

ISSN 1001-3695

計算機應用研究

1992

3

APPLICATION RESEARCH OF COMPUTERS 《計算機應用研究》雜誌



《計算機應用研究》雜誌辦刊單位

四川省電子計算機應用研究中心	新疆電子計算中心
貴州省科委計算中心	甘肅省計算中心
安徽省計算中心	廣西計算中心
吉林省計算中心	山東省計算中心
河南省計算中心	青海省測試計算中心
四川省電子學會	雲南省電子計算中心

《計算機應用研究》雜誌社董事會

董事長：周寶渝
董 事：唐 珍 孫傳江 陸慰椿 馮德成
吳地興 鄭國基 陳建嶺
黎 蓉 閔長榮 黎瑞常

《計算機應用研究》雜誌編輯委員會

主任委員：張執謙
副主任委員：李澤民
委 員：賈洪鈞 曾光初 龔宇清 張國棟
羅海鵬 劉鐵軍 崔振遠 李文華
楊劍波 余 凱 劉啟茂 張湘金

1992 年第 3 期(總第 47 期)

出版日期：1992 年 5 月

本期責任編輯：張 鋼

計算機應用研究(雙月刊)

(公開發行)

國內統一刊號：CN51—1196

主 編：張執謙

副主編：李澤民

編輯出版：《計算機應用研究》雜誌社

通訊地址：成都市人民南路 4 段 11 號附 1 號

郵政編碼：610015

印 刷：新都一中印刷廠

訂 閱 處：全國各地郵局

總 發 行：成都市郵政局

郵發代號：62—68

云南省电子计算中心简介

云南省电子计算中心(又称云南省计算技术研究所),始建于1978年6月。是国家科学技术委员会和云南省人民政府投资兴建的一个从事计算机应用研究和技术服务的科研机构。

中心现有职工210人,其中大专毕业人员占90%以上,在183名高级工程师、工程师和程序设计人员中,具有硕士以上学位或曾经在国外学习、进修过的中青年科技人员约占三分之一左右。有一支专业结构、知识结构和年龄结构比较合理的专业技术队伍。

中心目前设置的从事研究与开发、服务与经营的机构主要有:大型计算机应用研究室、微型计算机应用研究室、软件开发研究室、应用基础及理论研究室、CAD技术研究室、智能化产品开发研究室、遥感技术应用研究室、情报资料室、计算机应用人材培训部、技术开发服务部(又称电脑中心)、电脑及电子仪器维修服务部、电梯工程部、科技彩印厂以及联办的深圳爱德威光电控制设备制造公司。

中心的主要任务是:1、作为云南省的科技信息处理中心,按国家科委和省科委的布置,实施全省科技信息系统的建设;2、作为云南省计算机应用研究、服务中心,面向全省提供数值计算、数据处理和自动控制方面的服务,承担为各有关部门开发、研制各类计算机应用系统和信息处理系统;3、作为云南省的软件开发中心,面向全省提供开发各种软件的软、硬件环境,开发研制各类软件工具和应用软件;4、作为云南省计算机应用人员培训中心,面向全省需要培训各类计算机应用人才,提供各种培训条件、普及计算机知识;5、作为云南省遥感技术应用研究中心,面向全省提供各类遥感卫星图像处理和技术服务,开展遥感技术的应用研究。

中心拥有较强的技术装备。从国外引进的大型计算机系统自运行以来,开机率和CPU利用率均名列全国同类机器的前茅;数台先进的SUN工作站上装有一批优秀的CAD软件包和其他软件;遥感图像处理系统和暗室作业设备为全省各方面的需要提供了高质量的卫星图像;小型计算机系统、实验机器人、多套微型计算机系统以及各类仪器设备和开发系统保证了各项任务的顺利完成。

至1990年底,共完成科研、开发项目230余项,在国际国内(包括国家、省、部级科技进步奖和其他类型奖)获奖项目30余项。为全省培训各类计算机应用人员近万名。1991年开展的科研、开发项目50余项,其中有国际合作研究项目、国家部委委托的研究项目、省的“八五”科技攻关项目及自然科学基金项目、青年科学基金项目以及各部门委托开发的项目。其内容包括各类信息系统(管理信息系统,办公自动化系统、决策支持系统、数据采集处理系统等)与自动控制系统及生产过程自动化的研制开发,软件工程及软件开发技术的研究,数据库技术(如数据库的自动跟踪、多介值数据库等等)的研究与开发。各类应用软件的开发、智能化产品(如智能化医疗设备、控制设备、微机施肥设备、智能化的粮油、烟叶收购设备等)的开发与研制等等。

地址:中国云南省昆明市教场东路28号

电话:54045 53358 邮编:650223

电报挂号:昆明3404 电传:64028 YSTEC CN

重 要 公 告 (一)

据新出版的《中国科技论文统计与分析》年度研究报告获悉:本刊《计算机应用研究》杂志已进入中国科技论文统计源。

《中国科技论文统计与分析》是国家科委下达给中国科学技术情报研究所的重要科研项目。1988年以来,该所经综合评定筛选,从国家正式批准的3052种科技期刊中,选出了1189种期刊作为中国科技论文统计与分析的统计源。通过两年实践,再次征求各有关方面意见,1991年的统计源选刊数量确定为1230种,其选刊标准是期刊学术水平,期刊编辑质量、发行范围、技术内容等多项指标。在1230种科技期刊中,计算技术与自动化领域有28种期刊,本刊《计算机应用研究》杂志榜上有名,选为中国科技论文统计源之一。这是本刊读者、作者、编者及各方互相支持、共同努力的结果。

为了全面统计、分析、掌握有关数据、信息,并利于存档、建库、索引,现本刊特发出以下公告:

1. 凡在本刊1991和1992年各期上发表有学术论文的作者,请尽快将其姓名、性别、出生年月、职务、职称、职业、专业、学位、工作单位、通讯地址、邮政编码等信息函告本刊,本刊不胜感谢,并将赠送一件纪念品。

2. 今后来稿,请作者务必专页注明包括上述内容在内的有关信息及寄稿日期;课题经费来源、全文字数、插图数量、编号及在文中的位置。中文标题须译为英文,标题下须附摘要及关键词。文末宜列出参考文献,并按《科学技术期刊编排规则》GB3179—82书写。

3. 来稿请用钢笔工整誊抄或用计算机打印并标注每页字数;图纸请用描图纸转绘或用激光照排纸打印,图上文字宜用激光照排字处理;程序清单宜打印成半栏(行宽7cm)或通栏(14cm宽),采用硫酸纸(即半透明描图纸)激光照排处理则更佳。外文宜用铅笔注明文种并采用印刷体书写、打印,切勿用草体。

4. 本刊不受理复写稿、复印稿或铅笔、圆珠笔誊抄稿。

5. 稿件寄出半年后若未见录用,请作者函询或自行处理。来稿一律不退,业务请自留底稿。

6. 稿件一经刊出,酌致稿酬。对时效性很强的稿件,本刊办理优先快发业务,请作者在稿件中注明。

7. 本刊拥有工商行政管理部门正式批准的广告经营许可证,竭诚欢迎广大客户来人来函联系广告业务。

8. 热忱欢迎广大读者、作者、订户、同行、同仁和领导同志为本刊献计献策,提出合理化建议和批评意见。

衷心感谢广大订户、读者、作者、各联办单位、友好同仁、上级主管部门、业务领导部门、出版印刷单位对本刊的垂青厚爱 and 给予的热情关照、大力支持,并顺祝大家万事如意!

重 要 公 告 (二)

为了更快地推动、促进我国计算机产业的迅猛发展,及时地为计算机科研、开发、生产等部门提供计算机应用行业的最新发展动向,加速高新尖端技术、最新软硬件的开发、移植、引进,及时地为广大读者及计算机爱好者奉献更多更新的计算机专业技术资料,并使众多作者、译者脱颖而出,《计算机应用研究》杂志社拟从一九九二年四月起开展优惠出版各类计算机专业技术资料业务,欢迎广大作、译者踊跃赐稿。具体出版业务欢迎来函来电来人商议。联系人:本刊编辑部张钢编辑(邮编610015)

《计算机应用研究》杂志社启

计算机应用研究 第9卷 第3期(总第47期)

目 录

软件篇

关系代数的分块自然联接运算.....	罗伟其(1)
关系数据库关系规范化支持系统.....	宾晓华(5)
数据库设计自动化中的一种最佳编码方案	李彦明、黄声烈(8)
GKD-PROLOG/SUN 模块系统的设计与实现	严静东、金 芝、吴来源(10)
高级语言中超级返回的实现.....	谢果然、李 萍(15)
软磁盘文件加锁及高道局部特殊格式化扇区缝隙加密的技术.....	曹尔强、张 沂、李宝岩、潘继宏(17)
EGA/VGA 图形方式下 FOXBASE+ 屏幕叠加式菜单的实现	吴邦忠、夏 英(19)
一种用于自然纹理图像分割的并行算法	管旭东、刘健勤、王爱群(21)
微机上用 Turbo C 实现的动画技术.....	何汉武(23)
一个新颖生动的 XOR 动画实例.....	郭大伟(26)
Auto CAD 与其它程序快速交换图形数据方法	王建华(29)
绘制逼真图形的捷径	江 涛(31)
利用华光交互式图表编辑排版软件绘制框图	陈 康(32)
0-1 整数规划的枚举算法	陶友传、唐泳洪、程正澄(33)
C 语言与软中断方式实现读写端口信息	黄声烈、吴庆妍、赵冬林、李志英(34)
2.13 汉字系统实用外词组文件建立与分析	龔正科(36)
微机预测注射剂和滴眼剂有效期的方法	张鸿鸣、刘铁军、孙玉华(40)
从硬盘中恢复 2.13H24 点阵字库备份盘的一种方法	李修连(7)
通用论文稿打印程序	房红兵(28)

系统篇

条形码阅读器的设计实践	马在强、朱云、李尚明(42)
与 PC 机兼容的专用键盘的设计制作	周立、赵以钰、王晓红(45)
伪彩 B 超中微机的应用和软件的编制	唐 丹、王 沐(46)
EPROM 在控制系统中的应用	崔正德(49)

硬件篇

使用 GAL 的硬加密特性保护 EPROM 中的软件	黄银彪(50)
失电后保存内存数据的方法	吴汉文(52)
IBM-PC/XT J8 插槽上 I/O 地址的“扩展”	赖红威、赖君利(53)

维修篇

采用门阵芯片的 286 机的主板诊断	陈 亮(55)
CE-150 绘图打印机电力检测电路分析及常见故障处理	潘国军(58)
SENKEN 1KVA UPS 的维修	郭 毅(59)
高分辨彩色显示器故障维修一例	单昶贤(59)
解决实时时钟丢失故障一例	刘亮生(57)
软盘驱动器读错故障排除	宋靖涛(51)
利用冷却法修复微机一例	张 智(54)

信息篇

计算机图形标准展望	汪厚祥 夏 静(60)
简讯 4 则	(封 2、4、31、48)
重要公告	(封 3)

APPLICATION RESEARCH OF COMPUTERS

VO1. 9 NO. 3 (Total 47)

CONTENTS

SOFTWARE

The Operation about Block Natural Join of Relational Algebra	Luo Weiqi(1)
A Support System for The Normalization of Relational Database	Bin Xiaohau(5)
An Optimum Encoding Method on Automatic Design for Database	Li Yanming ,Huang Shenlie (8)
Designing and Realizing for GKD—PROLOG /SUN Modular System	Yan Jingdong and et al(10)
Realizing Return with Skipping A Grade or More on High—level Language	Xie Guoran, Li Pin(15)
A Technique of Locking Disk File and Encryption for Whole Disk by Formatting It's High Track between Sectors Specially	Cao Erqiang and et al(17)
Realizing for The Screen Iterative Menu of FoxBASE+ on EGA/VGA Graphic Display	Wu Bangzhong ,Xia Yin(19)
A Parallel Algorithm about Image Segmentation of Natural Veins	Guan Xudong and et al(21)
The Animation Technique by Realizing Turbo C on Microcomputer	He Hanwu(23)
A Novel and Vivid Animation Example of XOR	E Dawei(26)
The Method for Swap Drawing Data Quickly between Auto CAD and Other Programs	Wang Jianhua (29)
The Short—cut for Making Lifelike Graph	Jiang Tao (31)
The Plotting Block Diagram by HuaGuang Edit and Composition Software with Interactive Diagram	Chen Kang (32)
The Enumeration Algorithm about 0—1 Integer Programming	Tao Youchuan and et al(33)
Realizing for Read/Write Port Information between C Language and Soft Interrupt Processing	Huang Shenglie and et al (34)
Establishing and Analysis about Practical Other Word Group File of 2. 13 Chinese Operating System	Weng Zhengke(36)
The Predictive Method for Period of Validity in Injection and Titrant by Microcomputer	Zhang Hongming and et al(40)
A Method for Resuming 24 Dot Matrix Word Base of 2. 13H Back—up Disk from Hard—disk	Li Xiulian(7)
The Printing Program for Current Paper Manuscript	Fang Hognbing(28)

SYSTEM

Designing for Barcode Scanner	Ma Zaiqiang and et al(42)
Designing for The Special Keyboard Compatible with PC Computer	Zhou Li and et al(45)
Applying about Pseudo—colour B Ultrasonic Wave and It's Software Programming	Tang Dan ,Wang Mu(46)
Applying for The E ² PROM in Controlled System	Cui Zhengde(49)

HARDWARE

Protecting The Software in EPROM by The Specific Property Hard—encryption of GAL	Huang Yinbiao(50)
The Method about Preserving Data in Internal Storage after Power Turn—off	Wu Hanwen(52)
Expanded I/O Address on J8 Plug Board IBM—PC /XT	Lai Hongwei ,Lai Junli(53)

MAINTENANCE

Diagnosising Host—computer of 286 Computers by Gate —array Chip	Chen Liang (55)
Analysising for Testing Circuit of Power on CE—150 Plotter and Processing It's Common Fault	Pan Guojun(58)
Maintenancing The SENKEN 1KVA UPS	Guo Yi(59)
An Example about Maintenancing Fault of Colour—displayer with High—resolution	Shan Changxian(59)
An Example for Solving Drop—out Bug of Real—time Clock	Liu Liangshen(57)
Debugging The Read/Write Bug of Diskette Drive	Song Jingtao(51)
An Example about Repairing Microcomputer by Cooling Method	Zhang Zhi(54)

INFORMATION

Prospects for The Graphic Standardize of Computers	Wang Houqun, Xia Jin (60)
News	(Cover 2, 4, 31, 48)
The Important Announcement	(Cover 3)

关系代数的分块自然联接运算

暨南大学 罗伟其 (510632)

摘要 本文介绍一种对两个关系作自然联接的分块运算方法,利用该方法能较大幅度地提高关系数据库的自然联接运算的速度。

一、引言 关系数据库有许多优点,这是众所周知的。但是关系数据库也有一些不足之处,其中最主要的是处理的效率问题,如果不采取有效措施,处理的效率会相当低。譬如,在关系数据库设计时,为了减少数据的冗余,避免造成数据的更新异常,往往是把关系数据模式的规范化程度提高。规范化程度越高,在查询时就需要作越多的自然联接等运算来组装关系。而由关系代数知,当元组数较多时,自然联接运算是很费时的,以致造成查询的处理速度大大地降低。这时设法减少自然联接运算的时间就成了提高查询速度的关键。下面我们通过关系代数的讨论,设计出一种对两个关系作自然联接的分块运算的算法。这一方法使自然联接的运算时间大幅度减少,提高了关系数据库的查询速度。

二、关系自然联接的分块算法 为了行文上的方便,我们仍然陈述一下本文需用到的关系代数的笛卡尔积和自然联接运算的定义。

定义 1: 设关系 R 的模式为 (A_1, A_2, \dots, A_m) , (a_1, a_2, \dots, a_m) 是 R 的任一元组, S 的模式为 (B_1, B_2, \dots, B_n) , (b_1, b_2, \dots, b_n) 是 S 的任一元组。则 R 和 S 的笛卡尔积记为 $R \times S$, $R \times S$ 的关系模式是 $(A_1, \dots, A_m, B_1, \dots, B_n)$, $R \times S$ 由一切形如 $(a_1, \dots, a_m, b_1, \dots, b_n)$ 的元组组成。

定义 2: 设 B_1, \dots, B_n 是关系 R 和 S 的公共属性, R 的模式为 $(A_1, \dots, A_m, B_1, \dots, B_n)$, $r = (a_1, \dots, a_m, b_1, \dots, b_n)$ 是 R 的任一元组; S 的模式为 $(B_1, \dots, B_n, C_1, \dots, C_p)$, $s = (b_1, \dots, b_n, c_1, \dots, c_p)$ 是 S 的任一元组。则规定元组 r 和 s 的自然联接如下:

$$r \bowtie s = \begin{cases} (a_1, \dots, a_m, b'_1, \dots, b'_n, c_1, \dots, c_p) & \text{若 } b_k = b'_k \\ \varnothing & \text{(空元组) 其余} \end{cases} \quad (k=1, 2, \dots, n)$$

那么 R 和 S 的自然联接记为 $R \bowtie S$, 且 $R \bowtie S = \{r \bowtie s \mid r \text{ 是 } R \text{ 的元组, } s \text{ 是 } S \text{ 的元组}\}$ 。

由定义 2 知自然联接要做三件事:

(1) 计算 $R \times S$;

(2) 挑选出 $b_k = b'_k (k=1, \dots, n)$ 的所有元组;

(3) 去掉重复属性。

自然联接是组装关系的有效方法,它是关系代数中最重要的运算之一。下面我们就来讨论自然联接的分块算法。

定理 1: 设有关系 R_1 和 R_2 , R_{1i} 和 $R_{2i} (i=1, \dots, k)$ 分别是 R_1 和 R_2 的 K 个与 R_1 和 R_2 模式相同的互不相交的子关系,即 R_1 和 R_2 可分解为:

$$R_1 = R_{11} \cup R_{12} \cup \dots \cup R_{1K}, R_{1i} \cap R_{1j} = \varnothing (i \neq j, i, j=1, 2, \dots, k);$$

$$R_2 = R_{21} \cup R_{22} \cup \dots \cup R_{2K}, R_{2i} \cap R_{2j} = \varnothing (i \neq j, i, j=1, 2, \dots, K).$$

其中 \varnothing 表示空元组。则当 $R_{1i} \bowtie R_{2i} \neq \varnothing$ 和 $R_{1i} \bowtie R_{2j} = \varnothing (i \neq j, i, j=1, 2, \dots, K)$ 时,下式成立。

$R_1 \bowtie R_2 = (R_{11} \bowtie R_{21}) \cup (R_{12} \bowtie R_{22}) \cup \dots \cup (R_{1K} \bowtie R_{2K})$ 。(1) 并称(1)式右边表达式为分块自然联接表达式。

证明: 由关系代数知关系并与自然联接是满足分配律和结合律的。则由分配律和结合律有:

$$\begin{aligned} R_1 \bowtie R_2 &= (R_{11} \cup R_{12} \cup \dots \cup R_{1K}) \bowtie (R_{21} \cup R_{22} \cup \dots \cup R_{2K}) \\ &= [(R_{11} \bowtie R_{21}) \cup (R_{11} \bowtie R_{22}) \cup \dots \cup (R_{11} \bowtie R_{2K})] \cup \dots \cup [(R_{1K} \bowtie R_{21}) \cup (R_{1K} \bowtie R_{22}) \cup \dots \cup (R_{1K} \bowtie R_{2K})] \end{aligned}$$

又由假设 $R_{1i} \bowtie R_{2j} = \varnothing (i \neq j, i, j=1, 2, \dots, K)$ 即得 $R_1 \bowtie R_2 = (R_{11} \bowtie R_{21}) \cup (R_{12} \bowtie R_{22}) \cup \dots \cup (R_{1K} \bowtie R_{2K})$ 因此(1)式成立。

定理 2: 对作自然联接运算 $R_1 \bowtie R_2$ 的任意两个关系,都可以用式(1)作分块自然联接运算。

证明: 要证明定理 2 成立,事实上只要证明 R_1 和 R_2 能按元组和自然联接条件分解成 K 个互不相交的子关系 R_{1i} 和 $R_{2i} (i=1, 2, \dots, k)$, 且 R_{2i} 仅与 R_{1i} 有关而与 $R_{1j} (j \neq i)$ 无关即可。

由自然联接的定义知, R_1 和 R_2 是有公共属性的,且自然联接条件中所含的属性一定是公共属性中的属性。设 X 为自然联接条件中所含属性的集

合。那么R1的R2的分解步骤为:

首先,按X对R1作索引。其次,把索引后的R1按元组分为K个互不相交的K个子关系R1i(i=1, 2, ..., k)。同时要求把X属性值相同的元组必须划分在同一子关系中。

第三,取出任一R1i(i=1, 2, ..., k)中的第一个和最后一个元组中属于X的属性值,组成对R2作选择运算的条件E_i(i=1, 2, ..., K)。因R1i是互不相交的,所以选择条件E_i也是相互无关的。

最后,用选择条件E_i(i=1, 2, ..., K)对R2作选择运算,得到K个子关系R2i=δ_{E_i}(R2)(i=1, 2, ..., K)。若R2的K个子关系R2i(i=1, 2, ..., K)的元组数之和小于R2的元组数,即R2分为K个子关系后仍有多余的元组,则这些多余的元组可以去掉。这是因为这些多余的元组是与R1没有满足自然联接条件的元组,在作自然联接时,得到的是空元组,即这些多余的元组对整个自然联接的结果是无影响的。至此R2也被分为K个子关系。又因为R2i=δ_{E_i}(R2)中的E_i(i=1, 2, ..., k)是仅与R1i有关的互不相交的条件,所以R2i也是互不相交的,且R2i仅与R1i中的X属性集有关,而与其他子关系R1j(j≠i)无关,即R1i∩R2i≠∅,但R1i∩R2j=∅(i≠j, i, j=1, 2, ..., K)。

通过这样分块后,R1和R2以及它们的K个子关系R1i和R2i是满足定理1的条件的,所以R1与R2的自然联接可以按式(1)作分块自然联接。

三、自然联接的分块法与直接法的比较

设R1和R2的元组数分别为n和m,并把按定义进行的自然联接法称为直接法。

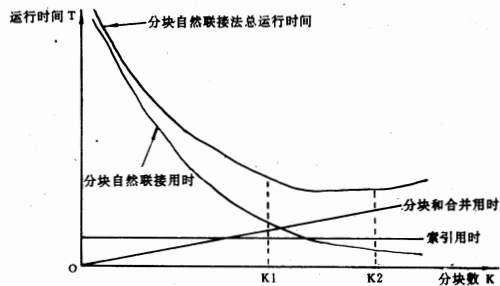
用直接法作R1∞R2时,在关系R1中顺序抽出各个元组,每次抽出一个元组后,就在关系R2的全部元组中寻找符合联接条件的元组,每找到一个就与R1中抽出的元组进行联接,在联接结果关系中形成一个新元组。所以需要作的元组操作总次数为nm。

对分块法由定理1和定理2知,把参与作自然联接运算的两个关系R1和R2分块后,可以把众多的R1i∩R2j=∅(i≠j, i, j=1, 2, ..., k)无效运算(即得到空元组的自然联接运算)排除掉。正因为避免了做这些无效自然联接运算,因此带来总运算时间的减少。为了讨论方便,设R1i(i=1, 2, ..., K)的元组数相同为r且n=Kr,即把R1等分为k个子关系,R2i的元组数为m_i(i=1, 2, ..., K),由分块时知 $\sum_{i=1}^k m_i \leq m$ 。

那么用分块法作自然联接的元组操作次数为 $\sum_{i=1}^k r m_i$

$$= r \sum_{i=1}^k m_i \leq nm/k, \text{即最多仅为直接法的 } 1/k.$$

因为分块法(等分时)最多占直接法同种运算元组操作次数的1/k,所以k越大,分块法占用的时间就越少,不是等分时,分块法作自然联接的时间也是随K的增大而减少。即分块法作自然联接的时间与分块数K成反比。那么是不是K可以取最大就一定能使总的运行时间最少,效率最高呢?显然不是,这是因为分块法除作自然联接运算外,比直接法还多作了三种运算:①索引运算,当R1选定后,用于索引的时间就是一常数;②把R1和R2分为K个子关系的分块运算,这种运算占用的时间与K成正比;③把各子关系作自然联接R1i∩R2i(i=1, 2, ..., k)的结果合并的运算,这种运算占用的时间也与K成正比。所以当K取值增大时,用于自然联接的时间是大幅度减少了,但与此同时用于分块运算和合并运算的时间也增加了一些。分块法总的运行时间与各部分运算时间的关系可用图一所示。



图一 分块法运行时间关系曲线图

从图一知,对作自然联接的关系R1和R2作适当地分块(如K落在K1与K2的范围内),则分块法比直接法作自然联接可以极大地减少总运行时间,提高运算效率。

四、分块自然联接法在关系数据库语言dBASE中的应用实例 dBASE(包括I、II、III PLUS、IV)的自然联接运算是由JOIN命令来实现的。我们知道JOIN命令的直接使用虽是方便的,但当记录数较多时(如150个以上)运行是很费时间的,甚至达到让人无法忍耐的地步。如果我们用分块自然联接来处理,则可以大幅度减少运行时间。

为了作直接法与分块法的效率比较,设有关系数据库R2和R3,R2和R3的结构见图二和图三。又

设 TRS 为 R2 和 R3 作自然联接运算产生的结果关系数据库,它的结构如图四。设 R2 和 R3 的联接的公共字段为 DM,R2 和 R3 的记录数设均为 600 条。自然联接的条件是 $R2 \rightarrow DM = R3 \rightarrow DM$

Structure for database,B;R2.dbf

Number of data records, 600

Date of last update ,05/20/90

Field	Field Name	Type	Width	Dec
1	DM	Character	21	
2	KMD	Character	3	
3	ZMD	Character	5	
4	XMD	Character	5	
5	ZYD	Character	8	
6	ZSL	Numeric	11	2
7	DSL	Numeric	11	2
8	YSL	Numeric	11	2
* *Total * *			76	

图二

Structure for database,B;R3.dbf

Number of data records, 600

Date of last update ,05/20/90

Field	Field Name	Type	Width	Dec
1	DM	Character	21	
2	ZRB	Numeric	13	2
3	DRB	Numeric	13	2
4	YRB	Numeric	13	2
* *Total * *			61	

图三

Structure for database,B;TRS.dbf

Number of data records, 600

Date of last update ,05/20/90

Field	Field Name	Type	Width	Dec
1	DM	Character	21	
2	KMD	Character	3	
3	ZMD	Character	5	
4	XMD	Character	5	
5	ZYD	Character	8	
6	ZSL	Numeric	11	2
7	DSL	Numeric	11	2
8	YSL	Numeric	11	2
9	ZRB	Numeric	11	2
10	DRB	Numeric	11	2
11	YRB	Numeric	11	2
* *Total * *			109	

图四

用分块自然联接法编写的程序如下:

1: * * 分块自然联接法程序

2:SET TALK OFF

3:INPUT 分块数 K=.TO K

4:T1=TIME()

5:SELE 2

6:USE R2

7:SELE 1

8:USE R3

9:GO BOTT

10:RN =RECNO()

11:KS=INT(RN/K)

12:JLS=RN-KS*(K-1)

13:INDEX ON DM TO IR3

14:SET INDEX TO IR3

15:T2=TIME()

16:I=1

17:DO WHILE I<=K

18:IF I<10

19:RE=STR(I,1)

20:ELSE

21:RE=STR(I,2)

22:ENDIF

23:IF I>1

24:SKIP

25:AM2=DM

26:ENDIF

27:IF I=K

28:P=JLS

29:ELSE

30:P=KS

31:ENDIF

32:COPY NEXT P TO R1&RE

33:AM1=DM

34:SELE 2

35:DO CASE

36:CASE I=1

37:COPY TO S11 FOR DM<=AM1

38:CASE I>1. AND. I<K

39:COPY TO S1&RE FOR DM<=AM1. AND.

DM>=AM2

40:CASE I=K

41:COPY TO S1&RE FOR DM>=AM2

42:ENDCASE

43:SELE 1

44:I=I+1

45:ENDDO

46:SELE 1

47:USE TRS

48:ZAP

```

49,T3=TIM()
50,I=1
51,DO WHILE I<=K
52,IF I<10
53,RE=STR(I,1)
54,ELSE
55,RE=STR(I,2)
56,ENDIF
57,SELE 2
58,USE S1&RE
59,SELE 3
60,USE R1&RE
61,JOIN WITH S1&RE TO TEMP FOR DM=B->DM
62,SELE 1
63,APPEND FROM TEMP
64,DELE FILE TEMP.DBF

```

```

65,SELE 2
66,USE
67,DELE FILE S1&RE...DBF
68,SELE 3
69,USE
70,DELE FILE R1&RE...DBF
71,I=I+1
72,ENDDO
73,T4=TIME()
74,T1,T2,T3,T4
75,CLOSE DATA

```

在640KB内存,20MB硬盘的IBM-PC/XT微型计算机上,用dBASE II PLUS系统运行上面的程序以及用直接法编写的程序(略),得到运行时间效率对比表如下(表一)。

表一 分块法与直接法做联接 JOIN 运算的效率对比表
说明:1、主动数据库 R3和被联接数据库 R2的记录数均为600个。

2、直接法运行时间 T1=40分26秒。

3、分块法的运行时间包括对 R3做索引用时36秒。

分块数 K	平均每块记录数	运行时间 T2	其 中		T2/T1	T2比 T1提高效率 %
			分块用时	联接用时		
K=4	150	13分48秒	1分7秒	12分5秒	0.341	193.0
K=5	120	10分45秒	1分24秒	8分45秒	0.265	276.1
K=6	100	7分59秒	1分33秒	5分50秒	0.197	406.5
K=7	85.71	7分31秒	1分42秒	5分13秒	0.186	437.9
K=8	75	7分13秒	1分52秒	4分45秒	0.178	460.2
K=9	66.67	7分3秒	1分59秒	4分28秒	0.174	473.5
K=10	60	6分50秒	2分6秒	4分8秒	0.169	491.7
K=11	54.55	6分48秒	2分13秒	3分59秒	0.168	494.6
K=12	50	6分47秒	2分19秒	3分52秒	0.168	496.0
K=20	30	7分46秒	3分14秒	3分56秒	0.192	427.0
K=30	20	9分12秒	4分23秒	4分13秒	0.228	339.5

在表一中的联接用时包含了合并运算 APPEND FROM TEMP 的时间。从表一中可以看出,分块自然联接法的运行时间比直接法的运行时间大幅度减少,运算效率大幅度提高。

参考文献

1. C. J. DATE,《An Introduction to Database Systems》,1986,Fourth Edition.
2. Stephen J. Hegner,“Algebraic Aspects of Relational Database Decomposition”, ACM. Symp. on Prin. of Database Systems,1983,PP. 400-413.

3. A. V. Aho, C. Beeri, and J. D. Ullman “The Theory of Joins in Relational Database.” Trans. on Database Systems 4,3, PP. 297-314, 1979.

4. 冯玉才编著:“数据库系统基础”,华中工学院出版社,1984.

5. 姚诗斌编著:“数据库系统基础”,计算机工程与应用编辑部.

6. 陈增武、金连甫编著:《汉字 dBASE II 关系数据库管理系统及其应用》,电子与仪表技术编辑部,1986.

实现复杂表达式显示及打印的翻译软件

辽宁鞍山钢铁公司废钢处计算机室赵消歧(邮编 114021)开发出一种可以实现复杂表达式(如各种化学、数学、物理公式)显示及打印的翻译软件,具有使

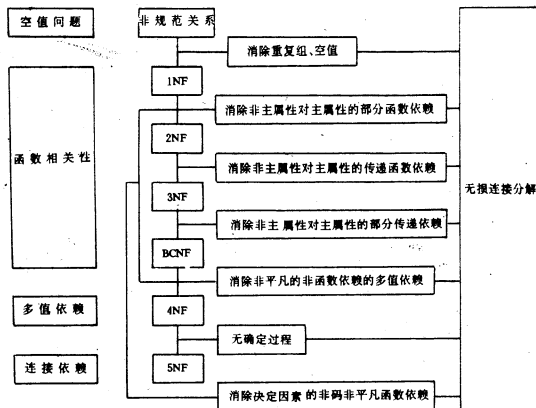
用灵活、适用范围广、处理速度快等特点,采用8088汇编语言编制,适用于PC及其兼容机。需此软件者,可同作者或本刊联系。(本刊)

关系数据库关系规范化支持系统

总后基地指挥部 宾晓华 (430010)

摘要:关系数据库的关系规范化(即范式设计)可有效地解决数据库的异常操作问题,在MIS的开发中有着重要的方法论意义。但是,由于其理论深度和复杂性,在实际工作中,它的应用受到限制。本文介绍的范式设计支持系统(FDSS)能帮助开发者越过上述障碍,支持用户设计出具有期望级别的数据库范式。

一、范式及范式的升级 范式揭示了一个关系框架中各属性之间不同类型不同层次的依赖关系,并由此说明数据库的异常操作问题。对一个信息系统来说,如果所设的数据库均具有良好的、合理的范式级别,那么系统的正确性将自动得到某种保证。但因范式的理论基础超出了一般的系统设计者的知识范围,而且验证范式级别和进行分解升级是件计算量大、过程繁杂的工作(见下图),致使范式设计难以实用化。



范式设计支持系统(FDSS)就是为了解决以上问题而建造的。它既要通过对话引导信息系统设计者将问题的背景和自己对问题的认识告诉FDSS,又要自动完成大多数的理论推导和验证,而不要求用户对范式有更多的知识。上图概述了FDSS的构成原理和控制过程。从大致情况来看,左边主要涉及对话过程,中间是希望达到的目标,右边是完成自动推导计算的方法及模型库工作的主要内容。

二、获取问题定义及用户意向的对话过程

1、空值及函数相关性的经验性与后验性 在信息系统的设计中,我们常常根据数据本身的用

途、性质来安排数据库结构。库结构的各字段之间存在一定的依赖关系。遗憾的是,这种依赖关系并不是一种语法性质,而是一种语义性质。就是说,我们无法从关系框架本身来得出这种依赖关系,而是要么通过一个具体关系来验证,要么通过设计者来亲自确定这种关系。前者说明了后验性,因为我们必须从已有的数据来推断其依赖性质,只要对这个数据库进行操作(增、删、改),这种依赖性质就可能发生变化;后者说明了经验性,因为要确定字段间的依赖性质需要用户对所涉及事物的分析、认识和理解,并且,即算用户确定了这种依赖性质,他也很可能无法从理性上判定这个库结构的范式级别。空值也具有与函数相关性类似的语义特征。空值指的是说目前尚不确知的值(如有生命的天体个数),或不存在的值(如没有分到住房的工作人员在该单位的住处),还有本来的空值(如空串作为无需解释的备注)。前面两类空值影响数据库的范式性质,因为,存在空值的数据库连1NF的条件都不满足。

2、函数相关性集的获取 设有关系框架R(A₁, A₂, ..., A_n)。为穷举其相关性,需考虑P(R)(R的幂集)上的一个关系F₁,它是P(R) × P(R)的子集,选自(2 * n) × (2 * n)个元素,如当n=8时,就有65536个元素,而从用户那里得到如此大的输入量是不可能的。幸而,在这些元素中,绝大多数是逻辑蕴含的,还有相当大一部分是恒假的,据此,我们分步骤减少论集的规模,直到合理的程度。

为说明问题,我们以下面关系框架为例,并用方括号表示系统的提示,用尖括号表示用户的输入,此外我们有时还将{A₁, ..., A₈}记作A₁...A₈:

部门	姓名	职务	应住面积	住处	面积	建筑类型	房租
A1	A2	A3	A4	A5	A6	A7	A8

第一步:通过对话获得主属性集,并自动删除其导出集。

[输入主属性集M]:——<A₁, A₂>

令: $D1 = \{M \rightarrow Y | Y \text{ 属于 } R - M\}$, $B1 = P(R - M)$. 当获得 M 后, 我们只需考虑在 $B1$ 上的关系 $F2$ 了. 可知, $B1 \times B1$ 的元素比 $P(R) \times P(R)$ 的要少得多. 例如, 在本例中规模由 65536 降到了 4096.

