

电脑与法律

独创性是著作权的核心 群 莲 (2)

电脑应用

条形码技术在商品经济中的应用

..... 陈 岗 (4)

计算机在机电设备进口批准证管理中的应用

..... 吴树林 (6)

软件纵横

Ada 语言语法制导结构化编辑器

..... 戴振喜 (9)

软件介绍

386 微机上位内存的利用 黄立山 (16)

NEW

多媒体微机技术概述 徐超汉 (17)

网络与通信

高级 UNIX 连网技术讲座 第一讲 UUCP

(UNIX to UNIX Copy) (中) ... 冯家宁 (20)

IDEA

Foxbase+使用扩展内存加速 ... 刘永强 (27)

大学生之页

也谈单色显示器的作图问题 任绥海 (28)

使用与维修

AST P386 微机故障维修 曹迎春等 (29)

CR-3240 彩色打印机维修一例

..... 金义炎 (29)

中学天地

384KB 内存的开发和利用 何管略 (30)

游戏乐园

反击战 茅吉等 (31)

C 语言游戏一则 梁宇翀 (32)

猜一猜 玩一玩 李 峥 (34)

病毒防治

漂亮女孩病毒的诊治 王江民等 (36)

Break 病毒的剖析 王延利 (37)

简讯

维普公司推出支持多种应用软件的字形卡

..... (16)

全国单片微机学术交流会通知 (38)

单片机与单板机

可编程控制器中数据块传送的实现

..... 徐巨善 (39)

名厂介绍

陕西省计算机 (集团) 股份有限公司 ... (42)

电脑用户

如何提高 FOXBASE+ 的统计运算速度

..... 冯建军 (43)

IBM PC 机 VGA 显示拍摄技术

..... 叶 宾 (45)

西文状态下特殊字符的打印 黄真康 (48)

《一个自动输出表格的多功能 dBASE 程序》

的修改 阎家富 (49)

如何在 DR-DOS6.0 下使用 2.13H

..... 徐洲痕 (51)

解除 CCED 的日期限制 刘 川 (52)

在 MS-DOS5.0 下使用高端内存的方法

..... 傅运林 (53)

再谈漂亮实用的动态标题程序 ... 王坚城 (55)

微机屏幕图象的压缩存储与恢复

..... 陈惠生 (56)

利用 PC-CACHE 提高 WPS 的运行速度

..... 黄 林 (57)

《正确用 C 语言读取数据》的补充

..... 肖晓斌 (58)

ADM 1.0Password 的彻底消除

..... 叶 武 (59)

PRG 文件结构打印一例 许立明 (60)

动画图形设计中的位图像操作与掩模技术

..... 鄂大伟 (61)

编读往来

读者来信 (8)

广告索引 (60)

独创性是著作权的核心

程序版权议论的续篇

本刊记者 群 莲

本刊第2、3期，连续议论到几个简短程序的版权问题。我们收到了受到批评的陈红新同志的来信，信中说：“从今天收到的《电脑》杂志上，突然看到沈凌同志反映《CEC-I模拟音响》系‘剽窃’其作品一信，让我大吃一惊。急忙找来《学生计算机世界》，发现果然有部分与沈凌同志的程序相同，本人分析后认为：造成此种结果是本人在摘录有用程序文章后未注明出处，时间一长误作自己的作品所致，在此，敬向沈凌同志、编辑同志及广大读者致以深深的歉意。另：余下的几列程序因与沈凌同志的几列程序同存一处，使我也弄不清是属摘录程序，还是自己创作程序。若是摘录他人程序，在此我先向未知作者致歉。同时，本人将吸取教训，对所有摘录文章注明出处及作者，防止此类事情的再次发生”。本刊记者就这件事，走访了华南师大微电子所承担着广东省科委下达的软件法律保护研究课题的王桂海高级工程师。下面是谈话的摘要。

记者：想来您已读到本刊连续两期刊登有关《CEC-I模拟音响》一文的议论。这里是该文作者陈红新给本刊的来信，您有什么看法？

王：是的，我一直关注这件事。读了陈红新来信，我有几点感想。

第一，他看到沈凌的信后，便急忙查找原因，及时向编辑写信；对另外的几则程序，也作了必要的说明，可以看出，陈对此事是认真的。

第二，他承担了责任，向有关方面致歉，表示要吸取教训，态度是诚恳的。

第三，他所作的解释是可信的。一是我国著作权法和计算机软件保护条例施行的时间都不长。对于把计算机软件置于版权保护之下和哪些方面受到保护，估计有相当部分的软件工作者还比较陌生。他们可以是软件方面的专家，但却疏于对软件法律保护的了解，所以，客观上产生某些违反著作权法的行为，并不太使人感到意外。二是计算机程序不象普通文字作品那样易读，那样有明显情节和个人风格。用于表示程序的指令种类很有限，在同样的语言下，要完成同样功能（例如都用于模拟声音），会导致程序有很大的相似。这一来，分辨程序尤其是十分简短的程序是否出于自己之手，便有些困难。象陈那样把抄录下来的程序混淆为自己编制的程序便可能发生。

但是，疏忽导致的错误，仍然是错误。正因为易于忽略，我们更要加倍注意。独创性是著作权的核心。自己署名的作品，一定要注入自己的心血，要真正是自己的劳动成果。抄袭是最典型的侵权行为，是绝对不允许的。我想，陈红新同志一定会吸取教训，今后会创作出许多具有自己特色的软件。

记者：在本刊第3期的文章中提到作品独创性的判断根据之一是作品与作者的能力相副。但人们根据什么去怀疑作者的独创能力呢？

王：这是一个模糊的，又是重要的概念，但目前没有一成不变的，明确的标准。个人的潜力，固然不可以低估，但也不是完全没有客观的尺度。一个作品在不引起怀疑时，谁也不会去查问作者本身所具备的条件（包括学历，阅历，经验、个人才华、客观环境等）。但是，世界上偏偏存在一些足以引起人们怀疑的人和事。人们有时甚至凭直觉也会产生疑问。如果有人投来一篇与他本人实际水平相差甚远的文章（例如，专业相距甚大，外语程度、文化程度不相称等等），编辑便会多加一些调查了解。这样做，是为了向他本人和广大读者负责，并不是对作者的不尊重。事实上，确曾发现过有人拿了别人的文章来充当自己的作品。这算是“把关”吧，可惜这个关也不是绝对把得住。公开发表了的作品，公众有疑问时，是可以提出质疑的。怀疑并不一定有根据，更不等于事实。一般文学作品，个人独创天地很大。作家的天赋、创作能力，没有什么限度，有时很难作出比较。曹禺 23 岁发表《雷雨》，冰心 26 岁发表《寄小读者》，巴金完成《家》的创作时也不到 30 岁。可从来就没有人对这些当时还很年青的作家的创作能力表示过怀疑。人们很自然地承认，这样的作者就能够完成这样的传世之作。文学史上对作者能力产生疑问的一个例子是苏联著名小说《静静的顿河》。作品第一卷问世时（1928 年），作者肖洛霍夫才 23 岁。这部作品题材取自 1912 年到 1922 年间俄国发生的许多重大历史事件，牵涉到大量的人和事，人们可以较为具体地比较这些内容和作者的实际条件。有人便提出，作者虽然已有几年的创作生涯，但当时仅担任过镇的革委会办事员和短期的武装征粮员。以这样的经历而论，怎能完成如此波澜壮阔而又极其深刻地反映整个时代的史诗般的巨著呢？经过很长时间，争论才平息。（这个问题牵涉的内容较多，这里不作详细介绍了）。中外文学中，类似这样的疑案也还有。

计算机程序是功能性的作品。一个程序的难度和它所必须依赖的知识基础、软硬件环境，是很具体的，确定的。它所要求创作的条件甚至是可以估算出来的（例如日本有人把编程人员的实际工龄——相应地是编程的经验——作为创作力量的表示之一，给出量化的等级划分），与文学作品相比，人们相对地易于比较作者能力与程序实际要求是否相副。在被指控为侵权时，是不是确实具备相应的开发能力（力度），便成为提供法庭判定作品独创性的依据之一。碰到这一情况，被指控者最好能从开发人员水平，用工的充分性（即是否有充足的时间），软硬件环境等方面，向法庭提供有足够说服力的佐证。

记者：一个作品被抄袭了，是不是只有被抄作品的作者（权利人）才可的对抄袭者提出控诉呢？

王：不是的，抄袭者把别人的作品用自己的名字来发表，是有意识的侵权行为，既侵犯了权利人的经济权利和精神权利，又欺骗了公众。这样，任何一个发现抄袭行为的人都可以对抄袭者提出控诉。这方面，有兴趣的读者可以查阅一下郑成思著的《知识产权通论》，那里对抄袭行为有较详细的分析。公众提出一般性怀疑，作者可以作答，也可以不作答。但如果是提出控诉，被告就一定要从法律的角度来应对了。我们主张软件开发者认真保留开发过程的全部记录，包括手稿、管理记录、各类文档（文件，文卷）、软件测试记录、各种技术讨论会记录、个人工作日志等等，这些都是判断谁是真正的软件开发者的第一手材料。

《电脑》第 2、3 期的讨论，虽然只是涉及几个很简短的程序的著作权，但这件事本身却很有意义。相信通过这次讨论，会进一步引起广大读者对软件法律保护的兴趣和关注。

记者：谢谢！

137

条形码技术在商品经济中的应用

南京师范大学 陈 岗

随着社会交流的扩大,在瞬息万变的商品市场中,人们对信息的需求量越来越大,同时更加依赖计算机对信息进行处理,条形码技术所具有的可靠准确、输入速度快、灵活实用、成本低、易于制作、无需专门培训等优点使其成为被广泛应用的一种计算机信息输入手段。

小小的条形码能包含很多信息,它已成为商品进入国际超级市场必备的“入场券”,为世界各国纷纷采用,随着国际通用条形码在世界范围内的迅速普及,没有条形码标志的商品不久就会失去立足之地。

条形码技术作为一种实用性很强的自动识别技术,广泛地用于自动化生产线中。

生产厂在每个部件上都贴有部件条形码标签,以标明该部件的生产厂家及批次等信息。由这些部件装配的生产过程通过条形码技术来控制。

工作时,工人将要装配的部件放在生产线的随行夹具上,当随行夹具将要装配的部件依次送到各个工位时,条形码阅读装置首先阅读该部件的条形码编号,并将其传送到计算机中,计算机向机械手发控制信号,机械手便将将要装配的部件装好并记下该部件的条形码。在测试工序时,该工序的条形码阅读器扫描每个部件条形码,在终端上立即显示出所测各部件的号码。若装配齐全,便开始对该部件进行测试。测试工序打印机将所测的装配各部件的号码及测试结果打印出来,送给包装工序。在包装工序上,条形码阅读器扫描包装箱上的条形码,从而统计出装配生产线的产量。

工厂生产线的使用条形码技术,给工厂管理带来许多方便。工厂产品库使用条形码技术可实现产品入、出库的快速管理,并及时统计出库存产品的种类及数量为工厂制定生产计划提供数据。

每种产品入库时将各种信息存档备案,出库时,用条形码阅读器扫描产品上的条形码,计算机可自动作出出库统计,可打印明细帐目,打印流水帐及打印库存表。还可以制定采购计划,核对明细表和流水帐,可以进行多方面查询,如查询某批产品的存量,某种产品的库存及流水账等。条形码应用于仓库管理中提高了人机交换速度,降低了人的劳动强度,还消除了由于人的因素带来的错误,保证了交互信息的正确性。

八十年代以来,计算机成本及价格急剧下降,在商

品流通领域中出现了基于计算机技术的 POS (Point of Sale 销售管理系统。)

POS 系统在发达国家及一些中等发达国家和地区已经广泛应用于餐厅、百货商场及一些零售机构。

目前国外的 POS 系统对商品的标识均采用条形码,我们可以从进口商品包装上看到一些排列着的有粗有细的黑白条,这就是代表该商品的条形码。现在国际上商品销售领域所使用的条形码主要是 EAN 码和 UPC 码, EAN 码主要在欧洲使用, UPC 码主要在美国使用,这两种码制是兼容的。

商品上的条形码所包含的信息,以力士香皂为例。力士香皂包装上的条形码为 5000186363017500 代表生产国为英国,0186 是力士香皂制造商的编号,363017 则是制造商为力士香皂编制的产品代码。不同国家的不同企业生产的产品都只有唯一的一个条形码值。

建立 POS 系统时,先将商品的有关信息如:名称、生产厂商、价格、规格等输入计算机。当顾客在超级市场里购买商品时,收款员只需用条形码阅读器——扫描所购商品上的条形码,计算机即可以自动进行阅读识别,确定商品的代码、名称、品种和制造商等信息,然后查找单价、累计等;把一个顾客购买的全部商品的条形码扫描输入后,计算机便会立即进行汇总结算,输出总金额。这在减少顾客等候付款时间,提高售货效率方面,比人工用键盘输入数码要快速、准确得多。这样可以减少或避免购、销双方因人为因素造成的矛盾,提高商业信誉,改善服务质量,同时还能实现对商品销售信息的分类、汇总、更新库存等处理,并对经营情况和营业员的工作质量进行分析。这对经营管理人员及时掌握市场动态,剔除滞销商品,确定合理库存,优化商品调运方案,加速商品与资金周转速度,充分利用有限资金,扩大经营,保证商业经营活动的顺利进行,都具有重要的意义。

POS 系统为商业带来了巨大的经济效益,安装 POS 系统将使以往的工作量减少 25%~30%。

制造厂家可以通过 POS 系统从销售商那里迅速获得产品的销售信息,及时调整产品结构,生产适销对路的产品,提高企业的经济和社会效益。

制造厂家还可使用条形码进行产品质量跟踪。当产

品进入市场后,有些产品需要维修。由于产品连续生产,先后出厂的产品在时间上有很大差距,超过保修期的产品和其它厂家生产的类似产品在社会上大量存在。保修人员可用条形码阅读器扫描维修部件上的条形码,便可知该部件的出厂日期及生产厂家等信息,确定是否给予保修。同时,维修部的信息的也可及时反馈给工厂,及时发现生产中的问题。

商品条形码标志的设计应考虑条形码扫描器的要求,商品包装形式,包装装潢的要求,以及条形码标志的印刷要求等。

商品的条形码标志形式主要有四种:直接印在商品的标纸或包装容器上;制成标签粘贴或悬挂在商品上;直接印在商品的外包装或运输包装上;直接印在物体上。

一般来说,标志形式的选择应从包装加工成本、装潢设计、条形码标志的印刷及不同的使用等方面综合考虑。

条形码标志印刷位置的相对统一对商品零售业及流通领域的自动化装置非常重要,它不仅可以提高工作效率,减少为寻找条形码而花费时间,而且还可以使商品包装满足不同的条形码扫描器的识读要求。

原则上,所有商品都可采用条形码标志。企业申请使用国际通用条形码(EAN、UPC),应有工商管理部门颁发的营业执照、有自己的商标、有支付能力等条件。

生产、经营食品的企业或公司,若其产品主要出口到美国和加拿大,应申请使用UPC条码,除此之外,原则上都应申请使用EAN码。生产、经营食品的企业或公司,若有产品出口到EAN成员国(或地区),同时有部分产品出口到美国、加拿大,这样的企业若有支付能力可同时申请使用EAN和UPC条码。

申请企业从中国物品编码中心或其在试点省、市设立的“分中心”或中国物品编码中心指定的地方标准化机构,索取并预填申请表。发申请表的机构对申请者进行初审,初审内容有,企业在必须在有商标的产品上申请使用条形码标志,必须拥有商标权。只有在国家工商行政管理部门登记注册的具有《企业法人营业执照》或《营业执照》的企业方可有权申请使用条形码。申请企业生产的产品是否适合使用条形码标志。填写的申请表是否符合要求,特别是年出口额及在美国市场的年销售额等需填写清楚。

中国物品编码中心对申请企业进行复查。对准予申请的企业,由中国物品编码中心或分中心统一填写正式

申请表(英文)。

准予申请的企业按收费通知单,将向UCC(美国统一编码委员会)或EAN交纳的外汇、国际邮电费和条形码系统维护费寄到中国物品编码中心。

EAN系统用户费为:

申请EAN用户资格(有权使用EAN-13),应交纳830.00美元。

申请EAN用户资格(有权使用EAN-13和EAN-8)应交纳1160.00美元。

我国企业加入UCC只须交纳一次性入会费,按企业产品在美国市场年销售额交费:

企业产品在美国年销售额	交费标准
155万美元以下	350美元

155~1000万美元 每百万元销售额,交纳200美元,(销售额不足百万元部分四舍五入至10万元),算出金额再加50美元,为应交的会费)。

1000~9900万美元	2050美元
--------------	--------

1亿~4.99亿美元	6050美元
------------	--------

5亿美元及以上	10050美元
---------	---------

申请EAN或UPC条形码的企业须交国际邮电费244元人民币(包括申请表、国际快递费、外汇邮寄费及中国人民银行收取的手续费);同时申请EAN及UPC条形码的企业交488元人民币,从第二年起,申请EAN条形码的企业,每年交国际邮电费110元人民币。

为了保护获得EAN和UPC条形码使用权的企业的权益不受侵犯,及条形码标志的有效性,建立国内的条形码应用、管理及质量保证系统,负责对外联系和处理涉外事宜,及时为企业提供国际国内条形码发展的最新信息及技术培训技术咨询服务,以保障我国条形码系统的正常运转,企业应按以下规定交纳条形码系统年度维护费用:

1.进出口公司	4000元
2.生产型集团公司	2000元
3.单个生产型企业	1000元
4.由中心统一解决EAN产品代码企业	500元
5.只需“中心”解决个别产品代码的企业	150元

企业无论何时办理申请手续,都要支付当年系统维护费的全额。

中国物品编码中心统一将申请表和外汇分别寄送UCC或EAN。经UCC或EAN批准后,中国物品编码中心将为UCC或EAN分配制造商编代码,并通知申请企业。企业即已可在其产品上使用条形码标识。

计算机在机电设备进口 批准证管理中的应用

北京清华大学科研处 吴树林

【内容提要】 本文着重阐述机电设备进口批准证计算机管理软件的实现过程, 软件的技术特色, 以及启用该软件后所产生的经济效益和社会效益。

【关键字】 高等学校, 信息系统, 办公室自动化, 机电设备。

计算机技术的不断提高, 有力地促进了管理科学、信息科学的发展, 因而对管理部门提出了越来越高的要求。受国务院机电设备进口审查办公室(以下简称国审办)的委托, 我们设计完成了机电设备进口批准证计算机管理系统(以下简称开证系统), 对于国审办所属的分布在北京及全国各地的共约一百个部门、地区的进口审查办公室(以下简称二级审查办), 能够独立将本单位终审内容用计算机开出批准证和进行统计工作, 并能定期以软盘传递或远程通讯的方式, 向国审办报送开证的原始数据, 从而加强了机电设备进口审批工作的管理。

下面介绍我们对开证系统软件设计的特点及实现的过程。

一. 系统的结构及功能

1. 系统的结构

开证系统软件是采用自顶而下的结构化程序设计的方法设计的。这种结构化的程序设计最基本的结构有三种:

- (1) 顺序结构;
- (2) 条件选择结构;
- (3) 循环结构。

由这三种基本结构(可以是只用其中的一种、二种或三种)构成能完成某种功能的模块, 一个功能模块还可以根据需要分成若干个能独立完成某种功能的子模块, 用这种方法设计的管理系统称为模块化层次结构的软件。其特点是层次分明, 结构清晰, 便于设计, 便于维护。

开证系统的功能流程见图1。

2. 系统功能的实现

(1) 基本数据管理功能模块

主要完成批准证的数据输入、修改、整理和转储工作, 同时还可以对开证系统软件的参数设置进行维护。因此在这个功能模块下又分别设数据输入、数据修改、数据整理、数据转储、参数设置等子模块。

参数设置子模块可以实现屏幕颜色、默认参数、密码口令等参数值的重新设定。由于不同的硬件配置或不同的软件环境, 屏幕颜色的显示效果可能有差异。我们为几种常用的汉字系统配置好屏幕颜色参数, 用户可选用, 也可以自行配置屏幕颜色参数, 使得开证系统软件能够适应各种硬件环境或软件系统, 以保持美观醒目的屏幕菜单显示。用户对一些固定不变的代码可以设置成默认值, 例如

