理后的边缘的亮度与原图中的边缘变化率成正比,这一模块实现有常用的LAPLACE、ROBERTS、PREWITT、SOBEL-ROBINSON、KIRSCH、Frei & Chen等算法。

(5)图象二值化模块 二值化是通过设定阈值把灰度图象变换为仅用二个值表示的图象,该模块给出了下列二值化方式:阈值法、双固定阈值法、P

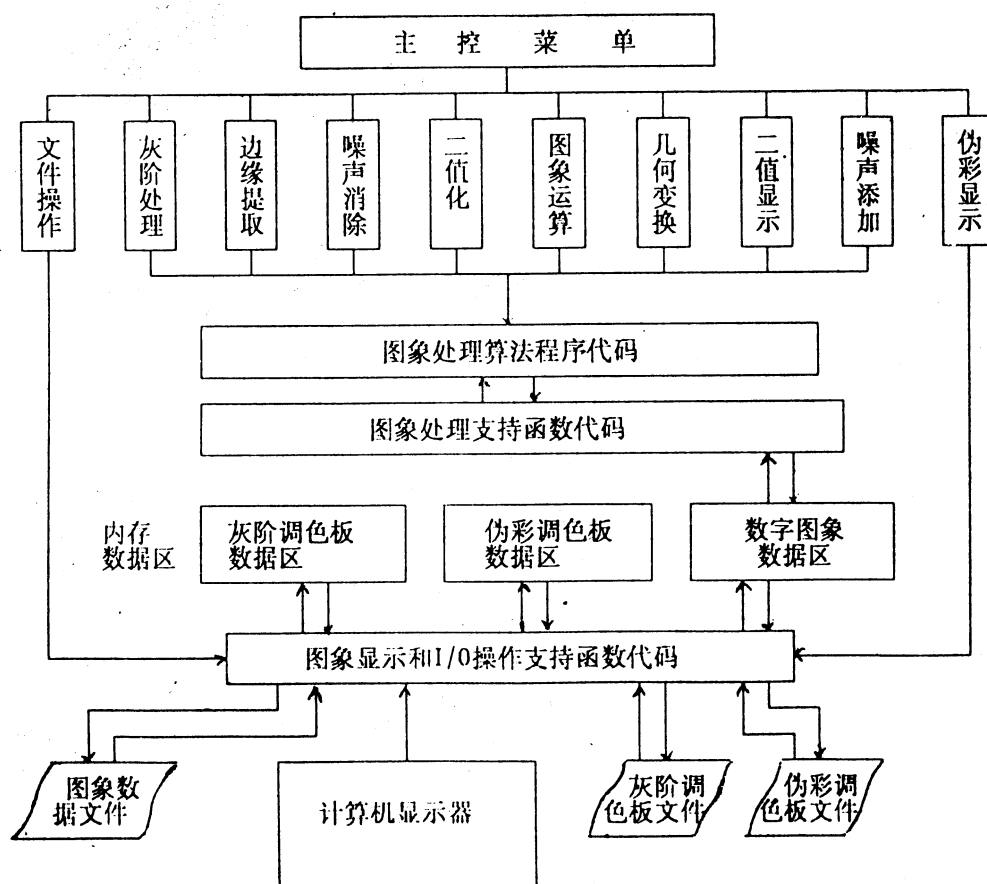


图 3

参数法、微分直方图法、灰度差直方图法。

(6) 图象运算模块可对二个图象进行加、减、乘、除算术运算和与、或、非、异或等逻辑运算。

(7) 灰阶图二值显示(dither, 有些文献译为抖动)模块模拟在单色显示器上用二值的方法显示灰阶图象。该模块有7种二值显示的方法及模式可供选择使用。

(8) 几何变换模块可对源图进行放大、缩小、旋转、镜像及镶嵌处理,变换中可以根据需要选择线性插值法或最小邻域法。

(9) 噪声添加模块 噪声是随处都有的,我们用程序来模拟产生出均匀、尖峰、高斯三种噪声图象,并可将这三种噪声叠加到别的图象中去。

(10) 伪彩显示 伪彩色是用彩色代替像素灰度值的一种技术,由于人眼对彩色的分辨力远远高于对灰阶的分辨力,所以这种技术可以用于识别较小的灰度范围的像素,该模块提供有五种伪彩方案可供选择使用。

从该软件的体系结构图可以看出,每进行一次处理后都将结果存入数据区中而冲去原有的图象数据,这使用户可以进行多个功能的复合操作。例如可对某图先进行灰阶均衡再进行 LAPLACE 边缘检测和二值化。

此外,从软件体系结构图还可看出,灰阶调色板和彩色调色板信息都是用外部文件的形式存储的,用户可以编写满足自己要求的灰阶调色板和彩色调色板文件来替换他们。

六、结束语

我们用近6千行C语言程序完成了这个软件。使用表明,本软件有下列优点:

(1) 集成度高,功能较全,该软件集成了空间域处理中的几十个算法,可在其中选择作单一或复合处理。

(2) 中文界面简洁,下拉式菜单式和热键响应相结合的工作方式,使用简便,非计算机专业人员,易学易用。

(3) 开放性好,图象格式采用普通的栅格文件格式,用户使用限制少;此外,用户还可以编写自己的灰阶和伪彩方案进行显示。

(4) 价格低廉,通用性好,不用专用的图象显示设备,直接在PC机上实现。该软件有130K可执行文件,可运行在DOS3.0以上及支持直接写屏的汉字系统(如四通汉字系统)环境中,TVGA卡要求有1M显示缓冲区。

总之,该软件的实现为在微机上普及数字图象处理技术提供了一个实用工具。(需要该软件者请与我们联系)

参考文献

- [1]《EGA VGA TVGA 高级微机图形编程指南与实例(下册)》,北京希望电脑公司,91年12月
- [2]王绍霖著,《数字图象处理》,国防科技大学出版社,1987年10月
- [3]R.C. 冈萨雷斯 P. 温茨著 李叔梁译,《数字图象处理》,科学出版社,1983年4月 (收稿日期:1995.4.3)

· 简 讯 ·

国家信息中心隆重推出 《中国计算机应用进展专辑》

为推进中国计算机应用进程,国家信息中心隆重推出《中国计算机应用进展专辑》。《专辑》汇集了50余家国内外著名计算机公司客户的应用实例和30多个行业的应用典型以及中外著名计算机专家、企业界名人如 Bill Gates 等向计算机用户的赠言,共计40余万字,邹家华副总理为《专辑》题写了书名。

《专辑》可以帮助计算机用户更好地认识各计算机公司和计算机品牌的优势,从而选择最佳合作伙伴和产品;可以帮助计算机公司找到商业机会和工作方向,以制定发展战略;可以使人们更好地认识计算机的作用。该书售价30元/册,邮挂费3元。需要者请与以下地址联系:

联系单位:国家信息中心《经济与信息》杂志社

联系地址:北京市西城区三里河路58号

联系电话:(010) 8526577 8502528

邮政编码:100045

开户行:中国银行总行营业部

帐户:《经济与信息》编辑部

帐号:5170100042

· 商 讯 ·

全球智能芯片市场前景

据美国加州专门从事计算机和信息的市场情报研究公司的研究表明,神经网络将是20世纪90年代最热门的技术,其年销售收入将从1991年的1.46亿美元激增到1998年2265亿美元。

该研究还包括了与神经网络不可分割的相关编程技术——模糊逻辑。

据称,神经网络的年销售收入将逐年上升,到1998年,其收入增长幅度将达到46%。此间,模糊逻辑市场将猛涨,其综合年增长率达76%。1998年,这两项技术的总收入将在100亿美元左右。

工业应用的增加刺激这两项技术的推广。对神经网络而言,虽然国防应用减少,而金融、医药应用将扩大。对模糊逻辑,工业应用和汽车应用将取代消费应用而成为最大的市场领域。

日本在模糊逻辑方面领先美国,并已用于多种消费电子领域。而美国最近掀起的模糊逻辑冲击波可望赶超日本。相反,美国在神经网络领域领先日本,而日本正在设法赶超美国。此外,欧洲也毫不示弱,飞利浦、西门子、汤姆逊也纷纷加码,力争后来居上。

(德森供稿)

BP 算法的模拟程序

国防科工委指挥技术学院 朱诗兵 李迎春

摘 要: 本文根据 BP 算法的基本理论, 以二值异或问题为例给出了 BP 算法的流程图, 同时编写了该问题的运行程序。该程序只稍做改动就可适应各种需要。

关键词: BP 网络, 模拟程序, 流程图

一、BP 算法

BP 算法以及基于该算法的多层神经网络模型是目前神经网络研究的重点之一, 并且已被大量地应用于模式识别、机器人控制等方面。BP 算法所采用的学习过程由正向传播处理和反向传播处理两部分组成。在正向传播过程中, 输入模式从输入层经隐含层逐层处理并传向输出层, 每一层神经元状态只影响下一层神经元状态; 如果在输出层得不到期望的输出, 则转入反向传播, 此时, 误差信号从输出层向输入层传播并沿途调整各层间连接权值以及各层神经元的偏置值, 以使误差信号不断减少。BP 算法实际上是求误差函数的极小值, 它通过多个学习样本的反复训练并采用最快下降法, 使得权值沿误差函数的负梯度方向改变, 并收敛于最小点。

BP 网络的作用函数采用 S 型函数 (Sigmoid): $f(s) = 1/[1 + \exp(-s)]$, 而误差函数为: $E(W) = (1/2) \sum (T_j^k - y_j^m)^2 = (1/2) \sum \epsilon_k$, 其中 T_j^k, y_j^m 分别为输入模式 j 下输出单元的期望输出值和实际输出值。设 BP 网络具有 m 层, 即 $1, 2, \dots, m$ 。令 y_j^m 表示第 m 层中第 j 个结点的输出, 而 y_j^0 (零层输出) 就等于 x_j , 即第 j 个输入。这里上标 m 不代表图形序号而是层号。令 W_{ij}^m 表示从 y_i^{m-1} 到 y_j^m 间的连接加权, BP 算法描述如下:

1、BP 网络状态初始化, 将各加权值和偏置值 (θ_j^m) 随机置到小的随机数, 可用均匀分布的随机数。这样可保证 BP 网络不被大的加权值所饱和。

2、从训练数据组中选一数据对 (x^k, T^k) , 将输入向量加到输入层 ($m=0$), 使得 $y_i^0 = x_i^k$ (对所有 i 端点), 式中上标 k 是图形号。

3、信号经过 BP 网络的前向传播, 即利用关系式: $y_j^m = F(s_j^m) = F(\sum W_{ij}^m y_i^{m-1} + \theta_j^m)$

计算从第一层开始的各层内每个结点 j 的输出 y_j^m , 一直到输出层的每个结点的输出 y_j^m 计算完为止。

4、计算输出层每个结点的误差值

$\delta_j^m = F'(s_j^m) [T_j^k - y_j^m] = y_j^m (1 - y_j^m) [T_j^k - y_j^m]$
(对 s 型压缩函数)

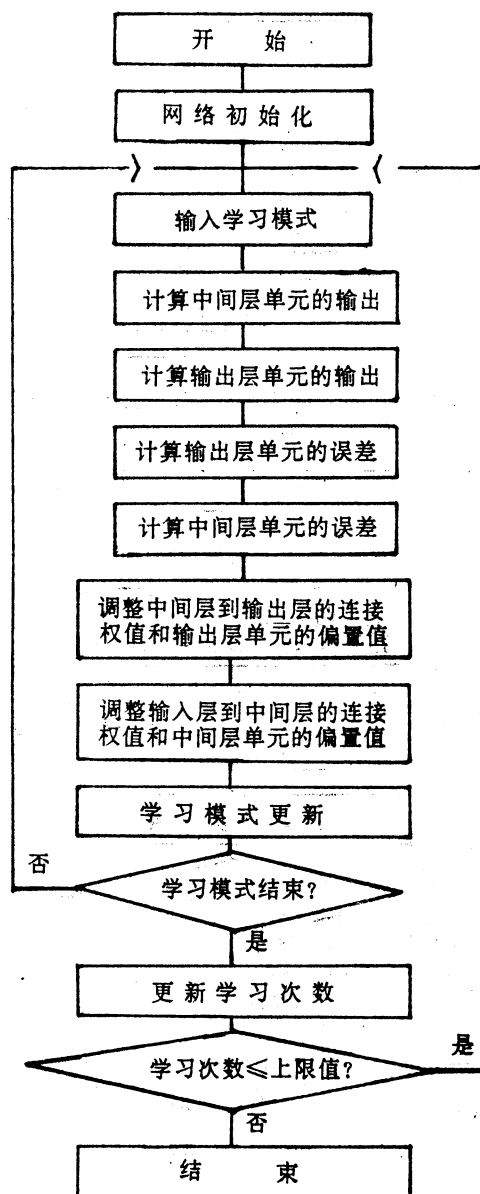


图 1

这个误差由实际输出和要求目标值之差获得。

5、计算前面各层每个结点的误差值

$$\delta_j^{m-1} = F'(s_j^{m-1}) \sum W_{ij}^m \delta_i^m$$

这靠逐层反传误差算得,其中 $m=m, m-1, \dots, 1$ 直至将每层内每个结点的误差值算得为止。

6、利用加权修正量公式

$\Delta W_{ij}^m = \eta * \delta_j^m * y_i^{m-1}$ 和 $W_{ij}^{new} = W_{ij}^{old} + \Delta W_{ij}$ 修正所有的连接。这里的 η 称为训练系数,一般取 $\eta = 0.01 \sim 1$ 的常数。

7、返回到步 2,为下一输入图形重复步 2 至步 7。

二、程序流程图及清单

下面以典型的二值异或问题为例来说明上述 BP 算法过程。采用三层网络来实现二值异或运算,其中输入层有两个神经元、隐含层有两个神经元、输出层有一个神经元。其流程图如图 1。

根据上述流程图,编写了 BP 算法解决二值异或问题的程序,其清单如下:

```
#include "stdio.h"
#include "dos.h"
#include "stdlib.h"
#include "math.h"
#include "conio.h"
#define IN 2          /* 输入层结点数 */
#define HN 2          /* 中间层结点数 */
#define ON 1          /* 输出层结点数 */
#define SPE 4         /* 训练样本总数 */
#define SIG0 0.00001
#define SIG1 0.99999
double u0=0.5;
double sigmf(double u) /* S 型函数 */
{
    double su;
    su=u/u0;
    su=0.5*(1.0+tanh(su));
    if(su>SIG1)return (SIG1);
    else if(su<SIG0)return (SIG0);
    else return (su);
}
double OT_IN[IN];      /* 输入层的输出 */
double OT_HD[HN];      /* 中间层的输出 */
double OT_ON[ON];      /* 输出层的输出 */
double W_IN_HD[HN][IN]; /* 输入层与中间层的连接权值 */
double CW_HD[HN];      /* 中间层的偏置值 */
double W_HD_OT[ON][HN]; /* 中间层与输出层的连接权值 */
double CW_OT[ON];      /* 输出层的偏置值 */
double DW_IN_HD[HN][IN];
double DCW_HD[HN];
double DW_HD_OT[ON][HN];
```

```
double DCW_OT[ON];
double OW_IN_HD[HN][IN];
double OCW_HD[HN];
double OW_HD_OT[ON][HN];
double OCW_OT[ON];
double TEACH[ON];      /* 输出层的期望值 */
double delta_OT[ON];   /* 输出层的误差 */
double delta_HD[HN];   /* 中间层的误差 */
double eta=0.3;         /* 训练系数 */
double beta=0.6;        /* 偏置值参数 */
double inival,ru0;
struct{
    int input[IN];
    int tch[ON];
} indata[SPE]={{1,0,1},{0,1,1},{0,0,0},{1,1,0}},);
main()
{
    FILE *fp;
    int k,m,times=3000;
    int loop=0,i,sig;
    double error,erlimit=0.01,wk,wkb,moment=0.4;
    double dmoment=0.03,wgt=0.2,intoff;
    init_wgt(W_IN_HD,IN*HN,wgt,0); /* 权值初始化 */
    init_wgt(W_HD_OT,HN*ON,wgt,0);
    init_wgt(CW_HD,HN,intoff,1); /* 偏置值初始化 */
    init_wgt(CW_OT,ON,intoff,1);
    ru0=2.0/u0;
    for(k=0;k<HN;k++){
        OCW_HD[k]=0.0;
        for(m=0;m<IN;m++)OW_IN_HD[k][m]=0.0;
    }
    for(k=0;k<ON;k++){
        OCW_OT[k]=0.0;
        for(m=0;m<HN;m++)OW_HD_OT[k][m]=0.0;
    }
    while(loop++<times){
        error=0.0;
        for(k=0;k<HN;k++){
            DCW_HD[k]=0.0;
            for(m=0;m<IN;m++)DW_IN_HD[k][m]=0.0;
        }
        for(k=0;k<ON;k++){
            DCW_OT[k]=0.0;
            for(m=0;m<HN;m++)DW_HD_OT[k][m]=0.0;
        }
        for(i=0;i<SPE;i++){
            for(k=0;k<IN;k++) /* 设置输入层的输出 */
                OT_IN[k]=(indata[i].input[k]) ? SIG1:SIG0;
```

```

for(k=0;k<HN;k++){
    /* 计算中间层的输出 */
    inival=0.0;
    for(m=0;m<IN;m++){inival+=(W_IN_HD[k]
[m]*OT_IN[m]);
    inival+=CW_HD[k]; /* 对中间层 K 的输入 */
    OT_HD[k]=sigmf(inival);
    } /* 中间层 K 的输出 */
    for(k=0;k<ON;k++){
        inival=0.0; /* 计算输出层的输出 */
        for(m=0;m<HN;m++){inival+=(W_HD_OT
[k][m]*OT_HD[m]);
        inival+=CW_OT[k];
        OT_ON[k]=sigmf(inival);
    }
    for(k=0;k<ON;k++){ /* 输出层的误差计算 */
        TEACH[k]=(indata[i].tch[k]? SIG1:SIG0;
        wk=OT_ON[k];
        wkb=TEACH[k]-wk;
        /* 输出层的期望值与实际输出值的差 */
        error+=(fabs(wkb));
         $\delta\_OT[k]=wkb*ru0*wk*(1.0-wk);$ 
    }
    for(k=0;k<HN;k++){ /* 中间层的误差计算 */
        inival=0.0;
        for(m=0;m<ON;m++){inival+=( $\delta\_OT[m]*$ 
W_HD_OT[m][k]);
        wk=OT_HD[k];
         $\delta\_HD[k]=inival*ru0*wk*(1.0-wk);$ 
    }
    for(k=0;k<ON;k++){
        /* 计算输出层与中间层的连接权值变化量 */
        for(m=0;m<HN;m++){DW_HD_OT[k][m]+=
( $\eta*\delta\_OT[k]*OT\_HD[m];$ 
        DCW_OT[k]+=(beta* $\delta\_OT[k];$ 
    }
    for(k=0;k<HN;k++){
        /* 计算中间层与输入层的连接权值变化量 */
        for(m=0;m<IN;m++){DW_IN_HD[k][m]+=
( $\eta*\delta\_HD[k]*OT\_IN[m];$ 
        DCW_HD[k]+=(beta* $\delta\_HD[k];$ 
    }
    }
    if(error<erlimit)break;
    for(k=0;k<HN;k++){
        /* 调整输入层与中间层的连接权值 */
        wk=moment*OCW_HD[k]+DCW_HD[k];
        CW_HD[k]+=wk;
        OCW_HD[k]=wk;
        for(m=0;m<IN;m++){
            wk=moment*OW_IN_HD[k][m]+DW_IN_HD
[k][m];
            W_IN_HD[k][m]+=wk;
            OW_IN_HD[k][m]=wk;
        }
    }

```

```

    }
    for(k=0;k<ON;k++){
        /* 调整中间层与输出层的连接权值 */
        wk=moment*OCW_OT[k]+DCW_OT[k];
        CW_OT[k]+=wk;
        OCW_OT[k]=wk;
        for(m=0;m<HN;m++){
            wk=moment*OW_HD_OT[k][m]+DW_HD_OT
[k][m];
            W_HD_OT[k][m]+=wk;
            OW_HD_OT[k][m]=wk;
        }
    }
    moment+=dmoment;
    if(moment>0.90)moment=0.90;
}
printf("η=%f\n",η);
printf("LOOP=%d\n",loop);
printf("ERROR=%f\n",error);
}
init_wgt(double w[],int s1,double inival,int flag)
{
    int i;
    double drand48();
    i=0;
    while(i<s1){
        *(w+i)=(flag)? inival*drand48():inival*
(drand48()-0.5)*2.0;
        i++;
    }
    double drand48()
    {
        double d;
        d=(double)rand()/32767.0;
        return(d);
    }
}