第二步: 处理单属性依赖.

[在非主属性集中, 哪一些单个属性可以决定另一单个属性]: $\rightarrow A5 \rightarrow A6, A5 \rightarrow A7, A3 \rightarrow A4$

将上面尖括号里的元素作成集合 $D2$, 并设:

$F3 = F2 - \{X \rightarrow Y | X, Y \text{ 是 } B1 \text{ 的任一子集}, t \rightarrow r \text{ 是 } D2 \text{ 的元素}\}$ 现在我们只需在 $F3$ 上考虑了. $F3$ 的元素个数已降到 200 个以下.

第三步: 排除压缩.

[在非主属性集中, 哪一个单个属性肯定与另外一些单个属性无关]: $\rightarrow A4 \rightarrow A5, A4 \rightarrow A6, A4 \rightarrow A7, A4 \rightarrow A8, A6 \rightarrow A7$

将用户回答内容做成集合 $D3$, 令:

$F4 = F3 - \{X \rightarrow YZ | X, Y \text{ 分别为 } D3 \text{ 的某元素的左、右部}, Z \text{ 是 } B1 \text{ 的任意子集}\}$ $F4$ 是对 $F3$ 的进一步的等价压缩.

第四步: 传递压缩.

对 $F4$ 再进行传递压缩, 形成 $F5$:

$F5 = F4 - \{X | X \text{ 是 } D1 \text{ 与 } D2 \text{ 之并的传递闭包}\}$

例如, 上例中, 由于 $A5 \rightarrow A6, A5 \rightarrow A7, A6 \rightarrow A7 \rightarrow A8$, 所以 $A5 \rightarrow A8$ 就被压缩了, 它可以自动推导出来, 无需从用户那里输入.

经过以上四步以后, $F5$ 的元素所剩无几 (甚至为空), 这时便可对 $F5$ 逐个询问用户, 它所表示的依赖关系是否成立. 设成立的元素组成 $D4$, 则我们最终得到了函数相关性集合 F , 它为 $D1, D2, D4$ 之并集, 还有主属性集合 M 这个副产品. F 和 M 是模型库中分解模型的重要参数.

3. 获得有关空值的信息. $FDSS$ 据此进行投影分解, 以达到 1NF:

[在 $A1 A2 A3 A4 A5 A6 A7 A8$ 中哪些可能出现空值]: $\rightarrow A5 A6 A7 A8$

4. 框架分解倾向性意见的获得

无损连接分解是范式升级的一种重要方法, 在大多数情况下, 系统可以自动完成这种分解. 但有时为了照顾用户的特殊需要或加快求解速度, 用户可以自行输入分解方案, 如对上面框架的一个使其达到 1NF 的分解为:

[对 $A1 A2 A3 A4 A5 A6 A7 A8$ 进行分解的倾向性意见是]: $\rightarrow A1 A2 A3 A4, A1 A2 A5 A6 A7 A8$

当然, 用户的分解是否是无损连接分解并且能

否达到所希望的范式, 将由 $FDSS$ 自行检验. 用户分解方案也是模型库所需要的重要参数选项.

三、模型库的管理与实现

1. $FDSS$ 模型库的管理 $FDSS$ 模型库是完成范式验证、关系框架分解升级、数据预加工等任务的, 它是 $FDSS$ 的核心. 模型库由一系列单元模型组成, 一个单元模型由五个部分组成: 模型标识 (及说明), 模型入接口、模型出接口、模型连接参考、模型体. 模型库的管理机构完成模型的登记、调度、参数连接、单元模型激发、结果数据管理等工作, 个别模型还需参与对话. $FDSS$ 的特点是高度的透明性, 这种透明性对不熟悉规范理论的用户特别重要. 用户不但可以不去熟悉模型的数学原理, 也可不去理会它们的调度和连接过程.

2. $FDSS$ 的单元模型 由于多数模型的数学原理均需大量篇幅来叙述, 这里只列出一些重要的单元模型的入出接口参数, 由此可以看出这些模型的功能:

①、相关性集的最小化

INSET (相关性集合 F) OUTSET (与 F 等价的一最小相关性集 F_{min})

②、相关性集的完备集

INSET (相关性集合 F) OUTSET (F 的自反传递闭包)

③、相关性集的非平凡化

INSET (相关性集合 F) OUTSET ($F1 = F - \{X \rightarrow Y | Y \text{ 是 } X \text{ 的子集}\}$)

④、测试一个分解是否是无损连接分解

INSET (关系框架 R , R 的相关性集 F , R 的一个分解)

OUTSET (YES/NO)

⑤、消除空值的投影分解一到达 1NF

INSET (关系框架 R , 相关性集合 F , 空值性集合 N)

OUTSET (R 的一个无空值性的无损连接分解)

⑥、消除部分函数依赖的投影分解——到达 2NF 的分解

INSET (1NF 关系框架 R , 主属性集 M , 相关性集合 F , 一种投影分解 S ——当采用人工分解时, 否则缺省)

OUTSET (R 关于 F 是否存在对 M 的部分函数依赖 $[Y/N]$, 为 N 时 R 是 2NF 的, 为 Y 时若 S 为空则输出到达 2NF 的投影分解, 否则验证分解 S 的连接是否是无损的, 并回答分解出来的两个关系是否有部分函数依赖)

⑦、消除传递函数依赖的投影分解——到达

3NF 的分解

INSET(2NF 关系框架 R, 主属性集 M, 相关性集合 F 的传递闭包的平凡子集 F1, 一种投影分解 S——当采用人工分解时, 否则缺省)

(OUTSET(R 关于 F 是否存在对 M 的传递函数依赖[Y/N], 为 N 时 R 是 3NF 的, 为 Y 时若 S 为空则输出到达 3NF 的投影分解, 否则验证分解 S 的连接是否是无损的, 并回答分解出来的两个关系是否有部分函数依赖[Y/N])

⑧、到达 BCNF 的无损连接分解

INSET(1NF 关系框架 R, 相关性集合 F)

OUTSET(R 的无损连接分解 {R1, ..., Rn}, 其中, R1 是 BCNF 的)

3. FDSS 的实现 FDSS 用 FOXBASE+ 作为系统控制、大多数对话和模型的实现语言。FOXBASE+ 的数组功能、字符串操作功能、宏替换功能、数据库管理功能补偿了计算功能的不足, FDSS 与用它开发的管理系统在同一语言环境下运行也有好处。此外, 用逻辑程序表达属性间的依赖关系并实现上述某些模型, 比用其它语言要简单一些。FOX 链接这种模型的方法是: 事先将输入接口参数以文本格式写盘, 之后用 RUN 命令调用 PROLOG, 读入上述参数并求解后将输出接口参数以文

本格式写盘, 退出 PROLOG 后 FOX 用 COPY...SDF 将文本文件转换成仅有一个字段的库文件, 供 FOX 直接引用。

四、结语 关系数据库设计既要考虑库结构的范式性质, 又要照顾数据组织的自然习惯, 它还依赖于设计者对数据背景的认识和理解, 因此数据库设计具有一定的能动性和随意性。为适应这些特点, FDSS 在功能和结构上作了重点处理。FDSS 的规模虽然不大, 却是一个完整的支持系统。它既适用于从事 MIS 生产的专业开发者, 也适合于一般的开发者。

参考文献

- [1] Wiederhold G.: "Database design", McGraw-Hill, 1977.
- [2] 郑若忠等: 《数据库原理与方法》, 湖南科技出版社, 1983.
- [3] 钟恢扶等: 《关系数据库的设计与应用》, 湖北科技出版社, 1986.
- [4] 宾晓华: "决策支持系统的功能与构成", 决策信息杂志, 1990. 9.
- [5] 宾晓华: "管理软件垂直推广所遇到的问题及解决方法"《微电子学与计算机》1990. 8.

从硬盘中恢复 2.13H24 点阵字库备份盘的一种方法

岳阳市四信箱 李修连 (414007)

2.13H 字库盘 HZK24-1 中存有 24 点阵宋体字库的前部分, 计有 362496 字节 (354K), 刚好占满 360K 软盘的整个数据区。

在备份盘 HZK24-1 损坏后, 从硬盘文件 C:\213\HZK24S 中拷贝出前 35K 至一张格式化的 360K 空软盘中, 这似乎是没有问题的。但实际的拷贝结果是出现盘空间不够的错误, 拷贝失败。但我采用如下覆盖法, 拷贝成功。

操作

```
C>DISKCOPY A: B;
C>REN B, HZK24H HZK24S
C>LABEL B;
C>DEBUG\213\HZK24S
-R CX
:8800
-R BX
:5
-N B, HZK24S
-W
Writing 58800 bytes
-Q
```

说明

① A 盘中是好的 HZK24-2, 存有 24 点阵黑体字库的前 354K。

② 改变文件名, 但切记不要删除盘上的文件。

③ 将盘的卷标改为 HZK24-1。

注意: 由于硬盘中的 HZK24S 有 477K, 因此必须要有足够的内存才能利用 DEBUG 来拷贝, 否则可通过编程来拷贝。

上面所说的实质上是一个“文件长度正好等于剩余盘空间”的拷贝方法。由于文件在盘上是以簇为单位分配空间的, 每簇有一个或多个扇区, 每扇区通常有 512 字节, 当文件填满一簇后, 才会再分配空间。因此, 在文件长度小于盘剩余空间时, 不会出现盘空间不够的错误。在文件长度等于盘剩余空间时, 为了得到被覆盖盘, 可用 DEBUG 或编程将文件的大部分拷贝到盘上以占用完盘空间, 再将整个文件覆盖到盘上。

以上操作是在 PC/XT 机上、DOS3.10 下进行的。

数据库设计自动化中的一种最佳编码方案

吉林工业大学 李彦明 黄声烈 (130025)

一、引言 随着近几年来数据库设计自动化理论与应用领域的不断深入,人们试想探讨一种编码效率高、数据冗余度为零的途径。事实上这是不可能的,因为数据库应用环境的变化,很难保证数据库设计自动化的一致性、完整性。本文论述有关这方面的理论,通过概率统计的数学方法来进一步论证一种最佳编码方案。

二、问题的提出 数据库设计自动化是指:从需求分析开始,到最后得到应用系统的整个开发过程,采用自动化设计方法,由计算机参与中间大部分开发活动,减少软件人员的大量复杂的重复劳动,从而有效地提高数据库系统开发速度,缩短开发周期;同时,系统的维护也必须使计算机参与,使得维护实现自动化。在这方面最主要的工作是CASE技术。数据库CASE技术主要包括四个内容,即CASE * Method、CASE * Dictionary、CASE * Designer和CASE * Generator。

CASE * Method实质上是数据库设计方法学的一个方法库,它是一个软件系统,用来解决数据库的设计问题;CASE * Dictionary是一个可移植的计算机化的多用户数据词典系统;CASE * Designer是为数据库设计人员和系统分析人员提供的一个高级的、高度图形化的用户接口软件;CASE * Generator是数据库自动化中最重要的内容,它的任务是根据字典信息和设计报告书自动生成出一个应用系统。

数据库本身的设计自动化最主要是方法库的设计,字典和生成器可作为一种辅助工具来决定与方法库的关系。该关系数据库通常是用自适应数据库方法来确定。这是由于数据库应用系统具有三大特性:(1)由于使用环境变化,数据库结构是时变的;(2)数据库应用模式是时变的;(3)从而导致程序也是时变的。要使得数据库应用系统能够自适应,就必须使得这三个时变内容能够自动变更,这就是所谓的自适应数据库系统。

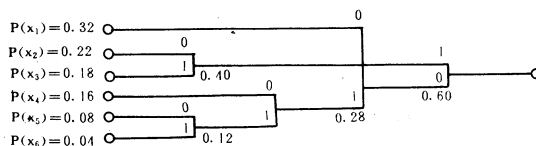
目前,实现这三大内容的自动变更的具体方法是:(1)数据字典方法;(2)格式文件自动生成方法;(3)通用报表方法。在软件工程和数据库应用系统中,数据字典通常作为分析,设计和维护的重要手段,且在数据库自动化模型中,增加了“编码”这个环节,这使开发模型提出了不少有益的概念、步骤、方

法和技术、为提高软件生产率,保证软件产品可靠性,指导人们开发各种类型的数据库系统起了积极的作用。

由此可见,高效率的编码技术及其代码用于数据库设计自动化中是非常重要的。

三、最佳编码方案 所谓最佳就是它的平均码长最短的编码。它的基本思想是按照信息出现的概率,赋予不同的码长,概率大的信息对应的码字短,概率小的信息对应的码字长。

设有信息 $x_1, x_2, x_3, x_4, x_5, x_6$,它们的概率分别是: $p(x_1)=0.32, p(x_2)=0.22, p(x_3)=0.18, p(x_4)=0.16, p(x_5)=0.08, p(x_6)=0.04$,则先将信息 $x_i (i=1, 2, \dots, 6)$ 按概率大小进行排序,大的在前,小的在后,然后将两个概率小的信息合在一起,作为一个新的信息,它的概率为两个信息概率之和,重复上述步骤,信息数目越来越小,直到最后两个信息的概率之和为1,其编码过程如下:



它的概率分布树形图如图1所示。

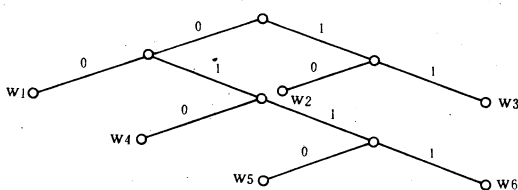


图1 概率分布树形图

其中: $W_1 \rightarrow 00, n_1=2$

$W_2 \rightarrow 10, n_2=2$

$W_3 \rightarrow 11, n_3=2$

$W_4 \rightarrow 010, n_4=3$

$W_5 \rightarrow 0110, n_5=4$

$W_6 \rightarrow 0111, n_6=4$

$n_i, i=1, 2, \dots, N(\text{码字长度})$

$W_j, j=1, 2, \dots, N$ (编码的码字)

由此得到平均码长 \bar{L} :

$$\begin{aligned}\bar{L} &= \sum_{i=1}^n p(W_i) n_i \\ &= 0.32 \times 2 + 0.22 \times 2 + 0.18 \times 2 + 0.16 \times 3 \\ &\quad + 0.08 \times 4 + 0.04 \times 4 = 2.32\end{aligned}$$

在一般情况下,编码的平均长度 \bar{L} 与编码的方法有关,为了说明各种编码方法的效果,我们定义编码的效率为:

$$\eta = \frac{H(x)}{\bar{L}(x)}$$

其中 $H(x)$ 为熵 (entropy)

熵的定义:考虑作为完全事件系的信息源为

$$x = \begin{pmatrix} A_1 & A_2 & \dots & A_N \\ p_1 & p_2 & \dots & p_N \end{pmatrix}$$

这里, $\{A_1, A_2, \dots, A_N\}$ 是消息的集合,各消息的概率用 $p_1, p_2, p_3, \dots, p_N$ ($p_i \geq 0, \sum_{i=1}^N p_i = 1$) 表示。这样一种信息源的信息量定义为:

$$H(x) = - \sum_{i=1}^N p_i \log_2 p_i$$

如果对数以 2 为底,则称信息量的单位为位 (bit)。即当 $N=2, p_1=p_2=0.5$ 时, $H(x)=1(\text{bit})$ 。

又定义冗余度为:

$$r = 1 - \eta$$

下面进一步说明编码方法与效率的关系。设有信息 x_1, x_2, x_3, x_4 , 其中 $x_i (i=1, 2, 3, 4)$ 在数据库设计自动化中可视为变量,字段名,文件名或表达式等。 $W(x_i)$ 表示它们的编码, $L(x_i)$ 表示码长, $P(x_i)$ 表示信息 x_i 出现的概率,它们的值分别如下:

x_i	x_1	x_2	x_3	x_4
$W(x_i)$	00	01	10	11
$L(x_i)$	2	2	2	2
$P(x_i)$	1/4	1/4	1/4	1/4

则它的平均码长为:

$$\bar{L}(x) = \sum_{i=1}^4 P(x_i) L(x_i) = \frac{1}{4} \times 4 \times 2 = 2(\text{bit})$$

信息熵为:

$$\begin{aligned}H(x) &= - \sum_{i=1}^4 P(x_i) \log_2 P(x_i) = -4 \times \frac{1}{4} \log_2 \frac{1}{4} \\ &= 2(\text{bit})\end{aligned}$$

所以上述编码效率为 1, 冗余度为 0。

如果上述 $W(x_i)$, 即编码不变, 只改变 x_i 出现的概率为:

$$P(x_1) = \frac{1}{2}, P(x_2) = \frac{1}{4}, P(x_3) = \frac{1}{8}, P(x_4) = \frac{1}{8}.$$

则它的平均码长为:

$$\bar{L}(x) = \frac{1}{2} \times 2 + \frac{1}{4} \times 2 + \frac{1}{8} \times 2 = 2(\text{bit})$$

$$H(x) = - \sum_{i=1}^4 P(x_i) \log_2 P(x_i) = \frac{7}{4}(\text{bit})$$

$$\eta = H(x) / \bar{L}(x) = \frac{7}{8} = 87.5\%$$

$$r = 1 - \eta = 12.5\%$$

这就是说由于只改变了 x_i 出现的概率, 该编码的信息量冗余度较大。如果在上述情况下, 改变编码方法, 不改变 x_i 出现的概率, 则进一步提高编码效率。

若编码方式变为:

$$W(x_1) = 0, W(x_2) = 10, W(x_3) = 110, W(x_4) = 111$$

则相应的码长分别变为:

$$L(x_1) = 1, L(x_2) = 2, L(x_3) = 3, L(x_4) = 3$$

那么, 它的平均码长为:

$$\bar{L}(x) = \frac{1}{2} + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{8} \times 3 = \frac{7}{4}(\text{bit})$$

对于信息熵 $H(x)$ 来说, 由于信息的概率没有变, 仍保持比特数为 $\frac{7}{4}(\text{bit})$, 所以编码效率为 100%, 冗余度下降到 0。

因此, 提高编码效率的原因是在编码中把概率大的信息用长度短的代码, 概率小的信息用长的代码, 这种编码不仅使它的平均码长 $\bar{L}(x)$ 最小, 而且使它的冗余度最小。不过这种编码的码字长度变化太大, 码字的结构比较复杂, 实现就困难。

为了简化码字的结构复杂性, 在数据库设计自动化中编码系统中, 往往采用编码的码字长度变化不宜过大的情况下, 确认编码方案合理性, 通过该信息 x_i 出现的概率, 提高计算机访问数据文件的高可靠性。

四、结束语 该文较系统地论述了在数据库设计自动化中一种高效率的编码方案, 探讨了一种应用于数据库自动化模型, 通过与概率分布相匹配编码的数学方法, 论证了最佳编码方案, 具有一定的参考意义。

GKD—PROLOG/SUN 模块系统的设计与实现

国防科技大学计算机系 严静东 金 芝 吴泉源 (410073)

摘要 模块化是大型程序设计的关键问题之一。本文在 GKD—PROLOG/VAX 解释器的基础上,详细分析了 PROLOG 模块系统的基本语法,知识的模块封装,信息隐藏等特性,实现了 GKD—PROLOG/SUN 模块系统,该系统以国际上流行的 QUINTUS—PROLOG 为蓝本,在解释器一级直接支持模块的动态及静态创建,模块间的交互及模块系统运行环境间的切换,使 PROLOG 适用于大型软件课题的需要,本系统不仅支持模块化的程序设计风格,也为实现面向对象的逻辑程序设计环境奠定了基础。

一、引言 模块化已经是许多传统语言的重要特性之一。如 PASCAL 的过程,Ada 的抽象数据类型,以及 C 的函数结构等都是实现模块化的手段,这些模块化特性主要有如下优点:

1. 大型软件系统由多个程序模块组成,这些程序模块可由不同程序员独立编程,编程速度大大提高,并且小的程序模块利于调试和维护。

2. 对模块的访问只能通过模块接口实现,而内部过程及数据结构是不可见的,体现了模块化语言信息隐藏的特性。

3. 可以对单一模块进行正确性和可靠性分析,从而为分析整个大型软件系统的正确性和可靠性提供了一定的基础。

4. 软件设计和实现过程仅考虑各模块的良好外部接口,而不用考虑其内部数据结构和算法,体现了数据抽象的基本原则。

由于逻辑程序设计语言语法简单,语义清晰,具有描述性风格和很强的表达能力,还具有人工智能所必需的不确定性推理能力。因此逻辑程序设计已成为人工智能和其它应用领域一种流行的程序设计风格,但至今在逻辑程序中,所有子句都是独立的,各子句的关系在推理运行时确定,整个推理过程是顺序搜索的。这种一个子句对应一个模块的退化的模块形式对构造小型应用系统的速成原型可能比较方便,但大多数 PROLOG 语言的单一模块结构却严重影响了它对大型软件的支持。目前对逻辑程序的模块化研究还欠深入,特别是对一些 PROLOG 动态特性问题还没有一致的看法。随着 PROLOG 应用范围的扩大,将模块概念引入 PROLOG 程序设计环境越来越受到广大研究者的关注。模块化也将成为国际标准 PROLOG 的一个重要组成部分,因此对 PROLOG 模块系统的开发是每个 PROLOG 实现者

的一个重要任务。

本文通过对 GKD—PROLOG/VAX 解释器的分析,确定了 PROLOG 模块系统的基本概念和语法,进而实现了 GKD—PROLOG/SUN 模块系统,并对模块化 PROLOG 程序的名冲突问题提出了相应有效的解决方法。该系统不仅完全支持模块系统的模块化,信息隐藏等特点及模块间的各种交互,且为实现层次结构的具有对象特性的模块系统奠定了必要基础。

二、系统概论 我系研制的 GKD—PROLOG/VAX780 解释系统自 1985 年问世以来,已在国内一些大学和研究机构广泛使用。1990 年,我们将其 C 化,并扩充后移植到 SUN 工作站上,主要将原来用数组静态申请栈空间及数据空间改变成用指针动态申请,从而使问题求解空间相应扩大,使其适合于 SUN 工作站容量大,速度高的特点。

GKD—PROLOG/SUN 模块系统是在此基础上进一步研究及实现的,该模块系统允许用户将一个大的 PROLOG 程序分成若干小的子程序(modules),每个模块有它自己的名字空间,模块间的交互由用户事先定义的界面决定。

2.1 基本概念 定义 1:一个模块(即一个模块文件)由模块说明,模块程序两部分组成。

(1)模块说明:给出模块开始标志,模块名及模块外部接口。模块外部接口是一组谓词,这些谓词在本模块中定义,并可供其它模块使用。

(2)模块程序:一组 PROLOG 子句(包括事实,规则和目标等)。

模块文件的 BNF 范式如下所示:

$$\langle \text{module} \rangle :: = : - \text{module} (\langle \text{atom} \rangle, \langle \text{PublicPredList} \rangle).$$

$$\langle \text{clauses} \rangle *$$

```

(PublicPredList)::=(predicate-arg-spec)
(predicate-arg-spec)::=[]
(predicate=arg-spec)::=[(<atom>/<arity>|
(predicate-arg-spec))]

```

其中(PublicPredList)为输出谓词表。

模块说明:—module()必须出现在一个文件的开头,以说明该文件是模块文件。

模块程序中的每个谓词都有模块标识,一个模块程序中所有的用户谓词可分为公用谓词和局部谓词两类,其中可以被其它模块调用的谓词是公用的,非公用谓词对于定义它们的模块是私有的,不能直接被其它模块调用,它对于其它模块是不可见的,公用谓词输入到其它模块后,就可在这些模块中使用。

定义2(谓词分类):

若P是模块M中出现的谓词名,则:

·P是M的公用谓词,当且仅当p在M中定义且出现在M的模块说明中的公用谓词表中。

·p是M的局部谓词,当且仅当p在M中定义且不是M的公用谓词。

·p是M的输入谓词,当且仅当p是其它模块定义的公用谓词,且动态输入到M中。

另外,关于模块中的谓词还有如下几个概念:

(1)p是M中已定义的谓词,当且仅当M中有一个子句,使p是该子句头的主函词,即子句有形式:p(……);—body。

(2)p在模块M中可见,当且仅当P在M中定义,或p是M的输入谓词,即M已从另一模块输入了该谓词,输入的谓词必须是公用的。

(3)在一个模块M中,除公用谓词可输出到其它模块外,其余谓词的作用域局限于该模块本身。这种局限的能力使信息隐藏特性得以实现。

两个模块间的交互通过模块的外部接口实现,即通过外部接口输入(调用)另一个模块定义的谓词,谓词的输入有两种情况:

(1)一个模块M1可以从另一模块M2中输入特定的谓词集,谓词集中的所有谓词在M2中必须是公用的。

(2)一个模块M1可以输入另一模块M2中的所有公用谓词。

2.2 命名规则 非模块化PROLOG的命名机制将被调用的目标名与其数据库中的过程名一致化,而模块化PROLOG增加了模块化特性,实质上就是一个谓词名扩展问题,即当引用一个目标时,系统就命名一个过程来证明这个目标,而可用过程在目标装入时就已确定,而不是在执行时确定,因此目标求解的模块名必须传递给目标调用的谓词,按各类谓

词的使用范围将目标与在调用环境中可见的过程名一致化。

因此对于模块中定义的谓词,根据其作用范围的不同,运用不同的命名规则,以指明谓词的作用域。对于局部谓词,它的模块标识是定义它们的模块的名字。这样就保证了各模块的局部谓词即使名字和参数个数相同也不会混淆在一起,保证了局部谓词作用域的局部性。而对于公用谓词,除了要标明其所属模块名外,还要说明它们可被其它模块调用。在模块程序的运行中,有可能发生名冲突,名字冲突的发生有以下几种情况:

(1)一个模块试图从M1中输入一个谓词,但该模块已从M2模块中输入了具有相同谓词名和参数个数的谓词。

(2)一个模块试图从M1中输入一个谓词,但它已包含了具有相同谓词名和参数个数的谓词的定义。

(3)一个模块试图定义一个谓词,但它已输入了具有相同谓词名和参数个数的谓词。

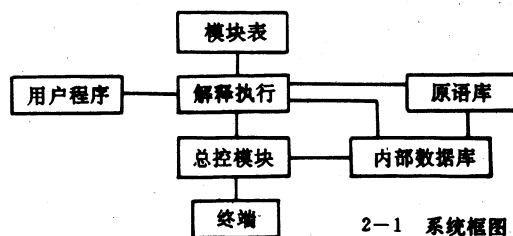
定义3(无名冲突的条件)

设M是任一模块,M要从M1中输入其公用谓词,要保证不发生名冲突必须满足下列条件:

(1) $\text{PublicPredList}(M) \cap \text{PublicPredList}(M1) = \varnothing$

(2) $\text{PublicPredList}(M1) \cap \text{local}(M) = \varnothing$

2.3 组织与结构 GKD—PROLOG/SUN 模块系统主要由总控模块,执行控制模块,原语库,模块表和内部数据库五部分组成。系统框图如图2-1所示。



2-1 系统框图

总控模块主要完成初始化数据库,调入原语索引及系统,用户定义的原语,准备某些数据,从终端上接受命令,将命令送给解释执行模块,执行命令。

原语库由内部谓词库,内部函数库组成;内部谓词库提供了所有内部过程,包括一般PROLOG的所有谓词和有关模块化的谓词,内部函数库提供了所有内部函数。

执行控制模块是整个系统的核心,其主要任务是对用户程序和命令进行解释,归结演绎以求得解,

主要操作是一致化与回溯。

模块表中存放有当前输入的所有模块名及相应的模块文件名,可在解释执行过程中动态改变,它可由内部谓词动态访问,以便随时了解系统中的模块信息。

内部数据库是在程序运行过程中动态形成的,记录了执行过程中系统中的所有模块及各模块内的信息,且保留了所有求解环境及所使用的信息资源。

在 GKD-PROLOG/SUN 模块系统中,内部谓词都作为全局名处理,即存在一个虚拟系统模块,所有 PROLOG 内部谓词都在该模块中定义,其它用户程序不能重新定义这些谓词,其它模块可隐式输入该虚拟模块,直接使用任一系统谓词,但对一些元级调用谓词,如 call/1, clause/1, assert/2, functor/2, retract 等,它们把谓词特性,子句或目标作为参数,这些目标和子句是动态约束的,运行环境不同,作用域也就不同,因此,这些内部谓词必须在程序执行时动态处理。

三、主要实现技术 3.1 数据结构

1. Hash 技术 所谓 Hash 技术,是指对于给定的一个关键字集合,设计一个函数,这个函数把关键字映射到 M 个连续的整数集合中,即 0, 1, 2, ..., M-1, 且把这些整数解释为 hp[0 : M-1] 元素的标号,因而在 GKD-PROLOG/SUN 模块系统中,将所有谓词与其所在模块名一起通过设计的 Hash 函数映射进行存贮,每个 hash 值标号都指向一个 hash 记录串链,这些 hash 记录的 hash 值相同,这样就较好地解决了同名谓词存贮管理冲突问题及各模块内信息的隐藏,从而获得执行的简洁性和高效性。

2. 索引技术 为解决模块存贮问题,我们采用索引技术,建立一个模块索引链,并且为每个模块记录保留两个空域,为将来实现面向对象机制时,指明所属关系及继承关系,该链表是动态的,可自动插入和删除任一记录,保证与当前内存环境相一致,此链表表头指针为 pmohead。每个模块记录结构如下:

```
structmodurc
{
    char          * charmo;      模块名
    char          fname[20];     模块所属文件名
    char          * isa;         面向对象中模块
                                的所属关系
    struct super-relation * super; 面向对象中的
                                父类关系链
    struct modurc * next;        下一模块记录
}
```

另外,具有相同模块名和参数个数的谓词定义过程通过 hash 记录中的 claus 域中的 next 域串链在

一起。子句记录结构如下:

```
struct claurc
{
    struct claurc * next;      指向子句头相同的下一子句
    bytes         varn;        子句中变量个数
    boolean        local;      指示子句中的变量是否为局
                                部变量,局部变量在归结过
                                程中其空间可以回收。
    struct noderc * sonl;      指向子句头结点
    struct noderc * term;     指向子句的根结点记录;
}
```

3.2 主要算法 在逻辑程序设计中有三个主要的算法,即语法分析算法,一致化算法和归结算法 GKD-PROLOG/SUN 由于引入了模块的概念,从而使这三种算法进行了相应的改变。

1. 语法分析算法:逐行分析用户的输入,识别输入的是命令还是文件名,若是命令则由归结算法解释执行,若为文件名,则对文件进行分析,建立文件的内部知识结构。

GKD-PROLOG/SUN 的语法分析算法完成了模块的识别,模块的静态输入及模块表的动态改变,每个文件都静态定义了一个模块。

2. 一致化算法:完成名和参数个数相同的谓词中所有参数的一致化,对参数逐个进行一致化。

在 GKD-PROLOG/SUN 中,每个谓词都有模块标识,但函词,变量,常量是全局的,不具有模块特征,因此一致化过程中,不比较两个 hashrc 中的模块名域,保证了它们的全局名特征。

3. 归结算法:将求解的目标与其动态求解环境中的子句按深度优先搜索和最左归结法则进行一致化匹配,以求得目标的解。

这里所提到的动态求解环境指的是在程序运行过程中动态确定的求解目标的模块环境,求解过程中所能运用的知识就是这一模块中的所有子句,与其它未使用的模块无关,但在执行过程中可以运用系统提供的原语动态调用从而使用其它模块中的信息。归结算法的流程图如图 3-1 所示。

在 GKD-PROLOG/SUN 模块系统中,内部谓词及系统操作原语都做为全局名处理,属于一个虚拟的全局模块 global,因此在各模块的归结过程中可直接调用,只有用户定义的谓词过程是各模块私有的,因此在归结时要根据动态改变的求解环境,在适当的模块环境中求解目标。