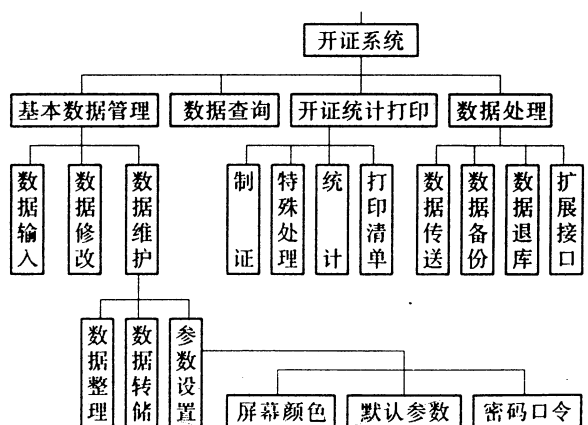


图1 机电设备进口批准证计算机管理功能框图

用户主管单位代码、到货口岸代码等。一旦设置好默认值,在数据输入时这些代码就不必再键入了。为了保证开证系统软件不被非法用户使用,在一些关键部位分别设置了密码口令,用户可重新设置密码口令值,如果此值设置成“0”,则表示取消该级的密码口令。

(2)数据查询功能模块

数据查询的设计主要考虑用户使用简单方便,又能满足要求。查询的方式是:按批准证号快速查询批准证的数据;按要求分类汇总;按组合条件查询。按组合条件查询是实现不同数据项的“与”操作和相同数据项的“或”操作。

(3)制证统计打印功能模块

在这个功能模块下分成制证、统计、打印清单等三个功能子模块。制证子模块实现批准证按规定格式单证制作或多证连续制作,单证制作为每次只打印一份批准证,多证连续制作则将用户指定的日期范围内的所有批准证连续打印出来。对于同一份批准证,制证只允许进行一次。对那些已经制证过的而仍需要对数据进行修改的批准证,可在此功能模块中进行特殊修改。特殊修改只允许修改指定的四个数据项,特殊修改完成后可在此模块中进行备注栏打印,即将特殊修改的内容打印在批准证的备注栏上。对于同一份批准证,特殊打印只允许进行一次。统计子模块是将当月的批准证数据按“引进项目和单机进口审批简况表”的要求汇总并打印出来。打印清单子模块可以打印输出指定日期范围内开出的批准证的清单。

(4)数据处理功能模块

根据执行功能的不同,数据处理功能模块又分为数据传送、数据备份、数据退库、扩展接口等四个功能子模块。数据传送子模块具有数据上报和数据接收两个功能。数据上报是由各二级审查办将每月审查批准进口的批准证的数据以远程通讯的方式报送给国审办;数据接收是由各二级审查办定期接收由国审办反馈的数据。可使用数据备份子模块做数据备份,然后将已作备份的批准证数据从开证系统的数据库中删除,即作退库处理。

扩展接口子模块是为用户准备的功能扩展的接口。当用户在使用开证系统的基础上,需要再增加一些功能时,可以将此菜单修改为用户所需要的内容,并按指定的文件名编写程序。当在运行开证系

统软件时,将通过此接口去执行用户编制的程序的功能。

二.开证系统的技术特色

用户使用一个软件,与机器打交道最多的是屏幕菜单的选择,工作量最大的是数据的输入。这两个问题是决定软件质量的主要因素。菜单的选择是用户界面最重要的组成部分,菜单的设计要力求简单,快速美观,一个快速美观、使用方便的菜单会给用户一个友好的感觉;而数据输入则要求迅速、准确。开证系统软件的设计较理想地解决了这两个问题。

1.菜单的设计美观清晰,菜单的选择采用光标控制键 ↓ 或 ↑ 控制一条光带来选择当前定义的菜单,选中后按回车键,系统则执行光带所指的功能。

2.数据采用代码输入,并设准确性检验。每个数据项的代码所对应的数据内容在提示行上显示出来,供用户选择。当代码较多时,可用光标控制键 → 或 ← 进行向前或向后翻页显示,选中后按回车键,然后输入代码,代码所对应的内容即自动填入数据项所在的位置上。代码的选择只能在该数据项的代码范围内进行。

在批准证号输入时设唯一性检验,申请金额设域值限制,如果违反,则在提示行上出现声动结合的警告提示,只有在用户按下光标控制键 ↓ 之后,才停止警告显示,等待用户作适当的处理。

在程序设计中,还有以下几个特点:

3.利用 FOX 系统的功能,使软件系统简单化。由于软件设计是模块化的层次结构,一个模块完成一个功能,所以模块(也即命令文件)个数就很多,直接影响了系统的处理速度。因此,我们采用了 FOX 系统的过程文件,一个过程文件可包括 128 个命令文件,仅用一个文件名表示。每个过程以 PROCEDURE <过程名> 开头,以 RETURN 结束。过程文件用 SET PROCEDURE <过程文件名> 打开,这个指令将把过程文件中的所有命令文件都读入一张表中。当使用 DO 命令调用此过程中的某一命令文件时,系统就自动搜索这张表,不必再到磁盘中去读取,从而节省每次打开文件所花费的时间。同时,由于磁盘目录中只以过程文件名字出现,从而节省了磁盘的相当空间。

4.组合查询灵活方便。用户在确定组合条件

时,可根据提示选择数据项号,并给定条件。系统自动根据用户给定的条件,实现了同一数据项的“或”操作和不同数据项的“与”操作的组合。当选择的组合条件太长,将要超过系统所限制的一条命令的极限长度时,在提示行上发出声动结合的警告,并终止用户继续选择查询的条件,同时显示出用户已选择的组合条件,用户认可后即迅速显示出查询结果。

5.便于维护。允许重新设置系统参数。

6.便于功能扩展。

三.使用开证系统,提高管理水平

1991年10月,国务院机电设备进口审查办公室发出通知:为加强机电设备进口管理,决定从一九九二年一月一日起,启用新的进口批准证[注1]。这种新的批准证,就是使用计算机管理,通过开证系统打印的。与此同时,国审办所属的分布在北京及全国各地的共约一百个部门和地区的进口审查办公室,都使用开证系统。并在现用的计算机中插入TurboFax传真通信卡,利用现有的电话,和国审办连接为具有数据通信功能的信息系统(图2)。

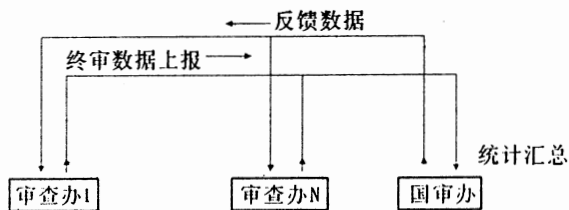


图2 机电设备进口批准证数据远程通信示意图

现在,在国务院机电设备进口审查办公室中,可以非常方便地与分布在全国各地的二级审查办进行远程数据通信,收集数据,及时进行处理和统计,同时还可以进行远程软件程序的传递。使用开证系统软件,有以下三点体会:

1.明显地提高了工作效率。按以往常规做法,每月各二级审查办用文字上报批准证发放情况,国审办按规定时间很难将全部数据收集齐。100多份报表要在短期内进行汇总统计,经常需要采用“人海战术”和“加班加点”来完成。使用开证系统后,只需用电话通信联系,几分钟就可以完成数据的传递,并且在很短的时间内就可以完成数据的汇总和打印,数据上报迅速,明显地减轻了汇总统计工作的劳动强度,提高工作效率。

2.提高了管理水平。过去由于时间紧,临时突

击搞统计,尽管花了大量的人力物力,数字也很难准确。在大量的数据中,要寻找某方面的信息,或按某种要求进行统计,效率很低,有的甚至可以说是很难实现。使用开证系统,各二级审查办可以保证数据的正确管理,并能及时迅速地国审办传递数据。国审办可以按时掌握全面情况,在最短的时间内提供出各种数据和各种汇总统计资料,从而为领导的决策及时提供准确的依据,大大地提高了管理水平。

3.提高了管理的规范化、标准化程度,有效地防止伪证、假证现象。国审办及各二级审查办,每年担负着大量的机电设备审批工作,由于各审批机构原来的证件格式不统一,各种假冒、伪造进口证件的现象时有发生,且给海关验放带来极大困难。开证系统启用后,全国范围内使用统一格式的机电设备进口批准证,并实现全国联网,保证了批准证的准确性。

4.提高了管理人员的基本素质。使用了开证系统,不但提高了管理水平,而且还加快了对管理人员素质的提高。进口批准证采用计算机管理,迫使管理人员不得不去学习计算机的硬件和基本原理,也不得不去学习最基本的计算机语言和计算机管理的知识。同时,由于采用计算机管理,就必须做到录入数据的标准化和规范化。为此,管理人员就必须提高自身的业务水平和管理水平,以适应当前工作的需要。

[注1]见国审办文件:国机审[1991]47号

[注]开证系统软件已于1992年9月25日通过了国家教委的技术鉴定 (139)

读者来信

编辑同志:

收到93年第1期电脑杂志,见到我写的《给COPY II PLUS5.2增加功能》一文在贵刊上发表了,可是文章中增加功能的入口修改部分漏排了。这样会给想使用此文介绍的方法修改自己程序的读者带来困难。

故请贵刊设法,改正这一错误,使发表文章的初衷得以实现。

改正部分为:将第55扇区读出,从98开始键入20 65 F7 20 F7 F7,并写回原扇区。

致

礼

杨建国

93年3月

Ada 语言语法制导结构化编辑器

解放军 39067 部队 62 分队 戴振喜

【内容提要】ASED (Ada structured Editor) 是面向 Ada 语言的语法制导结构化编辑程序,它是开发 Ada 软件的强有力工具。本文从结构化编辑技术的实用意义和一般原理入手,详细介绍了 ASED 的设计原理和实现技术,给出了 ASED 的系统结构和系统功能以及用户界面,还深入地介绍了系统的重要数据结构,最后指出了系统的特点。

一、引言

在软件开发过程中,频繁地用到两项最基本的工具:编辑程序和编译程序。编辑程序用以输入和修改源程序;编译程序则用来把这些源程序翻译成可执行的目标程序。通常,与用户产生交互作用的编辑程序完全不理解它正在编辑的节目的结构和语法关系。因此,它无法判断编辑过程中,所编辑的源程序是否发生过错误,只能在编辑完成之后,由编译程序查出程序中的错误。然而,编译程序与用户之间没有任何相互对话,在它对程序进行编译期间,用户不能对程序进行干预和修改。如果遇到错误,哪怕是最简单的错误,例如某个语句少了一个括号,也得退出编译,重新进入编辑程序来修改这个错误,然后再次进行编译。而前一次已经编译过的地方,一概推倒重来,即使你的错误是出现在程序尾部,再次编译时,也得从头开始。

编辑和编译的独立性,使得编译程序无法利用编辑操作的交互性质。从而导致:为了消除本来是很简单的语法错误,往往要进行多次的编辑、编译、再编辑、再编译这样的重复过程,耗费大量的人工和机时。

如何解决这一问题呢?

人们常说“防患于未然”、“把错误消除在萌芽状态”。这都是说应尽早地防止和消除隐患和错误,而且,防错更比纠错重要。纠错改错,莫如及早防错。把编译过程中对语法的检查工作前移到具有交互性质的编辑过程中来完成,使得通过编辑后所得到的源程序不再存在语法错误,这是解决问题的根本途径。语法制导结构化编辑程序便是这一途径上的产物。

语法制导结构化编辑程序与普通文本编辑程序一样是交互式的,但它比文本编辑的功能更强,使

用更方便。它熟悉语言的结构和语法细节,并能在语法的制约下,引导用户自顶向下,逐步求精,完成自己的程序设计。即使是只知语法梗概,不甚了解语法细节的用户,也可以直接利用结构化编辑程序设计和编辑自己的程序而不会出现语法错误。并且在编辑过程中还熟悉了语法。

利用结构化编辑输入程序时,不是逐个键入字符,而是以语法结构为单位,通过用户干预,一次导出一个完整的结构,用户只需键入如标识符、表达式等非结构性的正文短语,而语句框架、保留字、标点符号等固有信息都由结构化编辑程序自动给出。用户所输入的正文短语,编辑程序也会立即进行语法检查。这样,便保证了被编辑的程序在语法上始终是正确的。

结构化编辑用于程序开发过程中,至少具有如下优点:

- 1、避免语法错误,节省大量的人工和机时。
- 2、支持自顶向下,逐步求精的程序设计方法。
- 3、提供易于阅读理解、具有规范格式的输出形式,提高可读性。
- 4、提高输入速度,降低操作失误概率。
- 5、有利于程序设计语言的教和学、缩短学习语言的培训时间。这对于复杂的语言尤为明显。

Ada 语言是一种规模庞大,语法结构复杂的语言。学习、掌握和推广 Ada 语言需要比其它语言花费更多的时间。所以,语法制导结构化编辑对于 Ada 语言来说更能发挥其优越性。

尤其是:Ada 语言是一门重要的语言。它不仅是一种编码语言 (Programming Language),而且还可作为程序设计语言 (Program Design Language)。Ada 语言是一种全面的、通用的高级语言。它有丰富的定义数据类型的手段,有通常的

控制结构,有类似于 pascal 的那些过程和函数,有嵌套式的程序结构。Ada 的程序包体现了软件结构模块化的要求,提供独立的软部件;Ada 中的任务相当于进程,提供实时,并发和通讯功能;Ada 的异常处理功能可把检验和中断等引入语言中;Ada 把程序分割为可单独编译的单位,称为编译单元。Ada 语言是美国国防部耗资五亿美元,历时八年研制的全军统一的程序设计语言。目前,我国已有越来越多的计算机行家乐于使用 Ada 语言。特别是军内,大型软件系统都正在尝试用 Ada 语言开发。所以,研制 Ada 语言的结构化编辑程序有着十分重要的实际意义。笔者经过一段时间的努力,在微机上用 C 语言实现了 Ada 语言语法制导结构化编辑程序 ASED。本文将详细介绍 ASED 的设计思想和实现技术。

二、ASED 的设计原理

1、实现结构化编辑的一般原理

结构化编辑的概念起源于程序不仅是文本,尤其是严格定义的结构这一事实。在 70 年代初国外研制出的 EMILY 系统就是通过在其内部按一定结构建立一棵语法树来实现的。编辑过程就是用适当的产生式实例(模板)来代替语法树上非终结节点的过程。系统总是给出当前非终结节点可能的一组替换,用户从中挑选一替换规则,系统进行相应替换,然后又设置新的当前非终结节点,并给出新的替换集。EMILY 是纯语法制导的编辑器,即使是表达式也是按其语法定义逐步导出。对于那些较低层、不具有结构特点的语法成份也按语法逐步导出,使用起来就会显得十分繁琐,给用户带来不便,这是 EMILY 的不足。为克服这一不足,70 年代末 80 年代初开发的 CPS 系统采用了基于模板(template)和占位符(place holder)以及短语(phrase)的模型。模板是指具有一定结构的语法框架;占位符即非终结符;短语指不具有结构特性的字符串,包括标识符、表达式、赋值语句等。CPS 对具有结构的语法成份作语法制导,如 while、for、case 等语句都按模板输入,而对于短语则由用户直接按普通正文编辑方式输入,但并不放弃语法制约,用户每输入完一个短语,系统立即作语法检查,不符合语法的给出错误标志。我国科学院软件研究所在 80 年代中期研制的 C 语言结构化编辑器 CSSED 也是采用模板、占位符及短语的模型。

可以看出,事先定义语法结构的模板、采用模板驱动方式是实现面向特定语言语法制导结构化编辑程序的共同特点。

2、ASED 的设计原理

Ada 语言语法文本的规模比一般语言要大得多,其语法结构相当复杂。按结构化编辑的观点来划分,Ada 文法中,可以展开为新的结构的占位符共有 52 个,它们可以展开成 116 个不同的语法结构。如果按照模板驱动方式实现 ASED,首先要将 Ada 的语法造好 116 个模板,把这些模板嵌入程序当中,势必导致 ASED 系统过于庞大。而且,如果语法数据有错,为了改一个数据错,程序需要重新编译和连接。ASED 的设计策略是:将 Ada 的语法作为纯数据文件,独立于系统的程序而存在。当 ASED 启动工作时,首先将语法数据文件读入系统,并生成一棵语法树,以此代替语法模板。这样做,除了程序代码量大大减少,使系统显得紧凑和修改数据方便,不需重新编译和连接等优点之外,还有一个潜在的好处是:为实现通用结构化编辑程序打下了基础。显然,通用结构化编辑程序不可能将所有的语言的语法嵌入在系统程序当中,只能根据不同语言的语法建立不同的数据文件,这些文件与系统本身无关,当用户需要用到哪种语言时,系统将该语言的语法数据文件便读入系统,并变换成具有统一格式的语法树,以供系统作语法推导时使用。有关通用结构化编辑,笔者将在另一篇文章中专门介绍。

与其它结构化编辑一样,ASED 也是在语法制约下,引导用户进行交互式编程。系统一经启动,就自动给出当前非终结节点的一组替换 menu,用户从中挑选一项,系统进行相应替换,然后又设置新的当前非终结节点,并给出新的替换 menu。编辑过程中,用户不必担心语法细节,系统能自动插入保留字和标点符号等。ASED 在其内部除了建立语法树外,还建立一棵语法推导树。推导树包含了程序结构属性和显示输出属性,推导树由语法规则逐步导出。用户在终端屏幕上上进行编辑操作,所操作的对象是符合他们习惯的格式化的源程序,而在系统内部,操作对象就是语法推导树。每一个编辑操作首先在推导树上引起动作(推导、拉链、摘链等等),然后再反映到用户可见的源程序上去(展开、插入、删除等等)。编辑过程实际上就是对推导

树的推导过程。推导的依据就是语法树上的语法规则和系统接收到的用户命令。

由上可知,ASED的实现是采用语法树驱动方式,而不是模板驱动方式。必须强调:ASED的语法树定义,不同于其它结构化编辑的语法树,它只包含Ada语言的语法规则和语法结构,而不包含程序的诸多信息。语法树一旦建立,装入了Ada语法后,其大小和内容再不会发生变化。它的作用是当系统根据用户命令作语法推导时,提供语法依据。ASED的语法推导树才包含了用户程序的全部内容,推导树的大小是随着编辑的进行,逐渐增大的。程序越长,推导树越大。当推导树上所有的非终结节点全部被替换完了,编辑过程即结束,用户得到一个没有语法错误的源程序。

三、系统结构及工作流程

1、系统结构图

系统结构如图1所示。

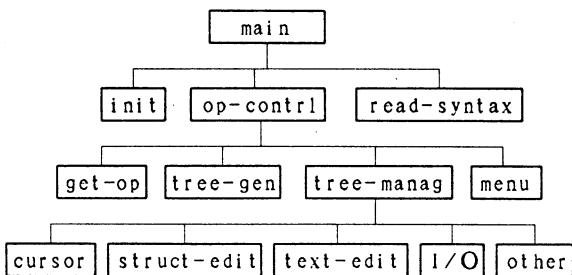


图1 ASED系统结构图

在图1中 main是主控程序;init是初始化模块;op-ctrl为命令控制块;read-syntax读入语法;get-op从键盘上获取用户命令;tree-gen为树生成模块;tree-manag为树管理模块;menu为菜单程序;最底层的各模块为命令执行模块。各模块之间的关系详见“工作流程”。

2、系统工作流程

本节根据图1叙述ASED的工作流程。系统的顶层是主控模块main程序,它首先调度read-syntax模块,建立一棵语法树,读入Ada语法,装入到语法树上;然后调init模块,建立一棵语法推导树,置其根节点root为当前节点,此时,从语法树读入第一个语法成份(comp-unit)作为当前节点root的长子节点,同时置该节点为当前节点,再调op-ctrl模块,至此,系统开启工作,准备接收用户命令,进行编辑。此后main程序反复调

op-ctrl模块控制系统按照用户命令的意图运行下去。

在系统的第二层,op-ctrl模块的功能是:接收命令、判别命令,转去执行命令。在接收命令之前,先要根据当前节点的类型,决定是否产生menu。如果是具有多种替换规则的占位符,则要先调menu模块,生成相应的menu,显示给用户,然后等待接收用户命令,如果是只有一种替换的占位符,则直接调tree-gen模块,进行语法推导,生成当前节点对应的子树;如果是正文短语占位符,则等待用户输入命令。

用户命令分为两类:一类是树生成命令,另一类是树管理命令。当接收到用户命令,经过判断,若属前者,则转tree-gen模块,由tree-gen程序根据语法树上的规则进行推导,生成新的子树,并将子树以正文形式显示给用户;若属后者,则转tree-manag模块,执行相应的功能。get-op模块专门用于接收用户命令。

系统第三层的tree-manag模块是一个多出口的树管理控制程序,它调用cursor、struct-edit、text-edit、I/O和other五个模块,完成光标控制、结构化编辑、正文编辑、输入输出和其它命令的控制功能。