```

三、结束语

本文所提供的 BP 算法程序已在微机 SUN 386 上调试通过。另外该程序只须稍作变动便可将 BP 网络规模扩大以适应其它各种应用需要,同时也可作为研究 BP 算法的软件工具,为各种改进的 BP 算法提供计算依据。

参考文献:

- [1]王军政编,《Turbo C 2.0 实用高级编程技巧》,北京科海培训中心,1992 年 5 月
- [2]周继成等编著,《人工神经网络——第六代计算机的实现》,科学普及出版社,1993 年 1 月
- [3]庄镇泉、王煦法等编著,《神经网络与神经计算机》,科学出版社,1994 年 10 月

(本文收稿日期:1995.4.14)

绘图软件中动态仿真调色板的设计

山东工业大学数理系 齐元华

摘 要: 本文采用直接对颜色寄存器编程的方法,模拟真实调色板的调色,以实现计算机全彩色绘图。

关键词: 颜色寄存器,颜色地址寄存器,TVGA 显示卡,调色板

PC、PS/2 系列微机的显示卡自 VGA 开始 (PGA, EVGA, TVGA 等),增加了 256 个颜色寄存器,可同时在一个屏幕上显示 256 种颜色。基于 256 色的绘图软件,比 16 色同类软件有了更宽的色彩范围,可以更加逼真的模拟自然颜色,因而近期 256 色绘图软件得到了广泛的应用。但是有些 256 色绘图软件采用了固定的颜色寄存器,而用户在启动软件时往往不知道将要设计的画面中要用到哪种颜色,不能满足绘制特殊画面的需要。另外,用 256 色来表示自然界的所有颜色是不够细致的。采用动态仿真调色板的方法,让用户用红绿蓝三原色自己调制所需要的颜色,并将调好的颜色储存在颜色寄存器,按照某幅图画的要求设计特殊的颜色寄存器可以克服上述缺点。下面以 TVGA 5D 模式为例说明如何实现上述功能。

一、TVGA 的 256 色模式及相关寄存器

TVGA 8900 提供了 5c(640×400)、5D(640×480)、5E(800×600)、5F(1025×768)五种 256 色模式,在这些模式下,图象的每个象点用 8 位表示。每个象点的低 4 位用以对 16 个调色板寄存器编

址,调色板寄存器的输出和象素的高 4 位联合使用对颜色寄存器编址,颜色寄存器中有输出数字—模变换器(DAC)的颜色代码,DAC 的输出送往监视器。如图 1 所示。

颜色寄存器是 18 位寄存器,每个由三部分组成,分别代表红、绿、蓝三色的比例,可以是 0—262144 之间的数值,代表 262144 种颜色中的一个。256 个颜色寄存器的颜色值决定了一屏可同时显示的颜色种类及数目。

对颜色寄存器的访问可通过 BIOS 功能调用和直接对颜色寄存器编程的方法。

1. BIOS 功能调用

使用 BIOS 调用功能号 10H,子功能 12H 可成块设置颜色寄存器:

AH=10H, AL=12H, BX: 第一个颜色寄存器, CX: 设置的颜色寄存器数,将代表红、绿、蓝比例的数据写入某缓冲区,把缓冲区段地址和偏移地址分别放入 EX, DX 寄存器,执行功能调用可将缓冲区中的数据写入颜色寄存器。值得注意的是这种方法必须同模式设置一齐进行,否则导致显示模式错。

2. 直接对颜色寄存器编程

256 个颜色寄存器由颜色地址寄存器统一编

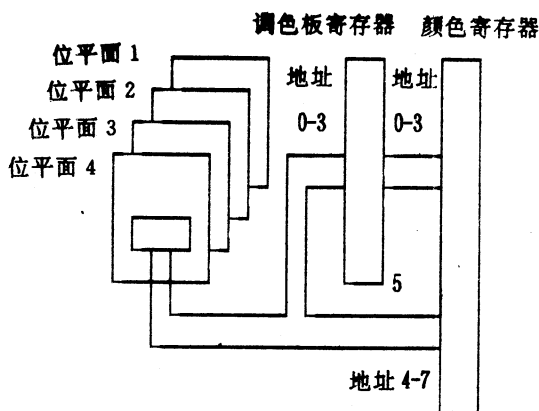


图 1

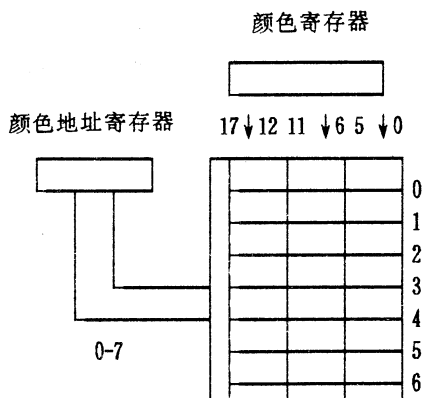


图 2

FOXBASE 伪编译文件的还原

兰州 508 信箱三分箱科研室[732850] 卓华宾

摘要:本文对 FOXBASE 命令文件执行部分及其调试命令机制的综合分析,以及剖析 FOXBASE 自身对加密 .FOX 文件还原算法,彻底将 .FOX 还原为源程序代码。

关键词:FOXBASE, 编译, 反编译, .FOX 文件, .PRG, 解密

一、引言

为了保护开发计算机 FOXBASE 应用软件,以及加快执行速度,2.00 版以上的 FOXBASE 中增加对源程序代码文件 .PRG 文件进行伪编译,生成 .FOX 代码文件;如在伪编译时带 -e, 则对 .FOX 文件进行多次变换,生成 .FOX 文件面目全非的加密文件,从中无法找到任何规律,要想解密,确实困难。设计 FOXBASE 者声称,从目标代码重新构造源程序几乎成为不可能。

而我们在工作中,经常需要对 .FOX 软件进行修改,不管是别人还是自己的,如果没有源程序则无法进行,再好软件只能是望之兴叹;要想进行修改,使 .FOX 文件能适应我们的需要,将 .FOX 文件反编译为源文件 .PRG 文件势在必行。

目前,在介绍 FOXBASE 反编译文章中,一般来说,比较彻底反编译方法有两种:一种是编一个专门的反编译程序,按照 FOXBASE 伪码文件格式及其所有的命令和函数内部代码对应生成源文件 .PRG 命令,但必须对这些内容进行大量跟踪分析,而这些内容均未公开,必须自行分析,其工作量可想而知;另一种方法是:利用 FOXBASE 所提供的调试命令,通过对它修改来进行反编译。本文就后一种方法对 FOXBASE 反编译的可能性进行研讨。

二、FOXBASE 的反编译功能及其命令的执行原理

1. FOXBASE 的反编译功能

FOXBASE 提供调度命令 SET ECHO ON 和 SET DEBUG ON 具有反编译的功能,只不过这两条命令非常脆弱,以至于在命令文件中有将它们关闭的语句时,它们的反编译功能也随之消失,即使命令文件没有关闭它,但反编译的结果只能按命令

执行过程中语句输出,而不是物理顺序进行反编译,使反编译结果非常杂乱,重复部分很多,不切实用。

2. FOXBASE 命令的执行原理

经过参考多方有关资料,了解到 FOXBASE 软件对命令文件执行部分程序代码如下:

```
-u4958 49eb
5EFE:4958 55      PUSH BP
5EFE:4959 8BEC     MOV BP,SP
5EFE:495B 81EC0202 SUB SP,0202
5EFE:495F 56      PUSH SI
5EFE:4960 833EEC0500 CMP WORD PTR [05EC],+
                    00 ;ECHO 标志
5EFE:4965 7470     JZ 49D7
5EFE:4967 833ED80000 CMP WORD PTR [00D8],+
                    00 ;DEBUG 标志
5EFE:496C 740B     JZ 4979
5EFE:496E 2BC0     SUB AX,AX
5EFE:4970 50      PUSH AX
5EFE:4971 9AEC020E2D CALL 2D0E:02EC
5EFE:4976 83C402   ADD SP,+02
5EFE:4979 833EFE0000 CMP WORD PTR [00FE],+
                    00
5EFE:497E 7405     JZ 4985
5EFE:4980 9A0A066E01 CALL 016E:060A
5EFE:4985 833ED60000 CMP WORD PTR [00D6],+
                    00
5EFE:498A 744B     JZ 49D7
5EFE:498C 833EE80000 CMP WORD PTR [00E8],+
                    00
5EFE:4991 7444     JZ 49D7
5EFE:4993 803EE10500 CMP BYTE PTR [05E1],00
                    ;加密标志
5EFE:4998 753D     JNZ 49D7 ;显示当前执行命令
5EFE:499A 8D8600FE LEA AX,[BP+FE00]
```

```

5EFE:499E 50      PUSH    AX
5EFE:499F 9AF8020E2D CALL   2D0E:02F8
5EFE:49A4 83C402   ADD SP,+02
5EFE:49A7 B8C025   MOV AX,25C0
5EFE:49AA 50      PUSH    AX
5EFE:49AB 8D8600FE LEA AX,[BP+FE00]
5EFE:49AF 50      PUSH    AX
5EFE:49B0 9A42040328 CALL 2803:0442
5EFE:49B5 83C404   ADD SP,+04
5EFE:49B5 83C404   ADD SP,+04
5EFE:49B8 8D8600FE LEA AX,[BP+FE00]
5EFE:49BC 50      PUSH    AX
5EFE:49BD 9A6C060328 CALL 2803:066C
5EFE:49C2 83C402   ADD SP,+02
5EFE:49C5 50      PUSH    AX
5EFE:49C6 8D8600FE LEA AX,[BP+FE00]
5EFE:49CA 50      PUSH    AX
5EFE:49CB B8FA33   MOV AX,33FA
5EFE:49CE 50      PUSH    AX
5EFE:49CF 9AB0009704 CALL 0497:00B0
5EFE:49D4 83C406   ADD SP,+06
5EFE:49D7 8B1EE605 MOV BX,[05E6]
                    ;取命令地址
5EFE:49DB FF06E605 INC WORD PTR [05E6]
5EFE:49DF 8A1F     MOV BL,[BX];取命令代码
5EFE:49E1 2AFF     SUB BH,BH
5EFE:49E3 D1E3     SHL BX,1;入口偏移
5EFE:49E5 D1E3     SHL BX,1
5EFE:49E7 FF9F0809 CALL FAR [BX+0908]
5EFE:49EB C706CC050000 MOV WORD PTR [05CC],
                    0000

```

可以看出:FOXBASE 在执行一条命令时,都要检测一下调试命令的状态,如果当前状态为 ON,则将当前要执行的命令输出到相应的设备上,然后再执行该命令;如当前状态为 OFF,则直接执行该命令,如此继续,直到结束。

三、FOXBASE 反编译方法

通过对 FOXBASE 命令文件执行部分程序分析,可以得到反编译方法:

1、将 FOXBASE 中判断调试命令状态的部分,使设置调试命令置为 ON,从而调试命令不再受执行文件影响,使要执行命令输出到相应设备。

2、截留所有命令,如果为结束命令,则予以放行,否则,一律以清屏命令替换,这就可以使反编译工作不再受命令文件中命令行的执行顺序影响,从而将伪编译代码按其物理顺序进行反编译。

四、修改 FOXBASE 软件,实行反编译及其影响

1、修改 FOXBASE 软件

掌握反编译方法后,下面就可以对 FOXBASE 软件进行修改,使 FOXBASE 软件成为自身的反编译软件,修改步骤如下:

```

C>COPY FOXPLUS.EXE FOX
C>DEBUG FOX
-A 4960
XXXX:4960 MOV AL,1
                MOV [5E0],AL
                MOV [D8],AL
                MOV [CC],AL
                XOR AL,AL
                MOV [5E1],AL
-A 49DF
XXXX:49DF CALL 4A75
                NOP
-A 4A73 JMP 4A84
                MOV BL,[BX]
                SUB BH,BH
                CMP BX,55
                JZ 4A81
                NOP
                NOP
                4A7E MOV BX,000E
                4A81 RET
-w
-q
C>COPY FOX FOXPLUS.EXE

```

2、修改后影响

修改结束,在 FOXBASE 状态下,执行.FOX 文件前,首先联通打印机,执行.FOX 文件后,会发现:执行结果是将命令语句从打印机打印出来,这证明上面修改是正确的,达到了反编译功能。但反编译结果输出到打印机上,这在实际应用中,对程序维护和修改很不方便,这是一个缺点;第二个缺点是:如果运行的.FOX 文件有多个子过程文件时,只能反编译其主程序,无法将其子程序反编译出来。

五、反编译结果改向输出及子过程文件反编译方法

1、反编译结果的改向:

利用 FOXBASE 提供打印机输出重定向命令,SET PRINT TO< 文件名>,很容易将本来向打印机输出内容转向所指定的磁盘文件名中,方法是:在运行.FOX 文件前,先执行 SET PRINT TO < 文件名> 即可。

2、对子过程进行反编译:

如果一个 FOX 文件有多个过程文件,那么它就有多个结束命令,执行这样的 FOX 文件后,首先要遇到第一个结束命令必定是主程序结束命令,而且执行是按物理顺序执行的,所以执行主程序后就退出,这也就是为什么只能反编译出主程序原因。如果我们知道 FOX 文件中包含的过程文件名,直接执行此过程文件也可以将其内容反编译出来,问题在于如何知道 FOX 文件包含哪些过程文件名。这就需要我们进一步了解 FOX 文件结构。

六、FOX 文件结构及过程文件名的获取

1. FOX 文件结构:

①FOX 文件无论是否加密都有一个 34 字节的文件头,对于加密的 FOX 文件,除去两个字节的标识符外,其余的都是经过加密后的代码,文件头中各部分所表示的意义见表:

.FOX 文件头结构

| 顺序号 | 字节数 | 用途 |
|-------|-----|---------------------------------|
| 1~2 | 2 | FOX 文件标识符,FB2A 表示加密,FB2B 未加密 |
| 3~4 | 2 | 内部控制用 |
| 5~6 | 2 | 过程文件中子程序个数 |
| 7~14 | 4 | 过程文件总长度 |
| 11~18 | 8 | 未用 |
| 19~34 | 8 | 随机密码钥匙 |

②在文件头后面的部分为文件体,它由多个块(即过程文件)组成(至少有一块),每一块又分为两部分:一部分为指令集,由 PRG 的命令变换而来;另一部分为变量区,存放前面指令中所用到的变量名称,每个变量占 16 字节。

每一块的头两个字节为指令集的长度,变量区头两个字节为变量个数。文件体后面的部分为过程文件名,非过程文件没有此部分。过程文件名是以它在过程文件中出现的先后次序存放,每个过程占 14 字节。

我们知道过程文件名存放位置后,编制一个程序是很容易获取的,但是如果 FOX 是加密的,文件名的所在地址面目全非,无法存取文件名,所以必须对加密 FOX 文件还原为非加密 FOX 状态,即要进行解密。

2. 加密 FOX 文件的解密:

解密算法首先利用 FOX 文件头中 16 字节的解密密钥码,进行一系列的运算(见流程图 1),得到一个译码基值,以此基值首先按流程图 2 算法产生 251 个字节的一级译码表,每产生一个译码值,基值就发生变化,变化后的值为产生下一个译码的

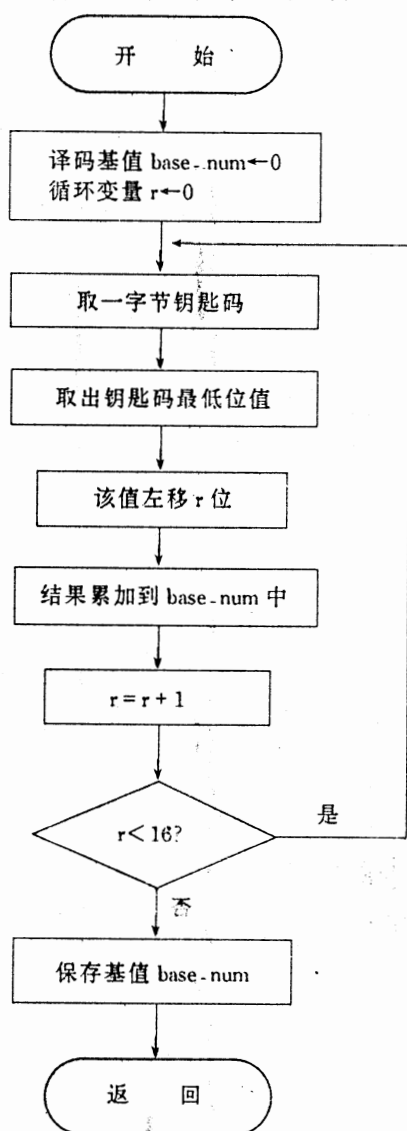


图 1 产生译码基值流程图

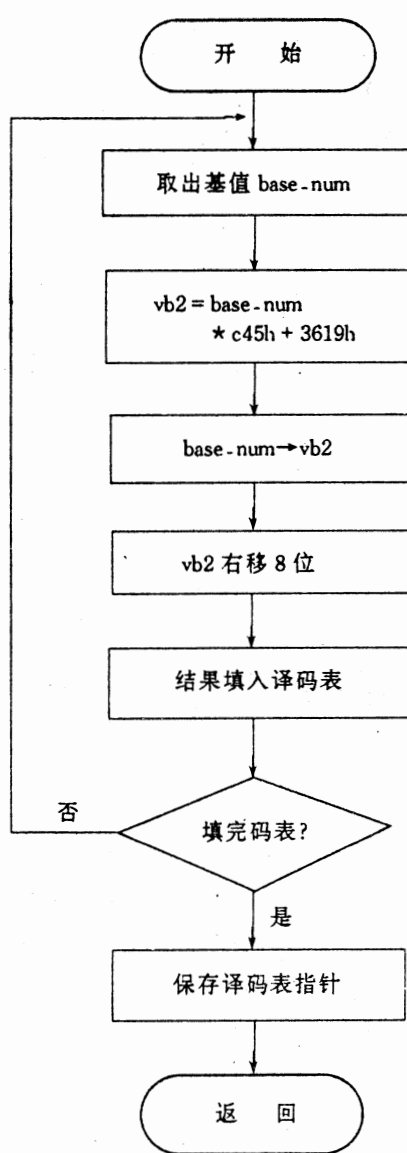


图 2 产生译码表流程图

基值。产生一级译码表后,再以一级译码表最后一个基值为二级译码表的第一个基值,再按流程图2产生257个字节的二级译码表。有了译码表后,便可以着手对加密.FOX文件码进行解密。首先用一级译码表对.FOX代码逐个异或,当译码表用完后,要重新从头使用,直到文件结束,然后同样再用二级译码表从文件开始进行第二次异或,直到最终形成的代码就是未加密代码。

3、过程文件名的获取:

从.FOX文件结构的文件头可知过程文件名的个数,每个过程文件名占用字节数,而且知道文件名放在文件的末尾,这就给我们正确获取过程文件提供途径。

七、反编译功能进一步完善及几点说明

1、由上可知,要对包含有过程文件的.FOX进行反编译,必须一个个运行过程文件,然而一个主文件可以包含过程文件多达128个,一个个运行,必定花大量时间,手续繁杂,易出错。笔者编写一个能自动执行所有过程文件的程序名为uf-1.asm。经编译后,生成UF-1.BIN文件,在经修改后的FOXBASE的点状态下,如下操作:

①LOAD UF-1 调入程序

②CALL UF-1 执行程序

根据提示输入.FOX文件,则自动生成主程序以及过程文件源代码。

2、几点说明:

经反编译而得到的命令文件与源文件.PRGR相比,功能是一致,但仍存在一些差异。

①源文件中注解部分无法得到,因为编译.