四、模块的操作 4.1 模块装入

1. 静态装入一个模块 静态装入一个模块必须先建立一个模块文件,在 GKD-PROLOG/SUN 模

块系统中,一个模块文件仅含一个模块程序,包括模块说明和一组 Prolog 子句,模块说明必须位于文件的开头,定义好模块文件,即可将它装入系统,系统对该文件进行语法分析,解释,并在内存创建了一个模块,该文件中定义的所有子句均放入该模块中。

一个模块的定义并不仅限于一个单独文件,因为一个模块文件可能包含装入其它文件的命令,若模块文件 M 包含一个内部命令装入文件 F,且 F 不是模块文件,则 F 中的谓词都装入模块 M 中。

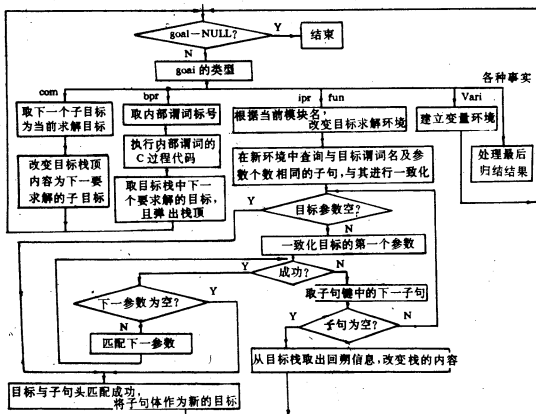


图 3-1 归结法流程图

2. 动态装入一个模块 除了创建和装入一个模块文件,还可动态定义一个模块,如将子句插入到一个指定的模块中,以这种方式创建的模块没有公共谓词,它的所有谓词均是私有的,即从外部模块不能调用该模块中的所有谓词。

为了使用另一模块文件中的公共谓词,在编写一个模块程序时可以用一个内部谓词动态装入另一个模块文件 M,这样在系统中就含有模块文件 M 的所有内容,同时将 M 中的所有公共谓词以及它们的定义都装入到调用该内部谓词的模块中。动态装入模块的内部谓词形式为:

`use-module (Filename)` 其中,Filename 必须为已特例化的模块文件名

另一类似的内部谓词为:`use-module (Filename, ImportPredList)`

其中,ImportList 是一个形为 Name/Arity 的属性表。

该谓词一方面将模块文件 Filename 装入数据库中,同时将 ImportPredList 中定义的公共谓词装入调用该谓词的模块中。

使用 `use-module/1`,当前环境模块装入的是动

态装入模块所包含的所有公用谓词,若它的公共谓词集改变了,则包含 `use-module/1` 语句的模块需重新装入,而用 `use-module/2`,则没有这个问题,因为它输入的谓词集是特定的公共谓词,使用 `use-module/2` 可使模块间的交互清晰,可靠。

4.2 谓词名扩展

在模块系统中,目标证明隐含地在当前模块环境中进行,但有时也可对个别目标证明指定另一模块环境。

“:”是一个二元操作符,定义为中缀形式,左边的参数表示一个模块名,“:”右边的参数是一个目标,“:”表示无论现在的模块环境是什么,“:”右边定义的目标必须在“:”左边定义的模块中求解,求解完毕再返回到当前模块环境。在执行“:”操作时,其参数必须是已特例化的。

引入了“:”二元操作后,一个模块 M 运行环境可见的谓词包括:

(1) 所有内部谓词;(2) 所有 M 中定义的谓词;

(3) 在另一模块 M1 的 PublicPredList 表中说明的谓词 p,且 M 已从 M1 中输入 p;

(4) 任意用“:/2”操作符与其它模块连接的谓词,不管该谓词是否为公用谓词如 `mod:make-object(-,-,-,-,-)`,即在模块 mod 中调用 `make-object/5`。这样就可忽略一般的模块可见规则,可方便程序的开发和调试。但它毕竟破坏了模块系统的信息隐藏特性,一般不要滥用。

4.3 执行环境的切换 关于执行环境的操作包括如下三个方面;

1. 显示当前环境 `active-module(P)`。

系统遇到这个目标就将其参数与当前所处的环境模块名进行匹配,将匹配结果返回,这时 P 特例化的值就是当前模块名。`in-module(P)`。

P 必须是已特例化的,表示一个模块名,否则报错,系统遇到这个目标时,就在模块表中寻找模块 P,若找到,则求解成功,否则失败。这个内部谓词供用户检查 P 是否是已有模块,以便用户确定应对模块 P 执行什么操作。

`current-module.`

该内部谓词顺序搜索模块表,列出模块表中的所有模块名。

`current-module(Mod,File).`

若该内部谓词的两个参数均未特例化,则列出当前模块表中的每个模块名及其对应的模块文件名,若两个参数均已特例化,则判断模块表中是否有这样的模块和相应的模块文件名。若一个已特例化,而另一个未特例化,则根据特例化的值找到相应的

变量值。

2. 改变当前环境模块 module/1 若系统当前所处的模块为 M1, 要使当前模块变为 M2, 可用内部谓词 module(M2), 但对 M2 有以下限制;

(1) 在顶级使用该谓词, 则 M2 必须是常量, 且 M2 必须已存在于模块中, 若为变量, 则系统报第一参数错, 若 M2 不在模块表中, 则系统执行命令失败, 不改变当前模块环境。

(2) 若在程序中使用该谓词, 则当执行这一命令时, M2 必须是已特例化的, 否则系统报第一参数错, 若 M2 已特例化, 但特例化后的模块名不存在, 则系统自动创建一个模块, 将其加入模块表中, 且将当前模块环境转到这一模块。

3. 改变数据库状态 系统提供了若干内部谓词对任意模块中的子句进行插入, 删除, 为了与非模块化 Prolog 系统兼容, 我们保留了原有 asserta/1, assertz/1, assert/2, retract 的功能, 而增加了新的谓词 assert/3, asserta/2, assertz/2, 具体功能如下:

asserta(c, m), assertz(c, m), assert(c, n, m): 将子句 c 插入到模块 m 中, 做为同名过程的第一个子句或最后一个子句, 或第 n 个子句。retract(c, m): 在模块 m 中, 删除与子句 c 匹配的最后一个子句。

4.4 其它与模块有关的内部谓词

1. 列出当前环境模块中的所有子句 list, list/1。

list: 列出当前模块中的所有事实与规则。

list(p/a): 列出当前模块中已有的子句头为 p 且参数个数为 a 的所有事实与规则。

list 不象非模块系统中列出数据库中的所有子句, 使信息不具有保密性, 修改后的 list 使当前模块能看到本模块内的信息而不能看到其它模块中的信息。

2. 列出有关谓词的信息。

current—predicate 列出数据库中所有已定义的谓词及其参数个数, 每个谓词名后都标明了其所属的模块名。

M:current—predicate 列出模块 M 中所有已定义的谓词及其参数个数, 每个谓词名后都标明了其所属的模块名。

M:current—predicate(p(—, ..., —)) 判断模块 M 中是否存在名为 p, 且参数个数与 p 相同的谓词, p 必须是常量或已特例化。

由于各模块间可以进行交互, 一个模块可以输入且使用另一模块中定义的谓词, 因此在内部数据库中存在的谓词具有不同的属性, 谓词的属性不同, 使用域也不同, 我们将谓词的属性分为三种;

(1) 谓词是内部的, 即该谓词在某一模块内定

义, 且仅允许在此模块内调用, 用 interpre 标记;

(2) 谓词是公用的, 该谓词在模块内定义, 且出现在该模块的模块说明中的公用谓词表中, 可被其它模块输入调用, 由 publicpre 标记;

(3) 谓词是输入的, 该谓词是在执行过程中从另一模块中输入的, 是另一模块中的公用谓词, 用 importpre 标记。

为随时了解谓词在执行过程中所具有的属性, 从而正确地使用各模块定义的谓词, 系统提供了内部谓词 predi—property(p, proerty), 以确定 p 的属性, 谓词中的 p 必须是常量或已特例化的变量, 若 property 未特例化, 则特例化谓词 p 的属性。若已特例化, 则看是否与 p 的属性一致, 若一致, 则求解成功, 否则, 失败。

五、结论与未来的工作 GKD—PROLOG/SUN 模块系统已在 SUN 工作站上用 C 语言实现了, 它允许大型软件系统的结构化, 信息隐藏和抽象数据类型实现。在每个模块文件的开始, 都有一个模块说明, 指明模块的接口信息, 以支持各模块间的动态调用。模块化系统的实现, 为实现面向对象 prolog 程序设计环境奠定了基础。用模块封装对象, 将消息作为目标, 向一个类发送消息就可以认为是要求在某个模块中实现对消息的处理, 即对目标的求解。对父类的继承可作为求解环境的动态扩充。我们今后的工作是在已实现的模块系统中嵌入面向对象的概念, 实现一个面向对象 prolog 程序设计环境。

参考文献

[1] D. T. Sannela, and L., "A. Wallen, A Calculus for the Construction of Modular Prolog Programs", Procc, SLP' 87, Sept, 1987.

[2] Rolanld Dietrich, "A Preprocessor Based Modules System for Prolog", Proc. TAPSOFT' 89, March 1989.

[3] Jan Chomicki and Naftaly H. Minsky, Towards "a programming environment for large Prolog programs", Proc. Symposium on Logic Programming, 1985.

[4] 金芝, 邓铁清, 胡守仁, "PROLOG 模块机制的一元级扩充的编译实现技术", 《逻辑程序设计与 AI 语言论文集》, 1991, 大庸。

[5] 《Quintus Prolog Reference Manual》, Quintus Computer Systems, Inc. Nov. 1987.

[6] L. Leonardi and P. Mello, "Combining Logic— and Object— oriented Programming Language Paradigms", HICSC 21, Hawaii, January 1988.

高级语言中越级返回的实现

江西工业大学计算机系 谢果然 李 萍 (330029)

摘 要 本文通过分析 Turbo Pascal 5.5 的过程调用与返回机制, 提出并实现了在 Turbo Pascal 5.5 语言的程序中越级返回的方法。该方法的原理可用于实现其他高级语言的越级返回。

一、引 言 一般来说, 在过程型程序设计语言中, 子程序的一系列调用, 比如子程序 A 调用子程序 B, B 调用子程序 C, C 调用子程序 D, 若要从 D 返回 A, 只能逐级返回, 即 D 返回其调用者 C, C 返回到 B, B 最后返回到 A。但研制较大型的软件系统往往需要从 D 直接返回到 A, 尤其是在系统运行中出现错误时, 我们总希望从出错点直接返回到主控程序。在研制符号演算系统 CASC 时, 就遇到这个问题。遗憾的是, 目前绝大多数高级语言都不提供这样的设施, Turbo Pascal 5.5 亦是如此。解决这一问题的做法是在出错点直接调用主控过程, 但这样做的不妥之处是显而易见的。这种递归调用将使系统剩余的栈空间迅速减少而导致因栈溢出的非正常结束。

通过分析 Turbo Pascal 5.5 的过程调用机制与返回机制, 本文提出了圆满解决这一问题的方法, 这种方法不只是针对 Turbo Pascal 5.5, 其原理也适应其它的高级语言。

二、过程调用机制与返回机制分析

1. 过程调用 Turbo Pascal 语言程序的执行代码在调用一个过程前, 先将各参数(值参自身或变参地址)依次进栈, 然后执行调用指令 CALL。CALL 指令执行过程是将其下一条指令的地址压入堆栈, 在 FAR 模式下, 先将指令的段地址(CS 之值)进栈, 再将指令偏移地址(IP 之值)进栈, 以便返回之用, 随后更新 CS, IP, 执行被调用的程序。其它高级语言类似。

2. 过程入口工作 执行 CALL 指令后, 即进入被调用程序入口。程序入口总是用如下两条指令:

PUSH BP

MOV BP, SP

这两条指令使 BP 和 SP 两个寄存器的地址指针指向内存中与堆栈段寄存器相应的同一位置。为在堆栈中寻址与说明局部变量作准备。在此 MOV 指令之后将在堆栈中为局部变量分配空间。若用编

译指令 {SS-} 关闭堆栈错误检查, 则不作堆栈检查, 直接为局部变量分配栈空间; 否则总是先检查是否有足够的堆栈空间来存储局部变量, 若有才为局部变量保留空间, 否则作出错处理。

3. 过程出口工作 在过程执行完退出过程之前, 执行如下三条指令, 返回到调用该过程的指令之后的那条机器语言指令。

MOV SP, BP

POP BP

RET

或

MOV SP, BP

POP BP

RET N

其中 N 是参数所占空间的字节数。

指令 MOV 将 BP 的值赋给 SP, 恢复了为局部变量分配堆栈空间之前的原堆栈指针值。在过程开始时, BP 等于 SP; 因此, 只要没有其它指令或子程序改变 BP 的值, 这就是在结束过程之前恢复堆栈指针最快的方法。第二条指令 POP 从堆栈中弹出一值给 BP, 从而恢复在过程开始的 BP 的值。RET 指令将栈顶的 4 个字节(在 FAR 模式调用下的返回地址)重置 CS, IP; 若 RET 后带 N, 则由 SP 加上 N 修改 SP 之值来释放参数所占的堆栈空间, 并使 SP 恢复到调用前的值。

举一个简单的例子[1], 考察一下在调用过程期间堆栈的情况, 图中右边的注释指出了过程调用期间不同时刻的 SP 和 BP 位置。左边的注释指出了和 BP 寄存器相关的变量位置。图中对应的过程说明如下:

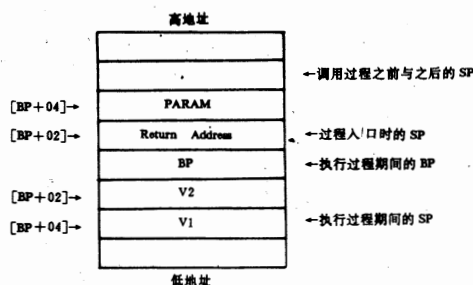
PROCEDURE TYPICAL(PARAM; WORD);

VAR V1, V2; WORD;

这个假想的过程带一个 WORD 型参数 PARAM 和两个 WORD 型局部变量 V1 和 V2, 在机器语言中, 过程的调用者执行如下指令:

PUSH AX; 一个 WORD 型参数

CALL TYPICAL; 调用过程



这里假定要传递的参数当前已在 AX 中。当 TYPICAL 开始运行时, SP 指定返回地址——即 CALL 指令之后的第一条指令的地址; 在堆栈中, 参数在返回地址之上。

TYPICAL; 先执行如下指令:

PUSH BP

MOV BP, SP

SUB SP, +04

PUSH 指令将 BP 的当前值压入堆栈保存, 正好在返回地址之下; MOV 指令将堆栈指针 SP 寄存器的值送到 BP, SUB 指令从 SP 中减去 4, 为两个 WORD 型局部变量 V1 和 V2 保留 4 个字节, SP 指向 V1 的第一个字节。

在 TYPICAL 的出口执行如下指令:

MOV SP, BP

POP BP

RET 0002

MOV 指令恢复 SP, 释放为局部变量 V1 和 V2 分配的对栈空间, POP 指令恢复过程开始时的 BP 之值, RET 告诉处理器从堆栈中删除返回地址, 将该地址重置 CS, IP, 并将 SP 加上 2 释放参数所占的两字节栈空间, 使 SP 恢复到调用之前的值。

三、越级返回的实现 通过以上分析, 要正确的从 D 返回到 A, 就是要编写一个模拟过程来模拟并完成 B 返回 A 的过程, 也就需要返回到 A 时应保证有正确的 SS, SP, BP, CS, IP 值, 并释放调用 B 后分配的一切栈空间; 同时必须保护数据段寄存器 DS, SS, SP, BP, DS 值是容易得到保证的。但我们知道 CS, IP 的值是不容许程序编写者直接修改的。我们所能采取的方法是: 模拟过程的出口工作完成过程 B 的出口工作, 并采用将返回地址移至 B 的参数区开始的 4 个字节以释放调用 B 后所分配的一切栈

空间, 按照这一思路, 我们编写了如下模拟过程。

; Turbo Pascal 过程说明

; Procedure Exitto (ODS, OSS, OSP, OBP, OPR, word); External;

; 调用格式:

; Exitto(ODS, OSS, OSP, OBP, OPR)

; ODS, OSS, OSP, OBP 为调用 B 过程前一刻的 DS, SS, SP, BP 值。

; PR 为 B 过程的参数所占字节数

; 调用 B 后, 由于已为 B 的参数 (如果有的话) 分配栈空间 PR 字节

; 保存返回地址占 4 字节 (FAR 模式), 故在 B 的入口处 $SP = OSP - 4 - PR$

; 这里需要将 SP 处的返回地址移至 $OSP - 4$ 处

EXITTO PROC FAR

PUBLIC EXITTO

MOV Bp, Sp

MOV BX, [BP+04] ; 取 PR

MOV DX, [BP+06] ; 取 OBP

MOV SP, [BP+08] ; 取 OSP

MOV AX, [BP+10] ; 取 OSS

MOV CX, [BP+12] ; 恢复 DS

MOV DS, CX

SUB SP, 4 ; SP 指向将建立的返回地址处

MOV BP, SP

Sub BP, BX ; BP 指向调用 B 的 B 的返回地址

MOV SS, AX ; 恢复 SS

MOV BX, SP

MOV AX, [BP+2] ; 将返回地址移至

MOV SS, [BX+2], AX ; 过程参数的开始位置

MOV AX, [BP] ; 在新位置建立返回地址

MOV SS, [BX], AX

MOV BP, DX

RET

EXITTO ENDP

; 取 BP 当前值

GETBP PROC FAR

PUBLIC GETBP

MOV AX, BP

RET

GETBP ENDP

软磁盘文件加锁及高道局部特殊格式化扇区缝隙加密的技术

长春邮电学院 曹尔强 张 沂 李宝岩 潘继宏(130012)

摘要 本文叙述用“软件黑盒子”对任意文件加锁及用高道局部特殊格式化扇区缝隙加密的技术。由于对文件和整片磁盘都加了锁,它能确保被加密文件中的秘密不被泄露,用 Copywrite, CW877 等解密软件也无法拷贝。又由于我们加了反跟踪,即使熟练地掌握了汇编语言的人也无法用 Debug.com 调试程序进行跟踪。

关键词 软件黑盒子,磁盘高道局部特殊格式化,扇区缝隙加密,文件加锁。

1. 引言 软磁盘的加密、加锁技术,五花八门,而且各有千秋。但有矛就有盾,加密与解密,历来都体现着矛盾的对立与统一。我们在总结前人工作的基础上,开发了一套自己的方法,供同行及感兴趣者批评、研讨。

2. 数学模型及程序设计原理框图

文件加锁原理公式为

$$y = f(x) \odot B$$

其中, X 代表源文, Y 代表密文, B 为密码;
f 表示某种算法, \odot 表示某种运算。

所谓“软件黑盒子”对文件加锁,就是基于上述基本原理。其设计原理框图见图 1。请注意,在这里,密钥 B 可取任意长短,且随机产生或键盘输入均可;算法也可采用加减乘除等四则运算,或者与或非、异或等逻辑运算,乃至左移、右移等处理均可。只需注

意解锁过程必须是加锁运算的逆运算即可。

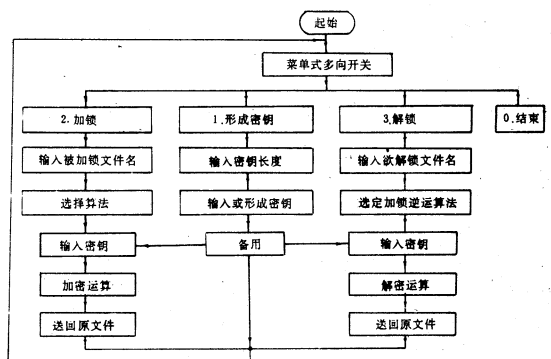


图 1 用“软件黑盒子”对任意文件加锁及解锁程序设计方框图

如果程序中不改变 DS 与 SS,则还可以去掉 EXITTO 的前两个参数。

下面是越级调用的一个样板程序,将 ODS, OSS, OSP, OBP, OPR 说明为全局变量。DSEG, SSEG, SPTR 为 Turbo pascal 提供的函数。有些高级语言不提供这一类函数,但这类函数容易用汇编语言实现(如上面的 GETBP)。

```

...
Procedure Main;
Var.....
Begin
.....
ODS:=Dseg;OSS:=SSeg;
OSP:=SPtr;OBP:=GetBP;OPR:=2;
Typical(4);
.....
End;
Procedure Typical(ParaM;Word);
Var V1,V2;Word;
Begin
.....
C;{调用过程 C}
.....
End;
```

Procedure C(.....);

...

Begin

...

D;{调用过程 D}

...

End;

Procedure D(...);

...

Begin

...

If Error Then Exitto(ODS,OSS,OSP,OBP,OPR);
{若出错,则返回 MAIN}

...

End;

四、讨论 在用 Turbo Pascal 5.5 实现 CASC 时,我们成功地使用了这个方法,获得了很好的效果。显然该方法作适当修改,亦可用于其它高级语言解决越级返回的问题。

参考文献

[1] 刘京等编译,《TURBO PASCAL 5.5 程序设计技术及库函数集锦》下册,中国科学院希望高级电脑技术公司。

这种加锁方式,对任何文件均可加锁,形式又灵活多样,使不知密钥及算法的人很难解。

为了保险起见,我们对整张磁盘进行了高道局部特殊格式化及扇区,缝隙加密。具体作法是局部特殊格式化高道扇区,在原来应该是噪声处的扇区缝隙处“随机地”加入密码作为我们自己的识别信息。然后在可执行文件的入口处加上一小段识别程序,只有识别无误才能进入执行真正的程序,否则,则指出为非法拷贝盘,从而达到进一步加密的目的。其程序设计原理框图示于图2。

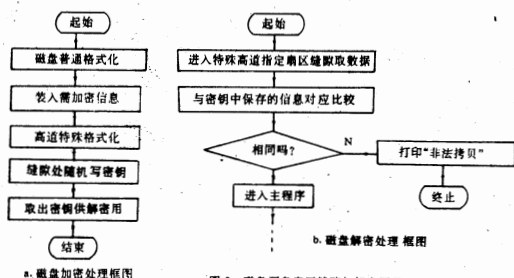


图2 磁盘扇区缝隙加密解密原理

这样,我们就为需要加密的文件加了两把锁,并把解锁的密钥放在另一张解密盘上。为了防止解密盘被截获失密,对解密盘本身也进行了缝隙加密,并采取了反跟踪措施。

由于扇区缝隙信号是些无法用常规方法拷贝的微弱噪声信号,而我们将殊加密的信息采用“真随机”的形式,就使我们的密码十分隐蔽。当我们的加密盘被中途拷贝、掉包时,我们一运行解锁程序马上就会发现。从而采取应急处理。

3. 试验及结果举例

上述加密方法,对任何文件均适用,假设母盘上有如下一段 BASIC 源程序:(P. BAS)

```

5H=3100:V=200
10 OPEN"COM1:9600,e,7,1"AS#1
20 PRINT #1,"J1"
30 GOTO 500
40 PRINT #1,"L-1"
50 FOR I=0 TO 3.1416 STEP .1
60 IF 3.1416-I<.1 THEN I=3.1416
70 X=H+200*COS(I)
80 Y=V+60*SIN(I)
90 IF I=0 THEN PRINT #1,"M",X,Y ELSE
PRINT #1,"D",X,Y

```

当加锁程序对母盘高道扇区加密信息识别无误后,即进入如下加密过程:

```
please input lock file name,c;p. bas
```

```
please input your word:1234567890
```

```
FILE LOCK OK!!
```

此时,再打印加密后的这段源程序为:

```
A>type c:d. bas
```

```
( f
```

```
8<aw>UxU
```

显然是不可识别的。

这时,假设将之拷贝到另一盘子上,而且用我们自己的解密盘解锁,则由于缝隙信息读出错误,则显示:

```
A>lockl
```

```
sorry! you are illegal copy!
```

如果是母盘本身,则运行解锁程序后,再打印 P. BAS,则自然会恢复它本来的面貌。

实际上,我们制作了一套、两张 $5\frac{1}{4}$ " 盘的表演程序,一张盘上放需加、解密的文件、程序,另一张盘上放高道格式化,加解密,加解锁,以及密钥等程序、文件,并具体对我们的技术使用说明进行了加密。有兴趣者,欢迎来拷贝被加了密的程序或文件,回去试图解密。

4、讨论与展望 我们的软盘文件加锁及磁盘加密技术,将“软件黑盒子”技术,磁盘特殊格式化高道扇区缝隙加密技术、反跟踪技术等结合起来,为需要加密的文件加了双层锁,对解密的钥匙又加了一把锁。磁盘扇区缝隙加密技术及反跟踪技术,保证了我们文件的不可拷贝性及缝隙密码的不可破译性;而“软件黑盒子”的可变长度随机密钥及算法的多样性,使“一次破译”的概率几乎为零,从而最大限度地实现了保密性。

当然,加锁、加密的方式、方法很多,经过一年多的探讨、摸索,我们认为这种组合方式最好。

预计本方法将在保密通信中大有用武之地。

参考文献

- 1、[美]Robert Jourdan,《IBM 编程指南》,电子工业出版社,1988.10.
- 2、[美]Steven Armbrust,《DOS/BIOS 使用详解》,电子工业出版社,1989.4.
- 3、夏东涛,《磁盘机原理与软件加密解密实用技术》,科海总公司培训中心,1988.5.
- 4、张载鸿,《局部网操作系统 DOS 高级技术分析》,国防工业出版社,1988.12.
- 5、王晓武,毛长凤,“IBM-PC 实用软盘加密原理与实现”,计算机世界,1990.3.
- 6、于功弟,路枝,“利用软件黑盒子对 PC 机文本文件加密的原理与方法”,计算机应用研究,1990,7(6):14-16.

EGA/VGA 图形方式下 FOXBASE+ 屏幕叠加式菜单的实现

上海现代信息技术研究所 吴邦忠 夏 英(200233)

屏幕叠加式菜单给用户一个良好的界面,现在许多从国外引进的工具软件、软件包绝大部分都采用这种菜单形式提供给用户选择有关操作,使用起来十分直观也相当方便。屏幕叠加式菜单的实现关键是如何正确地保存和恢复屏幕信息。用 FOXBASE+ 语言在文本显示方式下,编制屏幕叠加式菜单可以方便地使用系统所提供的 save screen 命令来保存原来屏幕菜单的信息,当返回上级屏幕菜单时,用命令 restore screen 就可以完成。但是 FOXBASE+ 提供的这种屏幕存取功能仅适用于文本显示方式,而不支持图形方式,由于中文操作系统(CCDOS)均都在图形方式下工作,所以这使国内众多的 CCDOS 用户在 FOXBASE+ 语言编制有关数据库管理软件的多级菜单时,往往都难于采用这种屏幕叠加式菜单方式,又在 EGA/VGA 图形方式下,用目前一些有关书刊介绍的“简单地直接存取显示存储器”的方法不能实现屏幕彩色图形的存取。

针对这个问题我们分析了 EGA/VGA 图形控制器的基本工作原理,在微机上进行一些摸索实践(286, VGA 高分辨彩色显示器),并根据 FOXBASE+ 能够用 call 命令可直接调用汇编语言模块的特点,编制了一个在图形方式下屏幕保存和恢复的汇编语言程序(S_RGRAPH, ASM 其源程序清单见附录)。为使大家能方便地阅读和理解该程序,在给出该程序使用方法之前我们先介绍一些 EGA/VGA 图型控制器有关方面的基础知识。由于 EGA/VGA 的高度兼容,且 VGA 图形控制器的端口地址与 EGA 相同,所以这里仅介绍 VGA 图形控制器:

1、显示存储器: VGA 显示存储器由 4 个平行的位平面所组成,且 4 个位平面合用相同空间地址,它们的首地址为:A000:0000H。显示存储器中第一位相对应于屏幕上的一个象素,而该象素位可以映射到 4 个位平面上(见图 1),象素的颜色由 4 个位平面上相对应的位值所确定,这样每个象素都具有值 000-1111B,相对应着屏幕显示的 16 种颜色。

2、操作定序和图形控制寄存器 VGA 图形控制器中有许多寄存器。其中操作定序寄存器和图形控制器见表 1,通过对这些寄存器的操作可以实现屏幕上象素的读写。

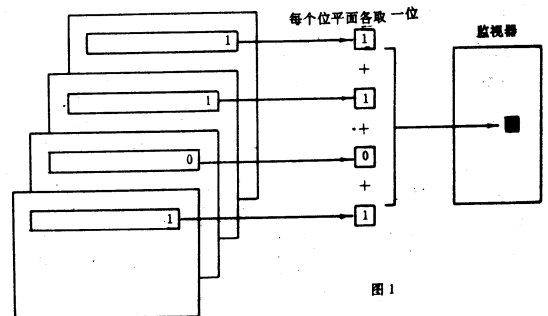


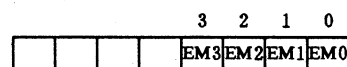
图 1

表 1:

端口	寄存器	索引号	索引端口
3C5	复位寄存器	0	3C4
	时钟模式寄存器	1	
	位平面屏蔽寄存器	2	
	字符发生器选择寄存器	3	
	存储器模式寄存器	4	
3CF	设置/重置寄存器	0	3CE
	使能设置/重置寄存器	1	
	彩色比较寄存器	2	
	数据循环寄存器	3	
	图选择寄存器	4	
	模式寄存器	5	
	地址控制寄存器	6	
	彩色无关寄存器	7	
	位分离寄存器	8	

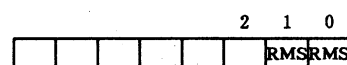
由于篇幅有限这里只介绍在程序中用到的位平面屏蔽寄存器和图选择寄存器。

位平面屏蔽寄存器:



当 CPU 送出一个字节到指定地址的显示存储器去时,这个字节可以同时写入 4 个位平面中去,也可以只写入到其中部分平面中去。对位平面的写入控制是由这个寄存器的 EMn(n=0-3)分别负责的。当 EMn=1 时,允许写入位平面 n(n=0-3)

图选择寄存器:



CPU 对显示存储器进行操作时,4 个位平面总是同时读出送到寄存器中的,因一次只能向 CPU 送一字节,此寄存器的 RMS 字段就用来选择 哪一个位平面中的读出数据字节送到 CPU 去,如二位 RMS=01,代表位平面 1 中读出数据字节送 CPU。

VGA 有四种写模式可以设置像素的值(EGA 三种)两种读模式,用来取出像素的值,通常 BIOS 中把写模式 0 和读模式 0 作为默认模式。我们编制的如下屏幕保存程序是逐个将 4 个位平面写入文件,屏幕恢复程序是将保存在文件中的 4 个位平面的值再相应写入 4 个位平面上去。

3、程序使用方法: 根据 FOXBASE+对汇编程序的要求,在使用屏幕存取程序前,应将该程序转换成二进制代码文件 S __RGRAPH. BIN。其步骤为:

- 1、输入源程序 S __RGRAPH. ASM
- 2、C>MASM S __RGRAPH;
- 3、C>LINK S __RGRAPH;
- 4、C>EXE2BIN S __RGRAPH

使用时须按 FOXBASE+调用汇编程序的方法,用 LOAD S __RGRAPH 将程序装入内存,要保存屏幕时用 CALL S __RGRAPH with“文件名/S”,要恢复屏幕时用 CALL S __RGRAPH with“文件名/r”。

由于程序是用汇编语言编制的,所以执行速度非常快,经多次使用效果很好。

注:文件名长度为 4 个字符,在 EGA 状态下,只要将源程序中 di 的值改为 28000。

C)

附录 S __RGRAPH. ASM 清单

```
code    segment
        assume cs:code
vga     proc far
        push ss
        push ds
        push bx
        mov al,ds:[bx]
        or al,al
        jz ex0
        mov di,38400
        mov al,byte ptr ds:[bx+5]
        mov byte ptr ds:[bx+4],0
        mov dx,bx
        cmp al,'s'
        jz s0
        cmp al,'r'
```

```
        jz r0
ex0:    jmp ex2
;
s0:     mov cx,0
        mov ah,3ch
        int 21h
        mov bx,ax
        mov ax,0a000h
        mov ds,ax
        mov dx,0
        mov cx,di
        mov si,0
lo0:    call s __c
        inc si
        cmp si,4
        jnz lo0
        jmp ex1
;
r0:     mov ax,3d00h
        int 21h
        mov bx,ax
        mov ax,0a000h
        mov ds,ax
        mov dx,0
        mov cx,di
        mov si,1
lo1:    call r.c
        add si,si
        cmp si,16
        jnz lo1

        jmp ex1;
s__c   proc near
        push dx
        mov dx,3ceh
        mov al,4
        out dx,al
        mov ax,si
        inc dx
        out dx,al
        pop dx
        mov ah,40h
        int 21h
        ret
s__c   endp
;
r__c   proc near
        push dx
        mov dx,3c4h
```


一种用于自然纹理图象分割的并行算法

西安交通大学人工智能与机器人研究所 管旭东 刘健勤 王爱群(710049)

摘要:本文提出一种基于统计随机场建模的并行算法,以完成对自然纹理图象的有效分割。该算法构成并行层次化结构,其特征向量可有效地表达空域纹理模式特征,从对羊皮纹理处理结果来看,该算法是高效和实用的,可作为皮革质量计算机分析系统的基本手段。

一、引言 随着人工智能技术的发展,计算机图象处理已从传统的灰度信息处理,发展到空间纹理特征的聚类分析,特别是通过二维随机场的数学建模可有效地反映模式属性。在随机场SAR模型的参数辨识基础上,笔者提出一种基于统计分析的并行分割算法,以有效地完成自然纹理图象的分割操作。

二、并行层次化的纹理图象分割 该算法为并行层次化结构,共分为三层:(1)并行随机场建模与特征提取、(2)基于向量分析的边缘检测、(3)边缘跟踪操作。其结构示于图1。