第四层的五个模块是具体执行用户命令,实现用户意图的一层。其中cursor模块执行用户的光标移动命令,完成上移、下移、左移、右移、移到行首、移到行尾、上移一页、下移一页以及作move操作时的光标定位等功能。struct-edit模块执行用户的结构化编辑命令,包括插入、删除、undo、剪辑等命令。text-edit模块执行用户的正文编辑命令,并负责正文短语的法检查。I/O模块执行用户的输入输出命令,包括输出Ada源程序,向磁盘输出树文件,从磁盘读入树文件。other模块执行其它命令,包括给程序编行号,去掉行号,提供Help命令等,以后还可以根据需要,再扩充命令。

四、系统功能

ASED提供如下功能

1、对于Ada语言具有语法制导的结构化编辑功能。具体内容是:

(1)对于具有结构特性的非终结符(即占位符)能实行语法制导下的结构化编辑,并且对那些有多种替换规则的非终结符能自动给出menu,列

出所有可能的替换规则,供用户选择;对只有一种替换但具有结构特性的非终结符能自动完成规则的替换,导出新的结构。

(2) 对于不具有结构特性的非终结符提供正文编辑方式,并能对它们作语法检查。

2、在编辑过程中的任何时候都能保证当前程序在语法上是正确的。

3、能创建和编辑新的 Ada 源文件,能编辑已存在的 Ada 源文件(指原来是用 ASED 编辑过的文件)。

4、能将所编辑的程序以源程序文件和树文件两种形式转贮到外存。

5、能将树文件读入系统,并恢复到前一次的编辑状态。

6、能方便地实施插入和删除操作;能在程序中添加注解;能插入空行;能把某程序段移倒(move)指定的位置(语法允许的地方);能对删除的内容实施 undo 操作。

7、能提供求助(Help)命令。

8、能方便地扩充用户命令集。

五、用户界面

1、屏幕分配

ASED 采用多窗口技术,将屏幕分为四个窗口,分别完成不同的功能。

(1) 程序窗口(program-window)

这是系统的主窗口,用于显示用户编辑的 Ada 源程序。此窗口的大小与屏幕相同。

(2) 正文编辑窗口(text-edit-window)

正文编辑窗供用户作正文编辑时用,位于屏幕下方,窗口的高为三行,宽为 60 个字符。

(3) 菜单窗口(menu-window)

此窗用来显示 menu,它位于屏幕右上角,其大小随 menu 的规模而定,菜单有几行,就开出一个显示几行的窗口。

(4) 剪辑窗口(clip-window)

在编辑过程中,用户有时可能希望对某块程序进行剪辑性修改。剪辑窗供用户作剪辑时存放子树之用。该窗位于屏幕右下方,大小随子树的大小而定。

以上四个窗口,只有程序窗口是常开着的,其余三个窗口都是当需要时,由系统自动打开,用完后即自动关闭,不会总出现在屏幕上。这样做的好

处是:用户不会觉得窗口太多而有碍视线,屏幕上“清洁”的,使人觉得既方便又简明。

2、光标控制

在结构化编辑中,光标不需要逐个字符移动,而是以语法单元为步长,一次移动一个语法单元,在内部就是移动一个树节点。在屏幕上光标总是位于某语法单元的首字符处。本系统利用小键盘上的←、→、↑、↓、pgup、pgdn、home、end 分别完成左移、右移、上移、下移、前翻一页、后翻一页、移到程序起点、移到程序末尾等功能。

3、操作命令

ASED 命令包括:占位符的展开、结构化编辑和正文编辑以及文件的输入输出命令。(详见《用户手册》)

六、数据结构

语法树和语法推导树 ASED 的主要数据结构,本节将详细介绍它们。

1、语法树

ASED 的语法树是用来存放语法信息的,它是 Ada 语法规则在系统中的树形表示。

(1) 语法树的构成

语法树是由子树构成的,每棵子树表示一个或一组语法结构。语法树的根叫 root,子树的根叫 subroot。Ada 语法中有 52 个具有结构特性的非终结符,即分别对应 52 棵子树的根(subroot),它们的替换与相应的 subroot 一起,构成 52 棵子树,这 52 棵子树与 root 一起构成 ASED 的语法树。下面给出子树的结构图(见图 2)。

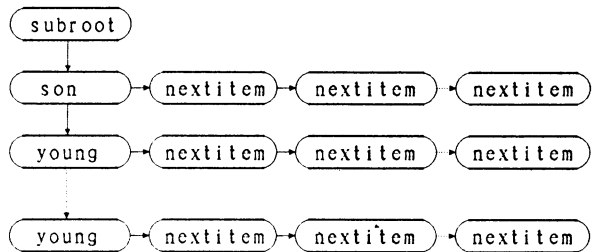


图2 ASED 的语法树子树结构图

(2) item 的定义:在 Ada 文法中,任何一个由 BNF 的元字符对“<>”或“[]”或“{}”括起来的字符串是一个 item;任何一个终结符或任意一组由空隔分开的终结符是一个 item。

下面举例说明:

例: `<body-stub>:: = <subp-spec> is separate;`

`| package body< pack- simp- name> is separate;`

`| task body< task- simp- name> is separate;`

此例中, “`< subp- spec>`”、“`<pack-simp-name>`”和“`<task-simp-name>`”分别是由元字符对“`<>`”括起来的, 所以它们都分别是一个 item; 而“`is separate;`”、“`package body`”和“`task body`”也分别是一个 item。

(3) 构成子树的原则

构成子树的原则是: 对于具有结构特性的某非终结符, 它自身作为 subroot, 它的第一条规则的第一个 item 作为它的 son 节点, 第一条规则的其余 item 依次作为 son 节点所指向的 next item 节点; 它的第二条规则的第一个 item 作为 young 节点, 第二条规则的其余 item 依次作为这个 young 节点所指向的 nextitem 节点; 它的第三条规则的第一个 item 作为下一个 young 节点……依次类推。

现在画出上例中的语法结构所对应的子树结构如图 3 所示。

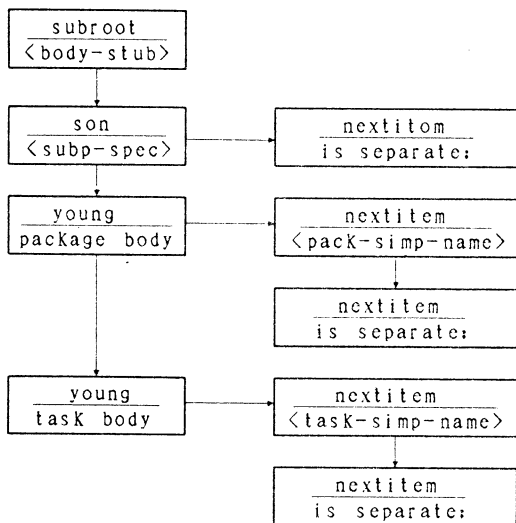


图3 Ada 语法实例对应的子树结构图

(4) 节点的组成

语法树的节点分为两类: 所有的 subroot 节点为一类, subroot 以外的其余所有节点为一类。

下面分别介绍:

(1) subroot 节点

subroot 节点组成如下:

`char * name;`

`int nson;`

`int lson;`

`struct nod * son;`

其中: name 指向该非终结节点的名字;

nson 表示该非终结符的替换规则的条数;

lson 表示该非终结符的所有的替换规则中第一个 item 的长度最大者;

son 指向该非终结符的第一条规则的第一个 item 节点。

仍以非终结符 `<body-stub>` 为例:(见图 3)

name 指针指向 `<body-stub>`;

nson = 3 因为它有三条规则;

lson = 12, 即 `package body` 的字符个数;

son 指向指向 `<subp-spec>` 节点。

(2) 子树的其余节点

在每子树中, 除了 subroot 外, 其余所有节点的形式是相同的。它们的组成形成如下:

`char * name;`

`int dx;`

`int dy;`

`struct nod * next item;`

`struct nod * young;`

其中: name 指向该节点的名字;

dx 表示该节点的起点在结构中的 X 方向的增量;

dy 表示该节点的起点在结构中的 Y 方向的增量;

next item 指向该节点的 next item 节点;

young 指向点的 young 节点。

由 `<body-stub>` 的语法结构, 很容易得出其对应的子树的各节点的成员的值。

例如: `<subp-spec>` 节点, 它的各成员的值确定如下:

name 指向 `<subp-spec>`

dx = 0

dy = 0

next item 指向 `is separate;` 节点 (见图 3)

young 指向 `package body` 节点 (见图 3)

2、语法推导树

语法推导树是在编辑过程中，系统根据用户命令和语法规则进行推导所得到的子树构成的树。

(1) 语法推导树的结构

ASED 的推导树的结构如图 4 所示。它既指出了各节点之间的链接关系，也表达了其用于 Ada 语法推导的物理含义。

由图 4 知：各节点之间的链接关系如下：

父节点与长子节点是双向链连接的，父节点与次子节点是通过长子节点来连接的，父节点与次子节点是通过长子和次子来连接的，……依此类推。而每个子节点都各自有一条链直接指向它的父节点。兄弟关系与此不同，相隔若干节点的兄与弟，一定是通过中间的兄弟来连接的。树的纵向、横向都是双向链连接的。

各层的物理含义如下：

第一层为编译 (comp)，这是 Ada 语法的根节点；

第二层为编译单元 (comp-unit)，这是子树的根节点；

第三层为构成编译单元的语法结构 (structure)，也是子树根节点；

第四层为构成语法结构的子结构 (substructure)；也是子树根节点；

.....

最后一层是构成语法结构的语法单元 (unit)，这是树叶节点。用户最终得到的是由这些树叶节点构成的 Ada 源程序。

(2) 结构化编辑的操作与推导树的推导对应关系

下面介绍一组实例，用以表述结构化编辑中，占位符展开的外部显示与推导树推导的内部表示的对应关系。各例中外部显示所对应的内部显示请对照图 5 中相同序号的分图。“外部显示”中，有下横线的表示光标当前所在的占位符，即当前非终结符；在“内部表示”中，有下横线的节点表示推导树上与光标对应的当前节点，有两道下横线的表示此节点已置为隐式状态，此节点在屏幕上不可见。图中用左斜线连接的两个节点是父子关系；用右斜线连接的两个节点是兄弟关系。

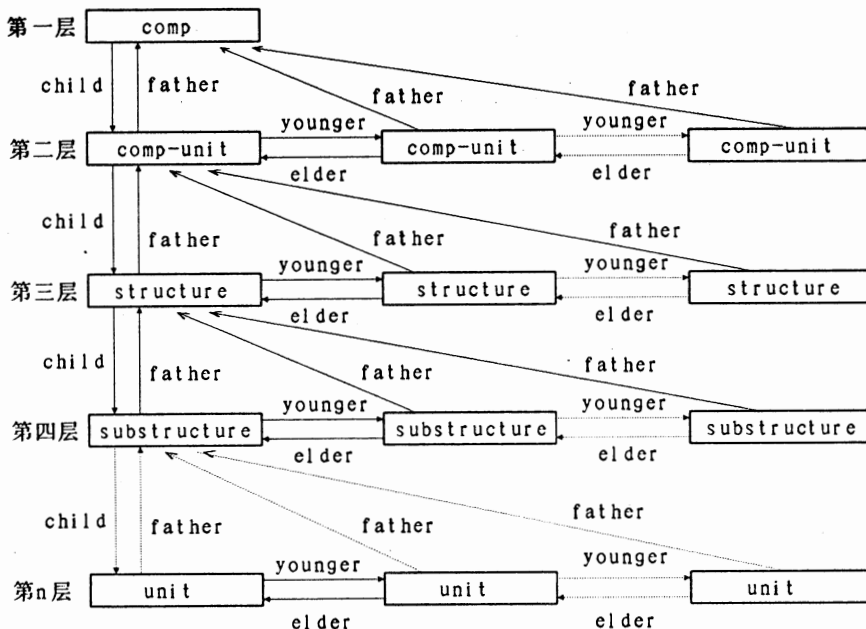


图4 Ada 语法推导树结构图

例 1:

begin 结构.

外部显示:

begin

<stmt>

end

内部显示见图 5

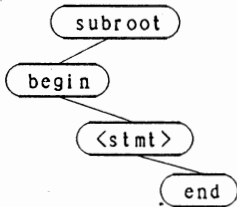


图 5

例 2:

将例 1 的 <stmt> 展开为 if 结构.

外部显示:

begin

if <condition> then

<stmt>

end if

end

内部显示见图 6

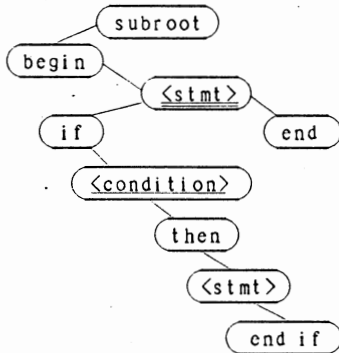


图 6

例 3

将例 2 中的 <condition> 展开为正文短语.

外部显示:

begin

if (a > b) then

<stmt>

end if

end

内部显示见图 7

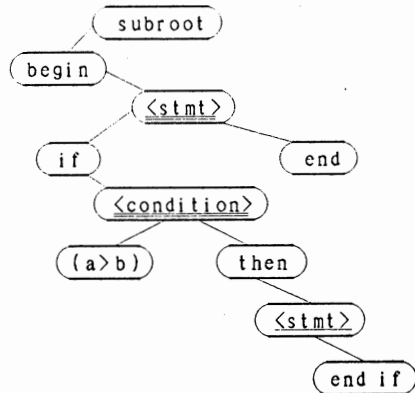


图 7

七、结束语

ASED 是经过精心构思而成的,它具有操作简便,响应速度快,功能实用等特点,是开发 Ada 软件的强有力工具.

设置语法树和语法推导树是 ASED 的独到之处.它使得语法规则可作为独立的数据输入系统,避免了庞大的语法模板(template)嵌入程序当中,并且使得对语法结构(如缩进格式等)的修改与系统程序无关;它还使 menu 可以自动产生,而不需要专门的 Ada 语法 menu 的数据模块.

由于语法数据的独立性,又使得实现 ASED 的算法与所面向的语言无关,给实现通用的结构化编辑提供了条件.

ASED 在运行期间,其推导树对内存的占用是实报实销,用多少申请多少,不用时,立即释放,使内存空间得到充分利用.

ASED 的内部结构十分紧凑,层次清晰,程序设计模块化,许多复杂的算法分割为非常单纯的小模块,一个模块解决一个问题.各模块内聚力大,模块之间耦合度小.有效地提高了程序的可读性.这必然给今后的使用和维护带来方便.

主要参考文献:

- [1] T.Teitebaum and T.Reps, "CPS-The Cornell Program Synthesizer: A syntax-directed programming Environment". ACM. vol24, No.9 (1981)
- [2] 卢慧琼等 "C 语言的结构化编辑器", 《计算机研究与发展》, 1986.7
- [3] 任承帮, "Ada 备忘录", 《计算机学报》1986.5
- [4] United states Department of Defence, "Reference manual for the Ada programming Language", ANSI / MIL-STD-1815-1983
- [5] 许卓群等 《数据结构》, 高等教育出版社
- [6] 王广芳等 《数据结构》, 湖南科学技术出版社

386 微机上位内存 的利用

安徽安庆石化总厂计算机站 黄立山

上位内存 (UPPER MEMORY), 是指 640-1024KB 之间的存贮空间, 这部分存贮空间是保留给显示和系统 ROM 用的, 其中未用的存贮空间, 用户程序也不能进入, 但在 386 微机上这部分内存资源也是可以利用的。COMPAQ 公司的 MSDOS SUPPORT 盘提供的一个实用程序 RUNHI.EXE, 就具有把 TSR(终了——停留——驻留内存) 程序装入上位内存的功能。RUNHI 程序不仅能将 DOS 的命令处理、设备驱动程序装入上位内存区, 还能将汉字操作系统的 CCCC.COM 等常驻内存模块装入上位内存区, 使启动汉字操作系统后, 基本内存的自由空间也能达到 580KB 以上, 与英文系统的用户空间基本一样, 使原来只能在英文条件下运行的大型程序现在也可以在汉字环境中正常运行, 不会再出现“内存不足”的警告。

RUNHI 程序需要 CEMM.EXE 驱动程序的支持才能运行, 先将以下语句写入 CONFIG 文件中:

```

    DEVICE=CEMM.EXE  RAM  NOEMS
FRAME=NONE  [I=MMMM-NNNN]

```

其中: RAM 是指定 CEMM 程序建在上位内存;
NOEMS 是不设置扩充内存(EMS);

FRAME=NONE 是指定不建立 EMS 页框:

I=MMMM-NNNN 是指上位内存中未使用的存贮块,如 VGA 彩色显示器不使用 B000-B7FF 单色文本缓冲区,设置 **I=B000-B7FF**,可释放 32KB 上位内存空间,如果还有别的未用碎块空间,“**I=**”的参数可以多次使用。

COMPAQ 机上位内存缺省空间为 96KB (从 C800-DFFF), 如果再释放单色文本缓冲区 (I=B000-B7FF), 并禁止 EMS (NOEMS), 不设页框 (FRAME=NONE), 则上位内存空间可增加到 180KB 左右, 相当于为基本内存扩展 15-30% 的运行空间。

设置了 CEMM 驱动程序以后, 就可以运行 RUNHI 程序, 命令的基本格式如下:

RUNHI 〈程序名〉

例: 将 CCDOS 2.13 系统模块装入上位内存。

RUNHI FILE3 D2

RUNHI CCCC

RUNHI CV26

也可以在 CONFIG.SYS 文件中使用 RUNHIMEM.sys 命令。

例:DEVICE = RUNHI.EXE VDISK.SYS 1024
512 12 8 / E

即将 VDISK.SYS 虚拟盘驱动程序装入上位内存。

用 RUNHI/SHOWALL 命令,可以显示驻留程序在内存的地址分布情况及可用空间。用 RUNHI/?命令,可显示帮助信息。

RUNHI 程序不仅适用于 COMPAQ 微机和 PC DOS 3.31 环境, 而且不受 DOS 级别和机型的限制, 也可应用于 DOS 4.0, DOS 5.0 等操作系统, 也可在其它型号的 386 微机上使用。

RUNHI 程序, 增强了 DOS3.3 的内存管理功能, 虽然没有利用 386 微机的全部内存资源, 但在实模式条件下的内存资源, 几乎完全利用起来了, 因此 RUNHI 是一个很有效的内存管理实用程序。 (142)

卅 卅

维普公司推出

支持多种应用软件的字形卡

北京维普电脑科技有限公司日前推出新型高精度矢量汉字字形卡。该卡拥有四十多种高精度的 256×256 简繁体矢量汉字，字形规范美观，每种字体都有空心字，立体字，旋转等多种特殊效果变化。该字形卡通用性强，字模接口公开，字库还原速度快。支持多种汉字系统及应用软件（如 CCED、WPS 等），并支持多种针式、喷墨和激光（包括 HP4 600DPI）打印机，能够打印出品质精美的文件。该卡字模接口公开，可供排版，字幕机等系统的二次开发使用。尤其是该卡具有与台湾用户面最广的华康字形卡相兼容的接口，支持华康字形卡的台湾软件将很容易改为简体版本，进入大陆市场，大陆软件也可与该卡配合进入台湾市场。维普公司正在与更多的软件开发商合作，以改变目前计算机汉字软件字库资源不能共享的不足。维普公司在北京市海淀区知春路 76 号翠宫饭店 201 室设有办事处。

联系电话: 2564422-201

联系人: 朱陶

143

多媒体微机技术概述

广州南方电子技术工程公司 徐超汉

人类获得信息的最有效最重要的形式是图象,它是周围的景物在人类眼睛视网膜上的映象。人类最方便的信息交流的方式是“说”与“听”。计算机是通过键盘与显示器以字符形式与使用者交流信息,而这一种交流方式显而易见是非常单调呆板的方式。解决人与计算机之间信息交流的最完善的方法是使计算机具有人类的智能水平,即具有人类的视觉、听觉与说话的能力,如何解决之?这是多媒体计算机技术需要回答的问题。

多媒体技术把电视式的视听信息传播能力与计算机交互控制功能相结合,创造出集图文、声像于一体的新型信息处理模型,使计算机多媒体化,具有数字式全动态、全屏幕播放、编辑和创作多媒体信息功能;具有控制和传播多媒体电子邮件、电视会议等视频技术新范围应用;具有计算机与家用电器一体化的多种功能扩展,这就是它的目的。