FOX文件,根本没有注解信息。

②反编译结果中命令和函数一律用大写字母为全称方式出现。

③如果原码文件中有不等于号“<>”,则反编译结果中用“#”表示,但对执行不影响。

④对使用到文件名,库名都用引号引起来,这也不影响执行。

⑤对设置颜色命令 set color to,如用到颜色,反编译结果用引号引起来,这些引号必须去掉否则影响反编译后所形成的.PRGR文件执行。

8. 结束语:

按上所述,笔者已成功开发了一套FOXBASE的反编译系统,经反编译的.FOX文件不管是加密还是未加密,都能反编译出来形成.PRGR文件,所有的.PRGR文件不经任何修改可直接运行;运行速度很快,笔者就一个包含有52个过程文件,10多万字节的主文件进行反编译,仅用4分钟;这与目前软件市场上出售的反编译系统相比,毫不逊色。笔者所修改前面所述的FOX-PLUS.EXE为单用户FOXBASE2.1系统中的启动程序,对于其它FOXBASE系统可能略有不同。由于UF-1.ASM源程序篇幅较大,在此从略,有需要者请与笔者联系。

参考文献:

[1]刘炳君,《FOXBASE伪编译文件的解密》,微计算机应用第2期,93年3月

(本文收稿日期:1995.6.19)

Drafting 实现机械制图辅助标注

(接下页)

次用到都重复绘制一遍,十分麻烦,效率不高。

可利用I-DEAS Drafting中所提供的symbol功能,实现这些常用符号的标注,步骤为:

(1)将需产生的组合符号做好;

(2)调用creat symbol功能,给符号取名,产生符号;

(3)用store命令存储所产生的符号到磁盘。

调用过程为:

(1)进入symbol状态;

(2)用命令creat instance,键入符号名;

(3)选比例、旋转角度,定位输出。

作者结合螺旋摆动器的设计过程,开发了常用辅助符号库,并以按用途分类、按使用频率分层的管理方式对图形符号库进行管理。如图2所示为所建图形符号库的部分例图。

四、小 结

本文基于I-DEAS Drafting实现汉字、常用符号标注的方法,是对I-DEAS Drafting进行二次开发的初步尝试,实践证明,对提高产品设计的计算机绘图效率是有效的。

参考文献:

[1]SDRC. I-DEAS Guide, 1992

(本文收稿日期:1995.4.4)

多媒体视频与音频同步的探讨

河南信阳空军一航院科研中心[464000] 贾建平

摘 要 本文根据多媒体对象数据的存放方式论述了纯软件实现视频与音频同步的方法。

关键词 多媒体 同步 音频 视频 播放 文件格式

一、引言

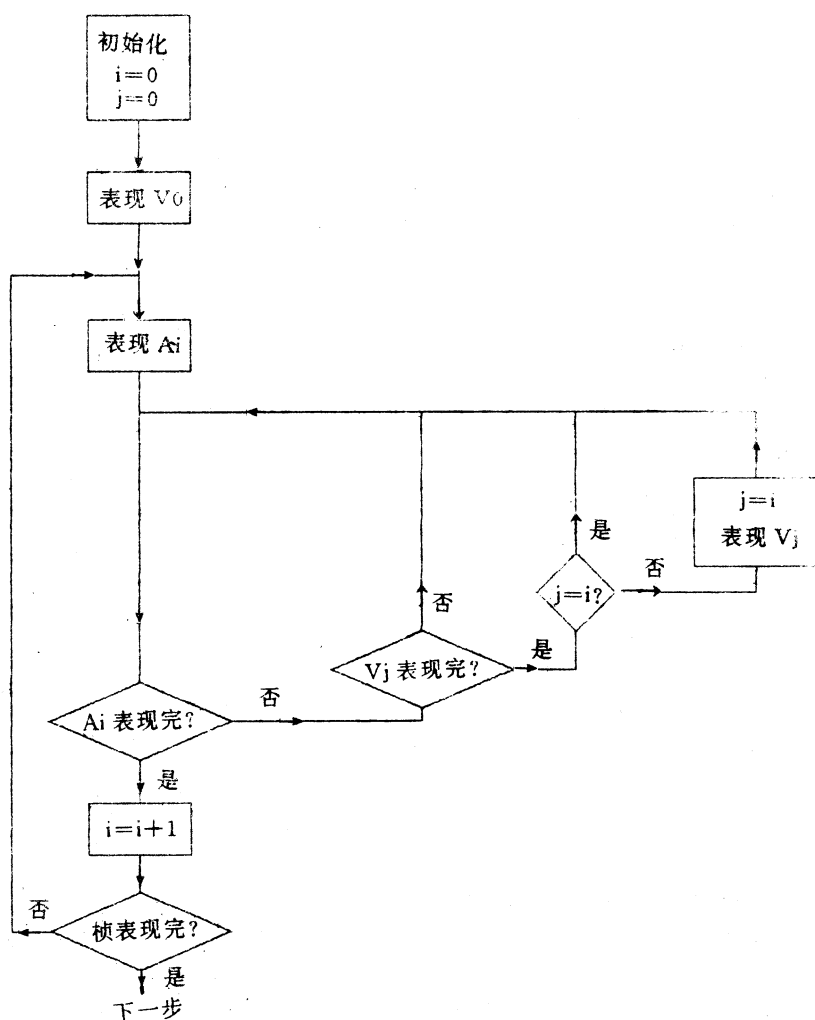
多媒体的主要表现对象是视频与音频数据。视频与音频对象在得以表现时,也就是动态地再现图像序列与声音时,应保持同步。比如:视频信息是正在讲课的教师,配音就必须与其口型符合。

由于磁盘读取速度、数据总线方式、计算机处理速度、显示缓冲写入速度、磁盘高速缓冲等的不同,可能导致视频信息显示的速度不同,也就是说,某种影响图象显示速度的因素变化就可能导致视频序列的播放速率与采集速率不一致。而模拟音频数字化时,是以固定频率采样形成数字波形音频的,并且音频的数据量较小(16位音频卡采样频率11025HZ 每秒采集的数据为22050字节),因此,固定频率的播放音频无论是以中断方式或后台DMA方式播放速度都是一致的。所以,播放环境不同于采集环境时,音频与视频就可能不同步。这种不同步现象在目前流行的PC多媒体开发平台上反映较为明显。这些多媒体开发平台开发的多媒体产品,如果使用环境不同于开发环境,动画演播与声音播放不同步,从而导致开发的多媒体产品只能是在特定环境下使用而难以推广。

我们知道,人类的听觉较视觉敏感,固定频率声音播放时,暂停及速率或高或低都使人难以接受,因此,多媒体对象中,音频对象的表现具有优先权,也就是说,视频及其它



图一



图二

媒体对象的表现时间要以音频的表现为准,视频及其它多媒体对象在表现时要同步于音频对象。

本文根据目前多媒体技术中媒体对象数据的分离或复合存储方式,讨论了媒体对象表现的同步问题。

二、音频与视频复合序列的同步

目前,有许多多媒体的数据文件格式(如MOV与AVI格式)采取了复合音频与视频数据的方式,如图一所示。各单媒体对象的数据被分割成固定时间间隔表现的离散子单元序列,而所有媒体对象同时表现的子单元复合在一起。这种情况下,每个子单元分隔点可作为一个参考点,也就是用于控制同步的同步点。同步控制的关键是:各媒体对象的表现同步于参考点。同步控制算法流程如图二。

①进行初始化,也就是打开相应文件、根据文件头及相关内容设置环境并把读写指针指向首帧。

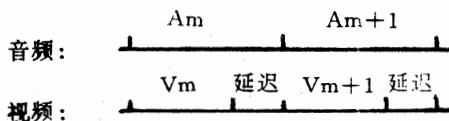
②然后依次表现各帧;在表现每帧时,如果音频帧表现完成,就表现下一音频帧,否则,判断当前的视频帧是否表现完成。

③如果没完成,则继续视频帧与音频帧的表现。

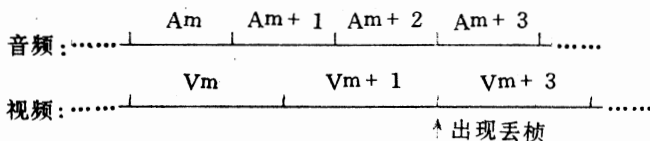
④如果当前视频帧表现完成,判断该视频帧是否当前音频帧的同步帧。

⑤如果不是,继续表现当前音频帧及与当前音频帧同步的视频帧(视频显示速度较慢时,该同步视频帧就可能非刚表现完视频帧相邻帧,也就是说出现丢帧);否则继续表现当前音频帧(视频帧表现快于音频帧表现,直到当前音频帧表现完成下一音频帧开始表现,下一视频帧才得以表现,也就是说视频帧表现出现延迟)。

从上述流程不难看出,视频数据播放快于音频数据时,视频播放就要等待(图三),而视频数据播放慢于音频数据时,视频播放就要采取丢帧的方法以便能够同步于音频数据的播放(图四)。



图三



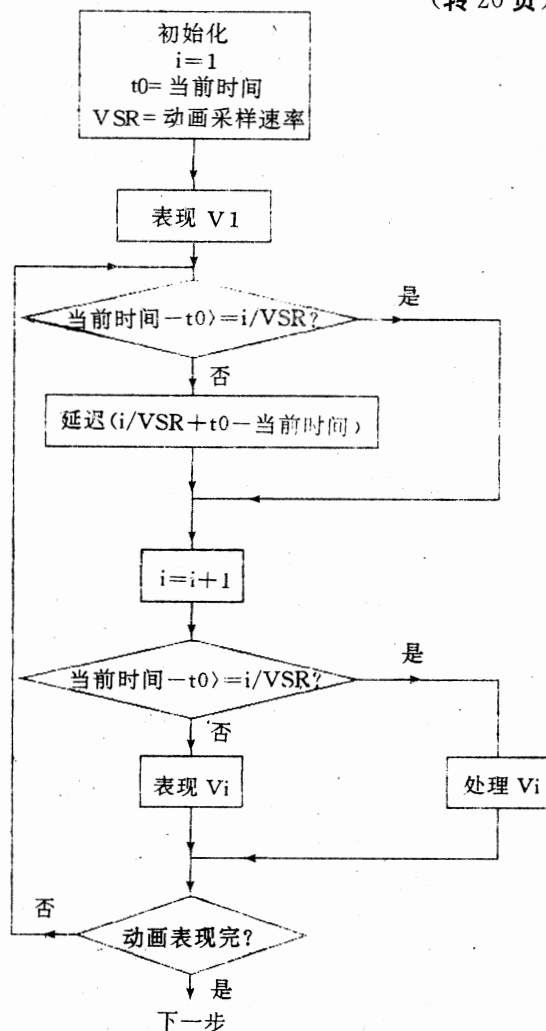
图四

这种媒体信息存储法对非正常播放(暂停、快进、快退等)较为方便,各媒体表现帧较易于定位,例如快进到第 m 帧,只需把读取指针简单的定位到第 m 帧信息所在位置即可。该种方法的明显缺点是:单一媒体对象的更换较为困难,各媒体对象数据的增删也不方便。

三、视频音频分离的同步

媒体对象的单独存放有利于各媒体对象数据的充分利用。但这又造成各同步表现媒体对象间除表现时间外没有共同的同步参考因素。因此,同步技术的关键便是各媒体对象表现的时间线,时间作为一个度量维度,单媒体对象与时间线联系在一起。同步算法维持一个公共的时间线,每个对象沿自己的局部时间线表示,当局部时间线与公用时间线的误差超出一定范围后,就采用丢帧或延迟的方式使局部时间线与公用时间线保持同步。

(转 20 页)



图五

高分辨屏幕图形的存贮和恢复方法示例

湖北省荆门供电局[448000] 樊启柏

摘 要:本文阐述了 VGA 及以上显示方式下屏幕图形存贮和恢复的多种方法的原理和特点,用 C 语言给出了实现上述多种存贮、恢复方法的实用程序。这些程序只需稍加改动可以很方便地移植到各种图形显示方式下进行屏幕图形的存贮和恢复。

关键词:VGA 显示方式 屏幕图形存贮和恢复 多种实现方法。

一、概述

随着计算机应用水平的不断提高,计算机图形在生产、生活实际中应用越来越广泛,特别自从 VGA 图形显示器推出以来,计算机图形显示的分辨率和颜色数目不断提高,产生的画面美观漂亮,给社会生产和人民生活带来了很多的方便,深受大家的喜欢。然而,由于图形分辨率和颜色数目的不断增大,也给图形的存贮和恢复带来了占用磁盘容量大,存贮、恢复速度慢等系列不利因素。怎样合理地解决这些问题呢?这是很多程序员比较关心的问题。事实上,很好地解决这个问题的绝对方法是难以找到的,但我们可以根据不同的实际应用要求,采用不同的屏幕存贮、恢复方法来达到理想的效果。本文就目前较通用的 VGA 介绍三种不同的屏幕图形存贮和恢复方法。这些方法各有优缺点,可供读者在实际应用中根据自己应用软件的实际要求去合理的选取或参考。

二、屏幕图形存贮、恢复方法

(一)屏幕像素扫描法

众所周知,屏幕上的图形是由许多像素组成的,其分辨率是由水平扫描线和垂直扫描线的数目决定的,如分辨率为 640×480 的屏幕图象,其每帧水平扫描线为 480,每条线有 640 个像素点,采用屏幕像素扫描法存贮和恢复屏幕图形就是按某种扫描规律依次读、写屏幕像素,将其数据写入磁盘文件(或从文件中读出)。该方法编程简单,通用性好,其缺点是存贮速度慢,占用磁盘空间大,适用于图象分辨率低、不需显示速度很快的场合。具体编程方法见程序清单(1)。

(二)屏幕图素代码描述法

为了提高屏幕图形的显示速度,减少图形文件的磁盘占用空间,我们可以将屏幕图形划分为是由很多基本图素组成的画面,每个图素用一个代码来表示(如圆用 01 代表、矩形用 02 代表),在存贮时

只需将其图素的代码、坐标位置、尺寸大小记录下来,而不必去存贮屏幕图形中的每个象素;在恢复时,依次将上述记录的图素数据使用作图函数显示出来,这样可以大大减少文件的存贮容量和提高存贮速度,但该方法编程较烦琐,通用性差,适用于实时性强,显示速度要求快的控制管理系统。具体编程实例见程序清单(2)。

(三)屏幕显示缓冲区直接读写法

屏幕显示缓冲区直接读写法是通过分析图形显示器的工作原理,利用计算机输入、输出指令直接对其控制寄存器进行操作,并对其显示缓冲区地址的数据进行直接读、写来实现屏幕图形的存贮与恢复。该方法存贮速度和通用性介于上述两种方法之间,使用较广泛,但编程复杂,图形存贮容量大,适用于专业绘图系统中。限于篇幅,现仅就 VGA 图形显示适配器为例,简单介绍其存贮原理(详细内容请参考有关专著)。VGA 图形显示适配器的显示页面由四个位平面组成,其起始地址均相同,由控制寄存器来控制其读、写,屏幕中每个象点的颜色是由四个位平面中各取一对对应位组成的。存贮屏幕图形时,可以通过读其地址寄存器的值,即从 A000:0000 的地址开始分别读出四个位平面寄存器的内容保存于磁盘文件中来实现屏幕图形的存贮;恢复时,可以通过改变其定序寄存器中彩色页面写允许值,从磁盘文件中依次将内容写入从 A000:0000 地址开始的四个位平面寄存器中,从而实现屏幕图形的恢复。具体编程实例详见程序清单(3)。

三、源程序清单及说明

程序清单(1):

```
#include "dos.h"
#include "bios.h"
#include "stdio.h"
union REGS r; FILE *fp;
```



```

/* -----在坐标(X,Y)处画点----- */
void pointxy(int x,int y,int cor)
{ r.h.ah=12; r.h.al=cor;
  r.x.dx=y; r.x.cx=x;
  int86(0x10,&r,&r); }
/* -----在坐标(X,Y)处写字符----- */
ccputs(int x,int y,char *str,int a)
{ while (*str) {
  _DX=(y-1)*0x100-1+x++;
  _AH=0x02;   BH=0x00;
  geninterrupt(0x10);
  _AX=0x0900+*str++;
  _BX=a;   CX=0x01;
  geninterrupt(0x10); } }

/* -----读坐标(X,Y)处像素----- */
int readxy(int x,int y)
{ r.h.ah=13; r.x.dx=y;
  r.x.cx=x; int86(0x10,&r,&r);
  return(r.h.al); }

/* -----存贮和恢复屏幕图形----- */
void loadsavetu(int dz)
{ register int i,j;
  char filem[13]; int buf[161][12],c;
  for(i=0;i<161;i++)
  for(j=0;j<12;j++){
    buf[i][j]=readxy(i+78,j+78);
    pointxy(i+78,j+78,0); }
  rectangle(78,78,238,90);
  ccputs(11,5,"File name:",15); /* 输入文件名 */
  gotoxy(10,5); scanf("%s",filem);
  if(dz==1)fp=fopen(filem,"wb"); /* 图形存储 */
  if(dz==0)fp=fopen(filem,"rb"); /* 图形恢复 */
  for(i=0;i<161;i++)
  for(j=0;j<12;j++){
    pointxy(i+78,j+78,buf[i][j]);
    if(fp==NULL)return;
    if(dz==0)clear();
    for(i=0;i<640;i++) /* 640*480分辨率 */
    for(j=0;j<480;j++){
      if(dz==1)fputc(readxy(i,j),fp);
      if(dz==0){ c=fgetc(fp);
        if(c!=0)pointxy(i,j,c); }
      } fclose(fp); }
}
/* ----- */

```

程序清单(2):

```

#include "graphics.h"
#include "alloc.h"
#include "stdio.h"
FILE *fp; int j1;

```

```

struct tu{ BYTE color,xs;
  int x1,x2,y1,y2; }gg[250];
/* ----- */
void drawtu(int hh) /* 作图函数 */
{ register i;
  switch(gg[hh].xs){
  case 1: i=abs(gg[hh].x2-gg[hh].x1);
    if(i==0)i=abs(gg[hh].y2-gg[hh].y1);
    circle(gg[hh].x1,gg[hh].y1,i);
    break; /* 画一个圆 */

  case 2: if(gg[hh].x2>gg[hh].x1&&gg[hh].y2>gg[hh].y1)
    rectangle(gg[hh].x2,gg[hh].y2,gg[hh].x1,gg[hh].y1);
    if(gg[hh].x1>gg[hh].x2&&gg[hh].y1>gg[hh].y2)
    rectangle(gg[hh].x1,gg[hh].y1,gg[hh].x2,gg[hh].y2);
    if(gg[hh].x2>gg[hh].x1&&gg[hh].y1>gg[hh].y2)
    rectangle(gg[hh].x2,gg[hh].y1,gg[hh].x1,gg[hh].y2);
    if(gg[hh].x1>gg[hh].x2&&gg[hh].y2>gg[hh].y1)
    rectangle(gg[hh].x1,gg[hh].y2,gg[hh].x2,gg[hh].y1);
    break; /* 画一个矩形 */

  case 3: setlinestyle(0,0,3);
    line(gg[hh].x1,gg[hh].y1,gg[hh].x2,gg[hh].y2);
    setlinestyle(0,0,1);
    break; /* 画一条粗线 */ } }

/* ----- */
void readwrite(int dz) /* 存贮和恢复图形函数 */
{ register hh; void *e;
  e=malloc(imagesize(0,0,240,72));
  getimage(400,385,638,450,e);
  putimage(400,385,e,XOR_PUT);
  setcolor(11); settextstyle(1,0,1);
  rectangle(400,385,638,450);
  if(dz==1)outtextxy(405,390,"please input filename (R):");
  if(dz==2)outtextxy(405,390,"please input filename (W):");
  gotoxy(51,25); scanf("%s",tum);
  if(dz==1)fp=fopen(tum,"rb");
  if(dz==2)fp=fopen(tum,"wb");
  if(fp==NULL){
    ccputs(55,28,"File not find",12);
    getch(); putimage(400,385,e,COPY_PUT);
    free(e); goto bh; }
  else{ putimage(400,385,e,COPY_PUT);
    free(e); }
  if(dz==1){ /* 从磁盘文件中读图形代码 */

```

```

fread(&jj1,sizeof(int),1,fp);
for(hh=0;hh<jj1;hh++)
fread(&gg[hh],sizeof(struct tu),1,fp);
setfillstyle(SOLID_FILL,0);
bar(1,1,638,449);
for(hh=0;hh<jj1;hh++)
setcolor(gg[hh].color);
drawtu(hh); /* 将图形代码恢复到屏幕上 */
if(dz==2){ /* 将图形代码写入磁盘文件中 */
fwrite(&jj1,sizeof(int),1,fp);
for(hh=0;hh<jj1;hh++)
fwrite(&gg[hh],sizeof(struct tu),1,fp); }
bh:{ }; fclose(fp); }

```

程序清单(3):

```

#include "graphics.h"
#include "dos.h"
#include "stdio.h"

int handle; char * buffer;
/* --- 存储显示缓冲区数据到磁盘文件中 --- */
void savedata(char * fname)
{ handle=_creat(fname,FA_ARCH);
buffer=(void * far)MK_FP(0xa000,0x0000);
outportb(0x3cf,4); outportb(0x3ce,3);
_write(handle,(void * far)buffer,2800);
outportb(0x34ce,2);
_write(handle,(void * far)buffer,2800);
outportb(0x34ce,1);
_write(handle,(void * far)buffer,2800);
outportb(0x34ce,0); close(handle); }
/* --- 将磁盘文件数据读入显示缓冲区 --- */

```

(接 17 页)

同步控制算法的核心是:对于动态媒体对象,假设其采样速率为 SR 秒/帧,播放速率为 R 秒/帧,当 $SR-R=0$ 时,播放速度适当,不用延迟也不用丢帧;当 $SR-R<0$ 时,播放速度慢,可能要丢帧;当 $SR-R>0$ 时,播放速度快,可能要延迟。下面以微机上 Autodesk 公司的 FLC/FLI 动画文件格式为例说明如何控制单媒体对象的局部时间线等同于公共时间线。