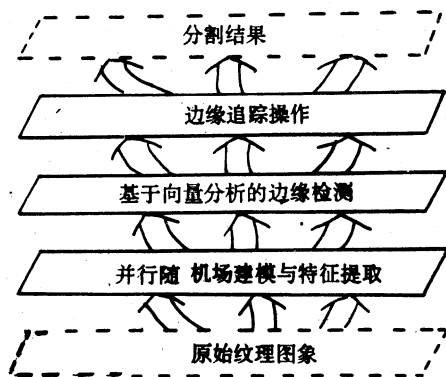


图1 算法结构

针对纹理图象,一般可采用空域统计法和频域分析法,通过二维傅里叶变换映射,可在频域内进行定量操作,但该变换的运算时间太长,且只能反映宏观特性,不易描述纹理模式的精细特征。空域统计法主要是根据统计决策理论,以统计模式识别机制,完成空域向量的分类,其中纹理特征的选择与提取是关键性的步骤,特征计算的结果直接决定纹理图象处理的效率和精度。

为有效地提取自然纹理特征,二维非因果SAR模型被用来表示相应模式向量的空间属性。通过系统参数辨识,构造局部纹理特征向量。向量空间的分布是分割判断的基础。

通过窗口的向量均匀性(uniform)判断,检测出边缘的存在,并加以适当标记。边缘跟踪操作着重解决边界的连续性问题,并消除相应边缘的语义二义性。

三、并行随机场建模与特征提取 本文采用一种二维非因果随机场模型——SAR(Simultaneous Autoregressiv)模型,通过空间邻域位置的特征分布,构造反映纹理空间特性的模式向量。

设二维矩阵 $\{g(x,y); x,y=0,1,\dots,M-1\}$ 表示离散 $M \times M$ 图象的灰度值, x,y 分别表示 x,y 轴坐标值,满足 $M \times M$ 阵列上SAR模型的 $g(X,Y)$ 可表示如下:

$$g(x,y) - \mu_g = \sum_{(i,j) \in N^{Q(a,b)}} Q_{a,b}(g(x+i,y+j) - \mu_g) + \sqrt{\rho_N} \omega(x,y)$$

```

mov al,2
out dx,al
mov ax,si
inc dx
out dx,al
pop dx
mov ah,3fh
int 21h
ret
r.c endp
  
```

```

;
ex1
mov ah,3eh
int 21h
ex2:
pop bx
pop ds
pop ss
ret
VGa endp
code ends
end VGa
  
```

$$x, y = 0, 1, \dots, M-1 \quad (1)$$

其中, N 为空间邻域。

$Q_{(i,j)}$ 为模型参数, 反映邻近像素点间在空间的数值分布关系。在特殊情况下, 对称邻域可用 $N: \{(i,j), (-i,-j) \in N\}$ 来表示。

$\omega(x,y)$ 为零均值、单位方差的独立同分布高斯随机变量。

P_N 为噪声的整体方差, 反映了纹理随机变化的程度。

本文中, 邻域 N 选取为两组:

$$N_1 = \{(1,0), (0,1), (-1,0), (0,-1)\}$$

$$N_2 = \{(1,1), (-1,1), (-1,-1), (1,-1)\}.$$

这两组邻域共同反映了纹理空间属性, 从两个方面反映了统计规律的空域测度, 对自然纹理的计算效果可看出, 这样的选择既减少了运算时间(比只用一个包括较多参数的模型的辨识, 计算开销明显减少), 又增强了向量的聚类性能。

在每个窗口内, 分别建立针对 N_1, N_2 的两个 SAR 模型, 利用最小二乘方估计, 可快速估计出该模型的参数: $\{Q_{(1,0)}, Q_{(0,1)}, Q_{(-1,0)}, Q_{(0,-1)}\}$ 与 $\{Q_{(1,1)}, Q_{(-1,1)}, Q_{(-1,-1)}, Q_{(1,-1)}\}$ 还可以计算出 P_{N1}, P_{N2} 。

由于 N_1, N_2 的对称性, 其参数也表现出对称性, 故选取 $\{Q_{(1,0)}, Q_{(0,1)}, P_{N1}, Q_{(1,1)}, Q_{(-1,1)}, P_{N2}\}$ 这六个参数作为纹理的特征向量, 以度量窗口内纹理的局部特征。

通过这六个基本参量, 纹理的粗糙程度、基元疏密、纹路走向等物理属性均得到了合理测度和反映, 它们是充分和完备的特征向量。

由于纹理特征的提取过程是局部计算实现的, 所以该层次的操作是并行分布式逻辑机制, 既是有效地, 又是快速的, 易于在计算机系统中以并行硬件结构实现。

四、向量决策的边缘检测与追踪 在特征向量空间中, 窗口的并行操作是比全局聚类机制效率更高的计算手段, 而且具有非迭代、计算复杂度低的优点。

分割过程以 $L \times L$ 的窗口扫描方式实现, 每次以 D 像素宽移动窗口。

窗口内的向量统计判断, 既反映了可能存在区域的内部属性和边缘特性, 又反映了最大置信度决

策过程。

通过对窗口内多维向量进行统计聚类, 在满足一定精度的前提下, 如聚类呈现出单类效果(即单类效应具有较大置信度), 则边缘不存在, 该窗口处于区域内部; 如聚类呈现出两类效果, 则系统以较大概率判定, 窗口内存在边缘, 并在更小窗口内实现边缘的精确定位和标记。

在边缘检测的基础上, 由并行窗口内的边缘有限追踪与精细化操作, 最终计算出连续的纹理图象区域边界, 完成图象的有效分割。

整个算法不仅是并行的, 而且是非递归层次化的。这样克服了传统操作的低效率、操作复杂度高、适应性差等缺点, 是一种适应面较多的实用方法。

五、实验结果 该算法已在 CIC-80286 计算机系统上以 Turbo-C 语言仿真实现。实验结果是令人满意的, 这说明该算法是高效的和实用的, 并对噪声有良好的抑制能力。

六、结语及应用前景 本文介绍的算法通过对随机场统计建模与参数的自适应辨识, 并采用并行计算机制, 完成了对自然纹理图象的有效分割。因此是一种行之有效的实用算法, 可做为面革(包括羊皮)质量计算机自动识别系统的基本软件之一, 用于生产第一线, 这对于充分保证面革质量分检的正确性和一致性以及降低工作人员的劳动强度有着十分重要的意义, 另外, 在皮毛、木纹、金属等有关涉及材料表面质量分析检测的场合, 该方法也有着相当广阔的应用前景。

参考文献

- [1] T. R. Reed and H. Wechsler, "Segmentation of textured images and Gestalt organization using Spatial/spatial - frequency representations", IEEE Trans on PAMI, vol. 12, no. 1, january, 1990, PP. 1-12.
- [2] R. J. Offen 主编, 许耀昌等译, 图象的并行处理技术, 科学出版社, 1989 年。
- [3] A. Khotanzad and Jong - yih Chen, "Unsupervised Segmentation of textured images by edge detection in multidimensional features", IEEE Trans. on PAMI, Vol, 11, no. 4, April, 1989, PP. 414-421.

微机上用 Turbo C 实现的动画技术

华中理工大学 机一系 何汉武 (430074)

摘要 本文从实用的角度讨论了微型计算机上用 Turbo C 语言实现的图形动画技术。详细地介绍了“图形页面互换”以及“画面存贮、重放”两种动画技术。文中提供了这两种方法的实例,并指出了这两种动画技术各自的优缺点。

一、概述 计算机图形学中的动画技术可以广泛地用于动画片的制作,辅助教学,游戏等方面。特别是随着计算图象仿真技术广泛地应用于现代工业生产中以来,人们对动画技术的兴趣便与日俱增。所谓动画技术是指屏幕上显示出来的画面或其中的一部分,能按一定的规则及要求,在屏幕上移动或变幻,从而使计算机显示的图形能动态变换的技术。

很多高级语言如 pascal, C 等都具备有图形处理的功能。特别是 Turbo C 2.0 提供丰富的进行图形处理所需的库函数,为微机上实现动画仿真提供了极大的便利。本文将实例的方式介绍用 Turbo C 实现的微机动画技术。

二、“画面存贮、重放”动画技术

1. 基本原理 这种动画技术的基本原理有点类似于幻灯片的制作过程。即把整个动画过程划分为一个片断,将每一片断作为一幅画在屏幕上一定的区域显示出来,然后把屏幕上的图象存在一个文件中。在动态显示时,按顺序不断地推出这些画面,就可以产生动画效果。

由此可以看出,这种方法在编程时的要点就是屏幕图象的存贮,以及画面的重放。这也是我们将这种动画技术命名为“画面存贮、重放”的原因。而在 Turbo C 中这两种操作主要用 Getimage() 和 Putimage() 函数来实现。

2. getimage() 和 Putimage() 的使用

(1) getimage(X1, Y1, X2, Y2, image)

其中 (X1, Y1), (X2, Y2) 分别是一个待保存的矩形区域信息的左上角和右下角的坐标。image 是用来存贮矩形区域信息的缓冲区。该语句的功能是把指定矩形区域内的图形信息存在一个缓冲区中。

必须指出,矩形区域的坐标应该容纳待保存在屏幕上的全部信息。其缓冲区 image 的大小(用字节数表示)可用 imagesize() 函数来确定。详细使用请参考 Turbo C 有关手册。

(2) Putimage(X1, Y1, image, operation-type)

其中 (X1, Y1) 表示矩形的左上角坐标。image 是保存有图象信息的缓冲区。Operation-type 表示缓冲区的信息和屏幕上原有信息进行操作的方式(如“与”,“或”“非”等)。该语句的功能是把已保存的动画对象在屏幕上指定位置显示出来。

3. 程序设计一般流程。 从前面所述可以知道,这种动画技术的编程分为两个阶段,一是生成画面与画面存贮,二是画面重放。这两个阶段可以分开进行。其程序流程见(图1),(图2)。

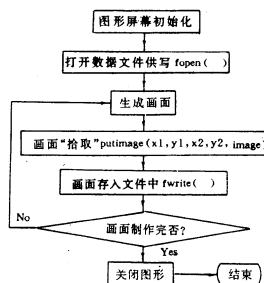


图1 生成画面

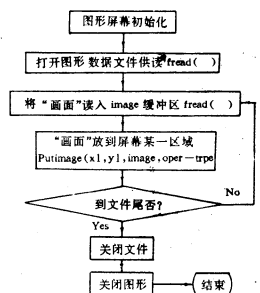


图2 画面重放

为了加快动画速度,可以先将图形数据依次读出,建立一链表,然后对此链表进行操作。这样不但可以增强动画效果,而且还可以通过对链表的特殊操作(如增删,重排次序等)实现画面的增加,删除,剪辑等技巧,不过,建立链表要占用计算机内存,而每一幅画所占内存有时就相当大。这方面的详细情况将在后面讨论。

三、“多页面”动画技术

1. 基本原理 计算机图形显示器往往有多种工作模式。在有些模式下,其图形有多个页面。例如 EGA640×350 模式下有两个图形页面。对有两个上显示页面的图形工作模式,可以采用“多页”

术来实现动画。其基本思想是将一页作为显示页,另一页作为绘图页。图形在绘图页面上作出后,迅速到显示页面去显示。这样显示页上总是有不同的图形在显示,看不到画图过程。由于两幅图形交换的时间间隔比较短,当视觉感觉不到这种间隔时,就产生了动画的效果。显然,时间间隔越短,其动画效果就越好。所以,尽量减了在绘图页面上的作图时间,是增强动画效果的关键。

2. Turbo C 中的编程要点 Turbo C 2.0 提供了两个库函数 Setactivepage() 和 Setvisualpage() 来实现显示页和作图页的互换。

Setactivepage(page-number) 的作用是将某页置为作图页。Setvisualpage(page-number) 的作用是将某页置成显示页。如果不特别指定,默认第 0 页为作图页,同时图形也显示在这页上。我们平时作图就是在第 0 页上进行的。

3. 程序设计流程 (见图 3) 假定计算机工作在有两个页面的图形模式下,页面登记号分别为 0 和 1。

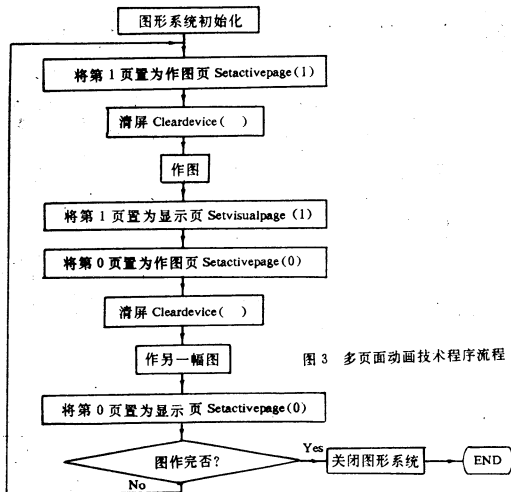


图 3 多页面动画技术程序流程

四、两种动画技术的优缺点 这两种动画技术在微机上的实现起来,其效果比较理想。“页面存贮、重放”式动画技术推出画面的速度比较快。如果事先将图形数据文件建成一顺序链表,那么推出画面的速度会更快。但这种方法内存开销很大。因为这种方式要建立一块缓冲区来存贮屏幕上的图形信息,在 CGA 640×200 的模式下,存贮整个屏幕需要约 16Kb 的缓冲区,而 EGA 的 640×480 模式则需要 120Kb 左右的缓冲区。一般地,图形分辨率越高,存贮一幅图的内存开销就越大。这在设计大的程序时

应该引起足够的重视。

“多页面”动画技术推出画面速度不如前者,但无需存贮图形的内存开销,且可实现实时动画。为了增强动画效果,在使用这种动画技术时,要注意尽量减短作图时间。这就需要优化算法。

有些时候,这两种动画技术可以结合起来使用。处理得当的话,可以收到意想不到的效果。

五、应用实例 如图示平面二连杆机构(开链)的动态仿真。已知连杆 1, 连杆 2 的长度。输入关节角序列 $\{\theta_1, \theta_2\}$, 使连杆在屏幕上动态地运动。

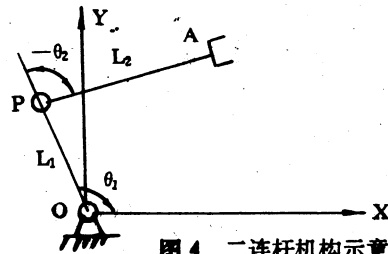


图 4 二连杆机构示意图

程序 2 为使用“画面重贮、重放”技术的例子,程序 1 为“多页面”动画技术的例子。两程序均在 sun 386, XT/286 上调试通过。

程序 1. “多页面”动画实例

Example of the mechanic (two link)
simulation using two graph pages.

#include<graphics.h>

#include<math.h>

double PI= 3.14159;

int x_cor= 350; /* center x */

int y_cor= 250; /* center y */

int l1= 100; /* link 1 */

int l2= 60; /* link 2 */

double sital, sita2;

int xa, ya;

main()

{

int gd= EGA, gm= EGAHI,

int i, times;

/* function prototype */

void viewport();

void set_thita();

void draw_link();

sital=sita2=0;


```

Printf("simulation times..."),
scanf("%d",&times);
initgraph(&gd,&gm,"");
getaspectratio(&xa,&ya);
for(i=0;i<times;i++) {
    setactivepage(1);
    clearviewport();
    set __ thita();
    viewport();
    draw __ link();
    setvisualpage(1);
    setactivepage(0);
    clearviewport();
    set __ thita();
    viewport();
    draw __ link();
    setvisualpage(0);
}

getch();
closegraph();
}

void viewport()
{
    setcolor(RED);
    rectangle(200,100,600,300);
}

void set __ thita()
{
    sital+=10.0; sita2-=10.0;
}

void draw __ link()
{
    float px,py,hx,hy
    double ratio,ss;
    ss=PI/180.0;
    ratio=(double)xa/(double)ya;
    px=11*cos(sital*ss);
    py=11*sin(sital*ss);
    hx=px+12*cos((sital+sita2)*ss);
    hy=py+12*sin((sital+sita2)*ss);
    px+=x __ cor; py=y __ cor-py*ratio;
    hx+=x __ cor; hy=y __ cor-py*ratio;
    px=(int)px; py=(int)py;
    hx=(int)hx; hy=(int)hy;
}

```

```

setcolor(RED);
setlinestyle(SOLID __ LINE,0,3);
line(x __ cor,y __ cor,px,py);
line(px,py,hx,hy);
/* draw the jt of the mechanic */
circle(x __ cor,y __ cor,6);
circle(px,py,3);
}

```

程序 2. “画面存贮、重放”动画实例

```

/* *****
Example of thr mechanic(two __ link)
simulation using getimage—putimage.
***** */
#include<graphics.h>
#include<math.h>
#include<alloc.h>
#include<stdio.h>
double PI= 3.14159;
int x __ cor= 350; /* center of x */
int y __ cor= 220; /* center of y */
int l1= 100; /* the 1st link */
int l2= 60; /* the 2nd link */
float sital,sita2;
char ch;
int xa, ya;
FILE *fp;
int size, t;
char *buf;
main()
{
    int gd=VGA,gm=VGAHI;
    int i,times;
    /* function prototype */
    void viewport();
    void draw __ link();
    initgraph(&gd,&gm,"");
    getaspectratio(&xa,&ya);
    size=imagesize(300,100,500,250);
    buf=malloc(size);
    if(size== -1)
        printf("not enough memory !");
    fp=fopen("c:\\link.dat","wb");
    while((ch=getch())!= 'q') {
        scanf("%f%f",&sital,&sita2);
        cleardevice();
        draw __ link();
    }
}

```

一个新颖生动的 XOR 动画实例

安徽商业管理干部学院

郭大伟 (233061)

当我们观察或设计计算机动画显示时都有这样的体验,就是在很多情形下,产生动画效果只需改变屏幕中一个前景运动物体在画面中的位置,而其背景和动画物体的形状一般是保持不变的。例如,当动画模拟一个在背景中快速移动的飞机或汽车时,因该物体是在屏幕中作平动,每次运动的结果只是显示位置的变化,而其图形本身形态并无变化,这时是否有必要每次对前景移动物体和背景图形进行重绘呢?回答是否定的。本程序所采用的方法是将动画物体的图像保存在存储区中,需要时快速从内存中拷贝回屏幕进行重显,并通过对该图像像素与背景像素进行 XOR(异或)运算,可使被前景所遮盖的背景图像部分还原,这种由“异或”操作所得到的动态图形我们称之为 XOR 动画,这种方法也是许多动画

显示和游戏程序设计时所采用的。

在 Turbo C 语言的库函数中,Putimage 函数是将内存中所存贮的图像回送到屏幕任意指定位置上,其中参数 OP 是一个组合算子,我们一旦定义 OP 参数为 XOR __ PUT,就可使内存中源图像同背景图像按位进行“异或”。XOR 运算的一个重要特性是它的“还原”作用,即当 XOR 操作后使像素值发生了改变,然后再作一次 XOR 运算又可把原像素值还原回来。利用这一特性,我们可以在屏幕上对一个运动物体连续作 XOR 运算,而不必担心背景图形的储存与还原问题。如果将其特性运用到动画技术上,即在动态物体的同一显示位置上连续进行两次 XOR 运算,然后在下一显示位置作同样操作,如此反复进行,就使前景运动图形产生了动画效果。

```

viewport();
getimage(300,100,500,250,buf);
fwrite(buf,size,1,fp);
}
fclose(fp);
clearviewport();
fp=fopen("c:\\link.dat","rb");
for(;;){
    if((t=fread(buf,size,1,fp))!=1){
        if(!feof(fp)){
            fclose(fp);
            printf("end of file");
        }
        else{
            getch();
            closegraph();
            exit(1);
        }
    }
    else{
        putimage(300,100,buf,COPY __ PUT);
    }
}
void viewport()

setcolor(RED);
rectangle(300,100,500,250);
}
void draw __ link()
{
    float px,py,hx,hy;
    double ratio,ss;
    ss=PI/180.0;
    ratio=(double)xa/(double)ya;
    px=11*cos(sita*ss);
    py=11*sin(sita*ss);
    hx=px+12*cos((sita+sita2)*ss);
    hy=py+12*sin((sita+sita2)*ss);
    px+=x __ cor;py=y __ cor-py*ratio;
    hx+=x __ cor;hy=y __ cor-py*ratio;
    px=(int)px;py=(int)py;
    hx=(int)hx;hy=(int)hy;
    setcolor(RED);
    setlinestyle(SOLID __ LINE,0,3);
    line(x __ cor,y __ cor,px,py);
    line(px,py,hx,hy);
    /* draw jt of the link mechanic */
    circle(px,py,3);
    circle(x __ cor,y __ cor,6);
}

```

作为示例,下面给出一个采用 XOR 动画实现的一个演示程序。该程序运行后首先在繁星闪烁的夜色背景上绘出一个由轨道环绕的蔚蓝色地球造形,然后一颗人造卫星由左至右不断从屏幕上掠过,屏幕下方同时显示有“COMPUTER WORLD”的放大字形,整个画面生动美观,笔者已将其作为开机时的“Hello”(问候)程序,有兴趣的读者不妨一试。

本程序已在 WYSE386(VGA 显示)上通过 Turbo C2.0 和 Turbo C++1.0 编译。(后附程序清单)

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#define IMAGE __SIZE 10
void draw __image(int x,int y);
void PutpixelDemo(void);
int main()
{
    /* request autodetection */
    int gdriver=DETECT,gmode,errorcode;
    void *pt __addr;
    int x,y,maxx,maxy,midy,midx,,i;
    unsigned int size;
    /* initialize graphics and local variables */
    initgraph(&gdriver,&gmode,"");
    /* read result of initialization */
    errorcode=graphresult();
    if (errorcode!=gok) /* an error occurred */
    {
        printf("Graphics error : %s\n",grapherrormsg(errorcode));
        printf("Press any key to halt : ");
        getch();
        exit(1); /* terminate with an error code */
    }
    maxx=getmaxx();
    maxy=getmaxy();
    midx=maxx/2;
    x=0;
    midy=y=maxy/2;
    /* sets the current text characteristics for graphics output */
    setcolor(YELLOW);
    settextrstyle (TRIPLEX __FONT, HORIZ __DIR,
4);
    settextrjustify (CENTER __TEXT, CENTER __TEXT);
    /* output a message to screen */
    outtctxy(midx,400,"COMPUTERWORLD");
```

```
setbkcolor(BLACK);
/* draws a arc around the earth */
setcolor (RED);
setlinestyle (SOLID __LINE, 0, THICK __WIDTH);
ellipse(midx,midy,130,50,160,30);
setlinestyle (SOLID __LINE, 0, NORM __WIDTH);
setcolor(LIGHTBLUE);
/* draw an earth __image on screen */
for(i=0;i<=13;i++)
{
    ellipse(midx,midy,0,360,100,100-8*i);
    ellipse(midx,midy,0,360,100-8*i,100);
}
/* draw the image to be grabbed */
draw __image(x,y);
/* calculate the size of the image */
size = imagesize(x,y-IMAGE __SIZE,X+(4*IMAGE __SIZE),y+IMAGE __SIZE);
/* allocate memory to hold the image */
pt __addr=malloc(size);
/* grab the image */
getimage(x,y-IMAGE __SIZE,x+(4*IMAGE __SIZE),y+IMAGE __SIZE,pt __addr);
/* Display some starts on the screen */
PutpixelDemo();
setcolor(WHITE);
/* Draw a solid single line around te screen */
setlinestyle(SOLID __LINE,0,NORM __WIDTH);
rectangle(0,0,maxx,maxy);
/* repeat until a key is pressed */
while (!kbhit())
{
    /* erase old image */
    putimage(x,y-IMAGE __SIZE,pt __addr,XOR __PUT);
    /* Calculate the next position to display */
    x=x+5;
    if(x>=maxx)
        x=0;
    /* plot new image */
    putimage(x,y-IMAGE __SIZE,pt __addr,XOR __PUT);
    delay(50);
}
/* clean up */
free(pt __addr); /* free memory */
closegraph();
```

```

return 0;
}
/*      DRAW __IMAGE;          */
/* Draw a source image to be grabbed and copied */
void draw __image(int x,int y)
{
moveto(x+10,y);
setcolor(14);
setfillstyle(1,4);
linere1(-3*10,2*10);
moveto(x+10,y);
linere1(-3*10,-2*10);
moveto(x+25,y);
linere1(-5*10,0);
fillellipse(x+13,y,8,8);
}
/*      PUTPIXELDEMO;          */
/* Display a pattern of random dots on the screen */

```

```

void PutPixelDemo(void)
{
int seed=1958;
int i,dotx, doty,h,w,color,maxcolor;
maxcolor=getmaxcolor();
w=getmaxx();
h=getmaxy();
srand(seed); /* Restart random # function */
for(i=0;i<5000;++i){ /* put 5000 pixels on
screen */
dotx=i+random(w-1); /* Generate a random
location */
doty=1+random(h-1);
color=random(maxcolor);
putpixel(dotx,doty,color);
}
srand(seed); /* Restart Random # at same # */
}

```

通用论文稿打印程序

阜新矿业学院 房红兵 (123000)

论文稿不同于其它稿件,通常是在固定的论文纸上打印,并且页号要打在每页的右上角。另外,由于论文中一般都有很多的公式和图表,行距不能统一,因此,若采用 Wordstar 或其它分页打印程序来打印论文,是不能满足要求的。为此,笔者编制了一个简单实用的论文稿打印程序(程序清单附后)。使用该程序,可自动将页号打在论文纸的右上角,并且允许使用者在文稿的任意位置分页。分页时,只需在分页处插入一空行,并在空行首部打入大写的字符串“PAUSE”即可。另外,为适应论文稿按章节划分的要求,在程序中增加了设定起始页号的功能,该功能可允许文稿按章节分别排版及打印。

本程序用高级 BASIC 语言编写,用 Quick Basic2.0 编译成 EXE 文件,在 CCBIO5 2.13H 汉字系统下运行,打印效果十分理想。值得注意的是,本程序打印的文稿应是标准文稿,即在文稿中不能有软回车及自动分页符号。

```

10 ' * * * * * PRINTLW. BAS * * * * *
20 ' * * * * * HOWARD 1991 * * * * *
30 KEY OFF;SCREEN 2;CLS
35 width "lpt1:",255
40 LOCATE 1,3;INPUT"请输入要打印的文

```

件名称",FN\$

```

50 OPEN FN$ FOR INPUT AS #1
60 LOCATE 3,3;INPUT"请输入起始页号";IP
70 LOCATE 5,11;PRINT"请把打印机准备好,然后按回车键...";
80 K$=INPUT$(1);IF K$(<>CHR$(13))
THEN BEEP;GOTO 70
90 LPRINT TAB(68);";";LPRINT USING"#
##";IP;LPRINT;LPRINT
100 LINE INPUT #1,B$
110 IF LEFT$(B$,5)<>"PAUSE" THEN
LPRINT TAB(3);B$;GOTO 150
120 LOCATE 7,11;PRINT"请把打印纸换好,然
后按回车键...";
130 K$=INPUT$(1);IF K$(<>CHR$(13))
THEN BEEP;GOTO 120
140 IP=IP+1;GOTO 90
150 IF EOF(1) THEN 170
160 GOTO 100
170 CLOSE #1
180 END

```


Auto CAD 与其它程序快速交换图形数据方法

四川仪器仪表研究所 王建华 (630700)

摘要 本文通过介绍 Auto CAD 绘图数据文件(PLT)的结构,提出一种快速、简单的其它程序读取 Auto CAD 生成的图形数据的方法。

Auto CAD 作为一高效的绘图工具,大大提高了在微计算机上进行各种图形设计和处理的效率。在某些应用中,其它程序需要利用 Auto CAD 产生、修改或绘出的图形,即进行图形数据交换。由于 Auto CAD 绘图文件(DWG)是以紧缩格式存储的,不同的机型的 Auto CAD 的实现采用不同的内部格式,所以其它程序是很难读出其数据并加以应用的。传统的 Auto CAD 与其它程序进行图形交换的方法是采用 Auto CAD 提供的具有统一格式的绘图交换文件(DXF)机制。在 Auto CAD 命令行中用 DXFIN 和 DXFOUT 分别读、写绘图交换文件 DXF,某些 Auto CAD 书籍中有 DXF 文件结构的详细介绍。DXF 文件是图形文件的完整表示法,信息量极大,结构及其概念十分繁琐,随着版本的升级,图素中还将加入新组及代码,所有这些特点使得其它程序在读取或构成 DXF 文件的难度较大。本文介绍一种特别简单的其它程序读取 Auto CAD 图形数据的方法,其特点是利用绘图机绘图产生的数据文件(PLT)。

一、有关知识 熟悉 Auto CAD 的都知道,Auto CAD 的图形由一个或多个图层组合而成,每一图层有相应的颜色和线型。不同的图层可以有相同的颜色号和线型。前面七层赋予标准色号,即:1、红(Red),2、黄(Yellow),3、绿(Green),4、青(Cyan),5、蓝(Blue),6、绛红(Magenta),7、白(White)。线型有各种点划线与实线(Continuous 线)。在使用绘图机绘制图形的时候,需指定每一颜色图层的笔号和硬件线型。如果硬件线型选取实线的话,则用户不必关心线型,Auto CAD 自动用实线线型以点划方式绘出图形中的点划线。

二、绘图数据文件(PLT)结构 在使用绘图机绘图时,Auto CAD 可将绘图数据写入指定文件,其它程序可读取此文件而达到利用 Auto CAD 所生成的图形的目的。PLT 文件由 Auto CAD 发送给绘图机的绘图指令构成。Auto CAD 支持的绘图机很多,画线、椭圆、文字等绘图功能强弱、指令格式都各有其特点,Auto CAD 在解决各种绘图机之间的差异上排

异求同,将各种复杂的指令都转化成简单的、各种绘图机都具有的几类指令,主要有:初始化绘图机(选择绘图单位、设置坐标方式,AutoCAD 一般采用绝对坐标方式。);设置抬笔方式;设置落笔方式;选择绘图笔;设置笔速;移动笔等指令。即圆、阴影等复杂绘图指令都用直线多次绘出,当圆的弦很小时,这些直线的弦就组成了圆,其它图形指令也同样都转化成了直线指令见 Houston Instrument, Hewlett — Packard 绘图机的相应指令列表:

指令种类	选笔	移动笔	抬笔	落笔	笔速
绘图机型					
HI	P	M	U	D	V
HP	SP	PA	PU	PD	VS

由上可见 PLT 文件结构是相当简单的,这也许是 Auto CAD 系统设计人员考虑软件兼容性的良苦用心之处。

三、应用举例 PLT 文件的简单性使得读取 Auto CAD 绘图数据的其它程序变得特别简单。程序 PLOT 是用 Turbo C2.0 编写的读取 PLT 绘图数据并将图形在屏幕上显示出来的例子。在 Auto CAD 产生 PLT 文件时,配置采用的是 HP7550 绘图机,图层颜色号等于笔号,硬件线型采用实线,所以程序中不必考虑绘图线型。HP7550 的 PLT 文件有类似如下的格式:...,PU,SP7,LT,PA1968,2961,PD,SP,...即各指令以“;”作结束符,纵横坐标用“,”分隔。此程序不必考虑笔速,也未考虑坐标之间的精确换算,而通过过程 GetXY 中的比例常数 P 来缩小图形,具体数字可根据实际情况而定,缺省为 30。程序首先打开命令参数给出的 PLT 文件(文件名后缀缺省为“PLT”),初始化图形方式;然后读取指令,用 MoveTo 模拟抬笔移动,用 LineTo 模拟落笔移动,用 Setcolor 模拟换笔,为了使 Auto CAD 的标准色号与 Turbo C 规定的色号对应,用数组 color[7]进行转换,因为绘图指令由两字符组成,用函数 get_code()返回指令的第二字符,用 U、D、A、P 分别来指示程序关心的指

令 PU、PD、PA、SP; 对不必考虑的部分用过程 Ignore 忽略。此程序编译成 EXE 文件, 用 PLOT<文件名>[比例因子]的格式调用程序, 如果不给出比例因子, 则取缺省值 30。例如 PLOT COLORS 40 就可读取文件 COLORS. PLT 来显示 Auto CAD 生成的图形, 并缩小 40 倍, 效果很好, 有兴趣的读者不妨一试。

此方法可用来解决 Auto CAD 与绘图机不匹配的问题, 即用 Auto CAD 所不支持的绘图机绘制 Auto CAD 产生的图形。首先配置时选择支持的绘图机型(如 HP7550), 生成 PLT 文件, 再用程序将此文件转换成所不支持绘图机的绘图指令进行绘图, 就可达到目的, 此时需考虑具体的比例变换。笔者曾经利用此原理为某单位设计一硬件接口电缆, 接在微机与绘图机之间, 直接对 Auto CAD 输出的指令流进行转换输出, 达到了绘图机仿真的效果。

程序 PLOT. C

/* 此程序通过 PLT 文件显示 Auto CAD 生成的图形, 绘图机选择 HP7500, 图层颜色号=笔号
调用格式: PLOT<文件名>[比例因子]

作者: 王建华 四川仪器仪表研究所

```

*/
#include<graphics.h>
#include<stdio.h>
#include<string.h>
#define TRUE 1
#define FALSE 0
int color[7] = {RED, YELLOW, GREEN, CYAN,
                BLUE, MAGENTA, WHITE};
char pen __up, ch;
FILE *plot __file;
int X, Y, P;
main(argc, argv)
int argc;
char *argv[];
{ char *filename;
  int drive, mode;
  char get __code();
  filename = strcpy(filename, argv[1]);
  if (strchr(filename, '.') == NULL)
    strcat(filename, ". PLT");
  /* 文件名后缀缺省为. PLT */
  plot __file = fopen(filename, "rt");
  if (plot __file == NULL)

```