多媒体技术所涉及的面很广,它是一项综合技术。本文仅从多媒体技术对微机体系结构的影响;交互式多媒体技术;多媒体数据压缩技术等方面来论述。

一、多媒体技术对计算机体系结构的影响

多媒体技术需要处理声音和图象信息,这类信息处理的特点是数据量大,而且要求具有实时性。实时图象和声音的处理需要有高速处理器,宽带数据传输装置、大容量内存和外存等一系列的硬件环境的支持,这与目前微计算机所能提供的硬件环境产生了相当大的矛盾。为了扩大外存容量需要采用 CD-ROM 光盘存储器;解决矛盾最大的 CPU/内存与外存之间传输频带宽度的方法之一是加宽系统总线的频带宽度。另一种办法是增强数据处理能力和增加附加电路板来专门处理系统 I/O 接口,这样需要应用数字信号处理器 DSP 和专门的 VLSI 芯片。解决传输频带宽度问题的更为通用和彻底的办法是对图象和声音进行压缩编码和解编码技术。

从软件来看,多媒体技术除了需要多任务实时操作系统和窗口系统的支持外,还需要有一个软件

环境来统一管理多媒体信息,同时,这个软件环境要便于用户使用,这就需要发展著作工具语言。此外,为了便于各媒体之间交换信息,需要制订多媒体信息文件的标准,建立接口协议等。

二、交互式多媒体技术

交互式多媒体技术的被认为 90 年代微机技术的又一次革命。那么,什么是交互式多媒体技术呢?根据 Lippincott 和 Robinson 在 1990 年 2 月份的 Byte 杂志上给出的不太严格的定义,概括起来就是计算机同多种信息媒体交互式地综合。例如计算机同显示屏幕的运动图象、视频光盘、CD-ROM、话语和音响等多种媒体建立逻辑连接,从而使整个系统具有交互性。计算机如果渗透到更广大的用户中去,必须建立起人机接口的多种媒介,信息的输入输出除采用标准的数据形式外,还需采用图形、图象及声音等形式,即采用彩电、音响、录像机和摄像机、电话等等,使人机交流合为一体。目前国际市场上具有代表性的多媒体产品有 Pro-750,该系列产品被称为 DVI (Digital video interactive) 与荷兰飞利浦等公司推出的 CD-I (Compact disk-interactive)。

1989 年,Intel 公司与 IBM 公司一起在国际上推出的 DVI 技术第一代产品 Pro-750,它有二个型号: DVI 开发系统; DVI 用户系统。

DVI 的开发系统和用户系统的硬件配置如图 1 所示。图 1 的中间部分是 DVI 的用户系统,它由三块专用的 DVI 接口板的个人计算机及相应的驱动软件组成。三块专用的 DVI 接口板是 DVI 视频板、DVI 音响板以及 DVI 多功能接口板。IBM PC/AT 或其兼容的微计算机作为工作平台,同时配有半高的 CD-ROM 驱动器,带有放大器和音响效果的 RGB 彩色监视器以及键盘成其它输入设备。DVI 的开发系统是在用户系统的基础上再配备与多媒体有关的视频信号数字化器(连到 DVI 视频板上)、音响数字化器(连到 DVI 音响板上),扩展的视频 RAM(连到 DVI 视频板上)、大容量光盘

(或硬盘)、磁带机、录像机、音响设备、监视器以及摄像机或扫描仪等外部设备组成。

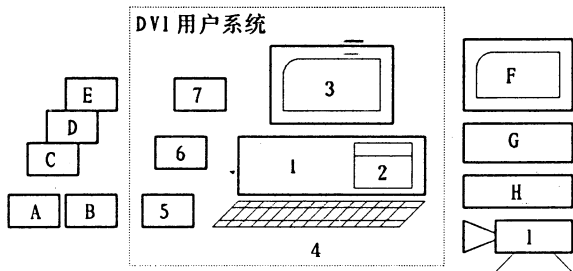


图1 DVI系统硬件配置图

A 和 CD-ROM 相似大容量的硬盘

B 半高的磁带机

C 视频数字化器

(连接到视频板)

D 音响数字化器

(连接到音响板)

E 扩充的视频 R
(冻结到视频框)

连接视频线
F 单色监视器

F 单色温枕器

G 录像机

H 音响设备

I 摄像机或扫描仪

— 22990 —

1.pro750 或 IBMPC / AT 兼容机

2.半高 CD-ROM 驱动器

3. 帶有放大器和音響效果的

RGB 彩色监视器

4. 键盘或其他的输入设备

5.DVI 视频板

6 DVI 音响板

7.DVI 多功能板

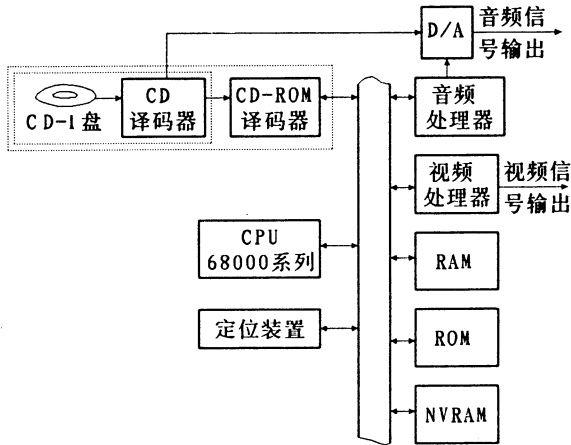


图2 CD-1 基本系统结构

CD-I 系统是交互式声音 / 图象 / 计算机多媒体的一种产品, 作为世界标准推向市场。该系统采用 5 寸 650MB 只读光盘 (CD-ROM) 将声、文、图、动画、静态画面和全运动屏幕等大量信息以压缩形式存贮在光盘上, 其压缩比大约 10:1。

CD-I 的基本系统结构如图 2 所示, 整个系统也称为 CD—译码器。CD 驱动器可以使用 CD-I 光盘, 也可用 CD-DA 光盘。驱动器本身的控制功能包括暂停、连接、停止、弹出盘片等。使用高度智能化的 CD-ROM 驱动器可使 CD-I 系统的性能进一步提高。驱动器连到由音频信号处理器、视频信号处理器、微处理器等组成的多媒体控制器 MMC (Multimedia controller)。

MMC 在光盘实时操作系统的管理下,去编译来自光盘的音频、视频、程序等数据,并把声音和图象分别送到音响设备和不同制式的彩电如 NTSC 制式电视机或 PAL 制式电视机,送到计算机的显示器。用户通过使用鼠标器、操纵杆一类定位装置移动显示屏上的游标,向 CD-I 系统发指令,实现交互性能。

三、多媒体数据压缩技术

多媒体计算机系统技术是面向三维图形, 立体声和彩色全屏幕运动画面的处理技术。为了达到令人满意的视频画面质量和听觉效果, 必须对视频信号和声音信号做到实时或准实时处理。实现实时或准实时处理技术的首要问题是如何解决计算机系统对庞大的视频和语言信号数据流的吞吐、传输和存贮。因此, 高效实时地压缩图象、声音等信号的数据量是多媒体计算机系统不可回避的技术关键问题, 它直接影响到多媒体计算机系统能否推广应用的问题。

值得庆幸的是数据压缩是可以实现的。事实上，数据压缩技术即压缩编码技术已经有几十年的研究历史，内容十分丰富，应用非常广泛。基于不同的应用场合产生了不同思路和技术途径的编码方法。

主要的压缩编码方法有如下几种:

• 电视编码

电视图象编码技术有模拟图象编码和数字图象编码二种途径。

• 变换编码

变换编码不是直接对空域图象信号编码，而是首先将定域图象信号映射变换到另一个正交矢量空间（变换域）产生一组变换系数，然后对这些系数量化、编码、传输。实践证明无论对单色图象、彩色图象、静止图象还是运动图象，变换编码是非常有效的方法。正交变换的种类甚多，有傅里叶变换、沃尔什变换、哈尔变换、斜变换、余弦变换、

正弦变换、还有具备统计特性的 K-L 变换。

• 混合编码

混合编码方法是综合 DPCM 编码和交换编码二种方法的优点而产生的一种编码方法。这种编码方法有计算量适度，抗干扰能力强，并能得到较大压缩比等优点。

• 矢量量化

矢量量化编码是近年来图象、语音信号编码技术中颇为流行的一种新型量化编码方法。它是一种限失真编码方法。对于经过映射变换后的数据分成许多小组，每组 K 个数可构成一个 K 维矢量，然后以矢量为单元，逐个矢量进行量化称矢量量化，其编码解码框图如图 3。

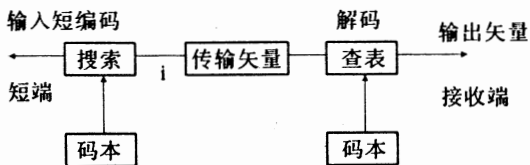


图3

• 子带编码

子带编码是一个实现高质量压缩编码的新领域，是一种在高压比下，信噪比最优的高质量编码方法。这个方法是把原始输入图象分裂成不同频段的子频带，对不同频段的子频带各自采用独立的预测编码方法。对于各子带，一般是用比全频带编码更加简单的预测器。

四、多媒体 PC 机

多媒体个人计算机 MPC (Multimedia PC) 是指融合高质量的图形、立体声、动画等多种媒体且有大量存储 CD-ROM 并由个人计算机控制的软硬件组合系统。它是特指微软公司的多媒体 PC 机，而 mpc (multimedia PC) 应是泛指的。

不是任何一种 PC 机都能作 MPC 的。最低的规格要求为下：

- CPU 80386—20MHZ (即 386SX)
- RAM 2MB
- 视频输出 VGA
- 软驱 1.44MB
- 硬驱 30MB

- 光盘规格 CD-ROM
- 声音规格 8 位 / 样本，音乐合成
- Windows 3.0 (多媒体版)

目前，已有各种硬件制造厂提供多种完整的多媒体 PC 系统，或者提供多媒体 PC 升级套件，许多公司也编写了大量视听软件及视听材料。具有代表性的多媒体 PC 系统有 IBM PS / 2 / ultimedia Model M57 SLC 和 Tandy 4033 LX Multimedia。

IBM PS / 2 / Ultimedia Model M57 SLC 系统的主要组成如下：

- CPU 386SLC
- RAM 4MB
- 视频输出 XGA
- 软驱 2×1.44MB
- 硬驱 80MB
- 光盘规格 CD-ROM XA
- 声音规格 16 位 / 样本，ADPCM 编码
- Windows 3.0 (多媒体版) OS / 22.0

Tandy4033 LX Multimedia 系统主要是由下述几部分组成：

- 386 处理器，运行频率是 33MHZ
- 4MB RAM
- 105MB 硬盘驱动器
- 二个半高 5.25 寸和一个 3.5 寸高密度软盘驱动器

驱动器

- CD-ROM 驱动器
- VGA 卡
- Creative Labs 公司的 Sound Blaster Pro 套件

中的声音卡

- Windows (多媒体版)

五、结束语

多媒体技术的出现将赋予计算机和电视机以新含意。当然，这里的电视机是泛指传统的信息传播媒介。所以，多媒体技术的发展可能是不拘一格多种多样的。从技术本身的发展来看，多媒体技术必然要走全数字化的道路，因为只有这样才能真正对多媒体信息进行交互控制，才能在多媒体信息之间建立逻辑联系。当前，全数字化的代表是 DVI 技术，由于 DVI 采用开放系统的方式，所以，它可以被移植到各种计算机系统的平台上去。90 年代中，DVI 技术将不断地完善和扩大应用的范围。

多媒体微机技术正在迅速发展，预计到 94 年大多数主流微机将用多媒体技术提供机内的计算机使用培训服务。多媒体微机系统的销售额将达到 100—170 亿美元。

高级 UNIX 连网技术讲座

广州昂立自动化工程公司 冯家宁

广州东山金城宾馆曙前楼 551 室 电话:7754888--3511

第一讲 UUCP(UNIX to UNIX Copy) (中)

回到例子上, Systems 文件的条目使用 ACU, 在拨号, 远端(compunerd)回答之后, 等待 in:(来自 login:). 然后发 uucp. 这就完成了 Systems 文件的注册过程, 并且 uucico 开始与远端的 uucico 对话. 系统也许不需要口令给 uucp 注册. 但是 uucico 是个注册的外壳, 是很难(不是不可能)受损害的.

回到本节的主题, UUCP 对设备来说是很灵活的. 在例子中 Systems 条目表明系统连接 compunerd 的次序. 这样, 连接 compunerd 的最先途径是通过 Internet, 它由设备字段的 TCP 说明. 注意有一个子字段(在这这是 e)是用于选择协议用的. 协议在另一节里加以讨论. 如果 Internet 的路径失败, 那么一下条目就会建议使用 Develcon 数据交换机. 这是此例中的次高速连接(9600 波特). 但时间字段只允许在周末或星期一至星期五下午 5 点到次日上午 9 点使用连接. 如果 compunerd 的数据交换机在工作时间里失败. 下一条目就是电话拨号了, 它没有时间上的限制. 这通常是连接一个 modem 或一组 UUCP 专用 modem 的电话号码. 但如果该号码忙音, 下一条目则是下一个号码. 但它只能在周末或工作日下午 6 点到 11 点间使用. 数据交换机的用户也许能在家拨通它, 而管理员却不让这样做.

文件一个叠一个, 看起来也许有点糊涂. 在本节的结尾, 我们用 ACU 条目作例子说明一下连接过程的大概.

首先检查 Systems 文件以决定远端机的条目. 由于 Internet 关闭, 又是在一个星期一的中午, 所以 ACU 条目被选中. 再从 Devices 文件中找到 ACU 的条目. 条目说明第一个选择是连接到 /dev/ttyl4 的 Penril 自动拨号机. Dialers 文件指示程序怎样通过 Systems 文件中规定的拨号与 penril 通信. 但是该拨号是首先经过在文件 Dialcodes 把字符串 boston 转变为 91617 而得到的. 在这里 9 代表拨通外线, 1 代表选定的长途电话公司, 617 则是 Boston 区号.

由于象 cu 或 tip 等其它程序也可使用与 UUCP 同类的信息, UUCP 内部调用函数可汇编成 STANDALONE 并包含在一个库子程序里给其它命令使用.

1.5 协议各层, 出错检测和恢复

UUCP 连接涉及两层通信协议. 高层的是 UUCP

协议. 它负责交换系统标识、发送序列数字并检查标志、与低层协议对话、建立文件传送、协议主/从角色的转换、断接手续. 在低层协议同意之前和在挂机命令发出之后, UUCP 使用一个简单的错误检测技术. 每一条消息都以一个同步字符(ΛP)打头并以空字符(Λ@)或换行符(ΛJ)结尾. 在这不做恢复处理. 一旦低层协议建立, UUCP 协议才用它做错误检测和恢复.

可选低层协议是 UUCP 的漂亮特征之一. 它可以多种不同媒质通信. 对选定的媒质选用适当的协议是很重要的. 当时在贝尔实验室的 Grey Chesson. 就为 UUCP 写了第一个协议 Packet Driver(PK)或 g 协议. 在此协议里数据由加有顺序号的帧表示. 发送者可在具有序列数字的一定窗口内发送帧. 接收者在检查核对和之后接收这些帧, 窗口随着数据向前移动. 拒绝接收或丢失确认信号可引起重发. g 协议对于用电话线以高至 9600 波特传送是很有效的. 在 UUCP 各版本中都有 g 协议或兼容的协议. 其它共同的协议包括排错或 c 协议. 这只是简单地发送一个计数字节和数据. 这些协议也会被作为通路的如 Internet 用到, 并与其自身的纠错技术相协调. DataKit(d), X.25(x) 和 Telnet(f) 中有这些协议. 在 TCP(t) 中有另一协议. 很多协议被设计作特殊之用. 在实际上, 安装在 UUCP 上协议的数量是任意数(也许是 20), 是一个协议数组的维数.

有两项输入用作选择协议. 第一项是在文件 Systems 中用逗号隔开的设备字段的子字段. 如在例子中机器名为 compunerd 有条目 TCP, e. e 表示所选的协议. 这可能是远端机 compunerd 上的 UUCP 也用 e 协议. 所有这些, 在 uucico 启动时就定下来了. 在呼叫机发送自己的名字后, 远端机用一张有效协议表(如 getf)作回答. 然后本地机根据可用性和优先性选择其一. 如果优先性在 Systems 文件里没有指明, 那么第一个有效的协议便被选中.

这些协议汇编进 UUCP 就象设备驱动程序汇编进 UNIX 内核一样. 有一个象 cdevsw[] 的表, 列有协议名(一个字符, 如 g)和给激活、读写报文或数据、关闭等用的入口点(函数名)子程序. 安装新的协议需要在表里增加新的条目, 提供指定的函数并重新连接 UUCP.

1.6 实现的规范(UNIX 与非 UNIX)

从字面上看, UUCP 是 UNIX 的工具程序; 但它已被

其它操作系统所采用。目前由 Berkeley 发行的版本可以有条件地汇编到 VMS/EUNICE 里。其他人(出名的有 Lauren Weinstein)已开发出在 PC 机的 DOS 上运行的 UUCP 兼容程序。我们这里要讨论 HoneyDanBer 的实现规范(它是接近标准的)。有很多参数在安装或调试 UUCP 或写与 UUCP 工作有关的程序时是必须考虑的。

UUCP 系统的细节是复杂的,因为有那么多的参数在安装甚至运行时要设置。通常要改变的项目都出现在 `parms.h` 的常量里(在 Berkeley 版是 `uucp.h`)。它们是:

AT&T Systems n.4.n BSD, n.4.n Edition(Vn), 等的 UNIX 版本

在不同的 UNIX 版本之间有很多很小(也有很大)的不同。UUCP 需要知道自己的运行环境才能去做合适的事。与系统位置和命名有关的一些主要的不兼容之处包括有关文件和现有标准的 C 库子程序。如果一些函数存在于三个 UNIX 版本,它们的 C 库子程序和系统调用在语法及一些语义上也会有所不同。虽然你可以在运行时确定操作系统的版本,使用条件语句选择正确的逻辑,但这时对编码、调试或处理来说都是不够的。

UUCP 的 Uid 和 Gid

虽然 UUCP 程序一般为由 uucp 注册的用户所有,但也有某种巧合,用户标识的设置程序同时被另一些设标志程序或根所调用。保守的办法是调用 `setuid(UUCP-UID)` 以防在某些情况下有假的用户标识。

时间的分解度睡眠

一个非常依赖 UNIX 版本的特征是时钟的分解度。有几种做法是让程序员用 `sleep()` 把一秒钟内的时间片拼凑在一起。这些涉及特别的设备驱动程序、`nap()` 系统调用、等待繁忙的循环甚至以给定的波特向一条 tty 线写数据。

磁盘工具程序

老版本 UUCP 的最大问题之一是当拷贝文件时遇到“空间不够”或“i 节点不够”时。`write()` 系统调用几乎不作检测,这样文件似乎被拷贝了而其实没有。现在的版本不仅检测是否有足够的空间可用。检测是节省时间和资源的并且能给出适当的诊断信息,如“远端没有足够的空间”而不是不太明确的“写出错”。系统间可以交换信息以进一步查明“写出错”的原因。随着系统数目的增长使得提供有意义的诊断成为必要。

最有效地检测文件系统的方法是通过可用的系统调用。其次,UUCP 安装时,可以指定一个程序替代 `ustat()` 系统调用。它以主/次设备作参数并返回自由的块和节点数。这样的程序可以用 shell 的 `df` 来写。

半公用或受限制的系统文件

为了调试,两系统间交换的信息可以向需要这些信息的用户报告。这些交换(特别是在注册开始时的对话)

也许包含不适宜公开的敏感信息。甚至系统的电话号码也是私有的,更不用说 uucp 的注册名和口令了。事实上电话号码也许是系统里存有的最敏感的信息了。确保隐私的一个办法是即使在调试时也不输出这些数据,但这对调试者来说并不是个好办法。只允许根用户得到调试信息也许是个办法。因此,可以定义一定范围的组,在这些组里的用户是可以得到调试信息的。

机器名的动态确定

每一个 UNIX 版本都有独特的方法确认其本身。`uname()` 和 `gethostname()` 系统调用和 `cat/etc/whoami` 就是可用的一切。奇怪的是许多 UNIX 机器都命名为 `unix` 或 `erenhon` 或 `kilroy`。

重试连接前的等待时间

Honey DanBer 用一个指数放弃算法决定两次联络远端的最短周期。每尝试一次,等待就加倍时间,直到一个最大值(常数)。这就使得在联络一个关闭的系统时占用拒绝使用的设备的时间最小。系统不可联络的时间越长,就越有可能继续不可联络。

远程执行命令时的环境 PATH

设置 PATH 变量是一个简单而有效的方法去限制远端的权力。这提供了一种机制限制直接使用路径名。Permissions 文件支持用灵活的办法决定一个或一组地点的远程存取权限。它取代了原来 USERFILE 的概念,它也是用于限制文件存取和注册的。Permission 文件允许使用主机并且它更为明确些。另一个显著之处是远端机可以隐含执行一给命令。多数地点允许执行 `rmail`,它是一个受限制的邮件命令,特别用于接收远端邮件。