首先初始化,打开动画文件,并根据文件头设置相应环境,然后依次表现各动画帧。每表现完一帧画面后,都要根据下面计算式

$\Delta = \text{动画播放使用时间} \Delta t$

—动画制作速率 VSR(秒/帧) * 该帧序列号 i 的值判断播放速度快或慢:①如果 $\Delta=0$,播放速度适当,继续下一帧的播放;②如果 $\Delta<0$,播放速度快,要延迟 $-\Delta$ 后再播放下一帧;③如果 $\Delta>0$ 但

```

void loaddata(char * fname)
{ handle=_creat(fname,O_RDWR);
buffer=(void * far)MK_FP(0xa000,0x0000);
outportb(0x3c4,2); outportb(0x3c5,8);
_read(handle,(void * far)buffer,2800);
outportb(0x34c5,4);
_read(handle,(void * far)buffer,2800);
outportb(0x34c5,2);
_read(handle,(void * far)buffer,2800);
outportb(0x34c5,1); close(handle);
outportb(0x34c5,0x0f); }

```

四、结束语

目前,随着计算机图形的广泛应用,其屏幕图形的存贮、恢复方法不断增多,在此基础上,各种压缩存贮、恢复方法也在不断涌现出现,但其根本性的原则不会变,这就是事先必须建立某种存贮协议,恢复时也应按同样的协议解码,至于那种方法为最佳,要因实际要求而选定。本文在这里介绍的三种屏幕图形的存贮、恢复方法是比较基本的方法,仅供读者参考,其目的是抛砖引玉,如有不妥之处,敬请同行和专家们指正。

参考文献:

- [1]徐蔓等编译,《C 语言实用图像处理、获取、处理、存贮》,1992 年。
- [2]Herbwel Schildt 著,《如何用 C 语言开发高级应用程序》,1991 年。
- [3]尹彦芝编著,《C 语言高级实用教程》,清华大学出版,1992 年。(收稿日期:1995.4.18)

$\Delta < \text{VSR}$,播放速度慢,但慢出的时间不足以丢帧,继续播放下一帧;⑤如果 $\Delta \geq \text{VSR}$,丢弃随后 Δ / VSR 个帧。

对于非正常播放,同步控制算法必须依各媒体采样速度来决定帧的定位。如果数据的存储是采用像 FLI/FLC 这样的相关压缩方式,媒体播放的快进与快退更需要大量的计算,这些问题已超出本文范围不再论述。

四、结论

本文论述的多媒体对象的同步方法采取了丢帧与延迟的方式。然而,由于目前计算机视频显示速度的限制,信息量较大的视频对象(如分辨率为 $640 \times 480 \times 256$ 播放速度为每秒 25 帧)与声音同步播放时丢帧现象较为严重。因此,单纯的同步控制算法并不能彻底解决音频视频的同步问题,只有解决了影响视频显示速度的所有因素,多媒体技术才会产生更好的效果。(本文收稿日期:1995.6.30)

NetWare 386 内存管理优化方法

华中理工大学自动控制工程系 周 斌 瞿 坦

摘 要:本文在分析 NetWare 386 内存管理机制的基础上,提出了优化存储器管理的方法。

关键词:动态内存分配、高速缓冲存储器、块化系数、可安装模块

一、引 言

Novell 网是美国 Novell 公司开发的一种高性能局域网系统。NetWare 局域网系统作为 Novell 网的核心技术,已成为世界上流行局域网系统的佼佼者。NetWare 386 是各种 NetWare 版本中使用最广泛的。它运行于 80386 以上的机器上,除具有 SFT 所具有的热修复,磁盘镜像,磁盘双工,TTs 等性能外,还具有整个网络服务器镜像后备功能。NetWare 386 采用平滑存储模式和动态管理技术,使系统的存储器资源得到了充分的利用。了解内存管理的概念,不仅可以帮助我们领会经常出现的与存储器相关的错误信息,从而迅速准确地排除错误;而且可以提示我们,什么时候服务器需增加内存,以及怎样运用 set 命令或其它方法来优化系统运行性能。

二、NetWare 存储器类型

NetWare 386 中存储器分四类:核心存储器(Kernel Memory)、永久存储器(Permanent Memory)、临时存储器(Alloc Memory)和高速缓冲器(Cache Buffer)。其中核心存储器在系统运行过程中始终保持不变而高速缓冲器的数量变动最大。

1. 核心存储器

在 DOS 状态下执行 SERVER.EXE, NetWare 装入模块在 1MB 的地址之上初始化一块服务器内存,装入操作系统内核。从内核开始运行到服务器关闭,核心存储单元始终保持不变。此时,尽管服务器运行在保护模式,而且在 NetWare 内核控制下,最低的 1MB 空间仍被 DOS 占用。如果管理员不需要用软驱装入 NLMs 或在 DOS 分区安装其它应用软件,就可用控制台命令 Remove DOS 将 DOS

移走。这样,这部分内存就可用作高速缓冲存储器。

2. 永久存储器

永久存储器指存放目录,电子报文等的通信缓冲区。永久存储器具有半动态特性。当 NetWare 初始化结束后,就建立了一系列的通信缓冲区。这部分永久存储器在运行过程中始终不变。当上网用户增加时,NetWare 可自动地分配高速缓冲器来增加缓冲区数量。内存一旦被分配或永久存储器,即使在通信负载降低后,也不能再被用作其它类型存储器,直至退出 NetWare。

3. 临时存储器

对于用户与服务器建立或拆除连接,文件的锁定与解锁操作或在 NLMs 运行期间,都会频繁地向 NetWare 内存管理器件请求内存。这类数据结构的生存期比较短,对内存的要求却很频繁,NetWare 将它们定义为临时存储器,并用空闲存储器链接表进行维护。举例来说,当一个 NLM 请求内存时,NetWare 就从链接表所指示的可用临时存储器池中,以“按需分配”的原则满足该 NLM 的要求。当 NLM 释放这部分存储器用尽时,就会从高速缓冲存储器池中窃取一部分。值得注意的是,这部分存储单元在使用完毕后,不再归还给高速缓冲器池。

4. 高速缓冲存储器

所有未被核心存储器、永久存储器、临时存储器占用的内存单元称为高速缓冲存储器。NetWare 运用高速缓存技术缩短文件服务器的访问时间。这种技术通过为自磁盘读入的数据提供内存缓冲器区实现。缓冲区数据将被保存,以希望用户以后会需要它。因为从内存中检索数据比从硬盘中读入数

需要快得多,所以如果增加高速缓冲存储器。用户所需数据在内存中的机会(命中率)就会增加,从而提高服务器的性能。操作系统以最近最少使用(LRN)算法来维护高速缓冲器。一般要求高速缓冲器占内存总容量的70%,最低不少于总容量的20%。

三、NetWare 对内存容量的需求

NetWare 初始化时,用少量的永久存储器建立其内部的数据结构,分配一小部分内存作为临时存储器,其余部分作为高速缓冲器。NetWare 内存管理成功的关键在于它确保有足够的存储空间作为卷上 FAT 表和目录表的高速缓冲存储区,否则 NetWare 就拒绝登录这个卷。

下面是一个计算 Netware V3. X 内存需求的简单公式:

所需内存总数 = $D + N + 2MB$

其中:

$D = (0.23 \times M) \div B$

$N = (0.32 \times M) \div B$

B = 块化系数(默认值为4);

M = 卷的总 MB 数;

D = DOS 卷所需的内存,若 $D < 2MB$ 则取 2MB;

N = 带有名字空间的 DOS 卷的内存。

由这个公式我们可看出,NetWare 的内存需求与块化系数有关,即平均每个存储块需 23 字节 RAM 作为文件目录、FAT 表等的高速缓冲器存储器。块化系数可供选择。对较大的块化系数,硬盘卷空间利用率低(文件最后一块平均有 1/2 容量被浪费),但所需的内存 Cache 容量较小;相反,较小的块化系数,硬盘卷间利用率高,但所需的 Cache 容量较大。实际选择块化系数是综合硬盘利用率和内存容量后决定的。一般来说,选择较大的块化系数对提高系统性能有好处;然而,如果块化系数达到文件平均长度的两倍时,系统性能就会大大降低。例如,对硬盘传输速率较高,且大多数文件长度较大,但内存容量相对较小的服务器,应选取较大的块化系数。

外部环境的变化,例如活动节点数,打开文件数的变化,网络上运行通讯协议的不同,以及服务器和工作站上运行的应用程序的不同,都会影响系统的内存分配和需求的变化。例如要想在服务器上安装运行 Btrieve,就需在占至少 1MB 空间;同样,若想在 NetWare 较好地运行 TCP/IP 协议,也需增加至少 1MB 内存开支。一般地,对学校这样的应用环境,NetWare V3. X 要求至少内存配置为 4MB;

对于工业应用,内存配置应在 8MB 以上。

系统管理员可通过加载可安装模块 MONITOR 来监视服务器内存分配情况,具体使用方法参阅相关的手册,对照上文很容易理解相关概念。此处补充一个 MONITOR 未公开的特性。它可帮助我们了解服务器 CPU 的利用率。装入命令为 LOAD MONITOR P(Enter)。这时在“Available Options”选项,选中此项,将会显示器 CPU 的平均利用率。

四、NetWare 内存使用优化

这里虽然只讨论内存使用的优化问题,但是对整个系统效率的提高都有意义。

内存使用优化的第一步应是在安装服务器之前,浏览要装入服务器的文件,估计文件平均长度,合理选择块化系数。尤其对较大硬盘的服务器,这一步更显重要。因为我们已经知道随块化系数增长,系对 Cache 的需求成倍下降。

内存优化的第二步是在系统进入稳定运行时,加载 MONITOR,查看系统正使用的临时存储器占系统已经分配的这类存储器的百分比。如果此值过小就运用 SET 命令加以调节。同时注意 Cache 高速缓存占总内存容量的百分比,如该值接近 20%,就说明必须为服务器增加内存配置。这里仅就两个比较重要的 SET 命令加以说明。

(1) SET MAXIMUM_ALLOC SHORT TERM MEMEMORY:NUMBER,此命令调整临时存储器所占内存最大值。NUMBER 有效范围是 50000 到 16777216 字节。缺省值是 2MB。该命令并不能禁止操作系统从 Cache 中窃取内存作为永久存储器。因为永久存储器用来存储连接表,事物跟踪,接收缓冲,等进程的重要数据结构,即使 Cache 已严重不足,系统也要保证先满足永久存储器的需求。

(2) SET MINIMUM_FILE_CACHE BUFFER:NUMBER,此命令设置最小的文件缓冲区占总内存容量的比例。NUMBER 最小有效值是 20,为了及时得到内存容量不够的报警信息,一般将此值设置为 40 左右。

内存优化的第三步是在系统维护期间尽可能地减少不必要的内存开支,以增大文件高速缓存。可根据实际情况采取以下几种方法。

(1)用控制台命令 REMOVE DOS 将 DOS 从文件服务器内存中移走,这部分内存将自动加入高速缓冲存储器池。

(2)卸下所有不需要的可安装模块。此外,有些

第三方开发的 NLMs 会造成服务器的性能下降。可以通过检查 NLMs 是否释放了对 CPU 的控制权来判断多任务的执行情况。方法是加载安装模块 INSTALL.NLM 编辑 AUTOEXEC.NCF, 在其最后一行加入控制台命令: "SET DISPLAY RELINQUISH CONTROL ALERTS=ON"。这样只要任何一个 NLM 占用 CPU 超过 0.4s, 服务器就向控制台发一条消息。对这种独占性很强的 NLM 只有在绝对需要时才可以加载。

(3) 当内存资源十分紧张时, 尽量减少卷上的目录数, 将文件集中到少数几个目录中。这种方法听起来不很现实, 但的确有效。最好在对硬盘管理时, 使目录结构合理化。因为每个子目录至少占用一个目录块, (目录块的大小等于 4KB) 所以, 只有一个文件的子目录和具有 32 个文件的目录需要一样多的存储器。如果合并了目录以使大部分的目录区拥有接近 32 个文件, 然后清除已删除的子目录

和文件, 系统将会释放出存储器。

总之, 内存管理的优化重心是增加高速缓冲存储器, 而解决因服务器内存量不足引起的性能变差或故障的最根本办法是增加存储器硬件配制。

五、小 结

本文在分析 NetWare 386 内存管理技术的基础上, 对内存管理的优化提出几点建议。在实践中应根据实际应用环境合理选择各项参数, 而不是简单地采用系统默认值。只有经过反复尝试、修改才可能达到适合自己系统的参数配制。

参考文献:

- [1] Troubleshooting NetWare For the 386 资料
- [2] 雪峰编译, 《NOVELL NETWARE 网络服务器故障检测和维护》, 北京科海培训中心

(本文收稿日期: 1995.7.5)

本刊明年改双月刊、自办发行

本刊以往交邮局代办发行, 从 86~95 年已有十年的历史, 交邮局发行一般比较省事, 但也不是一切如意。

自办发行就是由本杂志社服务部负责发行。订户直接把订购款汇到本刊, 杂志社直接把刊物寄到订户手中。很明显, 这种直接联系有利于订户、读者与本刊的沟通。

本刊为了增进与订户、读者的沟通, 正计划建立起订户资料库。为此将随订单附一个简要的问卷调查。以后本刊可以随时据此向订户发送信息资料。

96 年本刊改为双月刊, 将扩大篇幅, 增加栏目。按照纸张和出版费用的总水平, 定价亦相应调高。以往邮发, 发行费占去定价的 40%, 调价时, 发行费水涨船高, 增加了订户的负担。现在自办发行, 还可以试行对订户集体订购实行优惠。一个大号信封成本 0.40 元, 若一个信封装 2 本杂志, 就可省下 0.40 元。这 0.40 元就各归订户 0.20 元。于是我们定下的优惠条件是 2~4 本的集体订户每本减 0.20 元, 5 本以上的集体订户每本减 0.40 元。定价与优惠价详见下表。特别欢迎院校师生集体订阅。原意进行零售的单位, 包括原基层邮局邮亭门市单位, 请来函联系, 将给予更多的优惠。

| | 单 价 | 2~4 本 集体订户优惠价 | 5 本以上 集体订户优惠价 |
|--------|---------|------------------|------------------|
| 每 期 | 3.80 元 | 3.60 元 | 3.40 |
| 半年 3 期 | 11.40 元 | 10.80 元 | 10.20 元 |
| 全年 6 期 | 22.80 元 | 21.60 元 | 20.40 元 |

PC机与8031的并行数据通讯 及自校正标识符算法

哈尔滨工业大学 张乐 孙华

摘要:本文以研制IC卡读卡机为背景,介绍了PC机与8031并行通信的硬件机制,并提出了一种以自校正标识符算法为核心的软件通信协议,成功地开发出一个双机并行数据通信的最小系统,使多CPU的数据交换和传递变得更加简单、可靠。

关键词:并行通讯 IC卡 自校正标识符 PC机扩展槽

PC机与8031通过串口进行数据通讯已广为人知,但是在某些情况下,需要将8031系统板做成卡,以插槽形式插在PC机内,这就需运用PC机与8031的并行数据通讯技术。

PC机与8031的主频不一样,因此,它们的运行速度也就不一样,而并行通讯又不象串行通讯可以利用波特率来统一收发速度。因此,如何解决PC机与8031的同步问题以及如何避免通讯过程中的多数及丢数现象,这就必须遵循一定的通讯协议。我们在开发研制IC卡读卡机的过程中,成功地设计出一个双机并行通讯的最小系统,并开发出一套以间隔标识符和应答握手信号为核心的通讯协议来完成PC机和8031之间的数据传送。

整个系统由一个8031最小系统,两个并行数据通路和一个地址译码控制器形成的一块卡插在

PC机的扩展槽中来实现。硬件原理图见图1,其基本工作原理是这样的:通过对地址译码控制器的设置,使右边的两片74LS373和74LS244作为PC机的两个端口来完成PC机与I/O口的数据交换;8031最小系统也通过译码器对左边的两片74LS373和74LS244进行地址设定,把它们作为外部存储器来进行数据交换。双机通讯采用进出两个通道实现,互不干涉,因此可有效地避免了总线占用和总线冲突问题。

但是,在PC机发数给8031的过程中存在这样的问题:由于PC机欲发的数是键盘敲入,因此PC机发数速度远远慢于8031接数速度。此外,74LS373是锁存器,PC机每次发送的数均锁存在373的锁存端,直到下一个不同的数据来到时才能将旧数据刷新。因此,当键盘敲入相同数时,8031无法辨别是新来的数据还是上次锁存的旧数据,这样就容易造成丢数现象。针对这种现象,我们设计了一种标识符,间隔地插在每个要送的数之间,这样可把所送的不同数都分隔开。引入标识符SYMBOL后又面临新的问题:如果要送的数就是标识符本身,同样会造成丢数。为此提出一种自校正标识符算法,由子程序JUDGE()实现。

在介绍这种算法之前,先作以下几个约定:

1、由于PC机是八位数据总线,因此设所传送的数都是00H—0FFH间的数。

2、从键盘敲入的十六进制数均由“,”或“ENTER”隔开。即当PC机接收到键盘

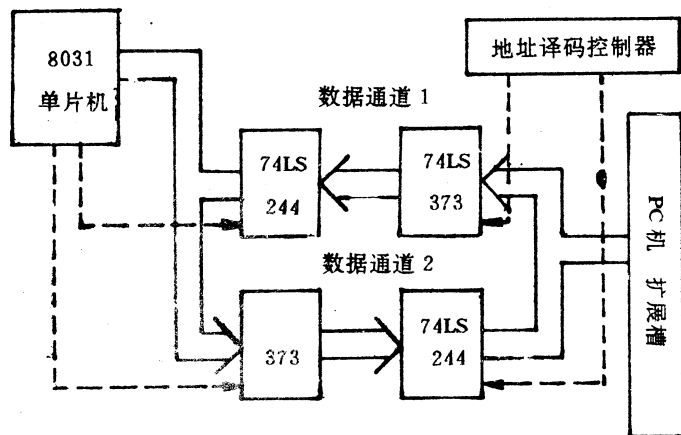


图1 PC机与8031并行通信原理图

敲入的“,”或“ENTER”时,则发送标识符来分隔发送的有效数据。(注:标识符本身也是 00H—0FFH 之间的数)。

3、几个相关的子程序:

GETKEYS()——从键盘获得十六进制数如 56H,0F0H 等。

GETKEY1()——从键盘获得“,”或“ENTER”,即把相应的标识符赋给标识符变量。

4、SYMBOL1,SYMBOL2,SYMBOL3 为用户设定的三种标识符取值。下面介绍一下自校正标识符算法以及单片机,PC 机利用这种算法进行应答通信的过程。

一、自校正标识符算法

引入标识符的目的在于分隔开相同的有效数据,所以标识符应插在连续发送的两数之间,且不同于前数和后数,即标识符不是固定的,而是动态的,变化的。但是由于标识符既不同于前数又不同于后数,加上初始设定的标识符本身,因此最多只有三种取值。算法应根据所发送的不同数据自动设置相应的标识符。

自校正标识符算法程序:

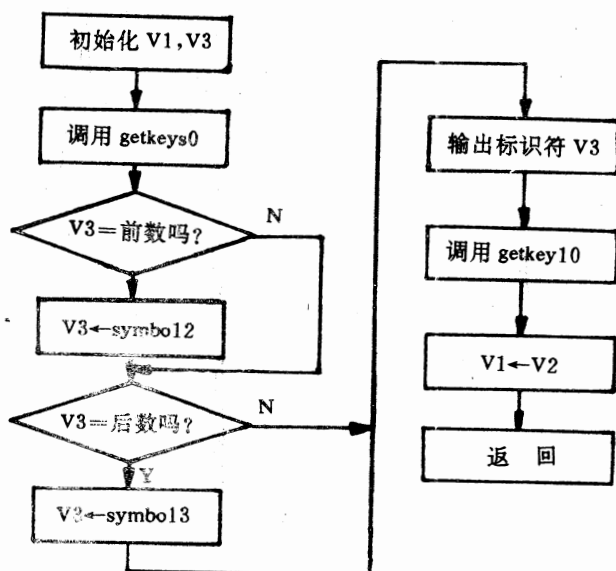
入口:V1——存放前一次发送的数据;

V2——存放当前待发数据;

V3——存放当前标识符值;

出口:发送标识符。

流程及软件程序如下:



对 PC 机而言:0x371 为数据通道 1 的出口地址,
0x370 为数据通道 2 的入口地址,
自校正标识符 SYMBOL 的三种取值
设为 0x8b,0x00,0x01;

```

void judge(int v2,int v1)
{
    int v3;
    if((v2==0x8b)||(v1==0x8b))
    { v3=0x00;
      if((v3==v2)||(v3==v1)) v3=0x01;
      outportb(0x371,v3);
    }
    else{ outportb(0x371,0x8b);}
}
  
```

```

getkeys()
{
    int s1=0;
    int k=0;
    int m=0;
    for(;;) { k=getche();
              if(k==27)exit(0);
              k=k-0x30;
              if((k>=0x00)&&(k<=0x09))
              {s1=s1*(0x10)+k;
               m=m+1;
              }
              else if((k>=0x31)&&(k<=0x36))
              {k=k-39;
               s1=s1*(0x10)+k;
               m=m+1;
              }
              else
              {printf("\nwrong input!");
               printf("\nreinput please...\n");
               s1=0;
               m=0;
              }
              if(m==2){ break; }
            }
    s1=s1&0x00ff;
    return s1;
}
  
```

```

void getkey1()
{
    int k,s2;
    for(;;) {k=getche();
              if(k==27)
              {exit(0);}
              else if(k==0x2c||(k==0xd))
              { break; }
              else
              { printf("\nwrong input!!!!");
               printf("\nreinput please
  
```

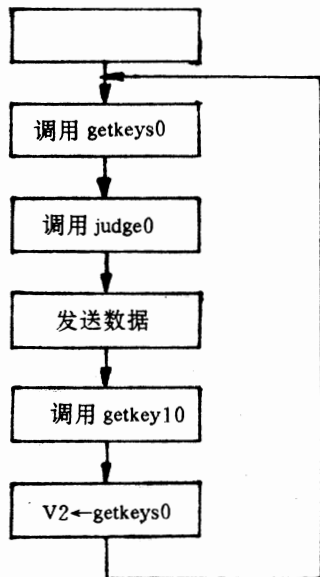
```

    this number... \n");
}
}
}

```

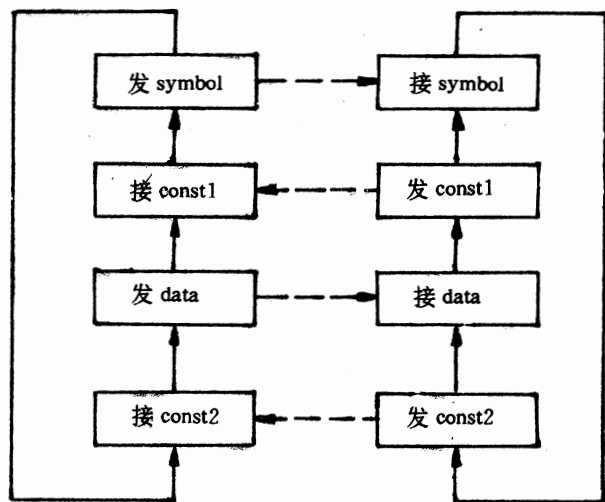
二、PC 机发数, 8031 接数。(8031 把接收到的数存放在内部 RAM 的 30H—7FH 寄存器单元中)。

流程图如下:



三、8031 送数, PC 机接数:

单片机发送一组存放在内部 RAM 中的数据不同于 PC 机发送由键盘输入的数据。因为键盘敲



SYMBOL——由 8031 发出, 表示 8031 准备发数;

CONST1——由 PC 发出, 表示 PC 机准备接收;

CONST2——由 PC 发出, 表示 PC 机接收完毕;

图 2 双机通信应答机制

击速度较慢, 所以有足够的缓冲时间让单片机接收到; 而 8031 汇编为机器语言, 其执行速度很快, 因此很容易造成丢数现象。解决这一问题的关键是建立完善的应答机制, 速度快的一方收到对方“接收完毕”信号后再继续发数, 使二者能协调统一。

PC 机与 8031 通信应答机制流程图如图 2。

在以下程序中设定:

CONST1 = # 86H;

CONST2 = # 87H;

BUSY = # 84H 表示 8031 或 PC 机正忙;

对 8031 而言:

4000H 为数据通道 1 入口地址;

6000H 为数据通道 2 入口地址;

V1 = 23H 寄存器单元;

V2 = 25H 寄存器单元;

V3 = 24H 寄存器单元;

用 C 语言编写的接口程序以及用单片机实现的汇编语言程序如下:

8031 接收数据, PC 机发送数据:

```

COM:  MOV DPTR, # 4000H
      MOVX A, @DPTR
      CJNE A, # 82H, COM1
      AJMP COM

```

```

COM1:  MOV 21H, A
      MOVX A, @DPTR
      CJNE A, 21H, COM2
      AJMP COM1

```

```

COM2:  MOV 23H, A
      MOVX A, @DPTR
      CJNE A, 23H, COM22
      AJMP COM2

```

```

COM22: MOV 22H, A
      MOVX A, @DPTR
      CJNE A, 22H, COM32
      AJMP COM22

```

```

COM32: MOV 23H, A
      MOV R7, 22H
      MOV R0, # 30H

```

```

COM3:  MOV DPTR, # 4000H
      MOVX A, @DPTR
      CJNE A, 23H, COM4
      AJMP COM3

```

```

COM4:  MOV 23H, A
      MOVX A, @DPTR
      CJNE A, 23H, COM44
      AJMP COM4

```

```

COM44: MOV @R0, A
      MOV 23H, A

```



```

INC R0
DJNZ R7,COM3
RET

8031 发送数据,PC 机接收数据:
PCCOM:  MOV 23H,#84H
        MOV 24H,#85H
        MOV 25H,30H
        MOV R0,#30H
        MOV R7,22H
PCCOM3: MOV A,24H
        CJNE A,23H,WAIT1
        MOV 24H,#00H
WAIT1:  MOV A,24H
        CJNE A,25H,WAIT2
        MOV 24H,#01H
WAIT2:  MOV DPTR,#6000H
        MOV A,24H
        MOVX @DPTR,A
        MOV DPTR,#4000H
PCCOM1: MOVX A,@DPTR
        CJNE A,#86H,PCCOM1
        MOV DPTR,#6000H
        MOV A,25H
        MOVX @DPTR,A
        MOV 23H,25H
        INC R0
        MOV A,@R0
        MOV 25H,A
        MOV DPTR,#4000H
PCCOM2: MOVX A,@DPTR
        CJNE A,#87H,PCCOM2
        DJNZ R7,PCCOM3
        RET

```

PC 机的收发通信程序:

```

void pccom()
{
    int v,k;
    int loop;
    int id=0;
    int flag=0;
    int sym=0x00;
    printf("\n");
    printf("\nSEND WILL BE CARRIED ON !");
    printf("\nplease input... \n");
    printf("press COMMA or ENTER to confirm. \n");
    v=getkeys();
    outportb(0x371,v);
    getkey1();
    outportb(0x371,0x8b);
}

```

```

v=getkeys();
if(v<=0||v>=0x50)
    {printf("\nwrong input! \n");
    v=getkeys();
    }
loop=v;
outportb(0x371,v);
printf("\n");
v=inportb(0x370);
do {    v=inportb(0x370);
    if(bioskey(1)) { k=bioskey(0);
        if(k==ESC) {exit(0);}
    }
    if(v==0x84)
    {
        printf("\n8031 is busy,please wait...");
        do {v=inportb(0x370);}while(v==0x84);
        id=1;
    }while(id!=1);
    sym=v;

do {
    outportb(0x371,0x86);
    do { v=inportb(0x370);}while(v==sym);
    vd=v;
    outportb(0x371,0x87);
    printf("%x ",vd);
    do { v=inportb(0x370);}while(v==vd);
    sym=v;
    sum=sum+1;
    } while(sum!=loop);
    flag=1;
    printf("\nsend finished!");
}

```

实际调试和运行的结果表明:这种以自校正标识符为核心的通信机制和通信协议有效地解决了PC机与8031之间的并行数据通信中所出现的问题,整个系统具有结构简单,高效可靠的优点,不失其借鉴和采纳价值。 (本文收稿日期:1995.4.3)

欢迎订阅
欢迎投稿
欢迎刊登广告

如何检测串行口

天津塘沽 707 所 高铁杠

串行口作为计算机与外设的主要接口之一,在计算机的通讯与控制系统中起着非常重要的作用。然而,如何对串行口检测与控制却也是程序员最感棘手的问题之一,当发生程序不能正常的对串行口进行数据的读取时,程序员往往在是软件问题还是口子的问题之间犹豫不决。因此有必要设计这样一个程序,它既能检测串口通讯是否正常,又能根据不同的串口配置(这里指诸如波特率、数据位等)进行自动设置以检测各种配置的串行口。本文就是在此思想的基础上开发了一个串口发送与接收的程序,它允许的串口为通常的 RS232,允许的各种参数基本满足了正常通信的各种配置,同时允许我们任意组合。这样,在控制系统中,如果双方采用串口进行通信,一旦任何一方出现问题,均可用此程序进行检测。即可用接收程序检测发送方的数据格式正确与否,也可用发送程序检测接收方接收的数据是否正确。如有必要,对程序稍加改动即可变为一个模拟发送器或接收器。

本程序在笔者的工作与程序设计中其作用得到了很好的体现。测试时可用两台微机的串口相连,发送方发的数据在发送方显示,接收方接收到数据后也显示出来,以便校对。本程序以 MSC7.0 编写,有关对串口的设置请参阅有关资料,程序在研华工控 386 机、Cmpaq 386/20e 等机上均已通过编译、连接、运行。最后说明,该程序只做检测,因为一般控制软件均采用中断方式接收数据,而本程序是以查询式接收的,所以本程序只能作为一种测试工具使用,为方便对中断接收感兴趣的读者,笔者也将用中断方式接收的程序附于文章后,以供参考,读者可根据自己的不同需求稍加修改即可满足各种情况下的中断接收处理。关于中断方式的设置请参阅有关书籍,笔者不再一一赘述。

最后,笔者对编者在审阅本文初稿时所提的改进意见表示衷心感谢。

程序 1: 查询式接收

/* 说明: 1. 程序可在 Msc7.0 下编译
2. 执行时选好参数可等待接收

3. 程序将接收到的字符显示出
4. 如要退出可按任一健

```

* /
#include<stdio. h>
#include<conio. h>
#include<bios. h>
#include<io. h>
#include<fcntl. h>
#include<dos. h>
#include<stdlib. h>
#define COM1PORT 0x3f8
#define COM2PORT 0x2f8
#define BAUD600 0x60
#define BAUD1200 0x80
#define BAUD2400 0xA0
#define BAUD4800 0xC0
#define BAUD9600 0xE0
#define NOPARITY 0x00
#define ODDPARITY 0x08
#define EVENPARITY 0x18
#define STOPBIT1 0x00
#define STOPBIT2 0x04
#define BITS7 0x02
#define BITS8 0x03
int port;
unsigned char bit;
unsigned char stop;
unsigned char pari;
unsigned char bord;
int addr;
unsigned init(int port,unsigned char config);
main()
{
    unsigned cc;
    int c;

    printf("输入通讯口号 1:COM1,2:COM2 \n");
    scanf("%d",&c);
    if(c==1)port=0;
    if(c==2)port=1;

    printf("输入波特率 1:1200 2:2400 3:4800 4:9600 \n");
    scanf("%d",&c);
    if(C==1)bord=BAUD1200;

```

```

if(C==2)bord=BAUD2400;
if(C==3)bord=BAUD4800;
if(C==4)bord=BAUD9600;
printf("输入数据位 1:7 位,2:8 位 \n");
scanf("%d",&c);
if(c==1)bit=BITS7;
if(c==2)bit=BITS8;
printf("输入停止位 1:1 位,2:2 位 \n");
scanf("%d",&c);
if(c==1)stop=STOPBIT1;
if(c==2)stop=STOPBIT2;
printf("输入校验法:1:奇校验,2:偶校验,3:无奇偶");
scanf("%d",&c);
if(c==1)pari=ODDPARITY;
if(c==2)pari=EVENPARITY;
if(c==3)pari=NOPARITY;
init(port,bord|pari|stop|bit);
if(port=0)addr=COM1PORT;
if(port=1)addr=COM2PORT;
cc=inp(addr);
cc=inp(addr);
cc=inp(addr);
while(! kbhit()){
    r;cc=inp(addr+5); /* 读状态口 */
    if(cc&0x1e){ /* 错误处理 */
        inp(addr);
        goto r;
    }
    if(cc&0x1){ /* 接收就绪 */
        cc=inp(addr);
        putchar(cc);
    }
} /* 可按任一健退出 */
}
/* 初始化串口 */
unsigned init(int port,unsigned char config)
{
    union REGS r;
    r.h.ah=0;
    r.h.al=config;
    r.x.dx=port;
    int86(0x14,&r,&r);
    return(r.x.ax);
}

```

程序 2: 查询式发送

/* 说明:

1. 程序可在 Msc7.0 下编译
2. 执行时选好参数可等待发送
3. 循环发送 1234567890
4. 程序将发送的字符显示出
5. 如要退出可按任一健

*/

```

#include<stdio.h>
#include<conio.h>
#include<bios.h>
#include<io.h>
#include<dos.h>
#include<stdlib.h>
#define COM1PORT 0x3f8
#define COM2PORT 0x2f8
#define BAUD600 0x60
#define BAUD1200 0x80
#define BAUD2400 0xA0
#define BAUD4800 0xC0
#define BAUD9600 0xE0
#define NOPARITY 0x00
#define ODDPARITY 0x08
#define EVENPARITY 0x18
#define STOPBIT1 0x00
#define STOPBIT2 0x04
#define BITS7 0x02
#define BITS8 0x03
int addr;
int port;
char bit;
char stop;
char pari;
char bord;
unsigned char buff[]="1234567890";/* 发送的字符 */
unsigned init(int port,unsigned char config);
main()
{
    unsigned cc;
    int i;
    unsigned char bb;
    int c;
    printf("输入通讯口号 1:COM1,2:COM2 \n");
    scanf("%d",&c);
    if(c==1)port=0;
    if(c==2)port=1;
    printf("输入波特率 1:1200 2:2400 3:4800 4:9600 \n");
    scanf("%d",&c);
    if(c==1)bord=BAUD1200;
    if(c==2)bord=BAUD2400;
    if(c==3)bord=BAUD4800;
    if(c==4)bord=BAUD9600;
    printf("输入数据位 1:7 位 2:8 位 \n");
    scanf("%d",&c);
    if(c==1)bit=BITS7;
    if(c==2)bit=BITS8;
    printf("输入停止位 1:1 位,2:2 位 \n");
    scanf("%d",&c);
    if(c==1) stop=STOPBIT1;
    if(c==2) stop=STOPBIT2;
    printf("输入校验位:1:奇校验,2:偶校验,3:无奇偶 \n");

```

```
scanf("%d",&c);
if(c==1)pari=ODDPARITY;
if(c==2)pari=EVENPARITY;
if(c==3)pari=NOPARITY;
init(port,bord|pari|stop|bit);
i=0;
if(port==0)addr=0x03f8
if(port==1)addr=0x02f8
while(! kbhit()){
    qq=inp(addr+5);
    if((bb&0x20)){/* 发送寄存器空 */
        bb=buff[i];
        _asm{
            mov al,bb
            mov dx,addr
            out dx,al
        }
        putc r(bb);/* 显示 */
        i+=1;
        if(i>9)i=0;
    }
    else goto qq;
}
```

```
unsigned init(int port,unsigned char config)
{
    union REGS r;
    r.h.ah=0;
    r.h.al=config;
    r.x.dx=port;
    int86(0x14,&r,&r);
    return(r.x.ax);
}
```

/* 说明:以下为中断方式接收的程序例子

其中 reint.c 调用 reinta.asm
将 reint.c 用 C 编译成目标文件
将 reinta.asm 用宏汇编 masm 5.0
编译成目标文件,而后以下式连接
即可生成 reint.exe
link reint+reinta;

*/

程序 3:中断式接收主程序(MSC7.0)

```
名称:reint.c
#include<stdio.h>
#include<dos.h>
#include<memory.h>
#include<string.h>
unsigned int addr=0x3f8;
unsigned char dch='1';
extern unsigned char dataflag;
extern void _interrupt_far newcom();
extern void _initcom(unsigned config,unsigned ff,unsigned
dd);
main()
{
    /* 设串口 1,波特 1200,数据 7 位,停止 1 位,奇校验 */
    initcom(0x3f8,0x0060,0x000a);
    /* 设置中断向量,初始化串口 */
    while(! kbhit()){
        if(dataflag==0xff){/* 收到字符就显示 */
            putchar(dch);
            dataflag=0;
        }
    }
    restorecom(); /* 恢复中断向量 */
    exit(0);
}
```

程序 4:中断式接收子程序(汇编)

名称:reinta.asm

```
_TEXT SEGMENT byte public _DATA SEGMENT
'CODE'
    _TEXT ENDS
const SEGMENT word public 'const'
const ends
_BSS SEGMENT word public 'BSS'
_BSS ENDS
_DATA SEGMENT word public
'DATA'
_DATA ENDS
DGROUP GROUP const,_BSS,_data
    ASSUME cs:TEXT,ds:
DGROUP,es:DGROUP,ss:DGROUP
    public _dataflag
    extrn _dch:byte
    extrn _addr:word
    _dataflag db 0
    old_int4_CS dw 0
    old_int4_IP dw 0
_DATA ENDS
_TEXT SEGMENT
    PUBLIC _initcom
_initcom PROC FAR
    push bp
    mov bp,sp
    push ax
    push bx
    push dx
```

```
push ds
push es
mov ah,35h
mov al,0ch
int 21h
mov old_int4_CS,es
mov old_int4_IP,bx
mov dx,offset _newcom
mov ax,seg _newcom
mov ds,ax
mov al,0ch
mov ah,25h
int 21h
mov dx,[bp+6]
add dx,4
in al,dx
```



```

and    al,0fch
out     dx,al
dec     dx
mov     al,80h
out     dx,al
mov     ax,[bp+8]
sub     dx,3
out     dx,al
xchg    ah,al
inc     dx
out     dx,al
mov     ax,[bp+10]
add     dx,2
out     dx,al
sub     dx,2
mov     al,1
out     dx,al
add     dx,3
mov     al,0bh
out     dx,al
in      al,21h
and     al,0efh
out     21h,al
sub     dx,4
in      al,dx
in      al,dx
in      al,dx
pop     es
pop     ds
pop     dx
pop     bx
pop     ax
pop     bp
ret
- initcom ENDP
PUBLIC _restorecom
_restorecom PROC FAR
    push    ax
    push    dx
    push    ds
    mov     dx,old_int4_IP
    mov     ax,old_int4_CS
    mov     ds,ax
    mov     al,0ch
    mov     ah,25h
    mov     21h
    in      al,21h
    or      al,10h
    out     21h,al
    pop     ds
    pop     dx
    pop     ax
    ret
_restorecom ENDP

indata proc
    push bp
    mov bp,sp
    mov dx,[bp+4]
i1:add dx,5
    in al,dx
    test al,1
    jnz i2
    pop bp
    ret 2
i2:test al,0ch
    jz i3
    sub dx,5
    in al,dx
    pop bp
    ret 2
i3:sub dx,5
    in al,dx
    mov dch,al

    mov dataflag,Offh
    pop bp
    ret 2
indata endp
_newcom PROC FAR
    push    ax
    push    bx
    push    cx
    push    dx
    push    si
    push    di
    push    bp
    push    ds
    push    es
    sti
    mov     ax,dgroup
    mov     ds,ax
    mov     ax,_addr
    push    ax
    call    indata
    mov     al,20h
    out     20h,al
    pop     es
    pop     ds
    pop     bp
    pop     di
    pop     si
    pop     dx
    pop     cx
    pop     bx
    pop     ax
    iret
_newcom ENDP
-TEXT ENDS
END

```

(本文收稿日期:1995.7.17)



广州年底将举办华南电子盛会

'95华南电子新产品、新技术博览会,将于1995年12月21日至24日在广州市广东国际科技展览中心隆重举行。本届博览会同全国电子信息中心等单位联合主办,得到了我国电子行业有关政府部门、研究机构、协会、新闻媒体的大力支持,届时将有百余电子生产商、贸易商展出当今最新计算机、通信、