```

{ printf("Error open plot file %s",
        argv[1]); exit(-1);
}
if ((P = atoi(argv[2])) == 0) P = 30;
/* 比例因子缺省为 30 */
drive = DETECT;
initgraph(&drive, &rhode, '');
setbkcolor(LIGHTBLUE);
while (! feof(plot __file))
  switch(get __code()) {
    /* PU */ case 'U': pen __up = TRUE;
              ignore(); break;
    /* PD */ case 'D': pen __up = FALSE;
              ignore(); break;
    /* PA */ case 'A': GetXY();
              if (pen __up == UE) moveto(X, Y);
              else lineto(X, Y); break;
    /* SP */ case 'P': ch = fgetc(plot __file);
              if (ch != ',')
                { setcolor(color[ch - '1']);
                  ignore(); } break;
    default; ignore(); break;
  }
  getch();
  closegraph();
}
char get __code()
{ fgetc(plot __file);
  return(fgetc(plot __file));
  /* 返回指令的第二个字符 */
}
ignore()
{ do ch = fgetc(plot __file); while
  ((ch != ',') && (! feof(plot __file)));
}
GetXY()
{ X = Y = 0;
  while ((ch = fgetc(plot __file)) != ',')
    X = X * 10 + ch - '0';
  while ((ch = fgetc(plot __file)) != ',')
    Y = Y * 10 + ch - '0';
  X /= P; Y = getmaxy() - (Y/P);
}

```

绘制逼真图形的捷径——快速灰度填充法

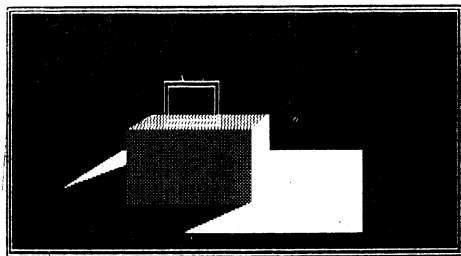
清华大学 江 涛 (100084)

摘要 利用该方法可实现屏幕上任意形状、任意大小的封闭曲线图形的任意灰度级快速填充,填充后,边界平滑,用于绘制各种逼真画面。

关键词: 任意形状,任意大小,任意灰度,快速,边界平滑。

一、说明: 以往要想绘制灰度级丰富的逼真画面,需要用到专门的高级图形显示器,对于在 IBM-PC 系列的个人机上实现则有些困难。我经过反复摸索,找到一种好方法,可以在普通显示器上绘出逼真图形,如人脸,风景等,配上计算机屏幕作图程序,便可在顷刻间,绘出一幅令自己都感到吃惊的生动画

面。所附的两张图是用该方法绘出,第一张,影星:铁爪·阿诺德,绘制用 20 分钟,第二张,空间积木,仅用 6 分钟。该方法在艺术图案设计,动画设计,辅助模型设计等工作中颇具实用价值,现介绍给读者。程序用 BASICA 编写,编译后运行。



二、方法: 该方法由九步完成:

第一步:构造 N 个灰度块,分别存于 C1%~CN%数组中。

第二步:在屏幕上定义一个矩形区,使欲被涂灰度的图形完全位于其中。

第三步:用 Get 语句,将整屏图象存于数组 Dark%中。

第四步:用 Paint 语句,将图形内部涂白。

第五步:用 Get 语句,将整屏图象存于数组 WIN%中。

第六步:取灰度级 M

第七步:用 Put 语句,将灰度级为 M 的方块填满矩形区。

第八步:用 Put 语句,将图象 WIN%与屏幕图象相与。

第九步:用 Put 语句,将图象 Dark%与屏幕图象相或。

[程序清单] 由于版面所限,略去。

三、程序说明:

语句 5030~5083:实现步骤(1)。

语句 6100:实现步骤(3),(4),(5)。

语句 6200~6500、5200~5210:实现步骤(6),(7)。

语句 6600:实现步骤(8),(9)。

四、变量说明:

MX,MY:封闭图形内一点的座标,。

LNx(1),LNy(1):矩形区左上角座标。

LNx(2),LNy(2):矩形区右下角座标。

(如需完整的绘图程序,可与编辑部或作者联系。地址:清华大学二十六号楼 407,邮码:100084)

《自动控制理论》试题库软件

西安交通大学工业自动化教研室李忠勇研制出《自控理论》试题库软件,具有编辑、建库、作图、查阅、求答、修改、显示、打印、自动成卷等丰富齐全的

功能,同时十分注重集成环境的构成和用户界面的友好性,用户使用该软件可“无师自通”,得心应手,十分方便。需此软件者,可同作者联系。(本刊)

利用华光交互式图表编辑排版软件绘制框图

西南财经大学信息系 陈 康 (610074)

华光作为一种比较完善的轻印刷系统已越来越受到广大用户的爱好,其中的交互式图表编辑排版软件也能给用户绘制一些图表带来很多方便。很多用户想用它,但苦于没有资料,不知从何入手。本人愿将在使用此软件中摸索出的一些经验供大家参考。其大致处理过程如下:

一、环境设置 用户在进入交互式图表处理后,屏幕右边窗口内为主控菜单,包括图形、文字、输入输出、显示选择和照排输出等处理功能。用户在绘制图表之前,应首先用显示选择功能设置显示环境参数。基本参数有:全图尺寸、基本字号。其中全图尺寸是相对基本字号而定,如字号为5,屏幕尺寸为(140,104),指定是屏幕宽为140个5号字,定为104个5号字。用户只有清楚这一点,才能正确的确定其绘制的图表大小。用户可根据具体要求自行设置。

二、绘制框架 参数设置完毕,用户可进入图形子菜单中的框图处理功能,此时右边窗口内有一些基本的图元,用户可选择所需图元进行框图的框架设计。具体的操作步骤为:①选择图元,即将右边窗口中光条移至所需图元上,敲回车。②图元定位,即将光标定于该图元在屏幕的位置,敲回车。用户可多次将所选图元定位于不同的位置上,而无需每次都在右边窗口内选择。框架中图元之间可以用箭头、折线和实线相连接,具体的操作方法基本上与其它图元相同,不同之处是这些连线需定位起始点和终止点。

三、文字处理 用户在框架设计后,要返回主控菜单,进入文字处理子菜单。文字字符不能直接填入其中,首先需要定义字符处理项,即定义用户所需填入字符的封闭区域,如字符不是填入封闭框区中,则首先要在上一步框架设计中使用无边框架图元来确定其字符所填区域。用户在一一定义完毕所有字符处理项后,方可进入文字填入方式。具体操作为:将右边窗口内亮条移至文字处理子菜单,敲回车,进入文字处理项定义方式,将光标移至需填入文字的图元,敲回车,定义文字处理项,将光标移至下一项中,同样处理;当所有项定义完毕,敲ESC键,进入

文字填入方式。

进入文字填入方式后,首先需设置参数,它包括:文字参数(字体、字号、字型)、页属性(指定上空、上下居中、指定下空)、行属性(居左、居右、居中、撑满、自动)、字距、行距等等。参数设置键为SHIFT-F3。用户只有在插入方式下才能填入字符,一项填入完毕,敲ESC键进入下一项。当用户将文字处理完毕,敲ESC键返回控制菜单中。

四、框架编辑和文字编辑 当用户处理完以上过程后,就可以照排输出,但如在以上操作中出错,就需编辑处理。编辑处理可分为框架编辑和文字编辑。框架编辑可在图形处理菜单下直接进入图形编辑子菜单,也可进入框图子菜单后进行编辑。基本功能包括:图元移动、图元修改、图元删除、恢复删除、图区移动、图区修改、图区删除等。这些功能使用起来都比较方便,用户只需根据屏幕提示就可完成。

文字编辑也首先需要定义编辑的文字处理项,在进入文字填入方式下才能编辑处理。这里有两点需指出:一是插入字符时,需将光标定位在插入位置的前一个字符上;二是修改不能直接进行,只能是先删除错误字符,后插入正确字符。

五、输入输出处理及照排输出 用户绘制好完整的图形,或在处理过程都可以进行输入输出处理。输入输出可以有两种方式,一种是直接以整个图表作为整体进行处理;另一种是可以将框架和文字分开进行处理。框架和文字文件的扩展名分别为TX和WZ。存贮的图表(或框架与文字)都可以随时调入屏幕,或存入磁盘上。

如要将绘制好的图形打印出来,必须将其变为二扫描文件,即必须调用照排输出功能,它在交互式图表软件的主控菜单中。在调用照排输出功能时,用户需给出输出文件名和排版参数。输出文件必须以S2作为扩展名,而参数可根据具体情况设置,或使用约定值。

以上所讨论的只是CGA显示方式下,华光交互式图表编辑排版软件的部分功能,另外还有很多有用的功能没有提到,但用户可以自己尝试使用,熟练以后会给用户绘制图表带来很大方便。

0-1 整数规划的枚举算法

武汉数字工程研究所 陶友传

(430074)

华中理工大学

唐泳洪 段正澄 (430074)

整数规划是数学规划的重要方面之一,许多重要的组合数学问题实质上都是从属于它,然而目前求解还存在许多困难,基本上是采用穷举法。0-1 整数规划是整数规划中最简单的一种,即变量只取 0 和 1 两种的整数规划问题。本文只限于讨论 0-1 整数规划的一个特例,它是装载问题整数规划模型 [1] 的一般形式。

$$\text{目标函数, } \max \sum_{i=1}^n C_i X_i \quad (1)$$

约束条件:

$$\begin{cases} f_1(X_1, \dots, X_n) \geq 0 \\ \vdots \\ f_m(X_1, \dots, X_n) \geq 0 \end{cases} \quad (2)$$

$$x_i \in \{0, 1\} \quad (i=1 \dots n) \quad (3)$$

其中, n 是整数变量的数量, m 是约束条件的数量, $C_i (i=1 \dots n)$ 是非负常系数, $f_j (j=1 \dots m)$ 是计划变量 $X_i (i=1 \dots n)$ 的函数, 它可能是非线性的。

1、分枝定界法 图 1 是深度为 3 的满二叉树。

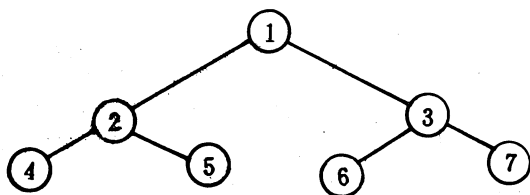


图 1 深度为 3 的满二叉树

如果一个深度为 $(n+1)$ 的满二叉树按上述方法顺序排列, 并对任一节点 i 存在:

(1) 当 $i=1$ 时, i 是根;

(2) 当 $i \neq 1$ 时, 如果 i 是偶数, 该节点存储的数据是 0, 否则存储的数据是 1。

那么, 从根节点到叶子的一条路径(以下简称路径)是 n 个 0-1 整数的组合, 因而 0-1 整数规划的完全枚举可等价于满二叉树的遍历。

设路径的节点依次为 $0, 1, \dots, n$, 节点 i 对应于二叉树的节点 $la(i)$, 并将路径节点 i 存储的数据赋

给 x_i , 令从路径节点 i_1 到 $i_2 (i_1 < i_2)$ 的收益为:

$$k(i_1, i_2) = \sum_{i=i_1+1}^{i_2} C_i X_i$$

首先定义让 $h^*(i)$ 表示过二叉树节点 $la(i)$ 的所有路径中满足的约束条件(2)的最大收益 $k(i, n)$, $g^*(i) = k(0, i)$ 。那么, 过节点 $la(i)$ 所能够获得的最大收益 $f^*(i)$ 可表示成:

$$f^*(i) = g^*(i) + h^*(i)$$

笔者希望估价函数是 f 是 f^* 的一个估计, 此估计由下式给出, $f(i) = g(i) + h(i)$

式中, g 是 g^* 的估计, h 是 h^* 的估计。

对于过节点 $la(i)$ 的路径来说, $g^*(i)$ 是确定的,

$$\text{即 } g^*(i) = \sum_{j=1}^i C_j X_j, h^*(i) = \sum_{j=i+1}^n C_j X_j \leq \sum_{j=i+1}^n C_j, \text{ 由此可}$$

以得到 h^* 的一个上确界, $h(i) = \sum_{j=i+1}^n C_j$ 。由此可以得到 f^* 的一个上确界 $f = g + h$, 其中, $g(i) = g^*(i)$ 。

设遍历到二叉树节点 $la(i)$ 时, 最优目标值为 optimal。那么, 如果 $\text{optimal} \geq f$, 更好的目标值不可能在通过二叉树节点 $la(i)$ 的路径中产生, 因而可以删去以 $la(i)$ 为根节点的子二叉树。分枝定界法的基本思想正源于此。

假设自顶向下搜索时, 人为地规定路径的节点对应的变量依次为 X_n, X_{n-1}, \dots, X_1 , 则分枝定界算法如图 2(a) 所示。程序用 Turbo Pascal 语言写成。

2、隐枚举法 穷举法是对如图 1 的树进行自上而下, 从左向右的搜索过程。隐枚举法只是在这过程中采用了“碰壁回头”的策略, 从而设法从此剪去尽可能多的树枝, 枚举的量尽可能的少。

“碰壁回头”策略的基本思想是: 由顶向下搜索时, 如果某个节点的值取 0 或 1 时, 约束条件得不到满足, 那么, 通过该节点的任一路径均不能使约束条件得到满足, 从而可以删去以该节点为根节点的二叉树。譬如, 如果某个 $X_i = 1$ 时, 机床刀库容量的约束条件得不到满足, 那么继续沿着该节点向下搜索就失去了意义。设搜索到节点 X_i 时, 某个约束条件得不到满足, 则 0-1 问题的隐枚举算法如图 2(b) 所示。将分枝定界法和隐枚举法结合起来, 就形成了

(下转 44 页)

C语言与软中断方式实现读写端口信息

吉林工业大学 黄声烈 吴庆妍 (130012)

吉林省开发建设投资公司 赵冬林 (130000)

吉林省计算机技术研究所 李志英 (130012)

软中断方式是指程序自行安排的中断方式,它常在 8086/8088CPU 执行到程序中 INT n(n 为中断向量)指令时产生,CPU 根据中断类别的编号 n 转去执行不同的中断服务程序。

MS-DOS 使用编号从 20H 到 3FH(中断向量表存放在 80H~FFH 区域中)的软件中断作为操作系统调用,为其它程序提供不同类型的服务。其中的 21H 号软件中断用于为用户提供系统功能调用;另外,还可以通过调用 BIOS 中的设备驱动程序,为其它应用程序提供服务。

由于提供的软中断方式足以满足实际应用场合,但是对于特殊的应用场合,为了直接控制读写端口信息等,需要自行开发和设计中断服务程序,可作为操作系统功能调用的一部分,或者作为 C 程序函数功能的一部分是一种切实可行的好方法。

本文将通过 C 语言与软中断接口方式,利用中断向量 60H(60H~67H 通常可供应用程序使用,可选择其一)如何实现读写端口信息为主线,论述其方法与步骤。

1. 程序设计要点

(1) • COM 文件的程序结构及加载过程

一个 • COM 文件的结构定义如下:

- ①该程序只能设置一个段,且不准建立堆栈段;
- ②该程序的长度必须小于 64K 字节;
- ③该程序必须预留 100H 空间,且在移位 100H 处是一条可执行指令;

④该程序被加载的起始标号必须由 END 语句说明为开始地址;

⑤若 • COM 文件是由几个不同的目标模块连接生成的,则要求所有目标模块必须具有同一代码段名和类别名(CLASS),且赋与公共属性(PUBLIC),而主模块应具有 100H 的入口指针并优先连接;

⑥该程序中的子程序必须具有近过程(NEAR)。

符合上述条件的汇编源程序,经汇编连接而生成的 • EXE 文件,可由命令 EXE2BIN 转换成内存映像文件 • COM。

PORT • COM 的汇编语言源程序如下:

```
_PORT SEGMENT
```

```
ASSUME CS, _PORT, DS, _PORT
ORG 100H
START: JMP STOVE
        PROC FAR
        PUSH DS
        PUSH DX
        PUSH BX
        PUSH AX
        MOV AX, CS
        MOV DS, AX
        POP AX
        OR AH, AH
        JZ POINTOUT
        DEC AH
        JZ POINTIN
INOUTEND: POP BX
          POP DX
          POP DS
          IRET
POINTOUT: OUT DX, AL      ,AH=0
          JMP INOUTEND
POINTIN:  IN AL, DX      ,AH=1
          JMP INOUTEND ENDP
        _POINT
        STOVE: MOV AX, CS
              MOV DS, AX
              MOV DX, OFFSET _POINT
              MOV AL, 60H
              MOV AH, 25H
              INT 21H
              INT 20H
        _PORT ENDS
              END START
```

一个 • COM 文件加载过程如下:

• COM 文件的加载过程是由 C 语言程序实现的。该程序执行过程是:如果 PORT • COM 文件在当前工作盘中存在,则加载该内存映像文件,否则提示使用 Port • COM 程序,并正常返回到操作系统状态下。下面是 C 语言加载一个 • COM 文件的部分程序。

```
#include<dos,h>
main()
{
    FILE *fpr;
    if((fpr=fopen("port.com","rb"))==NULL)
    {
        printf("Usage:port.com programming\n");
        exit(0);
    }
    fclose(fpr);
    system("port");
}
```

(2) 设置中断向量 60H

设置中断向量 60H 在 PORT.COM 汇编语言源程序清单的尾部。该设置软中断向量 60H 的部分程序如下:

```
MOV AX, CS
MOV DS, AX
MOV DX, OFFSET _POINT
MOV AL, 60H ;设置中断向量 60H
MOV AH, 25H
INT 21H
```

(3) C 程序与 PORT.COM 程序接口

当 PORT.COM 程序已被加载到内存后,通常用以下方式调用软中断 60H。下面是 C 程序与 PORT.COM 的接口程序。

```
OutinBIO60(Port,config,inout)
int port,config,inout;
{
    Union REGS regs;
    regs.x.dx=port;
    regs.h.ah=inout;
    regs.h.al=config;
    return int86(0x60,&regs,&regs);
}
```

C 编译程序中,对于标准函数 int86() 和 bcdos() 是分别用于调用 ROM-BIOS 和 DOS 的系统功能的。联合类型 REGS 是由 C 编译系统定义的,它的定义包含在 dos.h 文件中。从 REGS 的定义中可知,它的成员是以字为单位的 WORDREGS 结构变量 x 和以字节为单位的 BYTEREGS 结构变量 h。在这两种结构中均以各种通用寄存器作为它的成员项。在 8086/8088 处理器中可以作为操作数参加运算的通用寄存器有四个十六位的 Ax、Bx、Cx、Dx,它们还可以兼作八个八位的通用寄存器使用。另外,函数 int86() 的参数输入和输出定义为联合指针,所以与此相对应的实参应是联合体的地址 ®s。

2. 读写端口信息

下面是适用的全部 C 语言程序。该程序经编辑、编译、连接后执行时,可根据用户键入的端口地址、参数、进行读/写端口地址信息等操作。其中, port 表示端口地址, config 表示从端口输入/输出信息, inout 表示输入/输出, 当输入数值为 0 时, 表示向端口地址输出信息, 当输入数值为 1 时, 表示从端口地址输入信息。

适用的 C 语言源程序代码如下:

```
#include<dos.h>
#include<stdio.h>
#include<stdlib.h>
main()
{
    int port,config,inout;
    FILE *fpr;

    if((fpr=fopen("port.com","rd"))==NULL)
    {
        Printf("\aUsage:port.com program\n");
        exit(0);
    }
    fclose(fpr);
    system("port");
    printf("PORT=");
    scanf("%x",&port);

    printf("CONFIG=");
    scanf("%x",&config);

    printf("OUT and IN(0 and 1)=");
    scanf("%x",&inout);

    printf("RETURN VAR=%x\n",outindio60(port,config,inout));
}

outinbio60(port,config,inout)
int port,config,inout;
{
    union REGS regs;

    regs.x.dx=port;
    regs.h.ah=inout;
    regs.h.al=config;

    return int86(0x60,&regs,&regs);
}
```

2.13 汉字系统实用外词组文件建立与分析

新疆电子计算中心 龔正科 (830011)

摘要 本文通过对2.13汉字系统的深入了解和实践,首先通过具体实例介绍了建立一个外部词组系统的具体方法和步骤,然后通过反汇编分析了词组文件的结构和细节,最后给出建立一个实用词组库的途径和方法。

关键词: 操作系统, 汉字系统, 汉字 BIOS, 词组

2.13A~K 汉字系统的最大特点是能够适应各种硬件环境(如多种打印机,各种显示器和字库的多种安装方法等等)从而被广泛使用。该系统为汉字输入提供各种输入方法,但是无论那一种方法,输入一个汉字都要输入至少2~3键,这就大大地影响了输入速度,为此系统提供另一类输入方法——词组输入法。2.13为词组输入提供手段有联想、预选、内词组和外词组。外词组又分为一般词组和以拼音字母开头的词组。联想输入方法,在输入时每联想一个词要选择一次,有时显得不甚方便;预选输入方法,根据预选置入的内容,查表一样地选择所需汉字或符号,该方法一是预置的内容不可能很多,二是翻页查表输入仍然不是很方便;内词组方法,可以即编即用,但系统关机或者热启动后即消失,而且词组量不可能很大;外部词组方法,事先根据自己的需要建立起词典文件,通过系统提供的命令将词典文件转换为词组文件,如果每次系统启动时,运行词组文件,就可以使用自己的词组库了。这种方法输入速度很快,例如输入“新疆电子计算中心”一词,只要输入“xjdj”,然后选择“1”,击五次键将词组输入。系统中对外部词组提供两个命令(CZ 和 CZP),分别对应于

CZ.COM 和 CZP.COM 文件,前者对应于一般词组即词组的输入码编码是任意的,由用户自己定义,由于这种方式太一般,所以使用很不方便(一般不用)。后者对应于拼音方式,使用和建立都比较方便,本文将重点讨论这一方法。

一、新建一个外部词组文件 在2.13汉字系统下建立一个外部词组文件必须经过三个步骤,第一是建立过程,包括建立词典文件和建立词组文件;第二是安装过程,是指如何把词组文件安装到内存;第三是使用方法,指如何使用外部词组,分述如下。

1. 建立过程 建立过程包括两个内容,第一是建立词典,第二是建立词组。在2.13操作系统中词典文件和词组文件是两个不同的概念,词典文件是一个文本文件(.TXT)即可以在字处理程序下进行编辑的文件。而词组文件则是一个可执行文件(.COM),它是通过相应的文件生成器生成出来的。下面具体讨论操作方法。

(1)建立词典 词典文件可用任何字处理程序编辑方式建立,如可用WS,EDLIN,PE等建立词典文件,该文件的基本格式是:

词组,词组

即词组和词组之间用逗号隔开,每一行结束用回车结束。下面三个词典分别为两个汉字的词组(CZWZK2.TXT),三个汉字词组(CZWZK3.TXT)和四个汉字词组及多个汉字词组(CZWZK4.TXT)。

CZWZK2.TXT

爱护,爱人,安定,安全,安置,安装,按排,按照,按期

八月,办理,白天,半径,伴随,办法,办公,办事,帮助,包含,包括,包围,包装,保持,保存,保护,保留,保密,保守,保卫,保养,保证,饱和,报导,报到,报道,报废,报告,报刊,报名

.....

CZWZK3.TXT

按规定,按计划

八进制,百分比,办公室,北京市,必然性,必要性,编辑部,编者按,标志着,标准化,不得不,不可不,不能不,不要紧,不至于,波特率

参考书,操作员,出版社,出发点,创造性,存储器,打印机,大部分,大多数,大幅度,大规模,大体上,大学生

.....
CZWZ4. TXT

按时完成

边缘科学,标点符号,不可否认,不切实际,不相上下,不言而喻,笔尖轻触

参考消息,参考资料,操作规程,操作系统,参照执行,程序变换,程序结构,程序控制

.....
数据库应用系统,数据库管理系统,数据传输速率,数据库自动化系统,自动生成应用系统定标器的键被压下

.....
注意,下面的词典文件是错误的。

爱护 爱人 安定 安全 安置 安装 (词与词之间是空格)

CAD 计划, 80 中国, dBASE 数据库 (字母和阿拉伯数字不能作为词头)

(2)建立词组 建立词组文件的工作很简单,利用 CZP. COM 文件将词典文件转换为词组文件即可。值得注意的是 CZP 命令可以同时处理多个词典文件。在系统下键入 CZP 命令,系统提示输入词典文件,处理完后,要求输入第二个词典文件,如果没有要继续处理的词典,输入回车即可。完后系统显示处理的词组数和所占空间,之后系统提示输入词组文件名,该文件名的头两个字母系统自动提示为 CZ,后面的字母输入即可,注意不要输入后缀. COM,否则会出错。如无错误,词组文件就被建立。下面是就上述三个词典文件建立为 CZWZK. COM 词组文件的具体操作步骤。

C)CZP

~~~~~词典文件用拼音字头方式转换为词组文件~~~~~

请键入词典文件名: CZWZK2. TXT

(输入二字词词典文件)

请键入词典文件名: CZWZK3. TXT

(输入三字词词典文件)

请键入词典文件名: CZWZK4. TXT

(输入四或多字词词典文件)

请键入词典文件名: [RETURN]

(输入回车,表示不再处理)

词组 1732 个,总长度 8037 字节

(显示处理结果)

输入词组文件名: CZWZK

(输入 WZK, CZ 系统自动给出)

该词组文件已经存在 覆盖(Y/RETURN)? Y

(第一次没有该行提示)

CZWZK. COM

(转换成功显示)

CCZZ1

CCZZ2

C)

2、安装词组 词组文件的安装比较简单,在系统启动之后,运行词组文件即可,例如安装 CZWZK. COM 文件,在系统下操作:

C)CZWZK

之后 CZWZK. COM 词组文件就被安装。当然也可以把 CZWZK 命令放在 213. BAT 批处理文件中,其位置在退出 213 目录(CD\)-之前(注: CZWZK. COM 文件在 213 目录下)。当系统安装 2.13 汉字系统时,就自动安装词组。

3. 使用词组 当安装了词组文件之后就可以使用了,具体使用方法是:输入词组的头三个字第一拼音,对于二字词输入前两个字的第一个拼音,然后,

输入“;”,就得到词组状态,例如希望输入“新疆电子计算中心”词组,输入: xjd; 屏幕上显示:

词组: xjd [000]; 现阶段 1: 新疆电子计算中心

选择 1 即可。其中 x 对应“新”字的首拼音字母, j 对应“疆”字的首拼音字母, d 对应“电”字的首拼音字母。

二、词组文件的分析 词典文件通过 CZP 命令被转换成词组文件,要想了解词组文件就必须对其进行剖析。下面通过一个具体实例来分析。

例: 有一个文本文件 TT. TXT, 其内容如下:  
演示程序, 程序通过 CZP 命令将 TT. TXT 词典文件

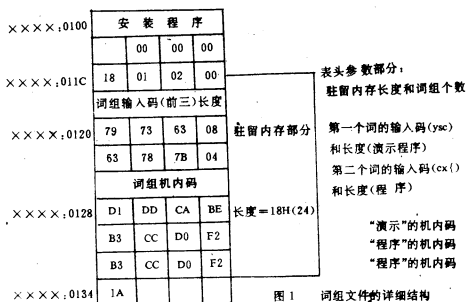
转换为 CZTT.COM 文件,我们可以通过 DEBUG.COM 程序来分析,反汇编结果如下。

```

; ***** 驻留程序 *****
3EAA,0100 8CCD    MOV    BP,CS      ;BP=当前代码段
3EAA,0102 BA2001  MOV    DX,0120    ;DX=索引表首地址
3EAA,0105 8B0E1E01 MOV    CX,[011E]    ;[011E]=词组个数
3EAA,0109 D1E1    SHL     CX,1      ;
3EAA,010B D1E1    SHL     CX,1      ;词组个数×4 字节=索引表长度
3EAA,010D 01D1    ADD     CX,DX      ;CX=机内码首地址
3EAA,010F B404    MOV     AH,04      ;
3EAA,0111 CD16    INT     16       ;键盘中断,装入词组
3EAA,0113 03161C01 ADD    DX,[011C]    ;[011C]=(机内码+索引表+表参数)长度,
; ***** 表参数 *****
3EAA,0117 CD27    INT     27       ;将其驻留内存
3EAA,0119 0000    ADD     [BX+SI],AL, ;00 00 00 为空白
; ***** 索引表 *****
7973    ADD     [BX+DI+73],BH, ;0120 索引表开始
3EAA,0122 63      DB     63       ;797363=ysc=输入码
3EAA,0123 086378  OR     [BP+DI+78],AH, ;08 词组长度(8 字节)
3EAA,0125 7B04    JPO     012C      ;索引表结束
; ***** 机内码 *****
3EAA,0128 D1DD    RCR     BP,1      ;D1DD="演"的机内码
3EAA,012A CABEB3  RETF    B3BE      ;CABE="示"的机内码
3EAA,012D CC      INT     3        ;B3CC="程"的机内码
3EAA,012E D0F2    ???     DL,1      ;D0F2="序"的机内码
3EAA,0130 B3CC    MOV     BL,CC      ;B3CC="程"的机内码
3EAA,0132 D0F2    ???     DL,1      ;D0F2="序"的机内码
3EAA,0134 1A      ;文件结束标志
    
```

通过反汇编程序的分析可以看出,一个词组文件分为四部分如图 1 所示,第一部分是安装程序,负责将词库驻留内存和用键盘中断装入词组;第二部分是索引表头参数部分,由于 DEBUG.COM 的开始偏移为 100,所以程序中实际值应该减 0100,程序中 1801,计算时应是 0118,减去 100 为 0018,转为十进制为 24,从图 1 中可以开出,要驻留部分的长度就是 24 字节。XXXX,011E 中的内容为词的个数;第三部分是索引表部分,它对每一个词组进行全索引,索引项长度为四个字节,前三个字节对应词的前三个字的第一个拼音字母,例如词“演示程序”的输入码是 ysc,对于二字词是前两字节为前两字的首拼音,第三个字节为“(”(7B),索引项的第四个字节为词组的长度(字数×2);第四部分为每个词组每个字的机内码,每个词之间顺序存放。第二、三和四部分实际上构造了一个输入码和机内码的对照表,统称为词库。这个词库被驻留内存之后,键盘中断就可以处

理,这就是外部词组原理。



三、实用词组库的建立 实用词组库的建立,本质上是建立词典的问题。在最初建立词典文件时,用户不可能说出许多词。我们可以借助其它一些资料建立词典文件,例如,系统提供的词典文件,联想的词典文件,长城机的词典文件和数据库内容等等。在建立起词组文件之后,对那些用到而没有纳入词典的词可以通过内部词组来不断增进。下面介绍具体方法。

1. 利用系统提供的三个词典文件 2.13 提供了 2ZC(二字词)、3ZC(三字词)、4ZC(四字词及多字词)三个词典文件,共计 5653 个词。这三个文件均在 7# 盘中,将这三个文件拷入到硬盘中,用 WS 软件对其进行编辑,找出适合自己的词,建立一个词典文件,然后转换为词组文件。

2. 取联想词典 联想本质上讲与词组没有太大的区别,也是事先建立了一个联想库,在建立联想库时也要通过相应的命令,将联想词典文件转换。这个联想词典文件的格式和词典文件是一样的,所以,如果系统中有联想词典文件,就可以直接拿来,通过 CZP 命令建立起词组文件。

3. 将长城机词组转为 2.13 外词组 在长城机上用惯了词组的用户,也可以将其词典文件拿来编辑。值得注意的是,原来的词典文件名是 WDLIB.CZ,词组文件是 WDLIB,转换文件是 WDCP.COM,而安装文件是 LOADCZ.COM。借助长城机的,是利用词典文件 WDLIB.CZ,其格式如下:

AHU,爱护 ARE,爱人 ADI,安定 APA,安排 AQU,安全 AZH,安置 AZH,安装  
APL,按排 AZH,按照 AQI,按期 ASI,按时  
BYU,八月 BLI,办理 BTI,白天 BHU,百货 BJI,半径 BSU,伴随 BFA,办法

显然这个词典文件和 2.13 汉字系统下的词典文件完全不同。利用这个词典文件进行编辑(用



WS)时,第一用块删除命令将所有的输入码删除,第二将分号改为逗号,第三将空格也删除掉,就形成2.13的词典文件。

4. 将dBASE文件转为外词组 在数据中,经常有一些站名,城市名,药名,厂名等文件,可以把这些名词变成词组,当进行数据库操作时,就可以用词组输入,这样既快又准确。下面结合一个具体例子来说明。

例如:有一个城市名文件如下:

| A1 | A2 | A3  | A4   |
|----|----|-----|------|
| 北京 | 上海 | 哈尔滨 | 乌鲁木齐 |
| 南京 | 广州 | 海拉尔 | 呼和浩特 |

将这个文件通过 COPY TO TT.TXT SDF 命令变成一个 TT.TXT 文件,结果如下:

北京上海哈尔滨乌鲁木齐  
南京广州海拉尔呼和浩特

这个文件与词典文件格式的唯一差别是,词与词之间没有逗号,把它编辑到老词典文件中即可。

5. 将内词组转为外词组 一个实用的词组库不是词组的个数越多越好,而是要满足自己专业的需要而又很少翻找为最好,除了用上述的方法收集初始资料外,还要靠平时的不断收集,如何进行这种实时收集呢?可以通过内部词组的方法。内部词组的管理通过按 ALT+F9 键,屏幕出现:

内部词组管理:A-添加,X-显示,Q-清除,\退出

选择 A 是向内部词组库中添加新词组,选择 X 是显示内部词组还剩多少空间,如果内部词组空间设置为 2KB(设置由驱动文件中 FILE1A 2,2 是指 2KB 内部词组空间),则开始显示为“空间 02048 字节”,每增加一个词组就减少一些字节。选择 Q 是彻底清除内部词组中词组。下面通过一个具体实例来说明词组收集方法。

例如:实时收集词组。在用 WS 编辑一个文本文件时,如下屏幕是一个操作序列。

#### 新疆电子计算中心是从事计算机应用和开发的专门机构

- ```

25 0 内部词组管理:A-添加,X-显示,Q-清除,\退出
1 内部词组管理:A-添加,X-显示,Q-清除,\退出 空间:02048 字节
2 编码:{内容:从光标起字符数 \
3 内部词组管理:A-添加,X-显示,Q-清除,\退出 (光标定在新的下面,输入 A)
4 编码:aaa 内容:从光标起字符数 4 \
5 内部词组管理:A-添加,X-显示,Q-清除,\退出(光标定在电的下面,输入 A)
6 编码:aaa 内容:从光标起字符数 4 \
7 内部词组管理:A-添加,X-显示,Q-清除,\退出(光标定在计的下面,输入 A)
8 编码:aaa 内容:从光标起字符数 8 \
9 内部词组管理:A-添加,X-显示,Q-清除,\退出(光标定在计的下面,输入 A)
10 编码:aaa 内容:从光标起字符数 6 \
11 内部词组管理:A-添加,X-显示,Q-清除,\退出
12 词组:"aaa[000]:新疆 1:电子 2:计算中心 3:计算机
  
```

在这个操作序列中,0~12行都是在第25行上显示的内容,为了清楚以序列形式列出来。0行是输入了 ALT+F9 之后的结果;1行是输入 X 的结果;2行是输入 A 的结果,其中 {{ 表示三位编码的位置, (=7BH 是该位编码的缺省值。从光标起的字符数是指你事先将光标定在的位置开始的词组字节数(一个汉字两个字节),回车后出现第3行。在第3行上,首先将光标定在“新疆”的新字下面,然后输入 A,编码输入 aaa,字符数输入 4,回车之后就建立了词组“新疆”,直到11行完成四个词组的建立。在11行上

输入回车就退出内部词组管理。然后在拼音、快速和首尾等几种字符输入方式中,输入 aaa 之后,再输入'之后,屏幕上出现第12行,四个词组出现。

上述过程就是词组的实时收集,有了这些词组,用 WS 将这些词组编辑到原来的词典文件中去,再用 CZP.COM 文件转换外词组文件,多次收集就得到一个适合于自己专业的词组库。我们知道内部词组一旦关机就丢失,词典文件并不是每天进行编辑,我们希望当内部词组文件大到一定程度时,进行一次编辑。如何存下内部词组呢?系统提供一个 CN.