非认识系统与主机的通信

为拒绝没有有效回邮地址的邮件,NOSTRANGERS 使 UUCP 拒绝 Systems 文件中没有的地方的任何联系。当远端注册之后,系统会回送报文“我不认识你”。

几个 uuxpt 及 uusched 进程同时运行

两个文件可以动态地限制执行中的进程数(如在高峰期)。uusched 用于执行 uucico。它随机地选择 uucico 被每一系统所调用的次序。

加锁文件所在目录与格式

文件加锁用于防止对于远端多路同时存取,以适应某些设备的互斥性。设备加锁被其它程序(`cu`, `tip`)使用,并且文件应在 `/usr/spool/uucp` 更为公用的位置上。子目录名在安装时指定。因为其它程序也可能用到加锁文件上的信息(加锁进程的标识),其格式也是参数化的。

1.7 用户介面(命令、调用和文件)

uucp

参数:

* -c 发送一个缓冲目录上的文件。默认是指定的实

实际文件

- * -f 不建立子目录。默认是建立任何需要的子目录
如: uucp 文件 remote!~/dir/文件

将建立子目录~/dir(如果它不存在)。-f 参数阻止建立。(~/表示远端系统 UUCP 的公共缓冲目录通常为 /usr/spool/uucppublic。默认任何系统可在其上写文件。

- * -g<级别>是一个字符,它为转送者指定相对优先级。ASCII 码值越小,优先级越高。

-j 使一任务标识打印在标准输出上。此标识可用于查询请求状态或取消请求。

- * -m 使当请求完成时发送邮件给用户。
- * -n<用户> 使当请求完成时通知远端用户。
- * -r 使当请求被缓冲时阻止 UUCP 调用 uucico。
- * -s<文件> 把转送意图的状态写入文件。
- * -x<N> 设置调试为级别 N。如果没有 N,那么输出最少的调试信息。N 范围从 0 到 9。

uux

参数:

- * -u 用 uux 的标准输入作为命令串的标准输入。
- * -a<名字> 用这里规定的名字而不是 uux 实际调用者的名字发通知。这在 uux 被用户标识设置程序调用时是有用的,真正的用户名被保留(如,mail)。
- * -b 以非零状态退出,并把命令的输入连同告示发送给用户。
- * -c 拷贝文件到缓冲目录而不是实际文件所默认的目录。
- * -g<级别> 设置单字符的传送优先级。较少的 ASCII 字符设置较高的级别。
- * -j 输入请求标识。标识可用于查询状态或从队列中取消请求。
- * -n 如命令失败不通知用户。
- * -r 请求被缓冲后阻止 uux 调用 uucico。
- * -s<文件> 把转送意图的状态写入文件。
- * -x<N> 把调试输出的级别设为 N。如不给出 N,则设为最小值 1。N 的范围是 0 到 9。
- * -z 如命令成功通知用户。

uucico

uucico 通常并不被直接执行,除非调试时。它是经过缓冲的 UUCP 程序请求和 usched 调用到的 UUCP 的主要功能部件。后面会给出 uucico 操作的对话例子。

参数:

- * -d<目录> 用所指目录作缓冲目录而不是默认的(通常为 /usr/spool/uucp)。
- * -r<角色> 设置 uucico 以主机(角色=1)或从机(

角色=0)开始。如以主机开始则远端系统必须用 -s 指定。如以从机开始, uucico 要求被连接到主机作为标准输入输出。

- * -s<系统> 是远端的系统名字,此系统作为远端 uucico 的主机。如机器为从机则此参数无效。
- * -x<N> 指定在运行时用 -xN 会话决定调试诊断的存在与范围。数字 N 越大,会话越多。

uuxqt

uuxqt 也是被其它命令(uux 或 uucico 放弃与远端连接后)所调用的(调试除外)。它从缓冲目录查找执行文件,并执行指定的文件(见后面关于执行文件的描述)。它有赖于文件 Permissions 和一些已经提到过的默认来决定什么命令会被远程系统执行。

参数:

- * -s<系统> 只为所指定的系统执行文件。
- * -x<N> 为调试提供诊断输出。

uulog

UUCP 维护日志文件,该文件记录了每次通信的开始和随后的事件(传送及与本地或远程命令执行有关的)。这些日志文件驻留在 UUCP 缓冲目录的 .Log 子目录里。子目录之下对每一 uux, uucp, uucico 和 uuxqt 又有一个子目录。在这些子目录里每一相关的远程系统有一个文件。如果不给参数 uulog 将输出所有日志文件。-x 参数使 uulog 将输出 uuxqt 的日志文件。-s 参数则指定一个特别系统。可以同时带几个 -s 参数。-f 参数使 uulog 的结果象 tail -f 那样输出系统日志文件,这些文件先由命令 -s<系统> 给出。

uuname

uuname 输出在 Systems 文件中所列的所有地方。通常此文件对用户是不可读的,但 uuname 为 UUCP 注册管理设置用户标识。-l 参数使 uucico 只输出本地系统名。

uustat

uustat 是监视 UUCP 队列、连接和活动的通用程序,也可用于注销请求。如果不加参数,所有的用户请求将被打印出来。

参数:

- * -a 列出所有的请求。
- * -m 报告最近向每一地点存取的状态和请求的总结。
- * -p 为与每一锁定文件和锁定目录有关的进程 ID 执行一个 ps。
- * -q 为每一机器列出请求队列、控制和执行文件的数量、它们的存在时间、连接状态、失败存取次数。
- * -r<任务号> “触摸”与任务号相关的文件,使它们变新而不至于过期。

- * -k <任务号> 从队排中取消任务号的请求。
- * -s <系统> 为系统报告所有请求状态。
- * -u <用户> 报告用户发出的所有请求状态。

uusched

如前所述,uusched 为每一工作的系统调用 uucico。uucico 的次序是随机的。-x 选择单数字参数以从 uusched 产生调试输出。-u 用于传递一个 -x 给 uucico。

uucheck

uucheck 是一个管理工具。它检查必要的 UUCP 系统文件和目录是否存在。如加 -v 参数,它也将产生一个 Permissions 文件的解释。在后面将描述 Permissions 文件。它为远端系统的命令和文件取得存取许可。-x 选择用于调试 uucheck。

uucleanup

在中途撤消文件传送或通信的出错会产生一些作废的文件。uucleanup 用于自动清理这些文件。程序在缓冲子目录寻找此命令行参数所规定或默认要老的文件。命令和数据文件的默认是 7 天,执行和未知类型文件为 2 天。出错会送一些消息给用户告诉他们的请求还未执行或警告已有存在(默认)一天的老文件。参数有:-C<天数>,-D<天数>,-X<天数>和-o<天数>。这分别用于决定在多少天后删除命令文件、数据文件、执行文件及其它文件。-W<天数>指定多少天后给用户发送警告。-m<字符串>指定警告消息。消息里头最好包括 UUCP 管理员的名字。-s 则告诉 uucleanup 系统的缓冲子目录。

uucleanup 起初是作为一个 shell 命令。后来经过代码开发,它位于缓冲子目录并处理该子目录里的文件,起着 UUCP 管理员的作用。它会检查文件以确定数据类型并加以标识。例如一个邮件信息里有收发者,是否转发,找不到接收者时是否退回原处。网络新闻在 UUCP 活动中占有较大比重,代码会识别网络新闻数据并在适当时发送它们。uucleanup 会发送慰问邮件给用户并签名“真诚的 uucp”。

1.8 周期性驻留监督程序

以下四个 shell 程序通常被 cron 周期地调用。

uudemon.admin

此程序送状态报告给 UUCP 管理员。

uudemon.cleau

此程序合并一些日志文件,清除过旧的文件。(最后一个字母 p 对某些 UNIX 版本来说会造成名字太长)。

uudemon.hour

顾名思义,此程序是每小时运行一次的。它通过 uusched 调用 uucico。

uudemon.poll

UUCP 可以联络被动系统,以使在那里排好队的任

务可以动作,就象是被动系统发起联络一样。这种按照安排好的、有规则的联络叫做轮寻。在 crontab 中放入 uucico 命令,可以有效地执行对少数地点的稀少轮寻。但对于象一天几次那样中等程度的轮寻,就要由 uudemon.poll 帮助管理工作了。每一 crontab 条目调用此程序,它使用轮寻文件(以后会讨论)向被动系统发出必要的呼叫。

1.9 其它程序

有几个程序是独立于 UUCP,但却是对其有用的。

uuencode 和 uudecode

这对程序使用户能把二进制文件编码为 ASCII 文件。这样文件就能以任意路径邮寄到另一用户,再由他解码把文件还原为原有的格式。在 uucp 执行跨越几台机器的请求时,该程序是有用的。uucp 现在可以通过几台机器转发数据了,就象邮件的转发一样。例如:

```
uucp myfile sys1!sys2!sys3!yourfile
```

在语法上是可以接受的。它产生一个 uux 命令在中间系统执行 uucp。这样,转发并非一个特殊情形,只是用一般远程执行功能而已。uuencode 和 uudecode 在非透明通信线上送二进制数据仍是有用的。

uuto 和 uupick

这对程序用于在用 uucp 命令转发整个目录树时,简化其工作。uuto 不仅发送文件(及目录的内容)而且也在邮件中告诉接收者怎样收集文件。文件被拷贝到公共的缓冲目录里。接收者执行 uupick 可以列出所收到的目录和文件。然后用户指定把它们存到哪,uupick 就把它们从缓冲目录里移到哪。

1.10 相关文件

以下文件位于 UUCP 的库目录里(典型情况下为 /usr/lib/uucp)。

Systems(L.Sys)

我们已经描述过 Systems 文件的许多细节。它列出了 UUCP 可以联络的系统以及用以联络的设备。在编译时间里,如 NOSTRANGERS 有定义,它也可以决定存取本主机的具体地点。它可以指定在一日之时,一周之日的时时间使用哪些设备存取具体系统,什么协议用于给定的设备。多条目的系统里可以指定不同的设备,电话号码等。一个条目最多只能对应一条线路。以(井)打头的行是注释行。条目中的字段用空格等或制表符分开。它们是:

系统	时间	设备	级别	号码	注册
----	----	----	----	----	----

系统就是远程系统的名字。在较老的 UUCP 版本里,系统名字只限 7 个字符。

时间则规定本条目所用的一天内的时间和一周内的星期几。星期几可以用头两个字符作缩写。第一个字母要大写。时间范围用 24 小时格式(两串 4 位数字中间用连字号连接)。它们跟在星期字符串之后。如不写出时间

范围,则假定为 0000-2359。任意的星期几/钟头范围在一条目中可用逗号表示。最后一子字段是连接远程系统失败后到下一次重试前的最小分钟数。它将取代已编译在 uucico 内的默认重试时间。该子字段是由分号隔开的。特别的替代有 Wk 代表 MoTuWeThFr,Any 代表 SaSuWk。时间字段的例子有

```
Any
Sa,Su,Wk 1800-0600
Any;30
```

设备是指连接设备的类型如 TCP, ACU 和 Micom。这些可以是内置的(801, 212, TCP, DK, Sytek, Unetrver)或在文件 Devices 中定义。此字段可以由一字段跟随,以选用协议。子字段用逗号隔开。如 c 协议通常用于 TCP 设备。某些设备字段为:

```
TCP,e
ACU
Develcon
```

级别说明设备的,在使用 tty 线(ACU、直接连接、数据 PBX)情形下,它是线路速度。当级别是不重要(如 TCP 情形)时用连字符(-)表示或用 Any 表示任意速度。

号码是远端机的“地址”:当用拨号设备在电话线上的拨号或数据 PBX 用到的名字或 Internet 的连接列程。在号码为电话号码的情形下,它可以用字母串作前缀以代替 Dialcodes 文件中的号码。如:

```
arizona55558632
```

注册是最后的字段,它用于与远程系统的注册并连接 uucico。注册字段并非总会用上。例如,一个 TCP 连接可以联络远端 uucico 而无需注册。即使要注册,也不一定要口令。注册字段也可用于联络一些远程数据 PBX,如为次级交换机提供名字。注册字段是些只有不同含义的字符串。头一字串是建立连接所等待的,第二字串是要发送的,第三字串是等待的,等等。如:

```
gin uucp rd sipglip
```

是在结到 gin 注册后,发送 uucp。在收到口令 rd 后发送 sipglip。发送的字符串用回车符结尾,除非尾处用 \C 指定。等待字符串可包含选择字符串。选择字符串由连字符引导,如:

```
login—in
```

将等待 login。如在超时时间内收不到(通常为 45 秒,取决于编译时的选择),则发送空字符(以回车符结尾)并等待 in。

这里是一个完整的 System 条目。它与远端两个 Micom 交谈。

```
james Any;20 ACU 1200 sanfrancisco5557330 "" "" ENTER "" ASS
N123\r\d\ d LASS-\r-LASS g\d\d GO \r\r\d
login:-\r-login:
elcor assword:sipglip
```

这样在 Micom 处就会有 ENTER CLASS 的提示。N123 和 g 是 Micom 的资源名字。

Maxuuscheds 和 Maxuqts

这些文件控制着同时执行的 uusched 和 uuxqt 的进程数。数字是最大进程数,用 ASCII 码表示。

Permissions (USERFILE)

Permissions 文件用于控制远端机文件的存取,远程命令请求的执行和注册许可。也有检测机制。Permissions 文件基本上用于修饰每一系统或系统组的默认状态。对远端呼叫和呼叫远端有两种指定方法。在 Permissions 文件中的每一条目里包含有一数字选择,用空格、制表符或续行符(\)所分隔的指派值。空行或以井打头的行将略去。任选关键字有:

LOGNAME 引导一些参数,当远端存取本地主机时,使用这里的注册指定。如:

```
LOGNAME = xuucp:yuucp:zuucp...
```

存取此系统时每一注册都必须在 LOGNAME 条目里提及一次且只提及一次。UUCP 安装程序为所有注册外壳为 uucico 的条目收集口令并创建一个默认的 Permissions 文件。

MACHINE 此关键字提供一组参数,用于远端被呼叫或当 uuxqt 代表一远端机执行命令时。在呼叫或被远端呼叫时,如打算共享参数,LOGNAME 和 MACHINE 关键字可以写在一个条目里。如:

```
LOGNAME = xuucp MACHINE = xunix
```

REQUEST 此关键字与 LOGNAME 或 MACHINE 关键字一起用,以指定远端是否可以请求本地主机发送文件。默认是:

```
REQUEST = no REQUEST = no
```

RAED 请求文件的能力可以被 RAED 关键字进一步限制。这里在任何被请求的文件前加上一串路径名作前缀。默认时 READ 的值是 UUCP 的公共子目录,通常是:

```
READ = /usr/spool/uucppublic
```

NOREAD 这可与 READ 联用以指明例外。如:

```
LOGNAME = juucp REQUEST = yes READ = /usr/joe
NOREAD = /usr/joe/private 允许一台机器以 juucp 注册并
请求 /usr/joe 里的任意文件。但 /usr/joe/private 是
例外。
```

WRITE 这与 READ 是相似的,默认为:

```
WRITE = /usr/spool/uucppublic
```

NOWRITE 这用于修饰 WRITE 条目,正如 NOREAD 修饰 READ 条目。

SENDFILES 此关键字以“yes”或“no”为值,决定当系统呼叫时(当 uucico 反转主/从角色,被叫系统等待工作时),为远端发送的文件是否排队。默认值是:

```
SENDFILES = no
```

这确保文件只有在本地发起呼叫并与远端有相当可靠的通信时才发送。

CALLBACK 这一关键字与 LOGNAME 联用。取

“yes”或“no”(默认)值。它使对话终止和远端机回话。这有助于确认机器标识。

COMMANDS 此关键字与 **MACHINE** 条目联用,以指定一组机器可在本地主机上调用的命令。默认的表在编译时决定,通常在 **rmail** 里。表里可以包含全路径名。当请求的命令没有指定路径名时,路径就取 **COMMANDS** 里的值,它取代了编译时默认的 **PATH**。特殊命令 **ALL** 允许执行 **PATH** 上的所有命令。如:

```
MACHINE=princeton COMMANDS=/usr/local/bin/lp:ALL:
/usr/berbell/bin/gm
```

允许名为 **princeton** 的机器执行路径和 **/usr/local/bin/lp** 及 **/usr/berbell/bin/gm** 上的任意命令。这样对 **lp** 和 **gm** 命令来说,路径名就不必提供。

VALIDATE 本关键字与 **LOGNAME** 条目联用,把机器名和注册名连在一起。条目:

```
LOGNAME=vineyards VALIDATE=yquem 要求
yquem 用 vineyard 注册。
```

MYNAME 基本上用于调试(及伪装),**MYNAME** 是与 **LOGNAME** 或 **MACHINE** 条目连用的,它告诉地机用不同的名字标识自己,这样:

```
LOGNAME=testuucp MYNAME=testunix
```

或

```
MACHINE=testunix MYNAME=debug
```

使本地主机以 **testunix** 为标识,响应以 **testuucp** 和 **debug** 注册的机器。通过这一机制,一台机器可在运行 **UUCP** 的同时呼叫它本身。

PUBDIR 这一最后的关键字用于重新定义 **UUCP** 的公共目录。默认写为:

```
LOGNAME=uucp MACHINE=OTHER
PUBDIR=/usr/spool/uucppublic
```

注意 **OTHER** 的使用。它可用于指定在 **MACHINE** 条目里没有列出的所有机器。但对 **LOGNAME** 却没有等价物,因为所有注册都必须列出。

这里是一个 **Permissions** 文件的例子:

```
#no secret files here
LOGNAME=uucp READ=/ NOREAD=/etc:/usr
/ber\
WRITE=/usr/spool/uucppublic:/tmp\
SENDFILES=yes REQUEST=yes
#these guys need to access the entire directory structure
MACHINE=bellcore:princeton:research\
READ=/ WRITE=/
#make sure these machines use the highly secret login since
#they have wide permissions
LOGNAME=suucp VALIDATE=vortex:ames
MACHINE=vortex:ames COMMANDS=ALL
```

Poll Poll 文件被被动系统(或注重节省开资的系统)的 **uudemon.poll** 用于调度轮寻。格式是每一条目以地点名开始,跟着是制表符,然后是轮寻的时间(0-23),之间以空格隔

开。如:

```
bellcore 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
borealis 0 13 16 20
blia 5
sdl 8 9 10 11 12 13 14 15 16 17 18 19 20 21
```

Contacts

此文件没有标准化,但却应该标准化。它包含有 **systems** 文件里所有地点上 **UUCP** 管理员的清单。每一条目包含有地名、管理员名、办公室电话和注册。这对解决故障问题是有帮助的。

下列 **shell** 程序帮助确保数据库被更新:

```
#!/bin/sh
IFS='
while read site name phone login
do
mail
site!
name < < !
Just checking to make sure you haven't been fired yet.
Please confirm your existence.
!
done
```

remote.unknown

在 **HoneyDanBer** 标准里,**NOSTRANGERS** 设为一个程序名,当远端连接一个 **Systems** 文件上没有的时,该程序就被执行。远端的名字作翁序的参量送出。默认时程序就是 **/usr/lib/uucp/remote.unknown**,它是一个 **shell** 程序,它把事件记到 **/usr/spool/uucp/.Admin/Foreign** 日志文件上。在 **System V R 3.2** 里 **remote.unknown** 改为二进制可执行文件。

以下所有文件位于 **UUCP** 缓冲目录的 **Admin** 子目录里。

audit

当远程系统连接上并且它的 **uucico** 以 **-x** 标志被调用时,本文件连同 **uucico** 的调试一起被输出。本文件通常会被 **uudemon.cleau** 周期性地截去,因为运行 **uucico -x9** 的调试器能很快地倾卸出很多废物在其上。为防止这样,参数 **RMTDEBUG** 可以被设为 **/dev/null** 而不是 **audit**。这并不总是必须的,因为人们很快就会讨厌 **-x9**。

errors

此文件包含记载 **UUCP** 声明的语句文本。

uucleanup

这里记载诊断。

xferstats

在此文件里,对每一文件传送给出一条目、其上有系统、用户、角色(主或从)、时间、过程号 **ID**、该过程所转文件数量、设备、传输方向、传送的字节数及所花的时间。例如:

```
obt2lroot S(10/3-1:42:16)(c,27947,1)[tty00]->49360/449.380=109.8
obt2lroot S(10/3-1:42:24)(c,27947,2)[tty00]->209/1.380=151.4
rutgersdaemon M(10/3-1:45:32)(c,29254,1)[TCP]->457/0.001=457000.0
rutgersdaemon M(10/3-1:45:38)(c,27947,2)[TCP]->260/0.001=260000.0
```

```

ob12root S(10 / 3-1:50:06)(c,27947,3)[tty00]-> 50162 / 456.380 = 109.9
ob12root S(10 / 3-1:50:12)(c,27947,4)[tty00]-> 209 / 1.340 = 155.9
attunixlattunix S(10 / 3-2:08:04)(c,29825,1)[tty02]<-229 / 13.440 = 17.0
attunixlattunix S(10 / 3-2:08:20)(c,29825,2)[tty02]<-152 / 13.080 = 11.6