广播、电视、音响、仪器、仪表、消费电子、电工器材、生产设备、电子材料等产品和技術。

为促进电子科技发展和交流,博览会同期还举行技术交流会。本届大会将吸引中外客商上万人次前往参观和贸易,成为电子厂商开拓广阔华南市场的有效途径。

参展总代理联络电话:(020)3348888转2504、2500
地址:广州童心路西胜街市科技中心科苑楼2504室
邮编:510091

利用 CCED 进行彩色打印的尝试

石油大学 代永壮 李继红

内容提要:本文描述了在 CCED 软件中,利用打印机本身功能实现排版及彩色打印的方法,并举例实现了部分功能。

关键词:打印;CCED 彩色

一、问题的提出

CCED 因具有方便的表格编辑功能而得到广泛应用,但 CCED 本身不含任何打印驱动程序,所以在打印文件或报表时,只能利用打印机本身的功能或其它软件的支持。现在,用户配置的打印机一般都带硬字库,象 EPSON LQ-1600K、LQ-1800K、STAR CR-3240 等,它们不但带有硬字库,而且支持汉字的各种变换。CR-3240 还支持放大汉字的平滑处理、彩色打印等,因此利用打印机本身的功能可以快速地打印出各种文件。利用 CCED 软件排版,需要用户直接输入打印机的控制码序列,给用户带来不便,并且用户手中的 CCED 软件大部分没有提供打印控制功能,为此,笔者针对 STAR CR-3240 彩色打印机,设计了一组软件,为 CCED 用户提供了方便的汉字变换功能和彩色打印功能。

二、设计思想

本软件为用户提供了一套易学易用的打印控制命令,它们以 ^ (不是 CTRL) 为标志,例如: ^R、^B、^G 分别表示红色、蓝色、绿色。用户可以直接输入到文件的相应位置。另外,设计了一组软件,将系统的 INT 17H 修改后驻留内存,取代 INT 17H。它们的功能就是识别打印控制命令,然后向打印机发送相应控制码序列,使打印机在此控制命令之后按用户设置的字型或颜色打印。

三、软件的实现

1. INT 17H 的修改

在新的 INT 17H 中断服务程序中,首先判断要输出的是否为打印控制命令,若是,则识别此命令并向打印机发送相应控制码序列,若不是,则调用原 INT 17H,输出该字符。

2. INT 17H 的替换

INT 17H 的替换是通过改变中断向量实现的。在程序中,首先将原 INT 17H 的中断向量从向量表中取出,放入系统为用户留用的位置,然后将修改后的中断服务程序的入口地址安装在 INT 17H 在向量表中的位置,并将该服务程序驻留内存。这样就达到了截取、取代 INT 17H 的目的。

3. 打印控制命令的识别

打印控制命令的识别是在系统调用 INT 17H 时,由新的 INT 17H 中断服务程序完成的。根据 INT 17H 的功能,系统在调用时,AH 寄存器存放功能号,AH=0 表示要输出字符,AL 寄存器存放要输出字符的 ASCII 码。所以在识别打印控制命令时,首先判断 AH 是否为零,若 AH 为零,再判断要输出的是否为“^”,若是,则将打印控制命令标志置为 1,然后再判断是何命令,识别之后转到相应程序段,输出该命令的控制码序列。

四、本软件的使用方法

本程序是用汇编语言编制的,它经汇编、连接成 EXE 文件,在打印文件之前运行本程序即可。打印控制命令可以在编辑状态直接输入,^ 作为控制标志和规定的字符组合成控制命令,但是,只要不是组合的控制命令,^ 和这些字符仍然都可以作为文件中的字符正常使用。

在本程序中只设置了 ^L、^S、^H、^R、^B、^C、^Y、^M、^G 等 9 个打印控制命令,它们分别表示设置倍宽打印、取消倍宽打印、选择黑色、红色、蓝色、紫色、黄色、橙色、绿色打印。如果读者需要扩展该命令集,依法设置需要的命令,在程序中做相应修改即可。

五、源程序清单

本程序在 AST386 计算机、STAR CR-3240 彩色打印机上,DOS6.0、MASM 宏汇编环境下运行通过。

(本文收稿日期:1995.5.26)

传真: 3851054 邮编: 510100

微机磁盘 CMOS 设置 出错常见故障排除5例

江苏省邳州港务管理局计算机中心[221300] 花 成

在现代微机系统中,系统之中设备的配置都是通过一个称作 SETUP 的设置界面进行配置的(在较低档的微机上,系统的配置通过主机板上的 DIP 开关来完成),即 CMOS 设置。其中包括软硬盘参数的设置,显示器类型、时间、日期等参数的设置,如果此部分由于设置不当,特别是软硬盘参数设置不当,将会引起整机系统不能正常引导、出现读写错误,操作速度下降等故障,下面就硬盘参数选择和配置不当所引起的故障作一简析。

故障现象一:一台微机,可以正常引导系统硬盘在工作过程之中,速度明显减慢,但其读写功能均正常。

故障分析及排除:

此台微机为 SUN-286 微机硬盘尺寸为 41M,多年来该台微机工作一直都是正常的,只是最近一段时间内才出现微机系统读写速度下降的现象。怀疑是受到了病毒的攻击,经使用 KILLV7.0,瑞星防病毒卡等检测,均报道未发现任何计算机病毒,后经仔细询问用户故障发生的过程,原来用户在机器使用过程之中,由于存放、使用时间长,电池 3.6V 可充电电池失效,造成 CMOS 中的硬盘参数丢失,经重新设置之后,便出现了系统读写速度下降的故障现象。打开微机电源,当系统 ROM BIOS 在自检过程之中屏幕上提示:

HITTO SETUP

时按下键进入 SETUP 画面设置界面,硬盘参数显示如下:

| TYPE | CYLN | HEAD | WPCOM | LZONE | SECT | SIZE |
|------|------|------|-------|-------|------|------|
| 17 | 977 | 5 | 300 | 977 | 17 | 41MB |

经查阅随机资料和有关使用说明书才发现是由于硬盘类型设置出错,正确的参数类型应为“41”。

| TYPE | CYLN | HEAD | WPCOM | LZONE | SECT | SIZE |
|------|------|------|-------|-------|------|------|
| 41 | 977 | 5 | 977 | 977 | 17 | 41MB |

在 SETUP 界面设置中,将光标横条移至硬盘参数设置上,利用<PGUP>键或<PGDN>键将硬盘类型由“17”改为“41”之后保存设置的参数,重新开

机。系统工作便恢复正常了。

故障原因:

读者可能要问,为什么硬盘的正确类型为“41”,设置成错误的类型“17”却也能正常引导系统呢?由于其 CYLN(柱面),HEAD(磁头数),SECT(扇区数),SIZE(尺寸)基本相同,故可以换用引导系统,如果这一点还能够理解的话,那为什么更换硬盘类型之后,系统能够正常引导,工作速度却明显的下降了呢?原来这原因主要在参数项 WPCOM 之上。

WPCOM(WRITE PRECOMPENSATION)称作“写入预补偿”,是由于为了防止磁盘“内圆柱”数据读写错误而设置的“预补偿”,“写入预补偿”只需要采用在靠近主轴的或其附近的圆柱上,外圆柱的信息一般不需要写入预补偿,此种补偿的实现是由硬件完成的,本题中由于用户误将盘的参数设置出错虽然尺寸基本相同,但其 WPCOM 却不相同,其值由正确的“977”变成了“300”即由原来的不需写预补偿操作变成了从“300”柱面开始进行写预补偿操作(WPCOM=977时表明写入预补偿操作从 977 柱面开始往高道写,由于此盘 CYLN 值为 977,故实际上没有“写入预补偿”),无疑在读写数据时,增加了硬盘系统的负担和开销,故而执行速度变慢了,由于其除了 WPCOM 的值不一样外,其它值参数基本上都完全一样,故而机器可以利用类型为“17”的参数正常引导操作系统了。(注:此机为 SUN-286)

故障现象二:一台微机在开机之后不能从 C 盘引导操作系统并且屏幕出现:

NON-SYSTEM DISK OR DISK ERROR

REPLACE AND STRIKE ANY KEY WHEN READY

按照出错提示信息在 A 驱插入系统引导盘,可以正常的引导系统并且能够转到 C 盘上,进行完全正常的读写操作。

故障分析及排除:

当接到此故障时,起先怀疑是由于系统 C 盘的 DOS 引导扇区记录遭到破坏而导致的故障。熟悉 DOS 引导机制的用户都知道,系统在自检完后,便运行 DBR(DOS BOOT RECORD)寻找系统两个隐含文件 IBMBIO.COM 和 IBMDOS.COM (DOSV3.30),若寻找不到这两个文件中的一个,便会给出上述屏幕显示的出错提示信息,况且从故障现象分析,可以从 A 驱引导系统并且转到 C 盘上进行“正常”的读写操作,从而判断,此类故障是由于 C 盘 DOS 中 DBR 区域遭到破坏而引起的。

可是,事实排障的结果证明,上述推测是错误的,系统出错的原因是由于硬盘参数类型设置错误而造成的,在机器引导过程中根据系统提示键键进入 SETUP 界面成的。在机器引导过程中为:

| Type | Cyl | Head | WPcom | LZone | Sect | Size |
|------|-----|------|-------|-------|------|------|
| 1 | 306 | 4 | 128 | 305 | 17 | 10MB |

经查阅随机说明书,发现机器硬盘正确的类型为“17”:

| Type | Cyl | Head | WPcom | LZone | Sect | Size |
|------|-----|------|-------|-------|------|------|
| 17 | 977 | 5 | 300 | 977 | 17 | 41MB |

将 C 盘类型设置成正确的数值之后存入 CMOS,系统便可以正常引导系统了。

故障原因:

当系统引导时出现这种故障是极易迷惑人的,因为从故障现象看,无论怎样分析,也不会想到是 CMOS 中 C 盘类型设置出错而引发的故障,如果从现象分析,按照 C:盘 DBR 遭到破坏而作为一般的 DOS 引导类故障排除的话,那么永远不能排除故障,而且还会真正地破坏掉盘上原来保存的重要的信息,可是为什么由于硬盘类型设置出错便会出现上述同 DOS 引导故障一样的出错提示信息呢?

原来,并不是所有类型设置出错都会出现这种故障现象的,一般情况下,硬盘的正常类型值和错误设置值相差较大的时候才会出现这类故障(主要是 CYLN,HEAD,SIZE 三个参数值),由于类型设置不正确,故而系统在引导过程中,在所设置的错误的 C 盘参数中找不到存放 IBMBIO.DOS 和 IBMDOS.COM 两个隐含文件的所在扇区,故而判断两个隐含文件不存在,(其实,这两个隐含文件存在正常类型的正常扇区上),系统按照错误的设置类型检测查找,当然查找不到,从而出现上述故障现象了。

故障现象三:机器开机之后,在自检过程中显示如下提示信息:

WAIT.....

C:Drive error

Press <F1> to Resume

按下<F1>键之后,屏幕显示:

NON-SYSTEM,DISK OR DISK ERROR

REPLACE AND STRIKE ANY KEY
WHEN READY

这类故障现象与故障二基本上一样,根据系统提示将软盘 DOS 插入驱动器之后,可以正常引导操作系统,但是转到 C:盘上时,出现下面提示信息:

INVALID DRIVE SPECIFICATION

故障分析及排除:

根据故障现象和提示信息,结合故障二,可以很容易地查出此类故障是由于硬盘类型参数设置出错而引起的故障,只不过由于设置类型出错甚大,造成系统连“作假”的机会也没有,只有给出

WAIT.....

C:Drive error

Press <F1> to Resume

及“INVALID DRIVE SPECIFICATION”的出错提示信息了。只要将磁盘类型设置成正确值之后,便可以排除此故障。

故障现象四:一台微机在自检完毕之后,屏幕上显示:

DRIVE NOT READY ERROR

INSERT BOOT DISKETTE IN A:

PRESS ANY KEY WHEN READY

故障分析及排除:

根据出错提示信息的指示,将 DOS 盘插入 A 驱之后,按回车键,系统可以正常引导,但是不能转到 C 盘上工作,如果对 C 盘操作时,屏幕显示:

INVALID DRIVE SPECIFICATION

出错提示信息,说明系统不承认 C 盘的存在。

重新引导系统在提示下键入键进入 SETUP 界面选择“STANDARD CMOS SETUP”一栏,发现硬盘 C 的参数类型为“NOT INSTALLED”非安装状态,将其设置成正确的硬盘类型,存入到 CMOS 中后,系统便恢复正常了。

故障原因:

如果不是人为设置错误或误操作造成的话,当出现此类故障时,一般情况下都是由于主机板 PARTY 电池失效造成的,该电池一般为 3.6V 锂电池,是可充式的,它的作用是在系统市电断掉的情况下,用来保存 CMOS 中设置的系统参数,以提供正常的服务,由于电池失效,致使所有 CMOS 中数据丢失,C 盘参数成为“NOT INSTALLED”状

态,系统在自检时不承认 C 盘的存在,故而出现故障。

故障现象五:一台 CHOST 386-SX/33微机开机之后,大约等待约30秒种的检测硬盘工作之后屏幕显示:

C:DRIVE FAILURE
RUN SETUP UTILITY
PRESS <F1> TO RESUME

系统不能正常引导。

故障分析及排除:

根据屏幕上给出的出错提示信息,按下〈F1〉键之后,屏幕接着显示:

DRIVE NOT READY ERROR
INSERT BOOT DISKETTE IN A
PRESS ANY KEY WHEN READY

将软盘插入 A 软驱之后,系统引导成功,但不能转到 C 盘上工作,系统不承认硬盘的存在。

此类故障也是由于 CMOS 中硬盘参数设置错误造成的,进入 SETUP 界面,查看其参数,显示如下:

Hard Disk c:

| | Cyln | Head | WPcom | LZone | Sect | Size |
|----------|------|------|-------|-------|------|------|
| TYPE 47: | 0 | 0 | 0 | 0 | 0 | 0 |

CHOST 386-SX/33微机使用的硬盘类型为“47”，即用户自定义型，从上面可以看出，其各项参数值均为“0”，将其改成正常值：

Hard Disk c:

| | Cyln | Head | WPcom | LZone | Sect | Size |
|----------|------|------|-------|-------|------|------|
| TYPE 47: | 530 | 8 | 65535 | 530 | 39 | 81MB |

[illegible]

存入 CMOS 之后,系统便可以正常引导系统了。(自定义类型的磁盘参数,用户需要记住,以防止 SETUP CMOS 信息遭到破坏时,以备恢复,如果将参数忘记,可以查阅硬盘说明书来配置安装,或将硬盘装在 386 以上机器上运行 SETUP 选择“AUTO DETECT HARD DISK”栏检测该硬盘的正确参数值)。

故障原因:

硬盘自定义类型参数变成全为“0”状态的原因可能由以下几点造成：

(1)主机板上锂电池失效,致使 SETUP 信息丢失所致

(2) 机器正在工作的过程之中, 突然受到干扰 (比如突然掉电) 所致

(3) 机器老化、性能不稳定所致

由于以上三条原因,便可能造成参数丢失成全“0”态,系统在引导时,便寻找不到类型为“47”的硬盘的正确参数而出现故障了。(注:当硬盘为自定义“47”类型时,一般情况下,自定义参数不存在或定义出错等时均会出现本故障现象,在这里不在叙述)。

通过上面的几例可以看出,CMOS 中磁盘参数类型的设置出错或不正确,都将直接影响到系统的正常工作,故障的现象也是多种多样的,用户在接到微机此类故障时,最好首先查看 CMOS 中磁盘参数的设置情况,而不要对机器硬件部分“大动肝火”,拆来换去,以免由于操作不慎,真正地给一个实际上不存在故障的微机带来了故障,造成一定的经济损失。

(本文收稿日期:1995.7.25)

(本文收稿日期:1995.7.25)

经营手册 设计参考 购买指南

欢迎订阅《电子产品世界》

《电子产品世界》是美国国际数据集团(IDG)和中国科技信息研究所共同创办,1994年与美国著名《电子设计》(ED)杂志结盟,是集高新电子产品及商情于一体的大型电子月刊。其宗旨是向中国业内的决策人员、工程师、采购商提供国内外最新、最深入的电子技术、市场、产品、采购、经营管理等方面的报道。

主要栏目有：市场透视、市场纵横、市场调查、

产品观察、产品之窗、产品热点、ED 专文、设计思路、技术专题、采购指南、经营之道等。

《电子产品世界》月刊,采用国际开本大16开,每期128页,每月19日出刊,定价4元,全年48元。

邮发代号:82—552

在邮局订阅有困难者,可与(100038)北京复兴路15号821室《电子产品世界》发行部联系,电话:8515544—2814。

DOS 命令巧用好处大

河南信阳空军一航院科研中心[464000] 贾建平 于冬梅

摘要:本文给出了几个巧妙组合 DOS 命令完成特定功能的例子。通过这些例子不难看出,对 DOS 已有命令的巧妙使用,可以避免不必要的重复性开发工作。

在一些公开发行的刊物上刊登的用户自己编程开发的一些本可以通过简单的命令组合完成的功能的文章不难发现:许多 DOS 用户不求甚解忽略了 DOS 命令的正确使用。本文给出几个巧妙使用 DOS 功能的例子,希望能起到抛砖引玉的作用。

一、删除目录树的一目录枝

6.0 以下的低版本 DOS 未提供直接删除一子目录的命令,要删除一子目录,首先必须删除掉该目录下的所有子目录与文件,然后才能删除掉该子目录,这种删除方法不但繁琐而且极易出错。下面,我们给出使用批处理完成直接删除子目录的方法。

在 C:\DOS 下建立批处理文件 DIRECTOR.BAT

```
@CD %2
@ECHO Y|DEL .
@CD ..
@RD %2
```

在 C:\DOS 下建立批处理文件 TREERD.BAT

```
@ECHO OFF
DIR %1/S|FIND "Directory" |SORT/R>C:\C.TXT
ECHO EXIT>>C:\C.TXT
COMMAND<C:\C.TXT
DEL C:\C.TXT
CD\
ECHO ON
```

在提示符后键入 TREERD,如果 TREERD 后跟子目录名,则删除相应子目录;否则,删除当前目录。

二、使 DOS 文件管理命令作用于当前驱动器中所有文件

DOS 的文件管理命令一般一次只能作用于一个目录,而有时我们希望能对当前驱动器的所有目录下文件进行某一操作,如删除所有扩展名为 BAK 的文件。幸好 DOS 5 及以上版本提供了列出所有目录及文件的命令 DIR/S,下面我们就给出利用该命令来完成所要求功能的批处理文件 TOALL.BAT。

```
@ECHO OFF
IF "%1"==" " GOTO ERROR
ECHO @CD %2>C:\DOS\DIRECTOR.BAT
ECHO @%1 %2 %3 %4 %5 %6 %7 %8 %9>>C:\DOS\DIRECTOR.BAT
DIR \S|FIND "Directory">C:\C.TXT
ECHO EXIT>>C:\C.TXT
COMMAND<C:\C.TXT
CD\
DEL C:\C.TXT
DEL C:\DOS\DIRECTOR.BAT
GOTO END
:ERROR
ECHO syntax: %0 cmdlist
:END
ECHO ON
```

它的使用方法很简单:若想删除当前驱动器下所有扩展名为 BAK 的文件,只需键入命令 TOALL DEL *.BAK 即可。

TOALL.BAT 的第3、4行在 C:\DOS 下建立一批处理文件 DIRECTOR.BAT(假设 PATH 语句包含目录 C:\DOS),其内容为

```
CD %2
```

TOALL 后面输入的命令(如:DEL *.BAK)第5、6行在根目录下建立一文本文件 C.TXT,该文本文件内容为当前驱动器下所有目录的清单及命令 EXIT。例如:

```
Directory of C:\
Directory of C:\DOS
Directory of C:\WPS
Directory of C:\WPS\JJJ
EXIT
```

第7行再次调用 command.com,并把 C.TXT 的每一行都作为一命令执行,这样就可以对每一目录进行 DIRECTOR.BAT 的操作,也就达到了使命令作用于所有目录的目的。

汉化 AutoCAD 10.0版配用 Q387的经验

刘 兵

目前 AutoCAD 得到了迅速的推广和应用,如今已有了汉化的 AutoCAD 13.0版本。AutoCAD 13.0版本要求机器的内存大。同时其所占的软盘也多。本人手上有汉化的 AutoCAD 10.0版本,共两张高密盘,安装在 ALR SX—386机上,其内存为 2M(基本内存 640K,扩展内存为 1820K)。在此机上运行时总会出现提示:浮点协处理器未装,便退出,拒绝执行 ACAD(注:我们学校已将浮点处理器给拿掉了)。后来经多方努力找到了 EM87和 Q387等一系列的仿真 80387浮点协处理器的软件,即通常人们常说的:以软件代替硬件的方法。后试用西文 AutoCAD 10.0版,(同样的情况,即都要求装浮点协处理器 80387,否则便拒绝执行 ACAD),先运行仿真 80387的浮点协处理器软件,再运行西文 AutoCAD 10.0版。结果是成功的,绘图效果也不错。

但用汉化的 AutoCAD 10.0版却会死机。经过分析汉化的 AutoCAD 中的 CAD.BAT 文件,发现它用了汉字驻留程度 Hlib.com 和 HzKey.com 文件,是否这与仿真浮点协处理器所占的高端内存有冲突呢?于是试着删去了 HzKey.com 及其它一些文件,重新启动,终于成功了,出现了汉字菜单和汉字命令提示,其绘图效果和西文状态一样。