## 微机预测注射剂和滴眼剂有效期的方法

吉林省计算机技术研究所 张鸿鸣 刘铁军(130012)

吉林省人民医院眼科 孙玉华

**摘要** ①微机预测注射剂和滴眼剂的重要意义及方法;②通过实验取得全部预测初始数据;③经典恒温法编微机程序所需全部计算公式;④微机程序框图和程序;⑤最优化及最佳结果。

许多注射剂和滴眼剂的稳定性能都很差,因此,只好定期重新配制。配制注射剂和滴眼剂是杀菌剂中经常性较繁重的工作,由此看来,预测药物有效期是贮存单位必不可少的一项工作。

预测药物有效期,总的原则是在若干个高于室温的温度下加速药物变质的化学反应。并在不同的时间取样测定其含量,求出药物在不同温度下变质的速度常数。然后,再根据 Arrhenius 温度与反应速度的关系式,求出室温下的速度常数及有效期。其具体预测有效期的方法可分为:经典恒温法、线性变温法、阶梯变温法、单测点法和初匀速法等。现以盐酸丁卡因注射剂为例,研究微机预测注射剂和滴眼剂有效期的方法。

盐酸丁卡因注射剂是盐酸丁卡因的杀菌水溶液,正常情况下含盐酸丁卡因为( $C_{10}H_{24}N_2O_2 \cdot HCl$ ) 0.5%或1%,其含量应为标准量的95~105%,是医院杀菌制剂室经常配制的注射剂。丁卡因含有脂链结构,易水解而失效,其水分解产物为丁氨基苯甲酸和二甲氨基乙醇。

通常,制剂手册注解中都说明本品不宜久藏。为确保制剂质量,减少配制工作的盲目性,我们采用微机进行计算和数据处理来完成对盐酸丁卡因注射剂有效期的预测。此法预测可以实现合理按期配制,既

能减轻制剂的工作量,节省很多药品,又能保证药剂的质量。

### 一、收集数据实验

数据是通过实验进行收集的,其具体作法是:

#### (一)仪器与试剂

1、仪器:PHS-2型酸度计(231型玻璃电极,232型饱和甘汞电极,使用前经标准磷酸盐缓冲液校准)、LB801型超级恒温器和731型分光光度计。

2、试剂:盐酸、磷酸二氢钾、无水磷酸钠和苯等,均要求二级规格。

#### (二)操作

调节恒温水浴为一定温度,在若干只纱布口袋内放入盐酸丁卡因注射剂安瓿数支(注射剂在杀菌后应立即藏于冰箱内)。将其口袋全部沉入水浴的水面下,立即记录时间。经过一定时间取出一只布袋,迅速放入冰水中冷却,取样分析丁卡因含量。再在七个不同温度下重复上述操作过程,便能取得全部实验数据。

#### (三)丁卡因含量分析

采用紫外分光光度法直接测定样品丁卡因含量,并准确汲取1%盐酸丁卡因注射剂安瓿内样品2.5ml(约含盐酸丁卡因25mg),放入250ml容量瓶内,加入到刻度,并混均匀。再准确汲取25ml此液放

COM文件专门用来存储和恢复内部词组,下面结合上例说明。

#### (1)存储内部词组

C)CN XD (XD是一个附加文件名)

C)DIR ?? XD

CZXD.COM (内部词组被存成CZXD.COM文件)

#### (2)恢复内部词组

C)CN CZXD

这样在每次关机时作一次存储,在开机时作一次恢复,当你需要编辑时,就用这些新词进行修改老词典文件。

在这种方法中要说明几点,第一,由于内部词组

文件和外部词组文件其结构是完全一样的,所以,上例中所存储的CZXD.COM完全可以作为外词组文件。第二,在每个词的编码中,其编码是任意的,可以是拼音,可以是五笔字型等等,但是一旦一个词组被编成一个指定代码,就必须记住它,否则就找不到该词组,因为词组文件不是.TXT文件,不能用WS和TYPE等形式查看。内部词组如果仅仅为了收集词组,我们建议就用aaa一个编码,可以用“>”翻页键来查看所有的新词。第三,在将新词编辑到老词典中去的时候,不存在什么拷贝的方法,而是在某一输入方法下(例如拼音),输入aaa'之后,选择1,2...,将一个一个的词输入到词典文件中去,然后通过翻页再将新词输入进去。

入分液漏斗内,加入苯 50ml 和  $\text{Na}_2\text{CO}_3$  试剂 2ml,并振摇一分钟,放置分层后,弃去水相,轻轻旋转分液漏斗,后放置分层,再弃尽水相。用 1:200 稀盐酸分两次(第一次 25ml,第二次 20ml)与苯相振摇各一分钟,放置分层后,合并所得水相于 50ml 容量瓶内,轻轻旋转分液漏斗,将取后的萃取液合并于容量瓶中,加 1:200 稀盐酸到刻度,混均匀,准确汲取 10ml 此溶液于 50ml 容量瓶中,并加 5ml PH6.0 磷酸缓冲液(将 20 克  $\text{KH}_2\text{PO}_4$  与 80 克  $\text{K}_2\text{HPO}_4$  溶于 1000ml 蒸馏水中,用  $18\text{NH}_3\text{PO}_4$  或  $10\text{N KOH}$  调至  $\text{PH}6.00 \pm 0.05$ ),用蒸馏水稀到刻度,混均匀。在实测最大吸收峰 314nm 处,用 1cm 比色杯测定吸收度。

## 二、计算公式

根据化学动力学原理建立经典恒温法所用数学公式,经推导整理如下:

$$\text{LOGC} = \text{LOGC}_0 - K_1 t / 2.303 \quad (1)$$

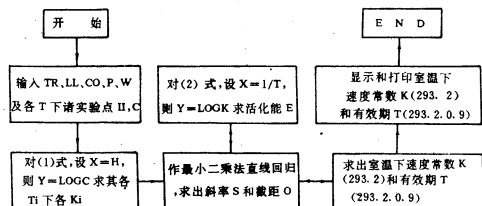
其中  $K_1 = (2.303 \text{LOGC}_{21} / C_{21+1}) / (t_{21+1} - t_{21})$  或  $K_1 = 2.303 \text{LOGC}_0 / C / t$

$$\text{LOGK}_1 = \text{LOGA} - E / 2.303RT_1 \quad (2)$$

其中  $\text{LOGV}_0 = \text{LOGA} - E / 2.303RT_1$  和  $V_0 = (C_0 - C) / t$

$$T(293.2, 0.9) = 0.1054 / K(293.2) \quad (3)$$

## 三、微机程序框图和程序



图一 微机程序总框图

1、根据计算公式,对直线方程作最小二乘法,并进行回归处理(代替作图),求得直线的斜率,从而求得  $T_1$  温度下的反应速度常数  $K_1$  和  $E$  或者开始升温温度下的速度常数  $K_0$  及室温( $20^\circ\text{C}$ )下的有效期  $T(293.2, 0.9)$ 。其经典恒温法预测有效期程序总框图如图一所示。

2、经典恒温法预测药物有效期微机程序是用 BASIC 语言编写的,并且是在 AST286 微机上实现的。其他四种有效期预测方法微机程序也是用 BASIC 语言编写的,并在同种型号微机上实现的。

## 四、最优化

1、盐酸丁卡因含量分析的方法很多,其中有银量法,中和法和非水滴定法等最为常用。但是它们都不是直接测定丁卡因药物本身含量的方法,因而不是我们所需要的方法。这里,我们采用的是紫外分光

光度法,可以不受水解产物等杂质的干扰,并且可以直接测定丁卡因含量。

在我们的实验中,比色液 PH 对吸光度影响很大,若 PH 取 4.5 将明显降低吸光度。本注射剂按上述含量分析,最后比色液 PH 可维持在 5.8 左右,适于分光光度测定。

在初均速法预测中,需要将吸度转换为浓度,  $A(1\text{cm}, 1\%) = 745$  为选用系数,得  $A = (1\text{cm}, 10\mu\text{g}/\text{ml}) = 0.745$ ,我们再根据 Beer-Lendert 定律,按  $C = A1\text{cm} / 0.745$  计算比色液中丁卡因浓度。再按稀释的倍率折算样品中丁卡因含量。

2、预测药物有效期方法中,所取室温的标准值得商榷。通常都取室温算术平均值  $20^\circ\text{C}$  或  $25^\circ\text{C}$ ,从而从本实验来讲,不同温度下丁卡因水解反应速度常数  $K$  或初均速度  $V_0$ ,均显示了随温度升高而迅速增大。现以  $15^\circ\text{C}$ 、 $20^\circ\text{C}$ 、 $25^\circ\text{C}$  和  $30^\circ\text{C}$  不同室温为标准,应用微机按四种预测方法进行预测,得到贮存有效期如表 A 所示:

四种预测方法对不同贮存温度有效期对比表 (A)

| 预测方法  | 不同温度下有效期(年)        |                    |                    |                    | 相对 $30^\circ\text{C}$ 有效期(倍) |                    |                    |
|-------|--------------------|--------------------|--------------------|--------------------|------------------------------|--------------------|--------------------|
|       | $15^\circ\text{C}$ | $20^\circ\text{C}$ | $25^\circ\text{C}$ | $30^\circ\text{C}$ | $15^\circ\text{C}$           | $20^\circ\text{C}$ | $25^\circ\text{C}$ |
| 经典恒温法 | 12.5               | 6.30               | 3.23               | 1.70               | 7.4                          | 3.7                | 1.9                |
| 阶梯变温法 | 12.0               | 6.13               | 3.20               | 1.71               | 7.0                          | 3.6                | 1.8                |
| 单测点法  | 12.1               | 6.06               | 3.10               | 1.62               | 7.5                          | 3.7                | 1.9                |
| 初均速法  | 11.4               | 5.72               | 2.94               | 1.54               | 7.4                          | 3.7                | 1.9                |

化学动力学理论指出,对于某一化学反应(活化能  $E$  一定),温度对化学反应速度的影响是指数关系  $K = AE(-E/RT)$ 。不同温度下,某一化学反应的速度常数比值由下式

$$\text{LOG}(K_1/K_2) = E(T_1 - T_2) / 2.303RT_1T_2$$

决定。其中  $K$  为温度  $T$  的速度常数; $K_1$  为温度  $T_1$  的速度常数; $K_2$  为温度  $T_2$  的速度常数; $R$  为气体常数 1.937 卡/摩尔·度; $E$  为该反应的活化能。

一般化学反应的活化能在 20~60 千卡/摩尔之间,若以 60 千卡/摩尔计算,则  $15^\circ\text{C}$ 、 $20^\circ\text{C}$  与  $25^\circ\text{C}$  的速度常数分别为  $30^\circ\text{C}$  的 0.56%, 3.4% 与 19%,相应地有效期将是  $30^\circ\text{C}$  的 178 倍、30 倍和 5.3 倍。即使以 15 千卡/摩尔较低活化能(温度对反应速度常数影响较小)的反应来讲,  $15^\circ\text{C}$ 、 $20^\circ\text{C}$  与  $25^\circ\text{C}$  贮存室温的有效期也是将分别为  $30^\circ\text{C}$  的 3.6 倍、2.3 倍和 1.5 倍。

根据以上预测结果与化学动力学分析结果均表明,以不同贮存室温来计算同一药物的有效期有很大的出入,并且无论反应的活化能较高或较低,  $20^\circ\text{C}$  的有效期都远长于  $30^\circ\text{C}$  的有效期。因此,为确保制剂质量,不宜采取年算术平均室温  $20^\circ\text{C}$  作为计算有效期的贮存室温标准。我们认为,药物有效期应当

## 条形码阅读器的设计实践

四川省电子计算机应用研究中心 马在强 朱云 李尚明 (610015)

目前在超级市场、图书馆、证件检查等领域,已广泛使用条形码阅读器作为信息采集、处理、传输、显示的工具。我国在条形码阅读器的需求方面,也已经呈现出一种上升趋势。国内好几家计算机公司已先后从国外引进整套或部分条形码阅读器,在市场上推广销售。

根据我们的实践,我们认为推动我国条形码阅读器的应用,可以采用引进和自我开发相结合的路子,特别是发挥我国软件开发的优点,减少投入费用,按我国自己的编码方案,开发出适合国情的应用软件。下面就我们对条形码阅读器的开发实践,作一介绍。

### 一、条形码阅读器的工作原理。

通过光笔扫描条形编码,生成随时间变化的信号。该信号与条形与间隔(白条形)宽度对应。一个单位相对宽度的条形表示0,两个单位相对宽度的条形表示1。间隔都是一个单位宽,一行条形编码的字节数,可依需要而决定。

通过计数间隔到条形和条形到间隔期间系统时钟周期数并与前导条形和间隔时钟周期数求出的参照相比较,得出编码。

条形码阅读器一般采用手工扫描条形码的工作方式。扫描速度不可能很均匀。为了避免误差,在判断条形宽度时,运用相对宽度概念,与先导条形宽度作比较。一般在先导条形的基础上,加上权值 $W$ ,若当前条形宽度 $<$ 先导条形宽 $\times W$ 则规定为0,若 $\geq$ 先导条形宽 $\times W_1$ 则定义为1。

最大/最小条形宽度由扫描速度和微处理器决定。对于象8031这样的单片机而言,最大条形计数等于 $2^{16}$ 时钟周期,若采用6MHZ晶振,则机器周期

$T=2\mu s$ ,最大条形计数 $2\mu s \times 2^{16} \approx 131ms$ ,人工扫描速度一般在76.2mm/S~762mm/S之间,若按扫描速度76.2mm/S计,最大条形宽度为9.98mm,最小条形宽度由条形的编码时间和扫描速度共同决定。例如,编码时间为3ms,扫描速度为76.2mm/S,则最小条形宽度为229 $\mu$ 。

在数据编码条形之外,还需要附加条形码,譬如在一行的条形编码两端各有一个单位宽及二个单位宽的条形作为初始参照,以及跟参照相邻的条形,决定扫描的方向,是从左到右,还是从右到左。双向扫描条形数据的处理由接口芯片完成,它在一行条形编码位的缓冲区中,找两个方向中的一个编码。

### 二、条形码阅读器的设计实践

#### (1) 阅读头

阅读头又叫光电扫描器,由光学系统和电路系统构成。简单光电扫描器其光学系统由光源、透镜和光阑等元器件组成,作用是当扫描器扫描时获取瞬间光信号。电路系统由光敏元件、放大整形电路和接口电路组成,主要作用是将光信号转换成电信号,然后进行放大和整形,最后以二进制脉冲信号输出给译码器。作为光电扫描器的光源常用的有发光二极管和激光器二种。低档阅读头可以用发光二极管作为光源,高档阅读头可以用激光器作为光源,除了上述光电系统外,其内部还要配备适当的光学和机电系统。另外也有采用磁性阅读头的,这就要求条形码制作在磁性介质上。

#### (2) 显示器

阅读器可以通过串、并口与宿主机联接,这样,就采用宿主机的CRT作为其显示器。阅读器也可以单独使用,自带一个显示器。这样的显示器可以由数

按30℃下贮存进行预测,这样才能得到所需要的最优优化贮存有效期。

### 五、最佳结果

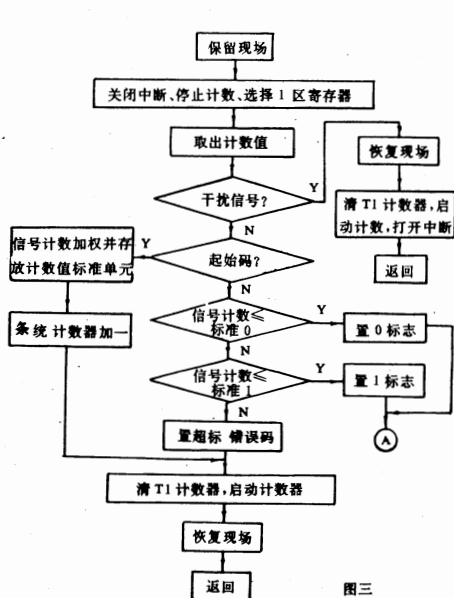
首先输入温度、时间和吸收度等实验数据,按经典恒温法求出不低于标示量的95%,30℃室温下贮存的有效期 $T(303.2, 0.95)=1.7$ 年。同理,按阶梯变温法,求出 $T(303.2, 0.95)=1.71$ 年,按单测点法,求出 $T(303.2, 0.95)=1.62$ 年,按初均速法,并可求出 $T(303.2, 0.95)=1.54$ 年。

显而易见,微机四种预测药物有效期的方法所得结果基本一致。按照盐酸丁卡因(Ig)、N/10盐酸(0.5ml)和注射用水(Q·S),合计总共制100ml盐

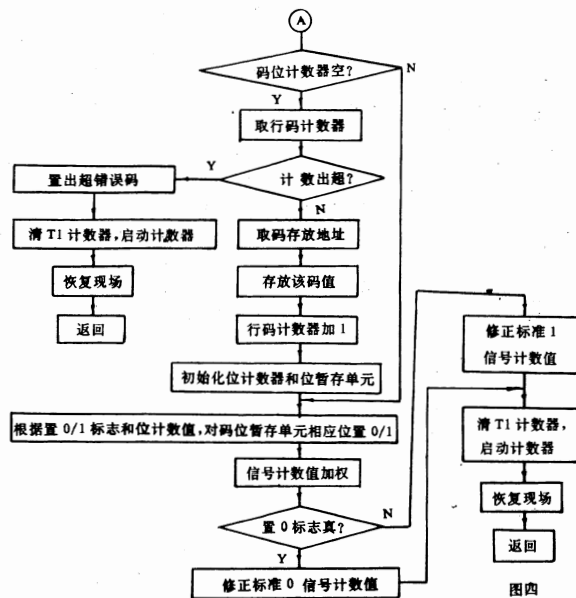
酸丁卡因注射剂的处方,又根据允许浓度降低到标示量的95%的规定,30℃贮存有效期 $T(303.2, 0.95)$ 应当定为1.5年,此为最佳结果。

综上所述,各种预测物有效期方法需要大量的数据处理,相当繁琐。自从计算机问世以来,可以应用计算机迅速完成这种繁重工作。甚至不必精通化学动力学原理的药房工作者只要会使用计算机,利用编制好的预测计算程序,按照计算机屏幕上显示的问号,按动键盘的有关键输入相应的实验数据,就能迅速显示出有效期的正确结果,并打印出来。因此,对保证质量配制注射剂和滴眼剂具有重要意义。





图三



图四

(上接 33 页)

分枝定界——隐枚举算法。详细描述略。

```

f1 = 0
for i1 = 1 to n do
if x[i] = 1 then
f1 = f + c[i]
for i2 = 1 to n do
if x[i] = 0
then begin
f1 = f + c[i]
if f > optimal then
begin
x[i]1 = 1;
goto 10;
}goto examine constraints
end;
end

```

```

else
x[i]1 = 0;
(a) 分枝定界法

```

```

for i1 = 1 to n do
if (c[i] = 0) and (i > j)
then begin
x[i]1 = 1;
goto 10;
}go to examine constraints
end

```

else

x[i]<sub>1</sub> = 0;

(c) 隐枚举算法

图2 0-1问题的枚举算法

3. 算法的比较 无疑,在枚举算法中,完全枚举法最简单,然而却是最不实用的。因为完全枚举法需要花费变量数量的指数次数遍历满二叉树,所需时间过长。

分枝定界法和隐枚举法各走极端:当约束条件较松时,分枝定界法极有利于删去满二叉树的部分枝叶,从而减少遍历二叉树的时间,提高速度;反之,当约束条件较紧时,隐枚举法有利于删去满二叉树部分枝叶。然而,在最坏的情况下,分枝定界法和隐枚举法所需的开销和完全枚举法的相同。

分枝定界——隐枚举算法将分枝定界法和隐枚举结合起来,充分利用了两个算法的优点,使得无论在约束条件是紧还是松的时候,都能删去满二叉树的部分枝叶,从而减少了遍历二叉树所需时间,提高了速度。

上述算法的优越性已在文献[1]的工作中得到验证。

### 参考文献

1. 陶友传,李培根,段正澄等:“柔性零件装载问题的建模及算法”,《组合机床与自动化加工技术》,1991,(3)。



## 与 PC 机兼容的专用键盘的设计制作

东南大学热能所

周立 赵以钰 (210018)

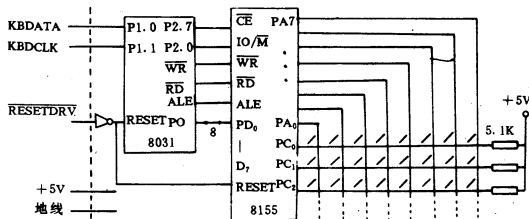
南京师范大学

王晓红

**摘要:**采用单片机设计、制作专用键盘,键可采用任一种开关,使其与 PC 机在软硬件上完全兼容。

**一、引言** 目前,与 PC 机兼容的工控机对于各种手动信号与指令输入有两种方式来实现,一种是采用 PC 机标准键盘,软件定义各键特殊功能;另一种是采用多个并口将来自各处的信号与指令读入。前者不适合众多的键盘类型(如薄膜开关、按钮开关等),不适合于地点分散的指令输入,而且对于某些未定义键要采取屏蔽措施(例如对 Ctrl—Alt—Del 热启动键,在未定义时一定要屏蔽),而后者会占用过多的 CPU 时间以及额外的硬件接口。笔者设计、制作了一个兼顾以上二者优点的键盘,键可采用多种类型,只要开通与断开反映阻抗变化就行,键盘接口采用 PC 机标准五芯接口,软件也与 PC 机完全兼容,对 PC 机 BIOS 而言,它如同标准键盘一样,对用户而言,设备面板上的薄膜开关,按钮开关等示意明确,有利于现场操作。

**二、方案的实现** 采用 8031 单片机为键盘管理 CPU,采用行列扫描法扫视键盘,软件去抖动,再对应按键位置查表得其扫描码,以串行方式发送给 PC 机。硬件线路如图一所示。



图一 键盘电路

软硬件要点说明:

1、按一键并释放之,PC 机的键盘中断服务程序就收到一个按键扫描码和一个释放键扫描码(等于按键扫描码加 80H),并转化为相应的键的 ASCII 码存放于键盘缓冲区中。101 标准键盘上的各键扫描码可用软中断测出:

```
NNL1:MOV AH,01H
```

```
INT 16H
```

```
JZ>NNL1
```

扫描码测试程序(1)

出口参数 AH 中即为按键之扫描码。

2、发送串行扫描码通过 KBDATA 线,由 KBDCLK 线的上升沿电平跳变引起主机键盘接口的 74LS322 串行移位寄存器将 KBDATA 线上电平读入并移位。KBDCLK 跳变由软件编程实现,其时钟速率要求低于主机的同步时钟 PCLK,下面为 8031 向主机发送扫描码程序:

```
OUTT:XCH A,R5
```

```
OUTP:MOV A,#0FFH
```

```
MOV P1,A
```

```
MOV A,P1
```

```
JNB ACC.0,OUTP
```

```
JNB ACC.1,OUTP
```

```
XCH A,R5
```

```
SETB C
```

```
MOV P1.0,C
```

```
ACALL CLK1
```

```
MOV R3,#08H
```

```
OUTPA:RRC A
```

```
MOV P1.0,C
```

```
ACALL CLK1
```

```
DJNZ R3,OUTPA
```

```
RET
```

```
CLK1:CLR C
```

```
MOV P1.1,C
```

```
SETB C
```

```
MOV P1.1,C
```

```
RET
```

发送扫描码程序(2)

3、对于采用单键承担复合键功能(例如单键用作屏幕硬拷贝,SHIFT 键+PRINT 键),软件上要连续发送相当于复合键的多个扫描码,同时,释放键的扫描码也不能丢,因为 ALT、CTRL 和 SHIFT 三键的按下和释放都对应着 DOS 管理键盘的相应标志位的位置 1 或清 0。

4、PC 机在冷启动或热启动时,BIOS 都将对键

## 伪彩 B 超中微机的应用和软件的编制

中国工程物理研究院应用电子所 唐 丹 王沐 (610008)

**摘要:**本文介绍了伪彩 B 超中微机的应用,较为详细地说明了 B 超帧存贮器和微机内存交换数据的方法,概要讲述了软件设计的方法和要点,着重讲述了 B 超图象打印驱动程序的设计。

**一、引言** 在伪彩 B 超的研制中,厂方要求 B 超具有图象冻结、存储、打印、伪彩色处理、图象存取和方式控制等功能,这些功能必须借助于微机才能实现,故此,我们在 B 超的研制中选用性能价格比较高的“PC-BOY”(和 IBM-PC 兼容)作机芯,所有的硬件实现和接口电路都放入“PC-BOY”的机箱,使之外观较为紧凑,软件采用“IBM 宏汇编”和“FORTRAN4.0”混合编程。

**二、微机在 B 超中的应用** 微机作为 B 超中的核心,用其本身丰富的资源能较为容易地实现存盘和打印,但由于 B 超图象是存在存贮器中的,打印和存盘必须要以微机内存作为中转站,故此,帧存和微机的数据交换显得十分重要,为满足数据刷新和动态

显示的需要,帧存设计为两组,每一组由两片静态 RAM 构成,两组分时工作,一帧图象数据刷新的同时,另一帧用来显示,这样就实现了图象的动态显示,图象冻结实际上就是停止帧存中数据的不断刷新,而固定显示某一帧存中的图象,它是通过键盘使微机给状态控制电路发一控制字实现的,图象的伪彩色处理是由彩色查表 LUT 来实现的,此 LUT 实际上由一片 2Kx8 的静态 RAM 构成,主要用作图象数据的转换,它是将原图象数据作为此静态 RAM 的地址,去查找预先放在 RAM 中的数据,改变此表格中的数据就能把原图象数据对应变换为想要加工处理的数据,色彩的变换实际上就是改变表格中的数据,使原 B 超图象的不同灰度可以任意改变颜色。

盘进行测试,其自检程序段如下:

```
F000,E2A1 CALL E5E6
F000,E2A4 JCXZ E2AB
F000,E2A6 CMP AH,AA
F000,E2A9 JZ E2B1
F000,E2AB MOV SI,FA06
F000,E2AE CALL EC4F
F000,E2B1 CLD
F000,E5E6 MOV AL,08
F000,E5E8 OUT 61,AL
F000,E5EA MOV CX,5C30
F000,E5ED LOOP E5ED
F000,E5EF MOV AL,C8
F000,E5F1 OUT 61,AL
F000,E5F3 MOV AL,48
F000,E5F5 OUT 61,AL
F000,E5F7 MOV AL,FD
F000,E5F9 OUT 21,AL
F000,E5FB STI
F000,E5FC XOR BL,BL
F000,E5FE XOR CX,CX
F000,E600 TEST BL,FF
F000,E603 JNZ E609
F000,E605 JMP E607
F000,E607 LOOP E600
```

```
F000,E609 IN AL 60
F000,E60B MOV AH,AL
F000,E60D MOV AL,C8
F000,E60F OUT 61,AL
F000,E611 MOV AL,48
F000,E613 OUT 61,AL
F000,E615 RET
PC 机自检键盘程序(3)
```

因而 8031 在扫描键盘矩阵的同时也要测 KBD-CLK,若其为低电平达到 20ms,则判定为 PC 机在做复位键盘自检,此时待 KBDCLK 线变为高电平后,8031 即将 AA 码送给 PC 机,就可通过自检,否则 PC 机喇叭响,且屏幕上出现“KB-ERROR!”字样,表示键盘自检未通过。

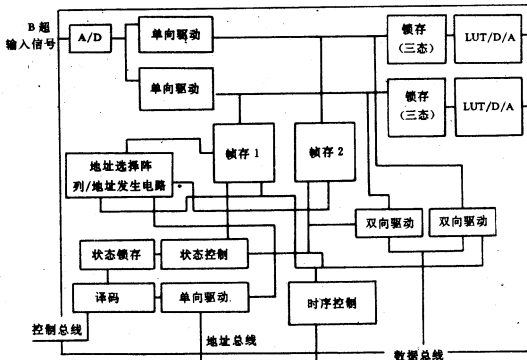
**三、结束语** 这样的兼容结构将有利于软、硬件设计规范化,同时可使在 PC 兼容机上用 101 标准键盘调试的程序在换用专用键盘后无需任何变化,规范化的设计同时也大大降低了软、硬件设计成本。

### 参考文献

[1]张载鸿:《IBM-PC/XT 硬件系统分析与应用》,科海培训中心,1990。

[2]何立民编著:《单片机应用系统设计》,北京航空航天大学出版社,1990.1

B超图象的扇形扫描有不同的角度和深度,B超中有一系列状态寄存器,这些状态寄存器控制B超扫描的方式,微机通过I/O口向这些状态寄存器发不同的控制字,从而改变B超的扫描方式。



图一 B超中和微机有关电路原理框图

三、微机和帧存交换数据的方法,和微机高速交换数据的方式通常有两种,一种是DMA方式,另一种是存储器映象方式,DMA方式是在微机中8237A的控制下,使数据直接发生交换,而不需CPU的干预,映象方式是把计算机内存中的地址空间分配给帧存存储器,在数据交换时,微机把帧存视为内存的一部分。结果帧存和内存的数据交换被看作是内存中两个不同数据区间的数据交换。

a. DMA 转送方式 PC中用的8237-5DMA控制芯片有4个通道,其中两个已由PC使用了,通道0用来完成系统单元的动态存储器刷新功能;通道2用来在磁盘驱动器适配器和存储器之间传送数据,目前,通道1和3尚未使用,通道1,2和3均已引出到系统总线上,可供安装在总线插槽内的接口板使用,参考文献[1,2]有详细的硬件和软件资料说明,这里只给出初始化宏汇编程序段。

初始化一通道,地址和长度

```

dmasd macro leg, pag, basadr
mov al, pag
and al, 0fh
out 83h, al; 页
out obh, 01001001b;
dout 0ch, al; 空写, 消除指针
mov ax, basadr
out 2, al
mov al, ah
mov 2, al
mov ax, leg
out 3, al

```

```

mov al, ah
out 3, al
mov al, 00000001b
out 0ah, al; 允许一通道请求
dout 226h, al; 请求
endm

```

DMA方式中不允许操作跨越64千字节的边界,故程序中一般应有“越页处理”的程序段。

b. 存储器映象方式 在B超中,帧存分配的内存地址空间为0a0000h-0a8000h,电路见图一,此电路可认为是一组门,在要交换数据时门打开,使微机把帧存能当成内存的一部分,内存中两个不同数据区的数据交换用数据串传送指令就可实现,下面程序为向帧存中灌黑色背景和灰阶的程序:

.....

```

dout 228h, 02h ] dout 为程序中定义的“送
dout 22ah, 0fdh ] 控制字”宏
dout 229h, 82h ] 打开三态门
dout 22ch, 0

```

```

mov ax, 0a000h
mov es, ax
mov c, leg1, leg=8000h
mov si, offset bj; 背景数据
mov di, 00
stat1: mov al, ds:[si]
mov es:[di], al
inc si
inc di
dec cx
jnz stat1

```

```

dout 228h, 2ah ]
dout 229h, 0c2h ]
dout 22ah, 0fch ]
dout 22bh, 48h ]

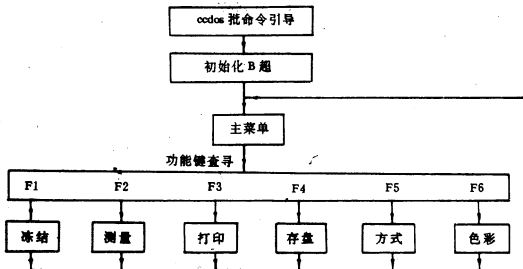
```

关三态门

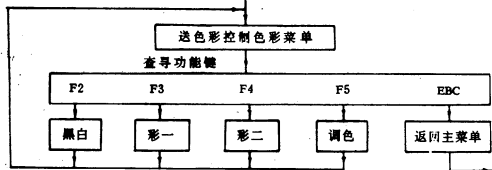
这两种高速数据的交换方式从速度上讲,DMA方式较快,而从硬件实现上来说,存储器映象方式相对简单,在B超中速度没有严格要求,故此,我们在B超中是采用存储器映象方式。