```

进程号前的 C 表示 uncico 作翁序响应。U 表示 uucp。X 表示 uuxqt。但这些程序都不会写入本文件。

以下是 UUCP 缓冲目录的子目录。

.Old

本子目录包含来自 .Admin 的日志文件的旧的拷贝和 .Log 的合并文件。清理驻留监督程序执行时,会改写此目录的内容。

.Status

本子目录为被存取的每一个选程系统建立一个文件。文件包含有状态代码、诱发次数、“unix 时间”、重试时间、状态的 ASCII 形式、机器名。例如,打入

```

cat /usr/spool/uucp/.Status/ 可以看到:
21 11 591847471 39600 CALLER SCRIPT FAILED alice
00 591862384 0 SUCCESSFUL attunix
40 591818100 0 CONVERSATION FAILED co
21 8 591832080 38400 CALLER SCRIPT FAILED decvax
20 591853847 0 WRONG TIME TO CALL ka9q
6239 591868143 900 LOGIN FAILED lion
160 591868050 39600 NO DEVICES AVAILABLE meritec
00 591677650 0 SUCCESSFUL oakley
62 591867664 900 LOGIN FAILED pyuxc
21 49 591861001 14400 CALLER SCRIPT FAILED star
30 591869693 0 TALKING tness7
19 2 591867065 600 REMOTE REJECT,UNKNOWN MESSAGE
utah-cs
00 591869733 0 SUCCESSFUL yquem

```

..Sequence

每一次控制的执行或数据文件的建立,都会包含有一个标识文件类型的前缀(C,D,X,),表示系统名字、服务级别(可能由-g 选项来指定)、一数字和地点名。数字可能是文件名的唯一部分。Sequence 子目录存有每一个系统最近使用文件的数字。数字是递增的。它们是十六进制数的 ASCII 表示。过去,一个单独的序列数为所有系统的递增。过度使用会导致数字回复(以 10 为基),导致命名重复。对此,一个解决办法是 Allan Watts' 的修改,它是基于 64 个数字的。这会产生一些不明其含义的文件名。在 Honey DanBe 方案中,每一系统分配一个序列号。这一序列号不应与对话序列号有冲突,它主要用于每一端的验证。如果文件 SQFILE 存在于 UUCP 公共目录,它含有如下形式的条目:

name count manth / day-hour:minute

例如:

kremvax 666 4 / 1-00:00

如果一个系统列于此文件,那么对话就被分配数字。它建立连接时,远端和本地端必须在对话序列计数上取得一致,否则连接取消。计数是以 10,000 为模的。这可以减少伪装的可能性和提高伪装被检测的可能性。不过,这很少会

用上。

.Xqtdir

远程请求的命令被放在此目录。

.Workspace

临时内部系统文件在此建立并销毁。

以下三种类型的文件在远程系统的缓冲子目录里由 UUCP 建立。

控制文件(C.)

控制文件包含执行传送的必要信息。它所含的行数与一条 uucp 命令所含的请求数

相当。一条 uux 命令使每一命令串里的相关系统产生一个控制文件。文件有 8 个字段给接收文件用,有 9 个字段给发送用。如:

```

S /usr/ber/.profile ~ /ber/.prafile ber -dc
D.0 644 ber 注意最后的字段是空的。开头 S 表示发送,R
表示接收。其次是源文件,接着是目的地。第四字段是本地
用户,接着是指明是否按所要求建立子目录。文件是传送
正本还是副本。下一个字段是包含有目的地拷贝的数据文
件名。在请求远端系统接收文件的条目里,此字段不出
现。若使用真实源文件(如本例),数据文件字段空下。下两
个字段是源文件模式和在问题出现时要通知的用户名(可选
用-n 选项)。最后是一个可略去的参数,它是报告传送状态
的文件,由-s 提供。

```

执行文件(X.)

作为 uux 命令的结果,这些文件被放在远端机器上。格式有点随意。文件的每行由一个字段开头,指定该行之后的含义。uux 程序实际上产生注释行,它们被插入到执行文件中以协助用户。行的类型有。

井是注释。

F 是一个有必要执行的文件。要到 X 文件存在,此文件才被处理。

I 是标准输入

O 是标准输出。

C 是要执行的命令。

U 是要通知命令执行状态的用户。

B 告诉 uux 当命令失败时把标准输入交还给用户。

M 是要写进执行状态报告的文件名。

R 是邮件的回邮地址(如不是一个用户本身,用-a 选项告诉 uux)。

Z 在命令失败时发通告(退出状态为非零)。

N 即使命令失败也不发通告。

在命令成功时发通告(退出状态为零)。

数据文件(D.)

这些文件没有固定格式;它们是资源数据的拷贝。它们可以由 uucp 用户在选择项-c 建立或由 uux 把一执行文件放到远端系统上。(待续)

Foxbase+

使用扩展内存加速

湖南省人民警察学校微机室 刘永强

我们在 FoxBASE+ 的点状态下输入各种命令时, FoxBASE+ 总要去磁盘上调用其覆盖文件 (如 MFOXPLUS.OVL), 尤其是在软盘上运行 FoxBASE+, 每一个命令输入进去后延了一段时间才执行。另外, 频繁读盘对磁盘寿命也有影响。鉴于现在的机器配有 1M 以上的内存已很普遍, 在这里我们的作法是让 FoxBASE+ 到扩展内存中去读覆盖文件, 其具体方法如下。

一、建立虚拟盘

在 DOS 系统配置文件 CONFIG.SYS 中加入一句:

```
DEVICE=VDISK.SYS 384K /E
```

重新启动机器之后, 系统即得到一个建立在扩展内存上的虚拟盘, 容量为 384 KB。如果启动时系统显示内存不够, 那么首先请检查一下你的机器是否有扩展内存,

即观察机器自检数据是否大于 640 KB。有的机器可将扩展内存划给 EMS, 如为这种情况只须在机器 COMS SETUP 中将 EMS 容量设为 0 即可。注意: 修改 CONFIG.SYS 后必须重新启动机器, 修改才会有效。

虚拟盘的容量可根据机器的配置进行设定, 虚盘盘符由 DOS 根据系统已有逻辑驱动器的个数依次向下分配, 我们在系统启动时显示的信息中可看到。为了叙述方便, 此处我们假设虚盘盘符为 E:, 当前路径为 C:\FOX, 使用的 FoxBASE+ 版本为多用户 2.0 版。

二、覆盖文件改名

建立虚拟盘后, 为了使 FoxBASE+ 在当前目录路径下找不到覆盖文件, 我们有必要将原来的覆盖文件改名。在 DOS 提示符下敲入:

```
C>REN MFOXPLUS.OVL MFOXOVL
```

改名是加速的关键所在。

三、覆盖文件拷入虚盘

这一步的工作是将改名后的覆盖文件以原来的名称拷贝到虚盘上。为方便起见, 可在 DOS 提示符下建立一个批处理文件 OVL.BAT:

```
C>COPY CON:OVL.BAT
COPY MFOXOVL %1\MFOXPLUS.OVL
```

```
^Z (按一下 F6 键即得)
```

再回车存盘即可。以后只要敲入 OVL E: 回车, 即可自动完成上述拷贝工作。

四、指定覆盖文件路径

为了使 FoxBASE+ 自动到虚盘上去读覆盖文件, 我们需要指定寻找路径。为此应用 3.3 版以上的 DOS 提供的一个外部命令 APPEND。该命令类似于 PATH, 但 PATH 只指定可执行文件的寻找路径, 而 APPEND 用来

指定非执行文件。具体作法是在 DOS 提示符下输入:

```
C>APPEND E:
```

回车即可。注意当前目录下要有文件 APPEND.EXE, 或用 PATH 指定该文件的目录路径。

此时执行 FoxBASE+, 就可以自动在虚拟盘上读取覆盖文件了。

以后每次使用时, 只须重复 3、4 两步。

如果不使用 APPEND, 则 FoxBASE+ 执行时会因找不到覆盖文件而出现一个不优雅的中断: 提示你输入覆盖文件所在的盘符路径。此时敲入 E: 也可使 FoxBASE+ 运行成功。

前述操作可放入批处理 FOX.BAT 中:

```
C>COPY CON:FOX.BAT
CALL OVL E:
APPEND E:
MFOXPLUS %1 -NOTIBM
^Z
```

以后使用 FOX 或 FOX <应用程序> 即可达到加速目的。

下面的情况值得一提:

有的汉字操作系统可以使用扩展内存, 如 2.13H、王码 5.0 汉字系统等。对于前者, 是把 16 点阵显示字库作为文件放入虚拟盘, 所以覆盖文件拷入虚盘问题不大, 只要虚盘有足够的空间; 而后者是分级把 16 点阵字库直接装入扩展内存, 它并不考虑虚盘的存在与否, 这就意味着如果我们直接往虚盘上拷贝文件, 将会覆盖破坏扩展内存中的字库信息。解决办法是:

建立两个虚拟盘, 对于王码 5.0 和 MFOXPLUS 2.0 来说, 可把前述 CONFIG.SYS 中建立虚盘的一条命令改为两条:

```
DEVICE=VDISK.SYS 240K /E
```

```
DEVICE=VDISK.SYS 144K /E
```

由于虚拟盘在扩展内存上是线性分配的, 我们可以使第一个虚盘的大小刚好能覆盖整个扩展内存中的字库, 那么我们使用第二个虚盘就十分安全了, 不过, 前面的有关操作都要改为对第二个虚盘进行。

现在的问题是不知字库在扩展内存中所占的深度, 第一个虚盘应设多大才合适呢? 设小了起不到保护字库的作用, 设大了则第二个虚盘的容量就小了, 可能装不下覆盖文件。为此可以反复试验:

根据字库文件字节数, 设第一虚盘为一个足够大的值, 则第二虚盘的容量为总扩展内存量减第一虚盘容量 (下同)。启动机器, 调用汉字进驻扩展内存, 然后用 CHKDSK 检查第二虚盘。如正常, 则减少第一虚盘容量, 否则应增加第一虚盘容量。再如前启动机器重试。如此反复, 即可找出一个最佳配对来。如果你的机器内存可观的话, 那么完全可以大方些。

本文介绍的方法, 对于那些要经常读覆盖文件的软件来说均有效, 如 dBASE 和 PCTOOLS 等, 完全可以如法炮制, 以便加快处理速度。

也谈单色显示器的作图问题

甘肃庆阳长庆一中

任绥海

《电脑》杂志 93 年 1 期“大学生之页”栏刊登的《单显微机的作图实现》一文，本人觉得大有商榷之必要。

首先，该文设置模式的方法是错误的。按屏幕的选择提示按键后，并不能实现相应的转换。例如，屏幕模式号为 0 时应是 40×25 黑白字符模式，而该文中说成是 80×25 模式；模式号为 5 时应为 320×200 黑白图像模式，该文中却说成是 640×200 模式；模式号为 6 时才是 640×200 模式，而该文中却说成是 729×324 。

其次，在程序中切换分辨率没有意义。所有的应用软件都会自动设置相应分辨率模式，故此举实属多余。不过，若除去设定模式的指令，程序中好象就没有多少东西属于作者自己的了。

再次，不知是排版出错还是作者笔误，程序中还有几处错误：

1、XOR AX, AX 指令同时出现两次

2、标号 AGAIN 后丢失冒号，汇编时会出错并给出警告信息

3、MOV AL, 02H / ADD AL, 88 H 这样的指令让人莫名其妙，为什么不直接写成 MOV AL, 8AH 呢？

4、根据有关资料介绍并经验证，送往 M6845 寄存器的第 1 个控制参数应该是 35H，而不是 38H，后者将导致屏幕显示杂乱无章。

最后要说的是，按这种方法设置后的屏幕分辨率实际上应该是 640×400 ，而 CGA 方式下最大分辨率不过 640×200 ，所以屏幕资源只能利用其中一半。更让人不能容忍的是它用的不是屏幕的上半部或下半部，也不是中部，而是占用整个屏幕。在这种情况下，你会发现所有原本完整的图形或字符都被纵向拉断了，原来的垂直线被弄成了间隔均匀的虚线。可以想象，这样的屏幕效果是很难让人接受的。本人经过摸索，发现若将 M6845 寄存器参数中的最大扫描线地址由 01H 改为 03H，则可使屏幕在上部和下部分别显示两幅完全相同的完整的图象，

这就相当于你在同时使用两个同步监视器，虽然这样的效果未必最佳，但总比不改要强得多。

为了便于读者使用该方法，现将经过整理的程序抄录在后，您可在 DEBUG 中直接输入，而且该程序的长度更短，只有 49H 字节。

程序清单：

```

100 JMP 010E
100 DB 35,28,2D,07,64,0D,64,67,02,03,06,07
10E XOR AX,AX
110 PUSH AX
111 POP DS
112 PUSH AX
113 MOV AX,006D
116 MOV [0410],AX
119 MOV AX,0006
11C INT 10
11E MOV AL,03
120 MOV DX,03BF
123 OUT DX,AL
124 MOV AL,29
126 MOV DX,03B8
129 OUT DX,AL
12A PUSH CS
12B POP DS
12C MOV SI,0102
12F MOV DX,03B4
132 MOV CX,000C
135 POP AX
136 OUT DX,AL
137 INC DX
138 LODSB
139 OUT DX,AL
13A INC AH
13C MOV AL,AH
13E DEC DX
13F LOOP 0136
141 MOV DX,03B8
144 MOV AL,8A
146 OUT DX,AL
147 INT 20
149
-RCX
:49
-NCGA.COM
-W
-G
-Q

```

147

AST P386 微机故障维修

盐城市计算机应用研究所 曹迎春 沈 勇

美国 AST 公司的 P386 微机是八九年开始在我国畅销的微机。这种微机因其功能较强, 兼容性好, 价格便宜。一度成了我国微机市场上的热销产品。这种微机的质量还是比较好的, 其故障多半发生在软硬盘控制器部分、彩显控制器部分和键盘接口部分。由于该机种的外围接口电路均做在一块底板上, 就给国内一般维修单位造成了一定的困难。现介绍我们维修中的一些做法。

由于软硬盘控制器和彩显控制器均做在其主板上,而且都采用了超大规模专用集成块,要对主板进行维修确实是比较困难。我们多数是采用加插控制板的方法来避开主板上发生故障的控制器,从而使机器工作正常。

1、 软硬盘控制器故障，四例。

其中三例是主机系统设置双软时认为软驱错, 而仅设置其中任何一个软驱时, 该软驱工作正常, 也就是该主机仅认可一个软驱。另一例, 软硬盘均不认, 主机无法工作。这类故障是 AST P386 微机的最常见故障, 对主板也难以维修。

排除方法：在市场上购一块 AT 标准的软硬盘控制卡加插到主板空插槽上，重新设置系统配置：

On Baird Floppy Adapter: Enable 改为 Disable

On Baird Hard Disk Adapter: Enable 改为 Disable

即使主板上提供的软盘控制器和硬盘控制器无效,将软硬盘信息号电缆同时改插到新加的软硬控制卡。软硬盘均恢复正常工作。这确实是一种花钱少的解决问题办法。

2、主机板 DMA 电路故障，一例。

故障现象为软盘驱动器不能工作，但用例 1 所述的方法无法排除。

其故障原因为: 主板 DMA 电路的 CH2 通道发生故障。AST386 微机采用专用大规模集成块 82C206 作为 DMA 控制电路, 外设向它发送 DMA 请求, 但它不能发出回答信号, 从而使外设不能进行数据传输。更换后故障排除。

由于 82C206 芯片为一只 84 脚方形芯片, 更换时有一定困难, 必须特别小心。

3、彩显控制器故障，仅一例。

其故障现象为无显示，主机发生一长七短声响。根据故障现象查阅随机手册发现为彩显控制器故障。

排除方法：加插一块 VGA 彩卡在主板任一空插槽上，将主板上的一个五位开关中的第五位拨到 ON 状

态。彩显信息号线改插到新加彩卡上。开机使用，工作正常。

4、键盘故障，仅一例。

故障现象为开机显示 03XX 错, 键盘失效, 键盘灯也毫无反应。用逻辑笔测得键盘插也的第 5 脚(电源)无电平, 其它脚正常。即主机不能向键盘提供+5V 工作电压, 造成键盘出错。这种故障多半是由于用户带电拔插键盘或接触不良造成的。

排除方法: AST P386 微机主板是通过一只电感向键盘提供+5V 电源。检查发现该电感开路。将该电感取下,用细漆包线绕上几圈后,焊接到原电感位置即可,也可以直接将电感两端短接。开机后,键盘工作正常。

CR-3240 彩色打印机维修一例

湖南省澧县农业银行 金义炎

一台 CR-3240 彩色打印机，其故障现象是：

- 1、利用打印机自检功能，自检打印正常。
- 2、联机用 WPS 系统打印白纸文稿，其打印正常。
- 3、安装蜡纸打印时，打印机报警，纸尽检测灯似亮非亮，打印功能失效。

从故障现象分析, 完全可排除打印机自身的硬故障, 此类故障可能与纸张检测信号电路有关。

故障排除方法:

卸下打印机控制面板，撤除固定打印机上机壳的三个螺钉，取下打印机上机壳，将橡胶辊二端的固定卡片拔掉，向右端上方拉出橡胶辊，在其橡胶辊下部左端有二个不同颜色的光电二极管 D1，D2，发现其 D1，D2 布满灰尘和碎纸片，用无水酒精将 D1，D2 清洗干净，待酒精挥发完后重新安装好各部件，开机一切打印正常。

该机的二个光电二极管 D1, D2 是并排安装在一个管脚里面, D1 用来发出光电信号, D2 用来接收光电信号。D1 发出的信号通过打印纸反射到 D2, 此时 D2 导通, 因而检测为有纸信号, 在未安装纸时, 由于橡胶辊呈黑色, D1 光电信号不可能反射到 D2, 此时 D2 呈截止状态, 纸尽检测灯亮。由于 D1, D2 积累灰尘, 使 D1 发出和反射到 D2 的信号大大减弱。在安装蜡纸打印时, 由于蜡纸呈深蓝色, 光的反射强度进一步降低, 因而 D2 管处于似通非通状态, 打印机报警。

384KB 内存的开发和利用

广东汕头金山中学 何管略

从位址 640KB 到 1024KB 之间, 有 384KB 的内存位置。它在逻辑上虽然只有一套编码, 而对用户来说却有可能是两种不同的物理空间。

首先, 它是某些系统的内存中不可少的只读存储区 Rom。尽管系统的基本(常规)内存, 有可能达不到 640KB 的容量, 而这个 384KB 的 Rom 空间, 却总是完整无缺的。它大致可分为三个 128KB: 最低位区, 固化有文本方式和图形方式下纯西文字符的字形点阵, 并用作单色和彩色屏幕显示的缓冲区; 最高位区, 固化有硬件设备的 BIOS 服务子程序; 至于中间位区, 早期产品用来固化 RomBasic 的解释程序; 而近期产品, 则用来设置对 Exp 内存的 EMS 页式管理规范。

对于内存资源为 1024KB 的 286 型机来说, 就增加了 384KB 的 Ram 空间。系统自动将它设置为 Exp 内存, 以 16KB 为一页, 共有 24 页, 由 Rom 区中的 EMS 负责管理。我们可以利用它来设置虚拟盘 Vdisk、或硬盘读写高速缓冲器 Cache。MS-Dos 能够严格区分这两套不同的物理空间, 通过 EMS 调度它们。

Dos 在 Exp 内存中交换数据, 比它向物理磁盘完成相同的读写操作, 的确快了很多。但是, 它所采用的是页式管理机制, 每当 Dos 向该区存取数据时, 都要以页为单位, 依次映射到设置 EMS 规范的 Rom 区中来。显然, 由于数据在 Ram 区和 Rom 区中的反复交换, 限制了系统的运行速度, 难以满足当前高档机, 对快速性能的迫切要求。

如果我们加载内存管理程序 Himem, 它会从 640KB 位址开始, 将原有 384KB 的 Exp 内存, 改造成为符合 XMS 管理规范的、速度更快、功能更强的 Ext 内存。当然这里所讲的两内存只是就管理方式的不同来说的, 而实际的物