我很想把这次成功的经历告诉大家,因为这对计算机图形辅助设计入门很有帮助。另外,说明一下 EM87和 Q387经实践对比,发现 EM87比 Q387应用更方便,且对机型和 DOS 的版本要求更低。仿真协处理器软件应用范围很广泛,不仅仅用在 AutoCAD 10.0上,对 AutoCAD 13.0版也同样适用,对 3DS 等软件也同样。(收稿日期:1995.6.21)

(接上页)三、模拟 UNIX 的垃圾箱

在删除 DOS 系统中的文件后,如果对硬盘进行了操作(比如存储文件)就可能不能再找回被删除的文件。其实,我们可以模拟 UNIX 的垃圾箱,即把被删除的文件保存到一个专门目录中。这样就可以给用户更多的挽救错误的机会。

首先在 C 驱根目录下建立目录 DUMPSTER。

然后,在 AUTOEXEC.BAT 中增加

```
ECHO Y|DEL C:\DUMPSTER
```

```
DOSKEY DEL=COPY %1 C:\DUMPSTER
$T DEL %1。
```

此后,每次用户使用 DEL 删除的文件都被转移到 DUMPSTER 下。

每次开机时,系统会自动删除垃圾箱中的文件。用户也可以用 DEL 命令删除目录 DUMPSTER 下文件,而被删除的文件将不会再出现在 DUMPSTER 目录下。

四、功能键的使用

DOS 提供了定义功能键的机会。我们可以把较常用命令以功能键定义,从而节省击键的时间。例如在 AUTOEXEC 中加入 prompt \$e[0;59;"c:\dos\debug";13p \$e[0;60;"dir ,/p";13p \$P \$G 并在 CONFIG.SYS 中加入 DEVICE=C:\DOS\ANSI.SYS(必须加入,否则定义不起作用),此后

每次启动计算机后,在 DOS 提示符下按 F1键都运行 C:\DOS\DEBUG 而按 F2键都表示输入了命令 dir ,/p。

五、DOSKEY 的使用

DOS 提供了 DOSKEY 宏定义,但很少有人注重加以巧妙地使用。下面,我们给出几例利用 DOSKEY 宏定义定义命令的例子,供广大用户、朋友参考。

1. 删除当前目录下除给出文件之外的所有文件

```
DOSKEY SDEL=ATTRIB +R %1 $T ECHO Y $B DEL *.* $T ATTRIB -R %1
```

该宏定义定义好后,如果在提示符下键入 SDEL *.WPS,将删除当前目录下所有除扩展名为 WPS 之外的文件。

2. DEL 只能接收一个文件名,下面的宏定义命令功能与 DEL 相似但可接收多个被删除文件的文件名

```
DOSKEY MDEL=FOR %F IN (%*) DO DEL %F
```

DOSKEY 较批处理执行速度快,对 DOSKEY 的恰当使用可为用户带来许多便利。当然, DOSKEY 也有诸多限制(如每个定义行受输入字符个数的限制),请参阅手册或在线帮助。

(本文收稿日期:1995.5.9)

口令动态输入的实现

河南信阳空军一航院科研中心[464000] 贾建平 陈剑萍

许多应用软件为了保密及限制使用者的权限都使用了口令字方式。然而,一些公开场合使用的软件(如:银行营业、财会报账、证券交易等)却存在着窃密者通过多次观察操作者的动作来获取口令的可能性。这儿,我们给出了一种口令输入动态变化的方法以对抗这种窃密行为。

口令动态变化主要有三种功能构成:1、每次进入时,重新随机地排列口令字各字符以作为本次口令。这样,每次进入尽管使用的是同一口令字,但要求的输入字符顺序却不同,增加了通过观察操作者动作获取口令的难度。也可让用户自己按排字符输入顺序,这样尽管增加了输入的时间,却大大提高了保密性,因为即使窃密者通过观察知道了口令

字的字符,但每次用户随机的输入使得窃密者很难观察到口令字各字符的排列顺序,即使是在获知口令字各字符的情况下,窃密者攻击成功的概率也仅为1/8! (后一种方法尽管需让用户首先输入8个数字的排列顺序,再输入按此顺序排列的新口令,但编程较为容易,故本文未给出具体子过程);2、记忆错误口令输入的日期、时间及所使用的错误口令,以便于提取窃密者的线索及进行必要的口令更换;3、每次进入向操作员报告此前发生的窃密事件。

下面就是采用 FoxBase+ 2.1实现的口令动态输入子模块(其中,假设系统获取口令的子过程为 getpassword) TEST.PRG:

```
SET PROC TO TEST
SET CONS OFF
SET COLO TO N/W
DIME A(8),B(8)
DO PASSEQ * 随机地排列1-8这8个数字
INSEQ=STR(B(1),1)+STR(B(2),1)+
STR(B(3),1)+STR(B(4),1)
INSEQ=INSEQ+STR(B(5),1)+STR(B
(6),1)+STR(B(7),1)+STR(B(8),1)
@5,15 SAY ' 请按提示顺序输入口令(' +
INSEQ+'):'
ACCEPT TO KEY
PW=""
DO GETPASSWORD WITH PW
SET COLO TO
@5,0 CLEAR TO 5,79
USE FKEY
I=1
DO WHILE I<=8
  A(B(I))=SUBSTR(KEY,I,1)
  I=I+1
ENDDO
KEY=A(1)+A(2)+A(3)+A(4)+A(5)
+A(6)+A(7)+A(8)
** 输入的口令非动态变化的口令,记忆该
口令及发生的时间并退出 **
IF KEY<>PW
  CDATE=DATE()
  CTIME=TIME()
  APPE BLAN
  REPL FKEY WITH KEY,FDATE WITH
  CDATE,FTIME WITH CTIME
  USE
  QUIT
```

```
ENDIF
SET CONS ON
IF RECC(<>0
  ?CHR(7)+CHR(7)+CHR(7)
GO TOP
DO WHILE .NOT. EOF()
  SET COLO TO
  @10,30 CLEAR TO 15,60
  SET COLO TO GR+/W
  @10,30 SAY FDATE
  @11,30 SAY FTIME
  @12,30 SAY ' 系统出现非法操作 '
  @13,30 SAY " 所使用口令为:" +
  FKEY
  SKIP
  IF .NOT. EOF()
    @15,30 SAY " 读下一非法口令(Y/
  N)"
  COND='Y'
  @15,49 GET COND
  READ
  IF UPPER(COND)='N'
    EXIT
  ENDIF
ENDIF
ENDDO
ENDIF
USE
SET COLO TO
SET PROC TO
RETURN
** 随机地排列1-8这几个数字 **
PROC PASSEQ
```

```
I=1
DO WHILE I<=8
  A(I)=I
  I=I+1
ENDDO
I=8
DO WHILE I>=1
  J=MOD(RND(),I)
  IF J=0
    J=I
  ENDIF
  B(I)=A(J)
  A(J)=A(I)
  I=I-1
ENDDO
RETURN
***** 产生一随机数 *****
PROC RND
NUM1=TIME(1)
NUM2=VAL(SUBSTR(DTOC(DATE()),
1,2))*100+VAL(SUBSTR(DTOC(DATE
()),4,2))
RETURN INT(NUM2*(VAL(SUBSTR
(NUM1,7,2))*100+VAL(SUBSTR
(NUM1,10,2))))
FKEY 库结构为:
字段名称 字段类型 长度 小数
1 FKEY 字符型 8
2 FTIME 字符型 8
3 FDATE 日期型 8
** 记录总长度 ** 25
```

(本文收稿日期:1995.5.9)

整理软磁盘经验谈

哈尔滨量具刃具厂计算中心 [150040]

王杰民

由于温度、湿度、灰尘、磁场、磁粉脱落、介质先天性缺陷、介质老化、操作失误等因素的影响,软磁盘使用一段时间后可能会丢失信息。软盘上文件及子目录频繁复制、删除,导致软盘上有许多文件分配的簇块十分零乱,当需要读取这些文件信息时,则要从多个簇块调出,这样就造成磁头的来回寻道读取,使系统的运行时间增加,同时也加快了软盘磁头的磨损。另外,文件存放的簇块太零碎,也加大了对误删文件进行恢复操作的困难。因此有必要对这些文件及数据的存储区进行归并处理。软磁盘工作的好坏直接关系到一个计算机系统的使用效率,对软磁盘文件的整理工作,可在对磁盘

文件做了较大的变动或用户认为必要的时候进行,至少年终时整理一次;反之,也应把硬盘上数据和程序及时地复制到软盘上,当硬盘有故障或误操作时不会造成大的损失。总之,硬盘与软盘应互为备份。可是,软磁盘整理容易被忽视,关于单用户 DOS 系统下软盘整理的文献报导亦甚少见。

磁盘整理的目的:①加强信号;②消除碎块空间;③重排文件顺序;④清除计算机病毒;⑤剔除坏磁盘。

磁盘整理的步骤:

1. 用 CHKDSK 命令检查软磁盘的完好性,坏盘不能再使用。

如果盘上有丢失的链簇,会报告:

Errors found, F parameter not specified.

XXX lost clusters found in XX chains.

convert lost chains to files (Y/N)?

这丢失的簇链是个陷阱,哪个文件赶上,哪个文件就会被剪裁,从而丢失内容。碰到这种情况,应将数据复制到其它盘上。至于这张丢失链簇的盘,应格式化后再用。

2. 检测、消除计算机病毒。

3. 用 COPY *.* CON 命令浏览全部文件内容,然后合并文件或删除无用文件。合并文件可用下述二条命令之一:

COPY 文件1+文件2(把文件2附到文件1之尾)

COPY 文件1+文件2+文件3 文件4

(把前3个文件依次连接后生成一个新文件4)

4. 用 PCTOOLS 的 SORT 功能将软盘文件排序。

可以按文件名、按扩展名、按文件长度、按文件建立日期和时间。可以按升序排列,也可以按降序排列。按 U 键确认修改,再按<ESC>键和 Y 键退出 PCTOOLS。文件经排序后查找容易,尤其盘上文件多时更是如此。排序前汉字文件名最好改成英文文件名,因为汉字文件名查找慢,且易出错。建议将系统文件、只读文件及其它不需要修改或修改后文件长度不发生变化的文件置于磁盘前部,而将常需修改或正在编辑的文件置于磁盘较后的位置。

存放系统软件和语言处理程序的软盘不必整理盘片空间和文件排列顺序,只需用 COPYWRITE 万能复制程序从同一个驱动器读出和复制回来即可,如盘片有问题,它会报告出错信息。

5. 在硬盘上建立一个子目录,将文件盘内容用 COPY *.* C: 拷入该子目录。

6. 用 COPY *.* A: 命令将硬盘子目录内容复制到已格式化的空软盘上。经过这样复制之后,软盘上的文件就存放紧凑了,能减少磁盘访问时间,加快程序运行速度。

7. 文本文件可用 TYPE 命令抽查软盘内容,如无问题,用 DIR>PRN 命令打印软盘文件目录,放入软盘保护封套内。此举的好处是,不用读盘即可随时查阅软盘目录。

8. 将软盘防写口贴上写保护标签,到此整理完毕。建议为每一盒每一片磁盘编号,以便索取。同一类性质的文件最好存放在同一盒或同一片盘中。

下面推荐几种工具软件,希望你整理软磁盘有帮助。

* KILL-70(杀除病毒)

该软件可以处理目前国内出现的各类病毒。它对已感染病毒的系统 and 文件进行检测和可靠的恢复,能准确地指出所感染病毒的名称及其位置。KILL-70 在设计上尽可能减少对用户的使用要求,对各种复杂情况的判定和处理均自动完成,并在屏幕上给出清晰明了的说明。

* Scan106/Clean106(扫描病毒/清除病毒)

该软件是检查及清除病毒的能手,可查出2149种病毒和所使用的软件技术及其属性。Scan检查并剔除病毒,MDISK 修理被病毒感染的主引导记录。对新的病毒,用VIRUSCAN可以找到并用CLEAN-UP删除。scan106有37种选项以满足用户的各种需要。与该软件功能相似的扫描病毒/清除病毒软件还有Scan112/Clean112。

* NORTON ANTI-VIRUS (诺顿反病毒软件)

诺顿系列软件以处理磁盘问题见长。其中NORTON ANTIVIRUS支持DOS平台和WINDOWS平台,它以3种方式驻留内存,最小方式只需占用1K内存。该软件能够防止正在运行的程序被感染,它还侦查根目录区,当被感染的文件从软盘向硬盘安装时,就能被发现并被消灭。

* NOVI(反病毒软件)

NOVI采用了先进的反病毒技术,是反病毒软件的换代产品。它不仅可以对已知病毒,还可以对未知病毒进行预防。NOVI的主要功能有:1.自动装入,后台操作;2.自动诊断及修复;3.自动装入磁盘根目录;4.下拉菜单;5.支持网络;6.可设警报信息;7.实时帮助。NOVI使用简便,扫描40MB不超过30秒,驻留内存只占用4K,能驻留内存高端。

* CPAV 2.0 (反病毒软件)

CPAV 2版能够侦察2000余种病毒。Vsafe和Vwatch驻留内存,时刻监视着系统,如有可疑的病毒,Vsafe立即报警,或清除或重新启动系统。一旦软盘被病毒感染,可用CPAV将病毒清除。要使文件终生免疫,可选用Imummize。还有一选项可以帮助你制作应急系统盘,一旦系统被病毒感染无法启动时,可用此盘将系统恢复。

* Fastback Plus for Windows (快速备份)

该软件用于快速备份和快速恢复之用。它有多种备份方式:增量备份、差分备份、全备份复制、全备份删除等。也有多种恢复方式:全恢复、选择性恢复、复盖性恢复等。各种备份过程和恢复过程都可以实现全自动化。

* COPYWRIT (万能复制程序)

它是一个功能很强的高级解密复制程序,专门用来复制加密软件。它的功能远比MS-DOS的DISKCOPY强,它先逐道、逐扇区调查加密软件的格式,然后复制出与源盘一样的复制盘。一般复制程序只能复制40个磁道,而万能复制程序可以复制42个磁道,另外它严格检测盘片,并报告出错信息。

* ALL OF COPY (复制大全)

该软件可以一对多,大小盘对应复制,可读校验,写校验,格式化校验。复制速度极快,可以把一张磁盘上的所有数据作为一个图象存储起来。

* SC (单驱动器磁盘复制软件)

SC软件有如下特点:1.可自动识别与使用系统所配置的扩展存储器(Extended)或硬件支持的扩充存储器EMS,从而使复制磁盘的速度加快。当系统有2MB以上内存时,单块磁盘的复制可以在内存中直接完成,因此复制速度极快;2.程序可自动将硬盘的剩余空间作为虚拟内存使用,因此一次可读入多个磁盘上的数据,从而简化了多个磁盘的复制操作;3.在复制磁盘的任何时间都可以按CTRL+C键或ESC键中断操作。

* DUPLICATOR (高效磁盘复制软件)

该软件的特点是:1.一次可将读入的单个磁盘上的数据复制到多个目标磁盘上;2.自动识别与使用系统所配置的扩展存储器(XMS)或扩充存储器(EMS),从而使复制磁盘的速度加快;3.磁盘复制中的各种参数设置采用全屏幕菜单方式,设置方法也简单到只需按光标键或回车键与Y和N两个键,因此更改参数非常方便。

* HD-COPY 1.7(高密盘格式化及复制软件)

该软件有以下特点:1.速度更快:比DUP等复制工具都快;还有一个FAT-selection,当它打开时,只对存有数据的磁道进行读写,不对空磁道浪费时间;2.更安全可靠:safe模式写或格式化前向用户询问;3.自动复制:RISK模式能自动识别驱动器门的开关,插入盘后无需按键即可自动复制;4.磁盘增容:对高密盘可进行10种格式化,可将1.2M盘格式化到1.476M,将1.44M盘格式化到1.722M;5.可5"盘和3"盘互相复制,可只读一次复制多张软盘;6.可将软盘镜像存储在硬盘上,也可将硬盘上的软盘镜像恢复到软盘上;7.可恢复零磁道坏;8.口令加密。

* Lock 93 (超级加密软件)

这是优秀的加密软件之一,在以下几点上有突破:1.在486/33M机及操作系统DOS 6.0上可以使用;2.高可靠性:加密子盘的制作和判读采用更可靠的思想 and 程序制作方法,采用更先进的反复制和动态反跟踪机制;3.高适应性:对高密、低密、3.5"盘均可识别并完成子盘的制作,从PC机到486机都可使用;4.多功能性:生成的加密子盘具有向硬盘安装的功能;5.面向用户,菜单驱动,界面友好,操作简单;6.成批制作:一次读入被加密软件,成批制作加密成品盘。

(本文收稿日期:1995.4.1)

含原字符的阴影处理技巧

随州市第一人民医院[441300] 李红胜

摘 要:本文介绍用 C 语言调用 DOS 系统资源,来实现含原字符的阴影的处理技巧。

关键字:原字符 阴影

在我们所接触的一些商品软件中,有些屏幕处理相当漂亮,比如立体窗口。其立体窗口的黑色阴影并非是简单地填充黑色,而是含有阴影覆盖处的原来字符,这样的阴影比简单地填充黑色要漂亮得多,立体感更强。

我们用 Foxbase、Clipper 或 Foxpro 设计应用软件时,由于系统本身的局限,所以无法达到含原字符的阴影的处理效果,这使很多使用 Foxbase、Clipper 或 Foxpro 作为开发工具的程序设计人员感到遗憾。

但是,如果我们利用 C 语言调用 DOS 系统资源,则可以轻而易举实现这一功能。

我们知道,屏幕上要实现阴影的方法是:在需要开窗口的位置上错位,设置阴影颜色后清屏,一般以黑色作为阴影色。

那么,含原字符的阴影是怎样实现的呢?屏幕上每个字符,在显示内存中是用两个字节来存储的,其中低位字节存放的是字符本身,高位字节存放的是字符的属性。如果能够获取阴影部分的字符,并且按阴影属性重新在屏幕上写一遍,则含原字符的阴影即可实现了。

DOS 中 ROM-BIOS 的 10H 中断是视频输入/输出,当 AH 为 8 时,可以读取光标处的字符及属性;当 AH 为 9 时,可以向光标处写字符及属性。用 C 语言编写一个读光标处字符及属性的函数 sread(),再编写一个向光标处写字符及属性的函数 swrite(),以上两个函数供主程序 main()调用,其中坐标处理及光标移动在主程序 main()中完成。具体请见所附源程序 SCHANG.C。

另附在 Foxbase 中可用的演示程序 SCH.PRG。

调用 SCHANG.EXE 的格式如下:

! SCHANG TOP LEFT BOTT RIGHT

参数说明:TOP 和 LEFT 为阴影部分左上角坐标;BOTT 和 RIGHT 为阴影部分右下角坐标。

```

* * * * *
* sch.prg 源程序
* 含原字符阴影处理
* FOXBASE 演试程序
* * * * *

set talk off
set stat off
set scor off
set esca off
* ----- *
if .not. file('sch. mem')
set colo to W+/b
clea
    i=1      && 以下双重循环为设置演试屏幕环境
    k=0
    do while i<=24
        j=40
        do while j<=79
            @ i,j say chr(180+k)
            k=iif(k<40,k+1,0)
            j=j+1
        enddo
        i=i+1
    enddo
save screen to cscr    && 存当前屏幕
save to sch all like cscr
else
    rest from sch
    rest screen from cscr
endif
* ----- *
kk=0
i=2
do while kk<>27
    ii=str(i,1,0)
    ! schang 8 20 18 61    && 调用阴影处理程序
    set colo to w+/&ii    && 设置窗口颜色
    @ 7,18 clea to 17,59    && 错位开窗口
    do while inkey()=0
    enddo

```

```

rest screen from cscr
kk=0
do while kk=0
kk=inkey()
enddo
i=iif(i<7,i+1,2)
enddo
*-----*
set colo to
set stat on
set scor on
set talk on
return

/* 环境: DOS6.20, AST386微机, UC DOS 希望汉字转行
如下程序用 Turbo C2.0编译运行成功。*/

```

SCHANGE.C 源程序

```

#include "dos.h"
#include "stdlib.h"
#include "conio.h"
main(int argc, char *argv[])
{ int left, top, right, bott, i, j;
  union scan {int c;
    char ch[2];
  } sc;
  if(argc<5) exit(0);
  /* 坐标处理 */
  top = atoi(argv[1])+1;
  left = atoi(argv[2])+1;
  bott = atoi(argv[3])+1;
  right = atoi(argv[4])+1;
  for(i=left; i<=right; i++)
  { gotoxy(i, bott); /* 移动光标 */
    sc.c=sread(); /* 读光标处字符及属性 */
    swrite(sc.ch[0]); /* 向光标处写字符及属性 */
  }
}