四、软件的编制 为了控制伪彩B超的工作和数据的交换,所有控制软件都采用“IBM-PC宏汇编”编程,为便于图象数据在内存中处理计算,主程序采用“FORTRAN4.0”编程,整个程序采用模块化结构,把程序按功能分为尽可能小的独力的模块,每一模块都可分别调试,整体看来每一模块是由汇编语言编成的子程序库。这些库可很容易的被“FORTRAN”语言的主程序所调用,作这些库的关键是子程序和主程序间的参数传递,本程序中是用“数

据公用区”和“帧结构”。一旦库作好后,主程序可根据需要灵活调用,下面为B超部份程序的流程图。



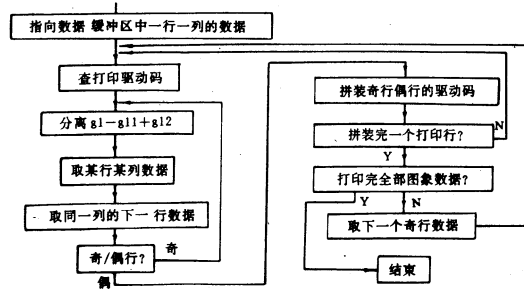
图二 主菜单程序流程图



图三 色彩菜单程序流程图

**五、打印驱动程序的设计** 作一个子程序设计例子,打印驱动程序具有一定的代表性,B超图象共由  $256 \times 256$  个象素单元组成,由于一个象素只占半个字节,两个象素拼为一个字节,图象数据在内存中占 32KB,图象的灰阶为 16 级,在程序中首先应开辟一个 32KB 的缓冲区,以备图象数据从帧存调入内存而占用,此子程序与其它子程序交换数据的方式为“数据公用区”,此程序复杂之处是字节的分离

和打印驱动码的拼装。



上面给出的是  $4 \times 4$  点阵为一象素的打印驱动程序的流程图,此程序在 MX-80 上运行,速度较快。同样,我们也可编制  $2 \times 2$  点阵的打印驱动程序。

**六、结束语** 伪彩 B 超中较为充分地使用了微机的资源,微机在 CCDOS 支持下,采用中文菜单提示,操作十分方便,得到用户的好评。CCDOS 在应用 B 超时,6845 的初始数据区和汉字提示要作相应改动。此 B 超的研制为 B 超国产化作了一些有益的探索。

#### 参考文献

- [1]: [美] L. C. 埃杰布赖奇 著 《IBM-PC 接口技术》
- [2]: 周明德 田开亮 著 《微型计算机接口电路及应用》

## 用 C 语言编写特殊的文件打印程序

编写程序时,根据各人的习惯,有人喜欢用小写字母,有的则喜欢用大写,但在阅读别人的程序时,一般习惯于小写。然而在编完一套软件要打印一份完整的程序清单时,则为了美观而用大写。有的杂志社则要求程序清单中的字母能用大写的尽量用大写。此外,大家在阅读有些软件类似 README 的说明文件时,在屏幕上显示时很整齐,可是在打印机上打印时却不整齐,甚至出现一些怪符号。这是因为在这些文件中使用了制表键(TAB)的缘故。

因此我们希望有一个较好的打印程序来解决以上提出的问题,能将文件中的大、小混用的字符全部

以小写或大写字符打印,并且能够按显示的格式将文件内容如实打印出来,排除文件中由于使用了<TAB>键后的干扰。

本文向您介绍的就是一个用汉化 C 语言编写的文件打印程序,该程序的基本结构较简单,读者一看就一目了然,只是在对文件中的 ASCII 值为 9 的字符(<TAB>)进行处理时有些技巧,您可以参考一下,也可以用类似的方法将文件中的特殊字符作相应处理使该程序的功能更完善。

需此程序清单者,请同作者直接联系(湖北省天门市 121 号信箱微机室 周日初 邮编 431734)。

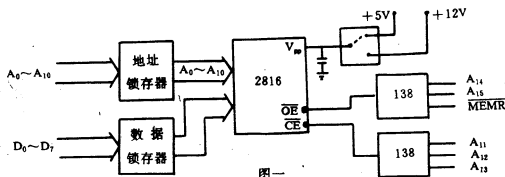
## E<sup>2</sup>PROM 在控制系统中的应用

甘肃省计算中心 崔正德 (730030)

在工业控制系统中,由于环境恶劣,电网波动、瞬间停电以及其它干扰等原因,经常造成控制程序非正常运行,使得控制系统失灵,这不仅危及仪器、设备和人身的安全,影响生产,而且控制系统的可靠性也大为降低,一般在较重要的场合和要求较高的控制系统中,多数采用多 CPU 系统,自动手动切换装置等技术手段来提高系统的冗余度,增强系统的可靠性。但是当控制系统受到干扰失灵时,控制系统中的控制参数(主要指中间参数)将因此而改变。在这种情况下,一般均需人工干预,才能使系统正常运行,这就影响了控制系统的快速反应和断点恢复能力。在这里本文介绍一种采用 E<sup>2</sup>PROM 器件随机存储实时控制参数,迅速恢复断点,提高控制系统的掉电恢复能力和控制系统可靠性、减少人工干预的方法。

我们知道,E<sup>2</sup>PROM 不仅具有 EPROM 的多次写入、擦除及掉电保持信息的功能外,而且具有对片内某一字节或一部分字节进行多次擦除、改写操作,这些操作都不需要专用的擦除或编程设备,只需在相应的管脚连接一简单的升压电路,在擦除或写入信息时,将电压升高,即可完成全片及字节的擦除、写入操作,E<sup>2</sup>PROM 的这一功能为我们随机存储系统参数,提供了有效而可靠的手段。

**一、硬件设计及功能实现** 根据控制系统的参数多少以及其它硬件要求可选取适当的 E<sup>2</sup>PROM。本文中我们选取 inter2816 2KX8bit 的 E<sup>2</sup>PROM 芯片,来实现现场信息的存储和保护。inter2816 芯片的管脚排列与 EPROM2716 芯片完全一致,只是在字节写入方式时  $\overline{CE}$  为低电平, $\overline{OE}$  为高电平,由  $V_{PP}$  端输入编程脉冲,幅值为 21V。Inter2816 与系统接线原理框图如图一。



图一

Inter2816 的擦除及写入操作约需 10ms 的时间,因此这种方法只适用于大滞后系统。其地址线  $A_0-A_{10}$  通过地址锁存器与系统地址总线  $A_0-A_{10}$  连接。数据线  $D_0-D_7$  通过双向数据锁存器与系统数据总线连接。 $\overline{CE}$  片选端和地址译码器连接, $\overline{OE}$  和  $V_{PP}$  端,则分别通过电平转换开关和 +5V 及 +12V 相连接,以满足字节擦除及写入操作的要求。

在系统中,Inter2816 的读过程为:当系统 CPU 选通 Inter2816 后,系统地址总线的有效地址被地址锁存器可靠锁存,地址译码器与 inter2816  $\overline{CE}$  端连接口译码输出“低”电平, $\overline{CE}$  有效, $\overline{OE}$  端由电平转换开关提供“低”电平有效读信号, $V_{PP}$  端输入 +5V 电压,这时 Inter2816 送出数据到双向数据锁存器,供 CPU 读取数据。

Inter2816 的写过程分两步:首先要对写入信息的单元进行字节擦除操作,这时系统除提供有效地址及有效片选信号  $\overline{CE}$  为“低”外,还必须使  $\overline{OE}$  为“高”电平, $V_{PP}$  端输入 +12V 的擦除电压,同时数据总线将 FF(H) 锁存在双向数据锁存器上,供 Inter2816 写入。其次,系统将需写入的信息送入双向数据锁存器,并提供与擦除操作相同  $\overline{CE}$ 、 $\overline{OE}$  信号以及相同的  $V_{PP}$  电压。这两次操作各延时 10ms 左右,即完成了字节的写入。可以看出单字节写入操作和单字节擦除操作过程相同,只是擦除操作过程为输入固定的 FF(H) 字节而已。

inter2816 还有另外一种擦除方式,这种方式只需在字节擦除的基础上,将  $\overline{OE}$  端电压提高到 +9V ~ +12V 之间,持续约 10ms 的时间,2K 字节将全部被擦除。

**二、软件设计** 软件设计应根据系统的需要,在任何时候及时恢复系统的正常运行,一般在掉电或 watch dog 使系统复位,程序将重新开始运行,这样在程序的适当位置调用读写子程序,将恢复断点所必须的参数写入 E<sup>2</sup>PROM,在程序开始执行时作一些简单的判断和参数恢复就可以完成系统的正常运行,以下以啤酒发酵过程温度控制为例做一简要说明,啤酒的发酵过程是一滞后极大的过程,因此有充足的时间存储中间参数,控制程序的部分框图见图二。啤酒发酵工艺曲线见图三。

## 使用 GAL 的硬加密特性保护 EPROM 中的软件

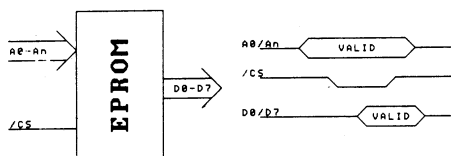
武汉大学无线电系 91 研 黄银彪 (430072)

**摘要:**本文介绍了一种使用 GAL 的硬加密特性来保护普通 EPROM 中的软件的方法,说明了其原理并分析了在以 8088/8086 为 CPU 的系统中的应用实例。

随着电子技术的发展,微处理器在现代经济领域的应用越来越广,大凡使用微处理器的系统都可分为硬件和软件两部分,对于硬件,由于器件的标准化,外观的透明性,使其结构在别人眼里一览无余。而软件则往往是产品设计者智慧精华之所在,比如具有独创性算法等,而在目前的应用中,软件的存储介质往往是可以直接读出其数据的 EPROM 芯片等,为了解决 EPROM 中软件的保护问题,我们使用了 GAL 的硬加密特性。

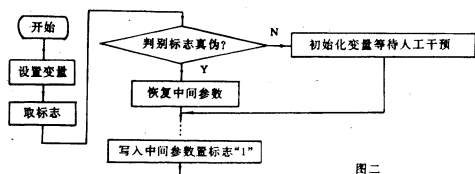
加密和解密系统都有两个类似的信息来源,1)明文或密文。2)密钥,系统可靠性的前提是所有的具体密钥不被窃密者获得,对明文的加密方法是用密钥对明文进行加密处理,得出加密信息后,再存贮或传输,密文是公开信息,但是由于窃密者不知加密所用的个体密钥及加密算法,因而,窃密者无法知道信息的真实内容。

普通的 EPROM 的应用电路及信号波形如图 1 所示,存于其中的软件极易被人读出后分析和复制。

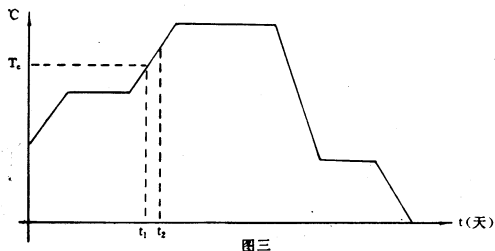


根据密钥原理,如果我们预先使用加密钥作用于要保护的软件,然后将所得到的软件密文存入 EPROM 中,就能防止他人直接得到软件代码,同时利用 GAL 的硬加密特性制成解密密钥直接做在硬件中,只要别人无法破译 GAL1 就不能获得解密密钥。这样既不影响微处理器工作,又能保护软件,根据这种思想,我们在一采用 8088 的单板机设计中采用如图 2 所示的电路。

GAL 是美国 Lattice 公司首创的据称具有不可读出其加密内容的可编程门阵列芯片,通过它可以实现多种复杂的逻辑功能。



图二



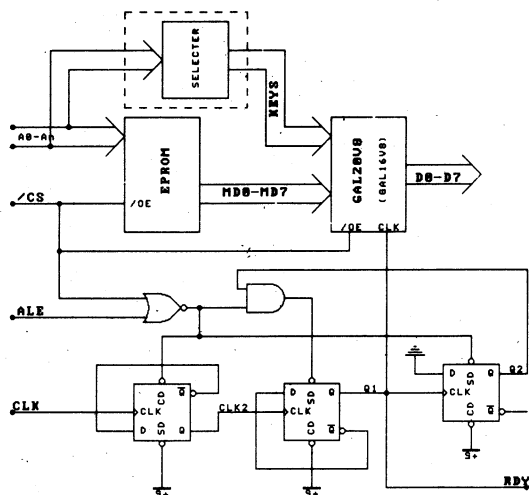
图三

醇时间,存储周期为  $T_1$ ,如图三所示,当系统在  $t_1 + \Delta t$  的时间内存储完中间参数,并置标志。段设在  $t_2$  ( $t_2 > t_1 + \Delta t$ ) 时刻系统受干扰,watch dog 使系统复位,这时程序首先取出标志,判断中间参数是否存储完毕,如果存储完毕,则取出段号,温度、发酵时间恢复程序的断点继续控制啤酒发酵过程。如果没有存储完毕,则系统关闭所有的阀门,报警并等待操作人员人工恢复断点控制发酵,一般中间参数的存储都在几十至几百毫秒之间,而存储周期  $T_1$  在几十分钟至几小时之间,因此没有存储完中间参数的情况极少发生,这样整个系统就能在受到干扰的状况下,迅速恢复断点,使系统正常运行,从而提高了系统的可靠性。

**三、结束语** 控制系统,由于和生产及精密仪器、设备等紧密相连,因此系统的可靠与否直接影响设备的安全和生产的正常运行,一个完善的控制系统不仅要精确控制仪器、设备的正常运行,而且应当能及时恢复处理受到干扰时的系统正常工作,相信随着 EPROM 的广泛采用,对提高控制系统的可靠性将起到积极的作用。

本系统采取周期性存储中间参数段号,温度、发

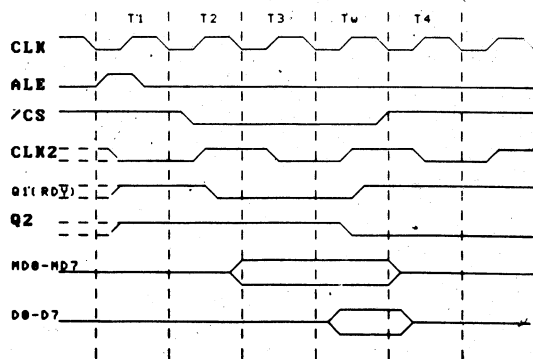




首先,当片选信号选中 EPROM,读信号有效时,EPROM 将数据送至 MD0—MD7 上,它是 GAL1 的输入,与此同时,/CS 有效释放了 D 触发器的 SET 端,此时 8088 的存储器读周期位于 T1 状态,当 T2 状态来到时,D 触发器翻转,RDY 变低,在 T3 状态 8088 采样 RDY,插入 TW 状态,同时,T3 状态使 RDY 上升,此信号接到 GAL1 的 CLK 端,使 MD0—MD7 的数据打入 GAL1 中的寄存器并同时经过解密密钥和固化在 GAL 中的解密算法作用后,送上数据总线,在 T4 状态,8088 完成对 EPROM 的访问周期。

第二个 D 触发器的作用为防止 RDY 再次翻转,产生多余动作。

Select 单元的作用在于选择不同的解密密钥。GAL 中可以存放多个密钥,EPROM 中不同地址的数据可以采用不同的解密方法,这样更加增添了破解密难度。



加入 ALE 的作用是为了确保每个 T1 周期各 D 触发器处于确定的状态,由于 8088 的每一个存储器周期总是要首先出现 ALE 信号,它将迫使两个 D 触发器置位,处于正常确定状态。根据 GAL 手册,其延迟时间为 25ns,当 CLK 频率小于 9MHz 时,可以满足 8088 要求的最小  $t_{\text{vcl}}$  (数据建立时间)延迟,如果 CLK 频率高于 9MHz,为保险起见,应采用 2 等待设计。

以上方案通过实际应用,证明切实可行,均获得了满意的效果。

## 软盘驱动器读错故障排除

巴陵石化公司 第一计算机应用研究所 宋靖涛

微机常出现从 C 盘启动正常,但在读软盘时读错的情况,这时屏幕上显示:

General failure reading driver A

Abort, Retry, Fail?

出现这种情况,说明软盘驱动器的机械部分有故障。因为软盘驱动器在机器加电自检时被检查,从硬盘能启动,说明软驱电路部分基本正常。其故障原因有以下四种:

1. 磁头太脏。这时需用清洗盘清洗磁头;
2. 磁头定位机构有故障。由于磁头小车的导轨

太脏,使其移动困难,寻道位置有偏差,导致读错,这时应用润滑油润滑小车导轨,使磁头能正确寻道;

3. 盘片夹紧机构有故障。盘片夹紧机构不能把软盘固定在主轴驱动轮上,软盘不能转动,这时会听到一种异常的吱吱声,它是由于主轴驱动轮和软盘摩擦产生的,出现这种情况,只要设法使软盘被紧压在主轴驱动轮上,能随主轴驱动轮一起转动即可;

4. 磁头方位角有偏差。需用校准盘对磁头方位角重新进行调整,直到能读出正确信息。

# 失电后保存内存数据的方法

四川省电子计算机应用研究中心 吴汉文 (610015)

**提要:**在工业自动控制系统和智能式仪器仪表的应用中,往往需要在系统失电时,将内存中的信息和数据保留下来,应用常见的数据保存方法,保存时间不长且保存的数据不完全可靠,本文提供一种实用的数据保存电路,使用 3.6V60mAH 微型镍镉电池,失电后使内存数据可保存半年时间。

在工业自动控制系统和智能式记录仪器仪表的研制设计中,如何保证在系统失电后内存数据能长时间可靠保存,这是设计者要考虑的重要内容之一。大规模静态 CMOS 存储器 6116 系列芯片,因其价廉而被普遍选用。6116 芯片本身具备数据保存特性,即失电后只要维持其片选端  $\overline{CS}$  和  $V_{CC}$  为高电平(2V 以上),则不论读允许端  $\overline{OE}$  和写允许端  $\overline{WE}$  为什么状态,其 I/O 端呈高状态,内存单元内数据保持不变,且此时芯片处于微功耗保持状态,其典型值仅为 0.1 mW。因此在电路设计时应充分利用其保持状态的微功耗特点,使有限的电池容量能得到充分利用,以延长保持时间。由于 6116 系列芯片仅有一个片选端,应设法将其与译码器输出端隔离。常见的数据保存电路,在实际中发现有两点不足:其一是失电后备用电池放电过快,仅能维持几天,比如有些待售的控制板,测量其上的备用电池,电压几乎已放到零;其二是内存中保留的数据不是绝对可靠,一些单元的内容有时会改变,笔者对这些电路进行过剖析,并针对上述不足提供一种可靠的数据保存电路,该电路使用 3.6V60mAH 微型可充电电池,能可靠保存数据达半年之久。现将电路介绍如下:

**一、延长备用电池放电时间** 常见的使用 6116 芯片保留数据的电路如图 1。(有些电路没有用三极管和稳压管,而是用锗二极管 D1 代替,如虚线所示。)在译码器内部,输出端与电源端  $V_{CC}$  之间总有静态电阻存在,由于这类电路中译码器的输出直接连在 6116 的  $\overline{CS}$  端,因此不论采用何种芯片译码,失电后  $\overline{CS}$  端的电位实际上是电池电压在 R1 与译码器静态电阻上的分压值,电池的放电回路是 R1→译码器静态电阻→电源→系统负载→地,为了使 6116 的  $\overline{CS}$  端能分到 2V 以上的高电平, R1 就不能用大电阻,一般整个放电回路总电阻只有几十 K,因而失电后备用电池维持不了几天。

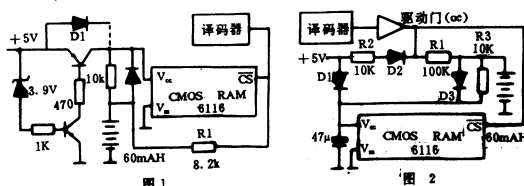


图 1

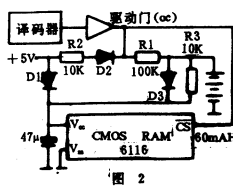


图 2

如果在译码器与 6116 的  $\overline{CS}$  端之间用 OC 同相驱动门(7407)隔离,如图二所示,则上述问题即得以解决。正常工作时, +5V 电源经 R2 与 D2 向驱动门提供灌电流,并通过 D1、R3 对备用电池充电。失电后,由于驱动门内部输出端与电源  $V_{CC}$  之间是开路的,而 D1、D2 的反向电阻又很大,备用电池除 6116 芯片外没有其它放电回路,不存在上述分压问题,备用电池的全部能量都用来保存数据而没有附加损耗,因而使 6116 的数据保持时间大大延长,实际应用证实,失电半年后,存储单元内的数据仍没有丢失。6116 的工作电压下限为 4.5V,故 D1 应选用在 50mA 电流时正向压降小于 0.3V 的锗二极管。

**二、失电时自动超前复位** 经过反复实验与分析,发现内存中某些单元数据丢失是由于 CPU 处于临界工作电压下非正常工作所造成的。系统失电后,由于电源滤波电容的作用,电源电压呈指数规律下跌,当下跌到 CPU 的临界工作电压时, CPU 的输出控制信号容易发生混乱,可能选中 6116,而此时地址线 and 数据线上的内容是随机的,有可能写一些混乱数据到存储器的某些单元中并保存下来。这个分析容易通过试验来证实,先按住复位键不放,再关断系统电源,即是使 CPU 在正常工作电压下复位,并且在整个电压下跌过程中一直处于复位状态,这种情况下数据保存的成功率为百分之百,若不复位

## IBM-PC/XT J8 插槽上 I/O 地址的“扩展”

中国保险管理干部学院计算机系  
湘潭师院科研处计算机室

赖红威 (410114)  
赖君利

一、问题的提出:现在利用 IBM-PC/XT 进行实时控制或过程控制的地方越来越多,而计算机并行处理,并行输入输出技术以其速度快又被广泛应用,IBM-PC/XT 计算机共有 64K 的并行 I/O 地址空间,系统内部及外设占去 1K 空间(0000-03FFH),其余提供给用户使用。用户在设计自己的控制系统中,往往与 CPU 之间的联络采用应答方式(如中断的扩展等),而 IBM-PC/XT 计算机只有 J8 插槽上具有应答功能,当在 J8 插槽上利用 I/O 口进行数据采集时,我们会发现可用的 I/O 口地址并非连续的,也就是说并不是所有提供给用户的 I/O 口都能够使用。

二、纠其原因与解决方法:从图一中可以看出, J8 插槽数据(XD0-XD7)与 CPU 数据(D0-D7)之间的传送是受数据收发器 U15 控制的,其中, U15 的

第一脚是方向控制端(DIR),它决定着数据是由 XD0-XD7 向 D0-D7 传送(DIR=0)还是反之(DIR=1)。而 U15-DIR 端的变化又是由  $\overline{INTA}$ 、 $\overline{CARDSLCTD}$ 、 $\overline{ROMADDRSEL}$ 、 $\overline{XIOR}$ 、以及 XA8 和 XA9 所决定。但是,从图二中可以看出,在 I/O 的读写周期中, U15-DIR 端的变化只由  $\overline{XIOR}$ 、XA8 和 XA9 来决定,而这三根线分别接至图一中 U23; A (LS27) 的 1、13 和 2 脚。U23; A 是一个或非门,其逻辑关系是:

$$Y = \overline{A+B+C} = \overline{\overline{XIOR} + XA8 + XA9}$$

U23; A 的输出接至 U23; B 的一个输入端,所以在 I/O 的读写周期中:

$$U15-DIR = \overline{\overline{XIOR} + XA8 + XA9} = \overline{\overline{XIOR}} + \overline{XA8} + \overline{XA9} \dots (1)$$

就直接关断电源,则有时个别单元的数据会丢失,因此,电路设计时应使系统具有失电自动超前复位功能,即是系统失电后,在 CPU 的工作电压未降到临界值时就让 CPU 自动复位并维持到电压为零。由于复位过程中 CPU 的控制线、地址线和数据线一直处于高阻状态,不会进行写存储器操作,存储单元的内容就不会丢失。

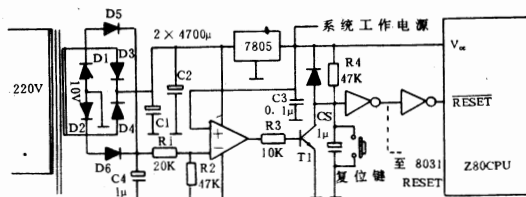
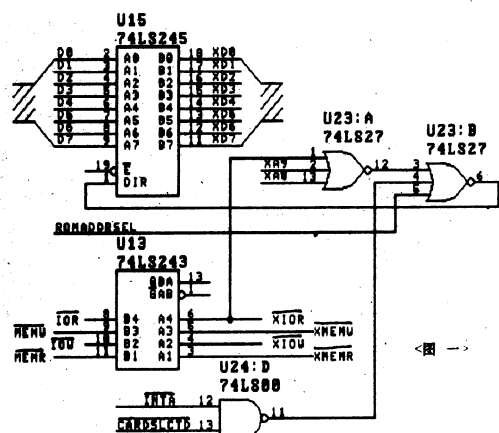


图 3

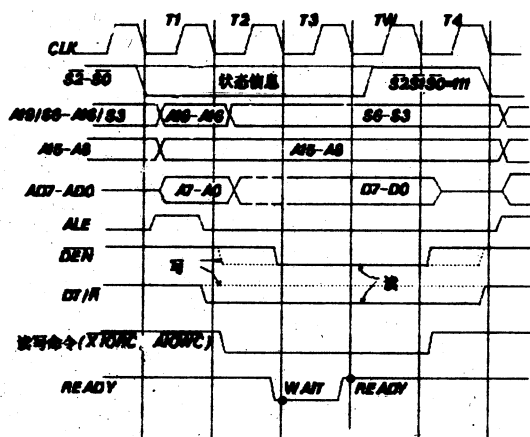
图 3 是在原系统电源基础上增加一条掉电检测支路,就构成了失电自动复位电路。图中 D1、D2、D3、D4、C1、C2 构成的全桥整流滤波电路,经过 7805 稳压后作为系统的工作电压,这部分电路是系统本身就具备的; D5、D6、C4、R1、R2、T1 和比较器是附加的掉电检测电路,交流电源经 D1、D2、D5、D6、C4 全

波整流滤波,再经 R1、R2 分压后,作为掉电信号送到比较器(LM324)的反相端,正常工作时比较器由于反相端电压谷点高于 5V 而输出低电平, T<sub>1</sub> 的集、射极之间开路。失电后由于掉电检测支路的滤波时间常数小,电压下跌速率比工作电压下跌快得多,当其下跌到低于 5V 时,比较器输出高电平, T<sub>1</sub> 饱和导通,其集电极为低电平,经过两级倒相器整形后送至 Z80CPU 的复位端,使 CPU 在正常工作电压下自动复位。由于比较器的同相端是接到工作电源上的,其电压下跌缓慢得多,因此自动复位过程能维持到工作电压为零。复电时,也有一个工作电压由低到高越过 CPU 临界工作电压的过程,但由于比较器反相端电压比同相端上升快,其输出为低电平, T<sub>1</sub> 集、射极之间开路,此时 CPU 的复位时间就完全取决于 R4 和 C5 的乘积。因复位时间常数比电源滤波时间常数大,待到复位结束时 CPU 的工作电压已经正常了,因此复电时不会发生存储器读写混乱现象。R2 的阻值需要在调试中确定,用双踪示波器分别接在比较器的两个输入端,用调压器将交流 220V 电压降至系统所允许的最低电压(约 150V),调整 R2 阻值,使反相端上的纹波电压谷点略高于同相端电压, R2 的阻值即被确定。

对于 8031CPU,其复位是高电平有效,复位信号应从第一级倒相器输出端引出,如图 3 中虚线所示。



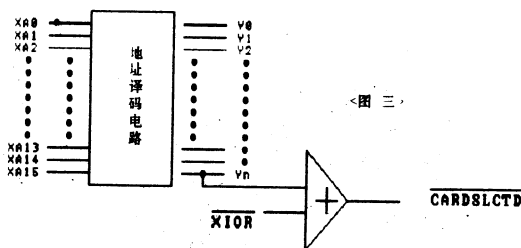
图一



图二

当 CPU 对 I/O 写操作时,  $\overline{XIOR}$  为“1”, 从逻辑表达式(1)中可以得出, 不论  $XA8$  和  $XA9$  为何状态, 一定是  $U15-DIR=“1”$ , 即  $D0-D7$  向  $XD0-XA7$  传送, 实现了 CPU 向 J8 插槽写操作。

当 CPU 对 I/O 读操作时,  $\overline{XIOR}$  为“0”, 从逻辑表达式(1)中可以得出,  $U15-DIR$  端不一定为“0”。而只有该端为“0”时, 即  $XD0-XD7$  向  $D0-D7$  传送, CPU 才能正确读入 J8 插槽上的数据, 因此, 只能使用  $XA8$  与  $XA9$  都为“0”的地址单元, 否则, CPU 在对 I/O 读操作时, 其数据通路会产生阻塞, 为了不产生阻塞现象, 我们将  $XA8, XA9$  与  $U23$  的 13、2 脚之间的联线断开, 并将  $U23$  的 13、2 脚分别接地, 使其  $U15-DIR$  端只受  $\overline{XIOR}$  的控制。这样, 不论  $XA8, XA9$  为何值, CPU 在读操作时都能将 J8 插槽上的数据正确读入。也就是相当于扩展了 I/O 的地址, 当然, 也可以象图三那样, 将地址译码后的输出端与  $\overline{XIOR}$  相或接至 J8 插槽上的  $CARDSLCTD$  端, 但这将给硬件带来很低的性价比, 特别是在多路采集系统中。IBM-PC/XT 中的  $XA8, XA9$  之所以这样接, 是限制用户使用基本外设的地址(200H-3FFH)。



图三

综上所述, 其意义不仅仅扩展了 I/O 的地址, 更主要的是避免了在您的开发的控制系统中, 无意中使用了  $XA8, XA9$  不同时为“0”的地址时, 给调试带来了烦恼。

#### 参考文献

- (1) 朱传乃等主编, 《微型计算机系统原理分析与维修》, 1989. 8, 科学出版社
- (2) 马道均等编, 《微型计算机维修指南》, 1990. 10, 中国广播电视出版社。
- (3) 李文中等编, 《微型机的维护与诊断技术》, 1988. 4, 清华大学出版社

### 利用冷却法修复微机一例

中国农业银行潼南县支行 张 智 (632660)

**故障现象:** 一台浪潮微机, 开机时工作正常, 10~20 分钟后显示器发生紊乱, 使其无法操作, 且很有规律性。

**故障检查与分析:** 出现以上这种具有规律性的故障往往是由于机内某一元件的热稳定性变差所致。其中尤以晶体管、瓷片电容等发生这种故障较为常见。利用交换法分别将显示器及彩色适配卡与

同型号的机器交换后, 确诊故障出在彩色适配卡上。

**故障排除:** 开机待故障出现后打开主机盖, 用酒精棉球放到各晶体管上, 显示器上没有变化, 然后逐个检查瓷片电容, 当酒精棉球放到  $C22$  上时, 故障立即消失, 证明该电容的热稳定性变差, 是引起故障的根源。更换该电容, 故障排除。

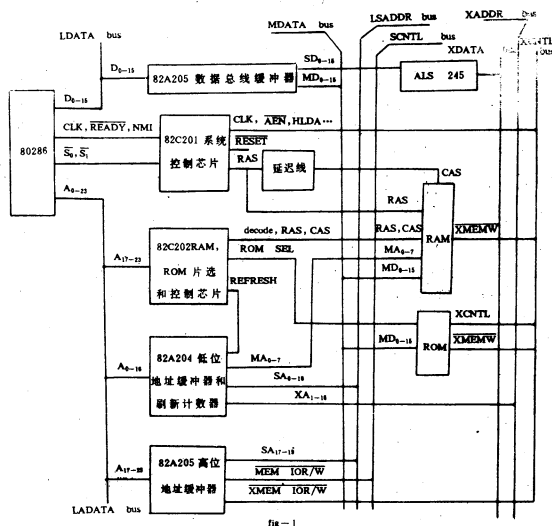
## 采用门阵芯片的 286 机的主板诊断

成都军区自动化工作站硬件室 陈 亮 (610011)

**摘要:**采用 80286 和 CHIPS 芯片的 286 微机虽然仍然采用总线结构,但在体系结构上与 PC/XT 机有不少的差别,由于大量采用 VLSI 芯片,主板被高度集成,在外观布线上甚至与同一级别的 PC/AT, PCXT/286 有很大的区别,使维修人员在维修采用 CHIPS 门阵芯片的兼容机主板时感到非常困难,特别是主机死机时,更感一筹莫展,本文讨论了这类采用 CHIPS 门阵芯片的 286 兼容机的维修方法。

一、采用门阵芯片的 286 微机体系结构 目前市售的 286 微机,除早期的 PC/AT,PCXT/286 外,绝大部分都采用门阵芯片,如东海 0530,长城 0530 都采用 CHIPS 芯片,用户拥有量很大的 AST286 微机采用与 CS8200 系列 CHIPS 芯片兼容的 LOGIC-STAR 芯片,这些采用门阵芯片的微机的体系结构如图 fig-1 所示:

fig-1 采用门阵的微机主板体系结构图



其中 82C201 和 82C202 都是控制芯片,这两个芯片实际上完成 AT 机中时钟芯片 82284 和总线控制芯片 82288 的功能,完成系统控制和时钟产生功能。82A203、82A204、82A205 中包含了许多与 IBM PC/AT 相兼容的设计中所要求的总线驱动器和缓冲器,它们都是在 82C201、82C202 的控制下工作的,所以检查门阵列类机器的底板要从分析 CPU 和这