理空间, 却都是相同的高位 Ram 区。

386 型机的内存资源, 随其档次而异。以 4096KB 的中档机为例, 系统自动将 640KB 用作基本内存, 其余的 3456KB 全部设置为 Exp 内存, 由 Rom 区中的 EMS 规范进行页式管理。为了提高运行效率, 通常总是加载内存管理程序 Himem, 将它改造成为 Ext 内存。

我们从操作实践中发现: 在内存资源大于 1024KB 的系统中, Himem 程序管理 Ext 内存范围的起始位址, 既有默认值, 又可定向推移。对于 384KB 的 Exp 内存, 如果不予开发、利用, 任其埋没, 非常可惜。兹将具体的操作方法, 简介于下:

(1) 在内存资源为 4096KB 的 386 型机上, 如果直接加载 Himem 程序, 它就从默认的 1024KB 位址开始, 管理此后 3072KB 的 Ext 内存。对于 640KB 到 1024KB 之间 384KB 的 Exp 内存, 它却置之不理, 而且此后也无法利用。

(2) 开发 384KB 的 Exp 内存的方法是, 通过系统配置文件 Config 中的 Device 命令, 首先在该机的 Exp 内存中, 建立虚拟磁盘, 使其容量刚好等于 384KB; 然后才加载 Himem 程序。这时它仍然管理 3072KB 的 Ext 内存, 而且两者都能分别进行正常的工作, 互不干扰。

(3) 值得注意的是, Himem 程序管理 Ext 内存的起始位址, 在内存资源大于 1024KB 的系统中, 虽然不会向低位址推移, 但却可以向高位址推移。如果我们首先在该系统的 Exp 内存中, 建立一个 1024KB 的虚拟磁盘, 然后才加载 Himem 程序, 则它将从 1664KB 位址开始, 管理此后 2432KB 的 Ext 内存。

不论我们采用以上哪种方案, 连同 640KB 的基本内存一起计算, 整机的 4096KB 内存资源, 全部都能得到开发、利用。

(150)

反 击 战

南京华东工学院 91621 班 茅 吉、黄 峻

茫茫宇宙，一艘飞船上正进行着一场激烈的较量。left 星球和 right 星球为了测试他们的太空防御系统，进行了一场模拟太空反击战。一颗毁灭弹呼啸着飞向某个星球，这时，只见此星球灵活地控制着它的太空防御板，终于挡回了这颗毁灭弹，很快，另一星球又将接受这一挑战……

用 A,Z 键控制 left 星球的太空防御板；
用上,下键控制 right 星球的太空防御板；
空格键可暂停,Esc 键可退出。

此游戏可一人玩，也可两人玩（一个控制 left，另一个控制 right）。

程序用 turbo pascal 5.0 编制，可在 PC 机上运行

程序清单

{Hit it back is a game of hitting a small ball moved, which may be hit by two men}

```
program hititback(input,output);
uses crt;
const long = 80; mode = 2;
var
  finish:boolean; rest:boolean; ch:char;
  name:string[20];
  lowleft,lowright,highleft,highright:0..200;
  i,number1,number2:0..200;
  speed,movx,movy,x,y,rad:real;
  soundcount:integer;
  vbuf:array[1..25,1..80,1..2] of byte absolute
  b000:0000;
  {vbuf is a buf for video directly with no cursor}
  label 9,99;
```

```
procedure goxy(asc2:byte);
var x1,y1:byte;
begin
  x1:=round(x); y1:=round(y);
  vbuf[y1,x1,1]:=asc2; vbuf[y1,x1,2]:=7;
end;
```

```
procedure upleft;
begin
  if lowleft > 4 then
  begin
    lowleft:=lowleft-1;
    gotoxy(1,highleft);write(' ');
    gotoxy(1,lowleft);write('#221');
    highleft:=highleft-1;
  end;
end;
```

```
procedure upright;
begin
  if lowright > 4 then
  begin
    lowright:=lowright-1;
    gotoxy(long,highright);write(' ');
    gotoxy(long,lowright);write('#221');
    highright:=highright-1;
  end;
end;
```

```
procedure downleft;
begin
  if highleft < 21 then
  begin
    highleft:=highleft+1;
    gotoxy(1,lowleft);write(' ');
    gotoxy(1,highleft);write('#221');
    lowleft:=lowleft+1;
  end;
end;
```

```
procedure downright;
begin
  if highright < 21 then
  begin
    highright:=highright+1;
    gotoxy(long,lowright);write(' ');
    gotoxy(long,highright);write('#221');
    lowright:=lowright+1;
  end;
end;
```

```
procedure inkey(var escape:boolean);
begin
  if keypressed then
  begin
    ch:=readkey;
    if ch=#0
    then
    begin
      ch:=readkey;
      case ch of
        #72:upright; #80:downright; end;
    end
    else
    begin
      case ch of
        #113:upleft;
        #122:downleft;
        #27:escape:=true;
        #32:begin nosound; repeat ch:=readkey until ch=#32;
      end;
    end;
  end;
end;
```

```
{main programme}
begin
  textmode(mode);
  write('please input the band (0-7)');
  readln(i);
  case i of
    0:speed:=0.03; 1:speed:=0.06;
    2:speed:=0.12; 3:speed:=0.3;
  end;
```

C语言游戏一则

广州华南师大计算机90级 梁宇翀

```

4:speed:=0.5; 6:speed:=0.8;
7:speed:=1.0; else goto 9
end;
9: clrscr;
number1:=0; number2:=0;
lowright:=5; lowleft:=5;
highleft:=20; highright:=20;
gotoxy(1,3);write(#219);
gotoxy(2,3);for i:=2 to long-1 do write(#223);
write(#219);
gotoxy(1,22);for i:=1 to long do write(#223);
for i:=lowleft to highleft do
begin
gotoxy(1,i);write(#221);
gotoxy(long,i);write(#221);
end;
randomize;
rad:=2*pi*random(65535)/65535;
movx:=2*cos(rad)*speed; movy:=0.5*sin(rad)*speed;
x:=long/2;y:=13;finish:=false;rest:=false;soundcount:=0;
repeat
soundcount:=soundcount+1;if soundcount mod 500=0 then
soundcount:=0;
sound(200*(soundcount div 50+1));
goxy(32);
x:=x+movx;y:=y+movy;
goxy(2);
inkey(finish);
if finish then goto 99;
if (y<4+abs(movy))or(y>21-abs(movy)) then movy:=-movy;
if (x<2+abs(movx))and((y<=highleft)and(y>=lowleft))
then
begin
number1:=succ(number1);movx:=-movx;
gotoxy(4,25);
write('left:',number1);
if number1 mod 9=0 then
begin
gotoxy(1,lowleft);write(' ');
lowleft:=lowleft+1;
end;
end
else
if (x>long-1-abs(movx)) and ((y<=highright) and
(y>=lowright))
then
begin
number2:=succ(number2);movx:=-movx;
gotoxy(68,25);
write('right:',number2);
if number2 mod 9=0 then
begin
gotoxy(long,lowright);write(' ');
lowright:=lowright+1;
end;
end;
until
(x<2)or(x>long-1)or(highleft-lowleft=5)or(highright-lowright=5);
clrscr;gotoxy(long div 4,12);
if x<2 then writeln('The left is failure!')
else if x>long-1 then
writeln('The right is failure!')
else
begin
writeln('You are all the best!');
writeln('You win the game!!!');
writeln('Input your names');
write('The left:');readln(name);
write('The right:');readln(name);
end;
99: nosound;gotoxy(long div 4,16);
write('press Y key to continue, N key to end. ');
repeat ch:=readkey until (upcase(ch)='Y')or(upcase(ch)='N');
if upcase(ch)='Y' then goto 9;
end.

```

在规定的时间内, 请你找出所有的定时炸弹。

游戏开始时, 先请你输入面积大小, 数字越大就越难玩, 接着请你输入速度, 数字越大就越慢, 一般在 286 机上以 200 为宜。按任一键后, 请你使用上、下、左、右键移动你的位置。如果你判断你所在的位置没有炸弹, 就请用空格键挖开地面, 反之就用回车键标记此地有一个炸弹 (红色星号)。如果你错挖到了炸弹 (白色星号), 则会失去一副防护衣; 如果你挖到了金钱 (黄色美元符), 则获得一副防护衣; 如果你什么也没挖到, 那么将会显示你所处的九宫格中有几个炸弹 (空格表示没有)。当你失去你所有的防护衣时, 就失去了这场游戏。

程序清单

```

#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <string.h>
char earth[62][22];area; /* 地 */
int total; /* 地 */
int hx,hy,t,speed,time,life; /* 人 */
main()
{
while(1)
{
init(); /* 初始化 */
while(time<10000 && total!=0)
{ show(); you op0; delay0(); }
ok0; /* 结束 */
}
}
init()
{
char ex,ey;
textmode(C80); textcolor(LIGHTGREEN); clrscr();
gotoxy(21,9);
printf("Please input the SPEED(0--10000): ");
scanf("%d",&speed); /* 输入速度 */
gotoxy(23,11);
printf("Please input the AREA (5--20): ");
scanf("%d",&area); /* 输入地方大小 */
gotoxy(28,14);
printf("Ready. A key to start.");
while(bioskey(1)=0) rand0;
bioskey(0); clrscr0;
}

```

```

total = area * area / 2;
ex = (80 - area * 3) / 2; ey = (24 - area) / 2;
window(ex - 1, ey - 1, ex + area * 3, ey + area + 1);
makebox(1, 1, area * 3 + 2, area + 2);
areainit(total); areashow();
hx = 2; hy = 2; time = t = 0; life = 3;
}
you_op() { /* 按键处理 */
    int key;
    while (bioskey(1)) key = bioskey(0);
    switch(key) {
        case 19200: if(hx > 2) hx--; break;
        case 19712: if(hx < area * 3 + 1) hx++; break;
        case 18432: if(hy > 2) hy--; break;
        case 20480: if(hy < area + 1) hy++; break;
    }
    textcolor(LIGHTGREEN); gotoxy(hx, hy);
    if(key % 256 == 13) {
        textcolor(LIGHTRED); putch(15);
        gotoxy(hx, hy); ++t;
        if(earth[hx - 1][hy - 1] == 15) --total;
    }
    if(key % 256 == 32) {
        if(earth[hx - 1][hy - 1] == 15) {
            if(--life == 0) time = 10000;
            textcolor(RED);
            gotoxy((3 * area - 10) / 2 + 2, area + 3);
            printf("Life : %d%c", life, 7);
            getch(); gotoxy(hx, hy);
        }
        textcolor(WHITE);
        putch(earth[hx - 1][hy - 1]); gotoxy(hx, hy);
        if(earth[hx - 1][hy - 1] == '0') putch(32);
        if(earth[hx - 1][hy - 1] == '$') {
            earth[hx - 1][hy - 1] = '0'; ++life;
            textcolor(YELLOW); putch('$');
            textcolor(RED);
            gotoxy((3 * area - 10) / 2 + 2, area + 3);
            printf("Life : %d%c", life, 7);
            getch(); gotoxy(hx, hy);
        }
        gotoxy(hx, hy);
    }
}
show() { /* 显示时间 */
    textattr(BLUE * 16 BLACK);
    gotoxy((3 * area - 10) / 2 + 2, area + 3);
    printf("Time : %d", ++time);
    textattr(BLACK * 16 GREEN);
}
delay0() { /* 延时 */
    int i, j;
    for(i = 0; i < speed; i++) for(j = 0; j < 1000; j++)
}
}
ok() {
    int mark = 0; char i;
    window(1, 1, 80, 25); textcolor(LIGHTBLUE); clrscr();
    makebox(21, 8, 61, 15);
    gotoxy(33, 10); textcolor(LIGHTGREEN);
    if(total == 0) mark = area * area * 50 / t;
    if(mark < 80) printf("Y o u L o s e!");
    else printf("Y o u W i n!");
    gotoxy(37, 12); printf("Mark : %d", mark);
    gotoxy(32, 14); printf("Go To Play (y / n) ?");
    while((i = toupper(getch())) != 'Y' && i != 'N');
    if(i == 'N') { clrscr(); exit(); }
}
makebox(x0, y0, x1, y1)
char x0, y0, x1, y1;
{
    int i;
    for(i = y0; i <= y1; i++) {
        gotoxy(x0, i); putch(186);
        gotoxy(x1, i); putch(186);
    }
    for(i = x0; i <= x1; i++) {
        gotoxy(i, y0); putch(205);
        gotoxy(i, y1); putch(205);
    }
    gotoxy(x0, y0); putch(201);
    gotoxy(x1, y0); putch(187);
    gotoxy(x1, y1); putch(188);
    gotoxy(x0, y1); putch(200);
}
areashow() {
    char x, y;
    for(x = 1; x <= 3 * area; x++)
        for(y = 1; y <= area; y++)
            { gotoxy(x + 1, y + 1); putch(177); }
}
areainit(int num) {
    char x, y; int i = 0, j;
    for(x = 0; x <= area * 3 + 1; x++)
        for(y = 0; y <= area + 1; y++) earth[x][y] = '0';
    while(i < num) {
        y = rand() % area + 1; x = rand() % (area * 3) + 1;
        if(earth[x][y] == '0') { earth[x][y] = 15; i++; }
    }
    for(x = 1; x <= area * 3; x++) for(y = 1; y <= area; y++)
        if(earth[x][y] == 15)
            for(i = -1; i <= 1; i++) for(j = -1; j <= 1; j++)
                if(earth[x + i][y + j] == 15) ++earth[x][y];
    while(1) {
        y = rand() % area + 1; x = rand() % (area * 3) + 1;
        if(earth[x][y] == '0') { earth[x][y] = '$'; break; }
    }
}

```

猜一猜

玩一玩

李 峥

以下的程序，是某个最流行的游戏，它的图形悦目，声音悦耳，控制灵活，最能体现玩者的智慧和敏捷。

猜一猜，这到底是什么游戏呢？

玩一玩，何不过过瘾，不过可别废寝忘食！

现介绍游戏的攻略如下：

一、该程序共 4 个控制键：“,”左移；“.”右移。至于“r”键和空格键起何作用，试试便知。

二、在每一级“round”中“level”从 0 至 9，玩过第 9 级“level”，则进入下一级“round”。

三、改 DELAY 值，可改变下落速度。

程序清单：

```
10 DIM B(3): DIM B0(3): DIM B1(3): DIM A(5, 2):
DIM C(5, 2): DIM D(30, 11)
20 KEY OFF: CLS: SCREEN 1
30 COLOR 8, 1
40 LOCATE 18, 14: PRINT "made by Z.Lee"
50 LOCATE 20, 14: PRINT "in Canton, 1992"
60 LOCATE 5, 12: PRINT "copyright reserved"
70 FOR I = 81 TO 241 STEP 40
80 LINE (I - 2, 103)-(I + 7, 112), 2, B: LINE (I - 1, 104)-(I + 6, 111),
1, BF
90 NEXT
100 FOR I = 1 TO 10000: NEXT I: SOUND 50, 5
110 LOCATE 14, 11: PRINT "?????": FOR I = 1 TO 10000: NEXT I
120 A$ = INKEY$: IF A$ = "" THEN CLS: GOTO 130 ELSE
70
130 A$ = INKEY$
140 LOCATE 12, 14: PRINT "Speed Choice:": LOCATE 14, 8: PRINT
"1-slow 2-medium 3-fast": LOCATE 16, 8: PRINT "Press BAR for
Normal speed"
150 IF A$ = "1" OR A$ = "!" THEN DELAY = 100000: GOTO
200
160 IF A$ = "2" OR A$ = "@" THEN DELAY = 10000: GOTO
200
170 IF A$ = "3" OR A$ = "#" THEN DELAY = 100: GOTO 200
180 IF A$ = "" THEN DELAY = 1000: GOTO 200
190 GOTO 130
200 CLS
210 LOCATE 12, 14: PRINT "COLOR CHOICE": LOCATE 15, 14:
PRINT "[BAR]-Lt Cyan": LOCATE 17, 15: PRINT "[Y]-yellow"
```

```
220 A$ = INKEY$
230 IF A$ = "" THEN COLOR 0, 1: CLS: GOTO 260
240 IF A$ = "y" THEN COLOR 9, 0: CLS: GOTO 260
250 GOTO 220
260 LINE (100, 100)-(105, 105), 2, B: LINE (110, 110)-(115, 115), 1, B
270 LINE (101, 101)-(104, 104), 1, BF: LINE (111, 111)-(114, 114), 2,
BF
280 GET (100, 100)-(105, 105), B: GET (110, 110)-(115, 115), B1:
GET (120, 120)-(125, 125), B0
290 CLS: LINE (117, 9)-(184, 184), 3, B: LINE (142, 9)-(159, 9), 2, B:
L0 = 0: SCORE = 0: LOCATE 14, 1: PRINT SCORE: LEVEL =
0: LOCATE 14, 34: PRINT LEVEL: LOCATE 10, 34: PRINT
ROUND + 1: IK = 0
300 FOR I = 1 TO 30: FOR J = 1 TO 11: D(I, J) = 0: NEXT J:
NEXT I
310 FOR T = 25 - ROUND * 2 TO 27 STEP 2: FOR S = -6 TO 2
STEP 2: PUT (160 + S * 6, 10 + T * 6), B1: D(T, S + 8) = 1: NEXT
S: NEXT T: S = -2: T = 0
320 LOCATE 8, 34: PRINT "round": LOCATE 12, 34: PRINT "level":
LOCATE 12, 1: PRINT "score"
330 IF L0 > LEVEL THEN L0 = 0: LEVEL = LEVEL + 1: LO-
CATE 14, 34: PRINT LEVEL
340 IF LEVEL > 9 THEN ROUND = ROUND + 1: GOTO 290
350 S = -2
360 FOR I = 1 TO 5: A(I, 1) = 0: A(I, 2) = 0: NEXT
370 K = INT((113 + LEVEL / 2) * RND(1)) + 1
380 IF K = 1 THEN A(1, 1) = 1: GOTO 520
390 IF K = 2 THEN A(1, 2) = -1: A(2, 1) = 1: GOTO 520
400 IF K = 3 THEN A(1, 1) = -1: A(2, 1) = 1: GOTO 520
410 IF K = 4 THEN A(1, 1) = -1: A(2, 1) = 1: A(3, 1) = 1: A(3, 2)
= 1: GOTO 520
420 IF K = 5 THEN A(1, 1) = -1: A(2, 1) = 1: A(3, 1) = -1: A(3, 2)
= 1: GOTO 520
430 IF K = 6 THEN A(1, 2) = 1: A(2, 1) = 1: A(3, 1) = 1: A(3, 2) =
-1: GOTO 520
440 IF K = 7 THEN A(1, 2) = -1: A(2, 1) = 1: A(3, 1) = 1: A(3, 2)
= 1: GOTO 520
450 IF K = 8 THEN A(1, 1) = -1: A(2, 1) = 1: A(3, 2) = -1: GOTO
520
460 IF K = 9 THEN A(1, 1) = -1: A(2, 1) = 1: A(3, 1) = 1: A(3, 2)
= 1: A(4, 1) = -1: A(4, 2) = 1: GOTO 520
470 IF K = 10 THEN A(1, 1) = -1: A(2, 1) = 1: A(3, 2) = -1: A(4, 1)
= 1: A(4, 2) = -1: GOTO 520
480 IF K = 11 THEN A(1, 1) = -1: A(2, 1) = 1: A(3, 2) = -1: A(4, 1)
= -1: A(4, 2) = -1: GOTO 520
490 IF K = 12 THEN A(1, 1) = -1: A(2, 1) = -1: A(2, 2) = 1: A(3, 2)
= 1: GOTO 520
```

```

500 IF K = 13 THEN 520
510 A(1, 1) = -1: A(2, 1) = 1: GOTO 520
520 GOSUB 980
530 A$ = INKEY$
540 IF A$ = "q" THEN S = 0: T = 0: GOTO 1390
550 IF A$ = "" THEN GOSUB 980: GOTO 710
560 IF A$ <> "," THEN 620
570 GOSUB 980
580 FOR I = 1 TO 5
590 IF POINT(160 + S * 6 - 1 + A(I, 1) * 6, 10 + T * 6 + A(I, 2) *
6) <> 0 THEN GOSUB 980: GOTO 530
600 NEXT I
610 S = S - 1: GOSUB 980
620 IF A$ <> "," THEN 680
630 GOSUB 980
640 FOR I = 1 TO 5
650 IF POINT(160 + (S + 1) * 6 + A(I, 1) * 6, 10 + T * 6 + A(I, 2) *
6) <> 0 THEN GOSUB 980: GOTO 530
660 NEXT I
670 S = S + 1: GOSUB 980
680 IF A$ = "r" THEN 770
690 FOR I = 1 TO DELAY * 10 - DELAY * LEVEL: NEXT
700 GOSUB 980
710 T = T + 1
720 FOR I = 1 TO 5
730 IF POINT(160 + S * 6 + A(I, 1) * 6, 10 + T * 6 + A(I, 2) * 6)
<> 0 THEN IF T < 3 THEN 1390 ELSE T = T - 1: GOSUB 980:
SCORE = SCORE + 100: LOCATE 14, 1: PRINT SCORE: GOTO
1050
740 NEXT
750 SOUND 100, .1: GOSUB 980
760 GOTO 530
770 GOSUB 980
780 FOR I = 1 TO 5
790 A1 = A(I, 1): A2 = A(I, 2): C(I, 1) = A(I, 1): C(I, 2) = A(I, 2)
800 IF A1 = -1 AND A2 = -1 THEN A1 = -1: A2 = 1: GOTO 880
810 IF A1 = -1 AND A2 = 0 THEN A1 = 0: A2 = 1: GOTO 880
820 IF A1 = -1 AND A2 = 1 THEN A1 = 1: A2 = 1: GOTO 880
830 IF A1 = 0 AND A2 = -1 THEN A1 = -1: A2 = 0: GOTO 880
840 IF A1 = 0 AND A2 = 1 THEN A1 = 1: A2 = 0: GOTO 880
850 IF A1 = 1 AND A2 = -1 THEN A1 = -1: A2 = -1: GOTO 880
860 IF A1 = 1 AND A2 = 0 THEN A1 = 0: A2 = -1: GOTO 880
870 IF A1 = 1 AND A2 = 1 THEN A1 = 1: A2 = -1: GOTO 880
880 C(I, 1) = A1: C(I, 2) = A2
890 NEXT
900 FOR I = 1 TO 5
910 IF POINT(160 + S * 6 + C(I, 1) * 6, 10 + T * 6 + C(I, 2) * 6)
<> 0 OR POINT(160 + S * 6 + 5 + C(I, 1) * 6, 10 + T * 6 + C(I, 2)
* 6) <> 0 THEN 710
920 NEXT I
930 FOR I = 1 TO 5
940 A(I, 1) = C(I, 1): A(I, 2) = C(I, 2): SOUND 1000 + I * 200, .1
950 NEXT I
960 GOSUB 980