```

```

for(j=top; j<=bott; j++)
{ gotoxy(right-1, j); /* 移动光标 */
  sc.c=sread(); /* 读光标处字符及属性 */
  swrite(sc.ch[0]); /* 向光标处写字符及属性 */
  gotoxy(right, j); /* 移动光标 */
  sc.c=sread(); /* 读光标处字符及属性 */
  swrite(sc.ch[0]); /* 向光标处写字符及属性 */
}
}

```

/* 读光标处字符及属性函数 */

```

sread()
{ union REGS r;
  r.h.ah=8; /* AH=8时 */
  r.h.bh=0; /* 显示页面为0 */
  int86(0x10, &r, &r);
}

```

/* 向光标处写字符及属性函数 */

```

swrite(char zf)
{ union REGS r;
  r.h.ah=9; /* AH=9时 */
  r.h.bh=0; /* 显示页面为0 */
  r.x.cx=1; /* 在光标处写1次 */
  r.h.al=zf; /* 字符本身 */
  r.h.bl=LIGHTGRAY|BLACK*16;
  /* 阴影属性 */
  int86(0x10, &r, &r);
}

```

参考文献:

- [1][美]Herbert Schildt 著, 郭兴社、戴建鹏编译, 《C语言大全》, 电子工业出版社, 1992
- [2]廖彬山、甘登岱、刘有军主编, 《PC中断调用大全》, 科学技术文献出版社, 1993 (本文收稿日期: 1995.3.29)



欢迎
订阅

科技与经济结合的纽带
科研成果转化的金桥

《适用技术市场》月刊

《适用技术市场》是国家科委技术市场管理办公室与湖北省科学技术委员会联合主办的全国唯一的国家级权威性技术市场月刊。她提供的信息权威, 介绍的技术成果真实, 传递的项目可靠。解决企业的技术观念题及时, 是企业、农民寻找项目开

发新产品的良师益友, 也是科研单位、管理部门及科技开发机构的必备刊物。

定 价: 每册2.50, 全年30元 (含邮费)。

邮发代号: 38-142

订 购 处: 全国各地邮局均可订阅, 也可直接汇款至本刊办理订购手续。

联系地址: 武汉市武昌洪山路2号湖北科教大厦D座15楼《适用技术市场》杂志社。

邮政编码: 430071

电 话: (027) 7822960 7824882

用@<行1,列1> CLEAR TO <行2,列2>语句产生动画效果

闰 锦

广大的微机用户使用 FoxPro 2.5 for Windows 数据库管理系统编写了不少的应用系统。本文描述的动态画面程序,可以使程序封面显得生动,别致。而且程序主要用@<行1,列1> CLEAR TO <行2,列2>语句实现,动画模块只有23句,简单实用,通用性好,易于实现。动画程序及调用它的示例程序如下:

```
*****
*      PROGRAM: 调用动画模块示例程序      *
*      LAST MODIFIED DATE: 05/09/95        *
*      FOXPRO VERSION: 2.5                  *
*****
clear
define window yan from 0,0 to 40,107 none in desktop
activate window yan
STORE "用 CLEAR" TO YAN1
STORE "语句" TO YAN2
STORE "实现" TO YAN3
STORE "简单" TO YAN4
STORE "动画" TO YAN5
STORE "功能" TO YAN6
set color to w/rb+
clear
length=20
endpoint=48
startline=7
endline=9
do blockmove with length,endpoint,startline,endpoint
@7,32 say YAN1 font' 中黑体 ',20
endpoint=65
startline=15
endline=17
do blockmove with length,endpoint,startline,endpoint
@15,51 say YAN3 font' 中黑体 ',20
endpoint=32
do blockmove with length,endpoint,startline,endpoint
@15,16 say YAN2 font' 中黑体 ',20
endpoint=72
startline=23
endline=25
do blockmove with length,endpoint,startline,endpoint
@23,58 say YAN6 font' 中黑体 ',20
endpoint=48
```

```
do blockmove with length,endpoint,startline,endpoint
@23,34 say YAN5 font' 中黑体 ',20
endpoint=23
do blockmove with length,endpoint,startline,endpoint
@23,11 say YAN4 font' 中黑体 ',20
WAIT WINDOW "请按任意键继续!"
clear windows
*****
*      PROGRAM: 用 CLEAR 实现动画          *
*      LAST MODIFIED DATE: 05/09/95        *
*      FOXPRO VERSION: 2.5                  *
*****
parameter length , endpoint , startline , endpoint
i = 0
do while .t.
do while i <= length
set color to gr+/g
@startline+1,j+1 clear to endpoint+1,i+1
set color to gr+/r+
@startline , j clear to endpoint , i
i = i+1
enddo
j = j + 1
i = i + 1
set color to gr+/rb+
@startline,0 clear to endpoint+1,j+1
set color to gr+/g
@endline + 1 , j + 1 clear to endpoint + 1 , i
@startline + 1 , i + 1 clear to endpoint + 1 , i + 1
set color to gr+/r+
@startline , j clear to endpoint , i
if i = endpoint
exit
endif
enddo
```

在程序运行时首先清屏,之后六个带阴影(阴影色可随意调节)的小矩形块依次从屏幕左侧出现并扫动至各自相应的位置上。

如果把动画模块中第13行的“rb”改为不同于背景的颜色时,可以造成“慧星”效果,即有长尾的拖动。通过变动前景和背景色,仿此可以编制出其它丰富多彩的类似动画的图面,读者不妨试之。

(本文收稿日期:1995.5.22)

“参数库”菜单实现方法及应用

江西拖拉机发动机厂 黄焕如 湖南师范大学物理系 王 玲

内容提要:本文介绍了一种将菜单内容及其相关参数设置在“参数库”文件中的通用菜单实现方法,并结合实例提出了菜单文件加密的应用。

关键字:菜单文件、参数、越界处理、涂色显示、加密

在开发计算机应用程序中,菜单设计是很重要的一环,程序设计者通常都习惯于制作一个较为通用的菜单实用程序,将应用程序和菜单内容及其相关参数分开,即便调换一个应用程序,也仅仅需要修改菜单内容和相关参数,给程序设计者带来很大的便利。本文将结合一个实例,详述这类通用菜单的实现方法。

一、菜单内容及其相关参数的设置和读取

要将应用程序和菜单内容及其相关参数分开,可以将菜单内容及其相关参数设置在一个单独的文件中,该文件可以是纯粹的 ASCII 码文本,便于调用和修改。为了加强菜单的通用性,如何设置相关参数将直接关系到菜单实现的通用程度,首先考虑到应用程序可能调用多个菜单,应设立菜单号;为便于程序读取菜单内容和进行越界判断,应设立本行菜单数量;菜单应允许在任何行内显示,应设立行号;对于菜单彩色背景,应设立显示列范围,即起始列和终止列;菜单本身字符和背景色彩应根据程序要求设置,应设立字符前景色和字符背景色。例如用字处理软件建立文件 MENU.TXT:

menu1 8 0 1 78 4 3 S_数据输入 X_数据修改 C_数据查询 U_数据删除 J_统计 D_打印报表 Z_作图 T_退出

menu2 3 1 20 58 3 4 L_初始化 B_备份数据 T_退出

以第一行为例说明如下:MENU1为菜单号;8为该行菜单数量,即8个菜单项目;0为菜单显示在屏幕的第0行,即首行;1—78为菜单背景列范围;4和3为菜单字符前景和背景颜色;其后依次为菜单内容,每项参数和子菜单必须用空格分开。颜色的定义通常为:

| | | | |
|---|---|---|---|
| 0 | 黑 | 1 | 蓝 |
| 2 | 绿 | 3 | 青 |
| 4 | 红 | 5 | 紫 |
| 6 | 棕 | 7 | 白 |

| | | | |
|----|----|----|----|
| 8 | 深灰 | 9 | 浅蓝 |
| 10 | 浅绿 | 11 | 浅青 |
| 12 | 浅红 | 13 | 浅紫 |
| 14 | 黄 | 15 | 亮白 |

第二行表示为第二个菜单,其参数定义和第一行完全相同。

菜单内容字符串结构如下定义:

```
typedef struct tmenue
```

```
{ char str[20];
```

```
int row;
```

```
int col; }TMENUE;
```

读取菜单内容及其相关参数程序:

```
FILE * pf;
```

```
int numb, line3, col1, co2, zcof, zcob, i;
```

```
int z_col, z_ul;
```

```
char cbuf[30], * fname;
```

```
TMENUE menue[10], * pm;
```

```
pf=fopen("menu.txt", "r");
```

```
/* 打开菜单内容及其相关参数文件 */
```

```
while (!feof(pf))
```

```
{
```

```
fscanf(pf, "%s %d %d %d %d %d %d",
```

```
cbuf, &numb, &line3, &col1, &co2, &zcof, &zcob);
```

```
for (i=0; i<numb; i++)
```

```
fscanf(pf, "%s", menue[i].str);
```

```
/* 读各项子菜单 */
```

```
if (strcmp(cbuf, fname) == 0)
```

```
/* 判断菜单号是否正确 */
```

```
break;
```

```
}
```

```
fclose(pf);
```

二、菜单内容对中显示和相关参数越界处理

通用菜单实现就必须对有关子菜单个数、各子菜单字符串长度等参数进行规范化处理,以便能自动地将菜单在屏幕指定行中对中显示,同时为了能避免由于菜单内容及其相关参数设置不合理而破坏屏幕正常显示,如各子菜单字符总长度超过屏幕列

宽等原因,应进行相应的越界处理。

```
z_col=col+(co2-col)/2;
z_ul=0;
for (i=0;i<numb;i++)
    z_ul=z_ul+strlen(menu[i].str);
    /* 各子菜单总长度 */
if ((z_ul+numb+1)>(co2-col))
    { putchar(7); return(-1); } /* 越界则退出 */
_setcolor(zcob); /* 设置菜单背景 */
rectangle(_GFillInterior,col*8,line3*18,co2*8
+8,(line3+1)*18);
```

三、移动光标选择各子菜单涂色显示

通常光标停在当前子菜单时,该子菜单相对于其他子菜单应有一定程度的彩色反向显示,在设计时首先介绍调用的几个子函数:

置光标位置子函数执行 INT 10H 中断,页号为 BH=0,IL 和 DL 分别为行和列。

```
curplace(row,col)
int row,col;
{ union REGS in;
in.h.ah=0x02; in.h.bh=0;
in.h.dh=row; in.h.dl=col;
col_val=col; row_val=row;
int86(0x10,&in,&in); }
```

设置字符前景和背景子函数中的 TEST—NA 初始值为0。

```
chacolor(fcolor,bcolor)
int fcolor,bcolor;
{ int col;
col=bcolor; col&=0x07;
col<=4;
text_na=fcolor;
text_na&=0x0f;
text_na|=col; }
```

显示字符串子函数在当前光标处逐个写字符,直到字符串结束,该子函数调用了置光标位置子函数和写一字符子函数。其中写一字符子函数执行了 INT 10H 中断,其中 AL=ASCII 码,BH=显示页,TEXT—NA=属性,CX=字符数。

```
writestr(glob,row,col)
char *glob;
int row,col;
{ int ichar; char cx;
while(*glob!='\0')
{ cx=*glob;
ichar=(int)cx;
curplace(row,col);
writechar(ichar);
glob++; col++;
if (col>79) col=col-80; }
```

```
}
writechar(Char) /* 写一字符 */
int Char;
{ union REGS in;
in.h.ah=0x09; in.h.al=Char;
in.h.bh=0; in.h.bl=text_na;
in.x.cx=1;
int86(0x10,&in,&in); }
```

```
fill_menu(row1,row2,place,fcolor,bcolor)
int row1,row2; /* 显示子菜单 */
TMENU *place;
int fcolor,bcolor;
{
int len,fcol,lcol;
chacolor(fcolor,bcolor);
len=strlen(place->str);
fcol=place->col;
lcol=fcol+len-1;
writestr(place->str,place->row,place->col);
}
```

只要将当前子菜单的前景色改成其他子菜单的背景色,背景色改成其他子菜单的前景色,就能达到彩色反向显示的目的。

```
fill_menu(row1,row2,pm,bcolor,fcolor);
pm=menu; /* 填当前子菜单 */
for (i=0;i<mnum;i++) /* 填其他子菜单 */
{ if (i!=pnum)
fill_menu(row1,row2,pm,fcolor,bcolor);
pm++; }
```

四、子菜单代码按键选单

一般采用移动光标键然后敲回车键的方法来选子菜单,但是有的用户习惯于敲子菜单的代码,即各子菜单的首个英文字母来选择子菜单。子函数 readcode() 执行了 INT 16H 键盘中断,返回时 AH=扫描码,AL=ASCII 码。

```
int readcode() /* 读一字符 */
{ union REGS in,ou;
int ix;
in.h.ah=0;
int86(0x16,&in,&ou);
ix=ou.x.ax; /* 高8位扫描码 低8位 ASCII 码 */
return(ix); }
```

将读得的键取出高八位扫描码和低八位 ASCII 码,经过适当运算后判断,如是某一子菜单的首字母,则将光标定在该子菜单位置并且使其反向显示。

```
lword=0x6d;
while (lword!=0x0d) /* 不敲回车键 */
```

```

{ keytest=kbhit();
  if (keytest!=0)
  { curplace(0,0);
    ix=readcode(); /* 读键 */
    lword=ix&0x00ff;
    hword=ix&0xff00;
    hword>>=8;
    if (lword==0&&hword==0x4b) /* <-键 */
    { if (pnum>0) pnum--;
      else if (pnum==0) pnum=mnum-1; }
    else if (lword==0&&hword==0x4d) /* ->键 */
    { if (pnum<mnum-1) pnum++;
      else if (pnum==mnum-1) pnum=0; }
    else if ((lword>=0x41&&lword<=0x5a)|| (lword>
=0x61&&lword<=0x7a))
    { if (lword>0x60)
      { lword1=lword-0x61; lword2=0x41+lword1; }
      else
      { lword1=lword-0x41; lword2=0x61+lword1; }
      pm=menue;
      cz='n';
      for (i=0;i<mnum&&cz=='n';i++)
      { cy=*(pm->str);
        iy=(int)cy;
        if (iy==lword1||iy==lword2)
        { cz='y'; pnum=i; }
        else pm++; }
    }
}

```

五、菜单内容及其相关参数的加密

众所周知,在生成或编辑 ASCII 码文本文件时,编辑软件将自动在文件末添写文件结束符“1A”,当调用文本文件时编辑软件一碰到“1A”时,就认为文件已到此为止,将不再搜寻“1A”以后的任何文本。

根据上述原理,巧妙地利用“1A”文件结束符,能在一定程度上保护菜单内容及其参数不会被修改。可使用手工方法插入“1A”,如用编辑软件写入(MENU. TXT);

123456menu1 8 0 1 78 4 3 S_数据输入 X_数据修改 (同前)

再利用 DEBUG 修改:

C>DEBUG MENU. TXT

-E105 1A

-W

-Q

也可以直接编制程序生成插入“1A”的文件:

```

char *ss="menu1 8 0 1 78 4 3 S_数据输入 X_数据修改
..... (同前)"

```

```

FILE *fp; int xx=0x1a;
int x1=0x0d,x2=0x0a;
fp=fopen("MENU. TXT","w");
fprintf(fp,"%s","12345");
fprintf(fp,"%c",xx);
fprintf(fp,"%s",ss);
fprintf(fp,"%c",x1);
fprintf(fp,"%c",x2);
fprintf(fp,"%c",xx);
printf("OK!");
fclose(fp);

```

显然暴露的部份仅仅是“12345”,其他信息都被隐藏起来了。调用时应先过滤掉“12345”,再调出菜单及其相关参数字符串。

fseek(fp,6L,0); /* 过滤掉12345和“1A” */

以上程序采用 MSC 7.0 编制的,稍加修改也适用于 Turbo C++。 (本文收稿日期:1995.4.13)

欢迎订阅《市场经济时报》

本报是国内最早以市场经济命名的,面向企业和会在国内外公开发行的综合性大报,具有大时空、大视野、高品质和实用性、操作性的特点,以对开4版(每周3报)的崭新面目向读者展示独特的风采。

本报邀请国内外著名经济学家、社会科学家、政府官员和企业界人士探讨我国市场经济进程中的热点及难点问题,将解开企业家、决策者和普通百姓在市场经济中的各种迷惘,她服务于企业,尽显企业家在市场经济大潮中的风采和英姿,向读者透露企业之子、巨商富贾的创业发展史;她同企业家共商“经营之道”,为您提供实用信息和市场预测。个性鲜明,雅俗共赏的红、蓝、绿、紫四个颜色的周末版,将为读者评说茫茫大千世界东西南北中千万种国情民情,讲述普通百姓人生苦辣酸甜咸五七八样过法活法,为繁忙劳作了一周的社会各界人士捧出一份精美的“周末精神快餐”。

本报传播市场经济发展的新消息,捕捉市场供求的新动态,评述市场变化的新趋势,反映消费者生产者的新呼声。

欢迎到邮局订阅,也可破月破季征订或直接向报社汇款。

单价:0.40元,月价:5.20元,季价:15.60元

半年:31.20元,全年定价:62.40元

邮发代号:11-77 联系人:冯爱莲

地址:长春市斯大林大街副111号 邮编:130021

开户行:长春工商银行自由大路办事处

帐号:144500-19

长度受限情况下 包含汉字字符串的打印方法

华中理工大学计算机系 金海 谢卫

摘要:本文介绍了一种简单实用的字符串打印方法,使得当需要打印的字符串中包含汉字信息,且报表栏的长度一定时,能够对字符串进行正常切割,所切的字符串不至于将一个汉字一分为二。

关键词:程序设计, FoxBase 语言, 报表打印

在使用 FoxBase 开发应用程序时,经常会遇到有关报表打印的问题。这时用户也许会碰到下面这种情况:报表栏的长度一定,而需要在该报表栏内打印的字符串却比报表栏长度要长。这时需要对字符串进行切割,将一部分内容移到下一行打印。如果这个需要打印的字符串中包含汉字信息,就存在一个如何切割字符串的问题,使得所切的字符串不至于将一个汉字一分为二。这时的关键问题就是要判断最后一个字符是否是汉字字符,我们编制了一个函数 HZPD 来判断在长度受限的情况下切割一个包含汉字的字符串是否正确。程序如下:

```

PARAMETER STRING
PRIVATE LENGTH, VALUE, POINTER
LENGTH=LEN(STRING)
IF LENGTH=0
    RETURN 0
ELSE
    POINTER=1
    VALUE=.T.
    DO WHILE POINTER<=LENGTH
        IF ASC(SUBSTR(STRING,POINTER,1))<=127
            VALUE=.NOT.VALUE
        ENDIF
        POINTER=POINTER+1
    ENDDO
    RETURN VALUE
ENDIF

```

该函数接收字符串变量 STRING,并且判断该字符串变量的最后一个字符是否是汉字字符。当最后一个字符不是半个汉字字符时(即切割正确时),该函数返回真值,否则返回假值。用户可以在自己的程序中按照下面的例子利用该函数:

```

BP=1
IF BP<=LEN(ZRZ)
    BA=SUBSTR(ZRZ,BP,8)
    IF .NOT. HZPD(BA)

```

```

        BA=SUBSTR(ZRZ,BP,7)+"
        BP=BP+7
    ELSE
        BP=BP+8
    ENDIF
ELSE
    BA=""
ENDIF

```

这里,我们需要打印字符串 ZRZ,变量 BP 是一个位置指针变量,用来指示当前打印的起始位置,BA 是在 ZRZ 变量当中按照固定长度(本例中为8)切割下来打印的字符串。用户可以采用以上的方法在长度受限的报表栏中分行打印长度超出报表栏长的字符串。

该程序简单实用,在我们研制的“QJS 通用档案管理系统”中应用效果良好,使打印出来的报表准确美观。

(本文收稿日期:1995.7.4)

广州中鸣公司严正声明

广州中鸣公司是 LED 大屏幕显示屏的专业设计和制作公司,为国内外用户设计和制作了大量的电子屏幕。作为广东领先大屏幕传播有限公司的股东之一,中鸣公司受领先公司的委托、设计、制作了当今第一块40平方米全彩色 LED 大屏幕电视屏。自今年四月十日开播至今,运作良好,深受国内外同行及客户的好评。

但是,不久前发现,有某家公司、个人和机构,带客户前往观看,妄称是他们设计、制作,蒙骗用户。为维护中鸣公司以及广大用户的利益,特此敬告用户,切勿上当受骗。如需设计制作电子屏幕,请来人来函与广州中鸣公司联系。

广州中鸣显示技术工程有限公司启