两个芯片(有些机器只采用 82C201 一片芯片)的输入是否满足条件,输出信号是否正常着手。

## 二、故障诊断方法

### 2.1 检查原理:当主板死机时,屏幕黑屏,无任何出错提示,无法借助于开机自检程序 POST 和其他软件方法检查,这时只能打开机箱用示波器看波形。如果主机死机,涉及的范围比 PC/XT 多一些,除了 CPU、基本 32KROM、8254 计数器、8237DMA、64K 基本 RAM 外,CMOS RAM 电路出错,8042 键盘控制器出错,页面寄存器出错也可能引起黑屏。这时,应首先从 80286 查起,只要从 CPU 出来的 $\overline{\text{MEMR}}$ , $\overline{\text{S0}}$ , $\overline{\text{S1}}$ 信号正常,则说明 CPU (80286)正常,然后可查看机器死机时的状态,机器的状态由下列几个信号决定,它们组合在一起决定 CPU 当前是处于什么工作状态,再检查对应工作状态下涉及的芯片是否确实是和 CPU 联系在该状态。这样逐渐深入,一般能发现故障点的位置,参考下表:

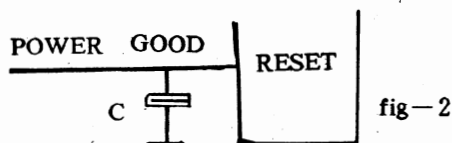
| $\overline{\text{COD}}/\overline{\text{INTA}}$ | $\overline{\text{M}}/\overline{\text{IO}}$ | $\overline{\text{SI}}$ | $\overline{\text{S0}}$ | 总线周期   |
|------------------------------------------------|--------------------------------------------|------------------------|------------------------|--------|
| 0                                              | 0                                          | 0                      | 0                      | 中断响应周期 |
| 0                                              | 0                                          | 0                      | 1                      | 未用     |
| 0                                              | 0                                          | 1                      | 0                      | 未用     |
| 0                                              | 0                                          | 1                      | 1                      | 空闲状态   |
| 0                                              | 1                                          | 0                      | 0                      | 暂停或停机  |
| 0                                              | 1                                          | 0                      | 1                      | 读存储器   |
| 0                                              | 1                                          | 1                      | 0                      | 写存储器   |
| 0                                              | 1                                          | 1                      | 1                      | 空闲状态   |
| 1                                              | 0                                          | 0                      | 0                      | 未用     |
| 1                                              | 0                                          | 0                      | 1                      | I/O 读  |
| 1                                              | 0                                          | 1                      | 0                      | I/O 写  |
| 1                                              | 0                                          | 1                      | 1                      | 空闲状态   |
| 1                                              | 1                                          | 0                      | 0                      | 未用     |
| 1                                              | 1                                          | 0                      | 1                      | 取指令代码  |
| 1                                              | 1                                          | 1                      | 0                      | 未用     |
| 1                                              | 1                                          | 1                      | 1                      | 空闲     |

这里所谓空闲状态是指 CPU 不需与外界交换信息,例如 CPU 正在进行数据处理,不需与内外存储器或外设打交道。

## 2.2.286 系统板常见故障分析:

(一)复位不可靠,有时多次复位才能启动,启动后一切正常。

(1)该现象与复位信号 RESET 和电源 POWER GOOD 信号有关,检查 RESET 信号,有跳变,但复位信号太窄,在 PG 信号输入处加了一个电容拉宽 PG 信号,如图 fig-2 则输出的 RESET 信号相应能可靠复位,故障消失。



(2)有些芯片的初态不好,在使用了三四年以上的机器上常出该毛病,而且多是由于 ALS244, ALS245, ALS373, ALS573 等总线驱动片的片选信号不正确所致。(3)有些 IC 片子上的累积电荷未能释放,造成电平混乱,使机器死机,对较脏的机器常出现该故障。

### (二)随机性死机

引起不规则死机的原因有很多,可能的原因如下:

1.时序配合不好,产生时序竞争,这可用逻辑分析仪观察芯片有无毛刺判断。

2.RAM 读时延时器不稳定。

3.系统总线输出特性不好,总线驱动芯片的带负载能力差,如果去掉一些负载卡,故障消失,就可肯定为是该类故障。

4.接触性故障,这类故障重插一次芯片即可排除。

5.8042 时钟或键盘接口部分出错。

时序配合不好在 286 兼容机中很常见,一是由于芯片速度不够,在进行到一定的总线周期时产生时序竞争,引起时序混乱,使系统停机。在采用逻辑分析仪检查时发现波形如下 fig-3,有毛刺。这种故障经常采用背芯片的方法排除。对 373,244,245,240,74s08,74f32 等通用小芯片,常用该方法排除故障。

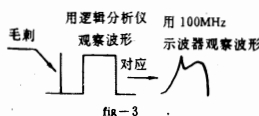


fig-3

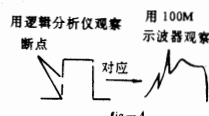


fig-4

另一个容易出现的问题是各不同厂家的芯片间匹配不好,用逻辑分析仪观察时出现 fig-4 波形,中间有断点。这类故障经常采用加旁路电容法去掉毛刺。

在一些劣质兼容机中,由于选用芯片不过关,时钟又选得很快,上述问题都容易发生。而一旦发生,故障点很隐蔽,检查很费周折。

对随机性死机,首先要排除接触性故障,它们容易发生在 CPU, PAL, ROM, 82C201, 82A203 等可插拔芯片上,然后利用各种速度切换初步检查是否有明显的时序竞争问题。第三步,检查各种时钟信号,例如键盘控制器 8042 时钟,SYSCLOCK 时钟。然后第四步,需用分析仪检查不同芯片之间转换,特别是在 F 系列——ALS 系列,ALS——F 系列转换时是否有干扰毛刺,最后应考虑 A/D BUS 驱动能力,芯片速度是否够等问题。

### (三)、系统丢设置:

如果不是软件操作错,该故障一般在 RT/CMOS 电路模块。(1)首先检查电池是否用完:在关机后,检查 14069 之电源和地之电压应在 4V 以上,否则电池应更换。(2)检查 PG 信号,在开机时,PG 作为 CMOS 存储器的允许端不得低于 4V,负脉冲宽度应  $>1ns$ 。(3)第三步应该检查 CMOS RAM 的供电电路的两个晶体管和电容是否有损坏。当电池用不了多久就漏电流汤时常是由于该部分电路短路所致。(4)开机复位时有脉冲干扰,冲掉了 RT CMOS/RAM 中的设置,如果开机有时能启动,有时死机,但开机后丢设置,肯定是该故障。

### (四)系统板故障影响软盘引导或读写:

这时应首先检查 8237 是否正常,或 DMA 地址生成信号是否正常,对采用门阵的芯片,控制该部分的是 82C201,需检查它的 HRQ1, HLDA1 以及 DACK2, DREQ2 信号。

### (五)因系统板而影响显示:

首先应检查 I/O CHRDY 输入到板内是否正常,然后检查,系统槽之第 A8-15(影响字形色彩)A16-19(影响字位置)之总线驱动块的驱动能力是否不足。

### (六)因主板影响打印和串行通信:

一般的故障都在 82C11 及外围芯片。

#### (七)死机故障:

对一开机就死机的故障,一般只能从检查总线槽的情况来进行进一步的判断,当然根据实际故障率的分布也可先查主板延迟线和 8042。若在开机后,A/D 总线上仅有少数几个脉冲,说明故障靠前,否则故障靠后,参考对应的 286POST 自检顺序,可估计死机位置。在停机后,数据线一般恒为定电平,而地址线有时为恒定电平,有时为连续脉冲。其中各芯片的检查方法如下:

##### (1)CPU 工作状态检查:

若输入 CPU 的 RESET、CLK、READY 信号正常,则说明输入信号正常。若进一步,A0-23,SO,SI,M/IO 有脉冲,则说明输出正常,即 CPU 可以认为是正常的。

##### (2)82C201 的检查:

1:M/IO,SI,SO,COD/INTA 的状态组合至少出现过一次 1011(用于读 ROM 中的内容)。

2:几个时钟信息 SYCLK,PROCLK,OSC,OSC/12,DMACK 均为脉冲信号。

3:DMA 信号必须处于无效状态,即:HRQ1 为低,HLDA1 为低,AEN1、AEN2 为高,DMAEN 为高。

4:内存刷新信号应正常,即:REFREQ 为低电平或为脉冲。

REFREFEN 为高电平或为脉冲。REFDET 为低电平或为脉冲。

5:系统奇偶校验不出错,即:IOCH RDY,IOCHK,PAR 信号为低,失效。

6:系统控制信号有脉冲,即:ALE,CRAS,

XBHE,AD DSDEN0,DSDEN1 均有脉冲信号。

7:系统复位信号(RESET3,RESET4)有正脉冲信号。

经过上述检查,即可判断 82C201 正常与否,但实际上,经过此番检查,已把主板的所有部分都涉及到了。有些机器如“东海 0530”只有该片子作为控制片,另有一些机器,如“AST286,浪潮 0530”还有另一个控制芯片 82C202,它的检查方法如下:

##### (3)82A202 的检查:

1:ALE 应至少有一脉冲。

2:CLK0,CLK1 在速度开关打到低速下应是脉冲。

3:SC0(用于高低速选择)为恒高电平,不允许为连续脉冲。

4:HSBM1(控制半读 I/O 操作)不允许为脉冲。

5:FSDMA(半读 DMA 控制)不允许为连续脉冲。

6:HLDA 应为低或为脉冲。

利用上述要求,可查 82A202 是否正常。

(4)82A203,82A204 的检查,它们满足以下条件时为正常。

1:PORTBRD,PORBTBWR 为高或低电平。

2:MRDC,IORC,IOWC,MWTC 为脉冲信号。

3:ADDSEL 行选信号为脉冲,且波形应规则。

4:CPU HLDA 不能恒为高电平,否则 CPU 一直让出总线控制权。

经过上述检查,基本上对底板上所有重要信号均作了检查,一般能逐步缩小故障范围,找到主板故障点,使机器恢复正常。

## 解决实时时钟丢失故障一例

解放军军事经济学院计算中心 刘亮生 (430035)

一台 IBM PC/286 主机开机自检时显示 CMOS RAM 诊断出错,检查时钟和硬件数据信息已不存在,重新设置参数表后,能引导系统正常工作。

该机采用的 CMOS RAM 芯片是 82C206,为主机系统提供日历时钟计数功能和系统硬件的配置情况。在开机时,BIOS 中的自检程序读取这些单元就可以知道系统的配置情况,如内存的大小,软、硬盘的容量、类型,显示适配器的工作方式等,并根据这些参数完成对各模块的初始化。由于通过 CMOS RAM 代替开关设置,使设置的选择和参数的设定有了更大的灵活性。该芯片在关机情况下由后备电池支持以保留上述常用的信息参数。

该部分故障主要从电池供电情况,CMOS RAM 输入控制信号和数据缓冲驱动器的检查下手。从现象上看,开机重新设置参数表后,主机工作正常,说明 CMOS 芯片及读出电路正常,重点检测电池供电电路。经检查电池接触良好,3.6V 正常。在电池供电情况下,测量 82C206 芯片的 75、32 脚电压只有 2V。这是因工作电压不足造成存储参数丢失,引起主机不能自举。由于开机+5V 电压正常,电容 C33 不会有故障,只能是 R1 阻值漂移增大,致使电池供电不足,拆下 R1 测其阻值为 360Ω 而色彩标称值是 150Ω。将其更换后,82C206 的 75、32 脚电压上升到 3V,主机自举正常,故障排除。



# CE-150 绘图打印机电力检测电路分析及常见故障处理

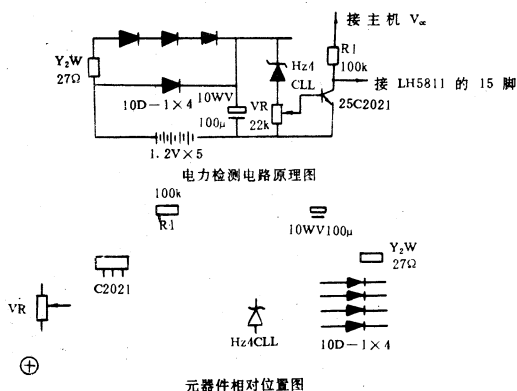
大庆石油管理局采油四厂计算机站 潘国军 (163511)

**摘要:** CE-150 是 PC-1500 系列袖珍计算机所配四色绘图打印机及 PC-1500 连接器部件, 使用效率很高。在维修中, 经常遇到主机显示屏出现“CHECK; 6”或“ERROR 80”等错误信息提示, 即使后备电池组电力充足, 打印机也不能正常工作, 本文就此常见问题谈一谈电力检测电路原理及故障的处理。

**故障现象:** 后备电池组电力充足, 开机绘图打印机也不能正常工作, 同时主机显示屏显示“CHECK; 6”错误信息。

**分析与处理:** 从屏幕显示的信息可知: 电池组电力不足, 用 EA-150 电源适配器给后备电池组充电, 开机试之, 打印机还是不能正常工作, 即使打印机能够完成初始化工作, 当打印机处于打印工作状态时, 显示屏显示“ERROR 80”错误信息, 提示用户电池组电力不足, 不允许打印机工作, 用万用表监测电池组在开关机时电压变化情况, 监测结果电压均为 6.5 伏, 电压没有变化, 说明问题不在电池组上, 而在电力检测电路上。

电力检测电路是由稳压管 HZ4CLL(4V)、电位器 VR(22K $\Omega$ )、电阻 R1(100K $\Omega$ )及晶体管 C2021组成的, 详见附图。



当电池组电压输出高于 6V 时, 稳压二极管导通, 电位器 VR 上获得 2.0V 以上的压降, 足使三极管 25C2021 饱和导通, 其集电极对地电压接近 0V, 即 LH5811 的 15 脚检测的信号为低电平, 系统检测

电池组电力正常, 打印机可以工作。

当电池组电压输出低于 6V 时, 稳压二极管虽然可以导通, 但电位器 VR 上获得的压降很低, 使三极管 25C2021 处于放大状态或截止状态, 三极管的集电极对地电压上升, 当其电压高于 1.4V 时, LH5811 的 15 脚检测的信号为高电平, 系统检测电池组电力不足, 封锁打印机控制端口, 不允许打印机工作。

根据上述原理分析可知: 当电池组电压充足, 而电力检测电路有故障, 系统将认为供电电力不足, 打印机不能工作。引起电力检测电路故障的原因有: 1、稳压二极管损坏。2、晶体管 25C2021 损坏。3、电位器 VR 接触不良。4、LH5811 的 15 脚内部损坏。

我们知道: 袖珍计算机系统设有电力检测电路, 其作用是避免打印机在工作中电池组过压放电, 当电力不足时, 提醒应用人员及时充电, 保护电池组, 延长使用寿命, 当电力检测电路有任一故障发生, 打印机不能工作。从四种因素发生的概率来看, LH5811 的 15 脚内部损坏故障率最小, 因为 LH5811 是一个多功能可编程的 CMOS 静态输入/输出器件, 属于大规模集成电路芯片, 精度很高, 功耗比较小, 所以, 一般情况下不会损坏。当遇到电力检测电路有故障时, 首先判断此芯片是否有故障, 方法为: 将 15 脚用导线接地即接电源负极, 开机后, 打印机能够初始化, 证明此芯片检测功能正常, 否则此芯片检测功能异常内部损坏, 更换芯片或用扩充接口的方法来修复故障机器, 致于其它元件用万用表测量, 就可以判断它们是否损坏, 这里不详细介绍, 本文介绍在维修中经常见到的故障因素——电位器 VR 接触不良, 由于该系统机器产品出现比较早, 以及应用环境等因素, 电位器 VR 表面灰尘较多, 受潮氧化等都会导致电位器的滑动触点与电阻膜接触不良, 处理过程, 用酒精棉球将电位器表面灰尘清除, 并左右旋转一下电位器的滑动触点, 调整电位器滑动触点的位置, 使触点地对地电压在 0.65V 左右即可, 打印机可以恢复正常工作, 不妨对类似此故障的机器试一试。

## SENKEN 1KVA UPS 的维修

中国工商银行佛山分行科技科 郭毅 (528000)

**故障现象** SENKEN 1KVA(型号 FBK-S102) 在线式 UPS, 在使用过程中, 突然出现报警声, 显示故障状态的 ALARM 红灯发亮, UPS 内散热排风扇停转, UPS 无电压输出, 连接在其输出端的设备掉电。

**维修过程** 断开连接在 UPS 输出端上的设备, 重新开启 UPS, 不成功, 故障现象与上述一样。再断开 UPS 的市电输入端, 用 UPS 内的电池组启动 UPS, 仍不成功, 因此确认是该 UPS 有故障, 并估计故障点在逆变器或在逆变器之后的部分。

拆开 UPS 的机壳, 断开电池组, 用万用表电阻档测量输入供电回路的 20A 速熔保险管  $F_1$ , 万用表针指到尽头, 改用直流电压档测量保险管  $F_1$  两端, 约有 100V 直流电压, 证明该保险管已熔断, 电容器贮存的电压跨于  $F_1$  两端。这时应注意使电容放电至安全电压下, 避免造成触电。

由于怀疑故障点在逆变器, 因此拆下 DK14116 板, 找出逆变器部件, 用万用表电阻档逐个测量分别焊接在  $FET_1 \sim FET_4$  位置上的逆变功率 MOS 管 2SK623, 若发现三个管脚与脚之间均导通或阻值很小, 则可认为该功率 MOS 管已损坏。SENKEN 1KVA UPS 的逆变器是接成全桥式电路, 每一桥臂 ( $FET_1$ 、 $FET_2$ 、 $FET_3$  和  $FET_4$ ) 由三个功率 MOS 管 2SK623 并联组成, 一般很少有整条桥臂上的三个功率 MOS 管都损坏, 而可能是一个或二个损坏。同时, 由于桥臂  $FET_1$ 、 $FET_4$  与  $FET_2$ 、 $FET_3$  是轮流导通和截止的, 因此若  $FET_1$  上的功率 MOS 管损坏, 则  $FET_4$  上的 MOS 管多数也会损坏,  $FET_2$ 、 $FET_3$  的情形也一

样。

找出故障管后, 换上好的 2SK623 和保险管  $F_1$  后, UPS 一般可恢复正常。由于 2SK623 难以找到, 可以用 2SK557 代替, 我单位用该方法维修的 20 多台该型号的 UPS, 一直运行良好。

有时在更换元件后开机仍不能正常工作, 观察 UPS 顶部的控制板 DK14167, 若显示故障状态的红灯 ( $LD_2$ ) 发亮, 则表明控制部分还有故障。此时测量控制板上测试点 TP12 的电压, 正常时 TP12 对 TP1 约为 +12V, 若测得远小于此值, 则可能是功率 MOS 管的驱动电路有故障。在控制板上,  $IC_{18}$  (TD62004P) 是一种多路晶体管驱动器, 是用来驱动功率 MOS 管的。 $IC_{18}$  内分别由 1 脚与 16 脚、2 脚与 15 脚……7 脚与 10 脚组成 7 个独立的驱动器, 8 脚为接地端。在判断故障时, 若 10 脚至 16 脚中测得某一脚已对地短路或电阻很小, 则一般认为该脚对应的内部器件已损坏, 而原电路中  $IC_{18}$  只利用了四个驱动器, 剩余 1~16、2~15 和 3~14 脚三个驱动器无使用, 所以当某个驱动器损坏时, 可用剩余的其中一个驱动器替代, 这样在不换  $IC_{18}$  的情况下有三次维修机会。完成上述维修后, UPS 应可恢复正常。

**故障分析** SENKEN 1KVA 型号为 FBK-S102 的在线式 UPS, 是没有静态旁路系统的, 其输出在什么时候均由逆变器提供, 因此, 当输出端的负载启动电流过大时, 对逆变器容易造成较大的冲击, 致使功率 MOS 管被击穿, 同时使速熔保险管  $F_1$  熔断。在功率 MOS 管被击穿时, 通过驱动变压器产生的高压有时会使用 MOS 管的驱动集成电路  $IC_{18}$  损坏。

## 高分辨彩色显示器故障维修一例

柳州铁路材料总厂 单昶贤 (545005)

**故障现象:** 一台 CTX-7 型高分辨彩色显示器屏幕呈红色, 无字符显示。

**分析与维修:** 从故障现象判断为图象缺色造成, 打开机壳查看, 显象管第 6、8、11 脚分别为 G、B、R 枪阴极。拔下管座开机, 用万用电表分别测量 6、8、11 脚的对地电位, 8、11 脚的电位正确均有 160V 左右, 而 6 脚只有 10V。关机用万用电表分别测量 6、8、11 脚的对地电阻, 8、11 脚的阻值正确均为无穷大, 而 6 脚有 3K 欧姆的电阻, 进一步检查 R621、R622。

D621、C622 这四个元件是好的, 仔细观察发现 D621 正端的焊点靠近 BG621-B 极的印刷电路连线, 用万用电表测量 D621 正端与 BG621-B 极之间的电阻有错为 0。因为 D621 正端与 BG621-B 极短路, 使得 180V 电压经 R622 后, 通过 BG621-BE 结骤然下降, 造成 G 枪偏置电位不正确, 以致图象缺绿色, 这个短路现象是由于电路板长期工作发烫, 使绝缘漆碳化造成的, 用小刀轻轻刮掉碳化层, 插上管座开机故障消除, 显示器恢复正常。

# 计算机图形标准展望

武汉海军工程学院计算机工程系 汪厚祥 夏 静 (430033)

**摘要:**本文介绍了 ISO 组织开发的计算机图形标准,分析和讨论了各图形标准间的相互联系和结构。最后提出图形标准在计算机图形应用中的重要性。

**关键词:**连接,元文件,结构,段,工作站,元素。

一、计算机图形接口界面概念 图 1 表示了一个标准的图形系统的应用程序模型。两个主要的接口分别是:应用程序接口——API 和虚拟设备接口——VDI。

应用程序接口 API 是一个标准的外部过程和子程序包,程序员利用此工具即可以开发适合自己应用需求的图形输入和图形输出,而不用涉及计算机系统实际物理配置。ISO 提供了三个 API 标准描述: PHIGS (Programmer's Hierarchical Interactive Graphics System), GKS (Graphical Kernel System) 和 GKS-3D。

计算机图形应用不可能都用同一个语言编写,因而对每个图形标准说明各种语言的图形接口是理所当然的,亦即语言连接——binding。ISO 公布了许多 binding 标准,诸如 Pascal binding GKS、PHIGS、Ada binding GKS、PHIGS 等。

Graphics Metafile) 和 CGM-3D,一个是图形描述数据标准化即 PHIGS 的档案文件格式 (Archive File Format)。

在不久的将来,操作员接口和硬件接口的标准化将成为研究的目标,图形数据交换的标准将演变为信息交换的标准化,亦即集正文、图象、声音和图形为一体的信息数据交换标准。

二、图形标准发展过程 1974 年,在国际信息处理联合会上法国率先提出了开发图形标准。同年,在美国国家标准局举行的 ACM SIGGRAPH 工作会上提出了制订有关图形标准的基本规则。

1976 年,英国计算机协会标准委员会向 ISO 提交了第一个图形软件包。

1977 年,ACM GSPC 提出了 Core (Core Graphics System) 图形系统。

1978 年,ISO 正式成立图形工作组,命名为 ISO/TC97/SC5/WG2。

1979 年,ANSI 成立一个图形委员会命名为 ANSI X3H3。同年,德国 DIN 提出了 GKS 图形系统,美国 ANSI X3H3 提出了元文件概念。

1980 年,ANSI X3H3 着手开发 VDM (Virtual Device Metafile),同时提出了虚拟设备接口 VDI 概念。

1984 年,ISO 将 VDM 更名为 CGM,并公布了 DP8632。同年建议 ANSI X3H3 更名 VDI 为 CGI。

1985 年,ISO 成立一个新的学术委员会 SC21,图形工作组划归 ISO/TC97/SC21/WG2。这一年,图形标准取得了突破性的成果,GKS 继 1983 年成为国际上第一个国家标准 ANS X3.124 后,又成为第一个国际图形标准 IS7942。同年,ISO 着手进行 3D 图形标准开发将 GKS 加以扩充修改,公布了 GKS-3D DP8805,委托 ANSI X3H3 开发 PHIGS。

1986 年,ISO 公布了 GKS-3D DIS 8805 和 PHIGS DP 9592。同年,ANSI 公布 CGM 为国家标准,ISO 公布 CGI DP 9636。

1987 年,CGM 成为第二个图形国际标准。

1988 年,GKS-3D 成为第三个图形国际标准,同年公布了 ISO DIS 9592。PHIGS 被 ANSI 接纳

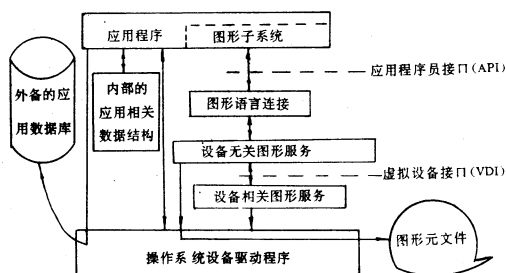


图 1 图形系统应用模型

一个公用的设备接口标准是十分必要的,它使得图形设备和软件就象接插件似的方便使用。在 VDI 界面,ISO 开了 CGI 和 CGI-3D。

在不同的系统里,图形数据文件的交换是十分必要的。因此图形数据文件的标准化是必需的。在这方面,有两条途径:一个是图形原语文件标准化,称之为元文件(metafile),ISO 开发了 CGM (Computer

为国家图形标准 ANSI X3.144。

1989年,PHIGS成为国际图形标准。

1991年,ISO公布了PHIGS扩充产品 PHIGS PLUS ISO/IEC DIS 9592-4。

基于 CGM 和 CGI 的 3D 图形标准开发也在进行,德国 DIN 在这方面作了大量工作。

### 三、图形标准概述

下面对现行的国际图形标准作一个简要性的介绍。

1. GKS GKS 是应用程序员接口标准,只支持 2D 图形操作,也可以说 GKS 是一个设备无关的应用程序工具包。

GKS 有六个基本输出原语,任意的图形都是由这些原语及相应的原语属性组合而成的。这六个原语(primitive)是:

- 多点标志(polyarleer):在指定的点列产生特定的符号。

- 折线(polyline):将指定的点列连成折线。

- 填充区域(fill area):在指定点列构成的多边形里填充特定的颜色或图案。

- 正文(text):在指定位置输出一定大小、一定方向、一定形体的字符串。

- 网阵(cell array):在指定位置输出一个平行多边形,该平行多边形是由许多具有不同颜色的小网组成。

- 广义绘图原语(generalized drawing primitive):产生一些特殊的设备特性输出,诸如曲线、圆弧等。

GKS 定义了六类输入设备每种物理设备即可以对应一种逻辑设备,也可以对应多种逻辑设备:

- 定位(locator):获取一个坐标点数据。

- 笔划(stroke):获取一串坐标点数据。

- 拾取(pick):从显示图形中拾取可见目标。

- 取值(raluntor):获取一个实数值。

- 选择(chvice):从众多可选项中选择项目。

- 字符串(string):获取一个字符串。。

GKS 的图形数据可以根据组成图形的原语和原语属性组合存入名为段(segment)的数据结构中。段是可操作的,诸如改变段的可见性、对段施加一个几何变换等。

一个典型的用 Fortran 语言实现的 GKS 系统平均占有 110KB,而用 C 语言来实现只占 15~20KB。因而在 PC 上,GKS 是非常有吸引力的软件工具。现在市场上的高档微机、图形工作站都配有 GKS 软件包。

2. PHIGS PHIGS 是基于 GKS 的概念而开发的一个新型图形标准,它引用了 GKS 的原语集合和粗

象工作站概念,也提出了许多新的思想和技术,使之成为一个便于实时处理、便于分布结构的高交互高动态图形标准。

PHIGS 比 GKS 更加用户模型化,它的数据组织机制和视图通道使系统能够方便地不经大的变化处理各种用户模型,因而称之为模型管理和维护工具。

PHIGS 有别于 GKS 的在于:

- PHIGS 的图形数据的基本存储单元是结构——structure,任一图形数据是一个根 structure 和众多子 structure 组成的有向无环图。因而 PHIGS 的图形数据机制是层次的,而 GKS 是线性的。

- GKS 的 segment 一经建立,其内容不许修改,而只能修改段的属性。PHIGS 的结构元素(structure elements)可以在任意时间修改。

- GKS 的原语属性与原语连接(binding)如图 2. a 所示。因此要修改段中和原语属性,只能放弃原来的段,重建一个新段。PHIGS 的连接方式如图 2. b 所示这种遍及时连接原语属性便于实时修改图形数据。

- GKS 只提供了有限的模型化转换,而 PHIGS 提供了一个模型化坐标空间,在此空间可以生成和维护图形模型。

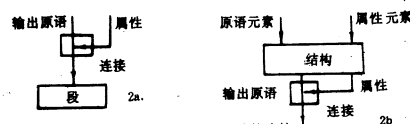


图 2 GKS 和 PHIGS 原语和原语属性连接比较

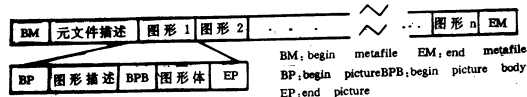


图 3 CGM 形式

PHIGS 立足于支持如下的应用需求:高交互、高动态的操作界面和复杂图象的快速屏幕更新。因而,PHIGS 对环境要求比 GKS 高,实现的技术难度也偏大。

新型图形工作站的出现,使 PHIGS 的应用在不断普及。现在的许多工作站、计算机和软件产品都提供 PHIGS 的工具包。

3. 其它的图形标准 CGM 提供了一种适合于存贮和恢复图象描述信息的文件格式。文件格式由一个有序的元素集合组成,元素用与设备无关的方式来描述图形。如图 3 所示。

CGI 是低级图形系统和设备相关图形设备驱动器与图形操作功能块之间的设备无关图形数据及相

应的控制信息交换的功能和语法性的说明定义。

它由如下元素(elements)组成:

- 生成图形原语。
- 用属性(attributes)控制原语外表形体。
- 获取图形设备性能、状态等信息。
- 控制图形设备。
- 产生图形输入。
- 生成和控制原语集合。

GKS-3D 是 GKS 的扩充,它在如下方面增加了 GKS 的性能。

- 3D 原语。
- 填充区域集合(fill area set)原语。
- 视图的 3D 转换通道。
- 3D 输入通道。
- 边界(edge)属性。
- 3D 几何属性
- 隐藏线和隐藏面移去。

PHIGS PLUS 增加了一些有关线和面的原语以及 B-样条原语,使之能显示科学运算的结果。象非均匀关系 B-样条原语在有限元模型 FEM(Finite Element Method)应用中是十分有意义的,广泛应用于模拟系统、计算机辅助几何设计和最优控制等领域。

**四、图形标准应用前景** 计算机图形学有广泛的应用领域,计算机图形的标准化工作在国际上已经取得了可喜的成绩。然而,计算机图形标准的应用情况却并非是乐观的,特别是在国内还没有引起计算机图形应用开发人员的足够重视。

应用图形标准开发图形应用是十分有益的,是计算机生产者、计算机应用开发者和计算机使用者都获利的最佳方案。

首先生产厂家可以循此方向设计自己的硬件和软件产品 PHIGS-LIKE。GKS-Like 的图形设备;用户选择软/硬件设备有了广阔的余地,也不用担心选择的软/硬件是否兼容,以前的系统和新的产品是否一致或兼容。

如果采用了 API 的标准,程序人员的开发的应用软件可以稍加修改地移植到其它系统里,甚至如果采用了 CGI 标准,API 的标准工具也可以方便地移植到其它计算机系统使劳动的成果受到了充分的利用和共享。

如果采用了 CGM 标准,图形数据在不同系统的交换是十分容易的。采用了 CGI 标准,便于将新的图形硬设备纳入系统里。

采用图形标准开发图形应用也便于图形系统的分布结构组织。即可以组织主机框架的图形系统,也

可以组织为分布的 CGI 工作站形式和网格式的图形工作站形式。如图 4 所示。

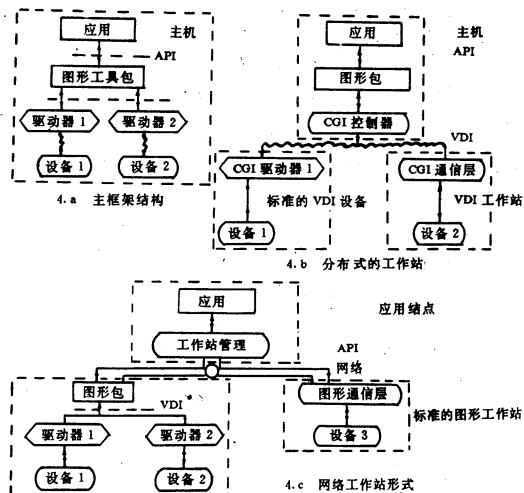


图 4 三种图形应用分布形式

虽然在实现标准时,要花去一些时间,但起到了一次劳动长期受益的效果,同时增加了系统的弹性、可修改性、可移植性。

#### 参考文献

1. Julian Gallop, "Standards in Computers Graphics", microprocessors and microsystem, Vol9, NO. 10, Dec. 1985, P. 494.
2. Peter R. Bona, "A Survey of Graphics Standards and Their Role in information Interchange", Computer, October 1985, P. 63.
3. Mike Heck, "PHIGS Hits the Market", computer graphics world, January 1986, P. 49.
4. Loftoon Henderson, "The computer Graphics nnetafile", IEEE CGQA, August 1986, P. 24.
5. Mienter Bakker, "Triangle Sets in prelers PLUS, a Valuable Link with Finite Edment Mokeling", computerGraphics forum, 10(1991), P. 61
6. American National Standards Institute, "Computer Graphics — Graphical Kernel System (GKS) Functional Description", ANSI X3. 124(1985).
7. American National Standards Institute, "Computer Graphics — Programmers Hierarchical Interatine Graphics System (PHIGS) Functional Description", ANSI X3. 144 (1988).
8. Madeleine R. Sparks and Julian R. Gallop, "Language Bindings for Computer Graphics Standards", IEEE CGQA, August 1986, P. 58.