```

```

970 GOTO 530
980 F = 0
990 FOR I = 1 TO 5
1000 IF F = 1 THEN 1030
1010 PUT (160 + S * 6 + A(I, 1) * 6, 10 + T * 6 + A(I, 2) * 6), B,
XOR
1020 IF A(I, 1) = 0 AND A(I, 2) = 0 THEN F = 1
1030 NEXT
1040 RETURN
1050 FOR I = 1 TO 5
1060 D(T + A(I, 2), S + 8 + A(I, 1)) = 1
1070 NEXT I
1080 I = T + 1
1090 IF I < T - 1 THEN S = -2: T = 0: GOTO 330
1100 FOR S0 = 1 TO 11
1110 DT = DT + D(I, S0)
1120 NEXT S0
1130 FOR S0 = 1 TO 11
1140 IF D(I + 1, S0) = 1 THEN 1170
1150 NEXT S0
1160 IF DT = 11 THEN IF I = 28 THEN 1170 ELSE I = I + 1:
GOSUB 1330: DT = 0: GOTO 1090
1170 IF DT = 11 THEN DT = 0: GOTO 1190
1180 DT = 0: I = I - 1: GOTO 1090
1190 IF I <> 28 OR IK = 1 THEN 1250
1200 IK = 1
1210 FOR P = 1 TO 100
1220 SOUND 4000, .001: SCORE = SCORE + 1000: LOCATE 14, 1:
PRINT SCORE
1230 FOR J = 1 TO 11: PUT (160 + (J - 8) * 6, 10 + I * 6), B, XOR:
NEXT J
1240 NEXT P
1250 FOR J = 1 TO 11
1260 PUT (160 + (J - 8) * 6, 10 + I * 6), B0, PSET: D(I, J) = 0
1270 NEXT J
1280 L0 = L0 + 1: SCORE = SCORE + 1000 * LEVEL: FOR S0 =
1 TO 11: SOUND 2000 + S0 * 200, .1: NEXT S0
1290 LOCATE 14, 1: PRINT SCORE
1300 FOR S0 = 1 TO 11: D(I, S0) = 0: NEXT S0
1310 GOSUB 1330
1320 GOTO 1090
1330 FOR M = 1 TO 2 STEP -1
1340 FOR P = 1 TO 11
1350 IF D(M - 1, P) = 1 THEN PUT (160 + (P - 8) * 6, 10 + M *
6), B, XOR: D(M, P) = 1: PUT (160 + (P - 8) * 6, 10 + (M - 1) * 6),
B0, PSET: D(M - 1, P) = 0
1360 NEXT P
1370 NEXT M
1380 RETURN
1390 LOCATE 12, 15: PRINT "game over"
1400 A$ = INKEY$
1410 IF A$ = "" THEN 290
1420 IF A$ = "q" THEN END
1430 GOTO 1400

```

漂亮女孩病毒的诊治

烟台轴承仪器研究所 王江民 宋继琛

该病毒将软盘 0 面 0 道 1 扇区(BOOT 区)DOS 引导程序修改,使其在启动机器时直接调用隐藏在盘中的病毒程序和 DOS 引导程序。在病毒主程序最后有以下字样,“A pretty girl came into the world on 1-9-68 I love her”。意思大概是“我爱 1968 年 1 月 9 日来到这个世界的漂亮女孩”,病毒名由此而来。

该病毒保留了磁盘 BOOT 中的 I/O 参数表和出错信息,所以粗略的查看磁盘 BOOT 区,容易被病毒蒙混过关。解毒软件 KILL50、CPAV1009、CLEAN80 都不能解除该病毒。

一、漂亮女孩病毒的特征及危害

1、若用染毒软盘启动机器(虽然软盘上无 DOS 的三个启动文件),病毒则抢在系统自举前感染硬盘,将硬盘 1 面 0 柱 1 扇区中的 DOS 引导程序修改,并将该扇区原有的全部内容和病毒主程序一起复制到硬盘某一空闲簇中,然后在 FAT 表相应部位写上标记,以免其它文件的覆盖。如这时我们对其它软盘进行读写,病毒也以同样方式将软盘感染。

2、染毒硬盘或软盘引导系统时,病毒程序将 0:413 地址处的内存总量减少了 2K,再将其自身潜藏于内存,病毒在内存共有两处,其首址分别为 9000:FA00H 和 9000:FE00H。

3、病毒感染标志为 C8C3H,位于硬盘 1 面 0 柱 1 扇区 2AH 处,位于软盘 0 面 0 道 1 扇区 2AH 处。

4、在染毒的浪潮机中,联想智能汉字系统一运行便死机。

二、漂亮女孩病毒的检测

1、要检测内存是否有漂亮女孩病毒,可用 DEBUG 查看内存 9000:FA00H 或 9000:FE00H 开始处是否有以下十六进制码“0E 58 03 C5 8E”,否则无病毒。

2、要检测软盘或硬盘是否感染漂亮女孩病毒,可用 PCTOOLS 查看磁盘 BOOT 区(DOS 引导区)2AH 处是否有病毒感染标志“C8C3H”,否则无病毒。

三、漂亮女孩病毒的清除

1、清除硬盘中的漂亮女孩病毒

由于病毒将硬盘原 DOS 引导区信息隐藏到硬盘某一扇区中,我们寻找起来较麻烦,但可用与硬盘相同版本的无病毒系统软盘引导机器,然后 SYS C:即可解除病毒。注意软盘系统版本号应与硬盘的系统版本号相同,而且 IBM-DOS 或 MS-DOS 出版厂家也应相同,不能混用。我们还应将病毒占据的一个簇清理出来,即 CHKDSK C:/F,这样在硬盘根目录下产生了一个名为 FILE0000.CHK 的文件,将此文件删除即可。

2、清除软盘中的漂亮女孩病毒

用 PCTOOLS 工具将一个正常软盘的 BOOT 区信息,写到有病毒的软盘 BOOT 区(一定要区分 360K、1.2M 或 1.4M),覆盖死病毒程序即可,步骤如下:

1)启动 PCTOOLS。

2)将无病毒同规格正常软盘放入驱动器内。

3)键入“F3”、再键入“E”,然后选择 A 或 B 驱动器,即可读出无病毒软盘

BOOT 区信息。

4)取出正常软盘,插入同规格有病毒盘,如有写保护,应去掉。

5)连续键入“F3”、“F5”、“U”键,即可解除病毒。

笔者还编写了查解漂亮女孩病毒的程序,由于程序较长,不便于刊出,但已存储于本期的磁盘中,名称为:KV18.EXE。有需用者可与编辑部联系。(该程序收集在第四期程序盘中)

Break病毒的剖析

辽宁省邮电学校 王延利

前不久,在用系统软盘启动微机时,出现了一种奇怪现象:读完A盘并启动机器后,屏幕上系统提示符不是A>,却是C>。这引起了笔者的注意,怀疑有某种病毒在作怪。为了安全起见,重新用另一张带有写保护的系统软盘启动微机后,并运行DEBUG,先查看了这张异常系统软盘的BOOT区,发现其中的引导程序已经改变,内容看上去有些类似硬盘主引导程序。于是又将硬盘的主引导程序打印出来,通过分析,证实了确实是一种新型的引导型病毒作怪。根据该病毒的激发条件,笔者取名为Break病毒。

一、Break病毒的特点

Break病毒与其它常见的引导型病毒相比较,具有以下几个特点。

- 1、不论它感染软盘或硬盘,都是将原来的引导程序或主引导程序覆盖掉,而不是象多数引导型病毒那样将原引导程序“搬家”。
- 2、任何一张染有Break病毒的软盘,不论它原来是否为系统盘,都能启动微机,并且启动后系统进入C>状态。
- 3、在对目标盘实施感染之前,不做的任何检查,因而对目标盘的感染重复进行。

二、Break病毒的感染方式

- 1、对硬盘的感染,是在用带有Break病毒的软盘启动计算机的过程中进行的。
- 2、对软盘的感染,是在Break病毒驻进内存后,每次对目标盘进行读写操作时进行的,如用DIR列目录,COPY文件到目标盘等。

三、Break病毒的识别

用“健康”的软系统盘启动后,运行DEBUG,并键入如下指令:

```
-A100
XXXX: 0100 MOV AX, 0201
        MOV BX, 0200
        MOV CX, 0001
        MOV DX, 0080; 软盘为 MOV DX, 0000
        INT 13
        INT 20
```

-G=100

将硬盘主引导扇区的内容读到内存XXXX:0200处,如果看到下面的内容,则表明该硬盘已经感染上Break病毒。

```
-U200
XXXX: 0200 CLI
        XOR AX, AX
        MOV SS, AX
        MOV SP, 7C00
        MOV SI, SP
        PUSH AX
        POP ES
        PUSH AX
        POP DS
        STI
        CLD; 此句之前与硬盘主引导程序同
        JMP 0290; 此3个字节为病毒特征, 正确主引导
                程序中此3字节为 MOV DI, 0600
        PUSH ES
```

用同样方法可读出软盘BOOT区的内容,如果内容同上,则该软盘亦感染Break病毒。

四、Break病毒的激发条件及破坏性

Break病毒在未发作时,软盘和硬盘中的数据 and 文件不会丢失。通过分析该病毒的代码发现,其激发条件为0:[0471]字节的高位(位7)为1,即按过Ctrl-C或Ctrl-Break键。同此,如果在磁盘读写过程中按下Ctrl-C或Ctrl-Break,将引起病毒发作。病毒发作时将内存0:0000开始的内容写入硬盘的指定扇区,从而破坏掉硬盘的DOS引导程序以及文件分配表FAT,造成硬盘不能启动和文件数据的丢失。

五、检测内存中的Break病毒

该病毒驻进内存后,0:[0413]单元的内存容量值会减少1K,病毒常驻在内存高端的XXXX:0000处。我们可以根据0:[0413]字的值,求出病毒的藏身之处,具体算法如下:

```
XOR AX, AX
MOV DS, AX
MOV AX, [0413]
MOV CL, 06
SHL AX, CL
MOV ES, AX
```

全国单片微机学术交流会

通 知

此时 ES 中的值即为病毒常驻的段址。此法适用于检测内存中任何一种修改 0: [0413] 字值, 这一类型的病毒。例如, 某机器内存容量为 640K, 即 0: [0413] = 0280H, 现在只剩下 027FH, 将 027FH 左移 6 位, 得 9FC0H, 此值即为 Break 病毒所在的内存段值。

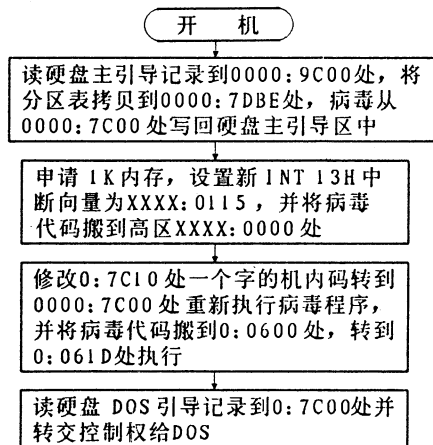
六、Break 病毒的消毒方法

由于 Break 病毒是覆盖型的, 因此, 只有再用正确的引导程序将病毒覆盖掉这一方法来对磁盘消毒。即将正确的主引导记录写回到主引导扇区 (分区表保留不变)。正确主引导记录可以从相同类型的微机上提取, 也可以从有关技术资料中得到。工具可使用 DEBUG, 或 PCTOOLS 或 Norton Utilities, 具体方法不占篇幅介绍。软盘上的 Break 病毒也可用相同的覆盖法消除, 但要注意 DOS 版本的一致性。

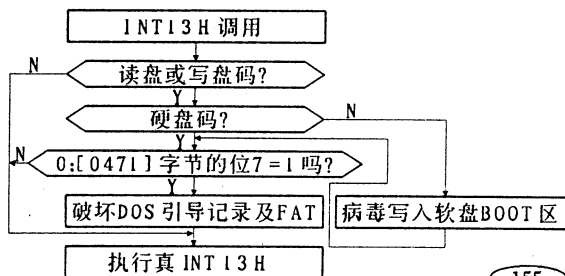
七、Break 病毒原理剖析

下面以方框图的形式给出 Break 病毒的原理剖析, 供大家在检测并消除 Break 病毒时参考。

1、病毒安装引导部分 (含硬盘感染)



2、软盘感染及破坏部分 (INT 13H 中断新入口)



中国微计算机学会单片微机学会全体理事会 1991 年天津会议决定委托广东省单片微机分会主办 1993 年全国单片微机学术交流会。现就有关事项通知如下:

一、1993 年全国单片微机学术交流会定于 1993 年 8 月 16 日~18 日在广州市召开。

二、从现在开始征收参加本次学术会议的论文。论文应在 1993 年 5 月 31 日前寄到本次大会秘书处, 以便付印。

三、征文请寄到如下地址:

广州市东风东路 729 号 邮政编码: 510090
广东工学院计算机应用研究室 余永权 唐平收
电话: 7766069—6693
FAX: 0086—020—7776592

四、征文的内容:

1. 最新单片微机的发展
2. 单片微机系统的开发
3. 单片微机在工业控制或智能化仪器中的应用
4. 单片微机数据、文字处理与网络
5. 单片微机应用技术
6. 数字信号处理器, DSP
7. 单片微机与通信
8. 单片微机在家用电器或其它领域的特殊应用。

五、征文的基本要求:

1、每篇论文的字数应用为 5 千字左右, 一般不得超过 6 千字。

2、论文中全部插图必须按照出版标准用图纸描画。

3、论文中公式的下标必须写清楚, 特别是英文 O 与数字 0 必须分清。

4、论文中所有的小标题分级如下:

一级标题用一、××××

二级标题用 1、××××

三级标题用 (1)、××××

四级标题用 a、××××

5、参考文献按出版标准列出作者, 文献名, 出版社名称, 日期 (卷号, 期号), 页号。

6、寄来的稿件应包括两部份

(1)、纸介质即打印或誊写好的内容。

(2)、磁介质即把内容录入磁盘寄来。(文本文件)

中国微计算机学会
广东省计算机学会单片微机分会

可编程控制器中数据块传送的实现

华南师大微电子所 徐巨善

可编程控制器是专门在工业环境下应用的计算机实时控制系统。为了尽可能缩短扫描周期,除了采用协处理器和位处理器以外,还采用实现数据块传送的硬件结构。《一个大型可编程控制器的电路分析》[1]和《可编程控制器中的位处理器》[2],详细分析了K3N可编程控制器的结构框图和位处理器的结构原理,本文主要介绍K3N可编程控制器的数据存储器及其数据传送的工作原理,着重分析利用Z80CPU的空操作指令,实现数据块传送的具体过程。

一、用户数据存储器及其数据的传送

用户存储器是存放用户程序和用户数据的数据存储器,用户程序是可编程控制器用来实现生产自动化的专用计算机程序。用户数据是与生产过程直接有关的参数和现场状态。用户数据有位数据和字节数据两种。其中输入输出控制量为位数据,表示开关量或状态等。输入位数据存放在输入映像寄存器X中,运算时存入输入继电器X;运算后输出位数据存入输出继电器Y,从输出映像寄存器Y输出。定时,计数运算中使用字节数据,存放在数据继电器D之中。字处理器和位处理器执行用户程序时,还会使用辅助继电器M,位数据的流向如图1所示。

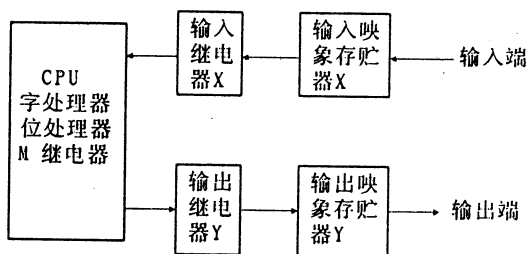


图1 输入输出数据流向图

可编程控制器在执行用户程序前,Z80CPU将输入端的位数据,全部连续读入输入映像寄存器X,再从输入映像寄存器X读出,并存入输入继电器X,然后执行用户程序。在用户程序执行过程中,字处理器和位处理器不断进行逻辑运算等处理,把运算的中间结果存入辅助继电器M,同时控制数据信号不断送到输出继电器Y。当用户程序执行完一个周期后,可编程控制器将输出继电器Y的数据信号全部连续传送到输出

映像寄存器Y,进而通过输出端传送到输出模板,最后到达被控制器件。可见,可编程控制器输出信号是在用户程序执行完一个周期后进行,而在用户程序执行期间,虽然输出继电器Y陆续改变,但实际现场控制信号不会改变,所以实际现场控制信号只有在用户程序执行完一个周期,数据到达输出映像寄存器Y后才能变化,而实际控制状态是否改变,由输入状态是否改变来决定,因而,可编程控制器的控制过程就是输入处理、程序执行处理和输出处理,这一系列操作的不断循环。

二、数据块传送的实现

数据从输入映像寄存器X读出,并存入输入继电器X;从输出继电器Y读出,并存入输出映像寄存器Y。这种数据的传送,采用了一种独特的数据块传送方法:在地址7800-7FFFH范围内,由Z80CPU执行空操作指令来实现。数据块传送的逻辑电路如图2所示。图中G9、G8(HM6147)为输入映像寄存器X和输出映像寄存器Y,是 $2K \times 1$ 位RAM。F6、F4(HM6116)为输入继电器X和输出继电器Y,是 $2K \times 8$ 位RAM,用作位数据只使用D0一位。D10、B9(TC40H374)为I/O锁存选择电路,根据控制字选择G9、F6或F4、G8。C9、C10(74LS244)为Z80CPU的空操作指令码00(NOP)和中断返回指令码C9(RET)的缓冲器。J9(74LS374)、J8(74LS684)分别为数据块长度锁存器和比较器。由Z80CPU提供地址总线A15-A0和数据总线D7-D0,由RD信号经门电路逻辑组合、D触发器B4及多路转换器D5(74LS157)等产生读写控制信号。下面分析实现这两种数据块传送的具体过程。

1. 数据从输入映像寄存器X读出,并存入输入继电器X,使用控制字FD,在端口2A00H(MOVSEL),写入I/O锁存选择电路D10、B9。Z80CPU在7800-7FFFH寻址时,即MOVXY使D10-1、B9-1的 \overline{OC} 有效,D10、B9的输出为FD,即D10-2、B9-5为0电平,D10、B9的其余输出为1电平。D10-2的0电平,经D9-13、11到G9-10,使 \overline{CE} 选择有效。又D10-15为1电平送到G9-8,使G9的 \overline{WE} 为1电平,即读有效。这样,G9-7的DO端输出数据到D4-5,由于B9-5为0电平,即D8-11、D4-4为0电平,数据经D4-5、6到F6-9,即输入继

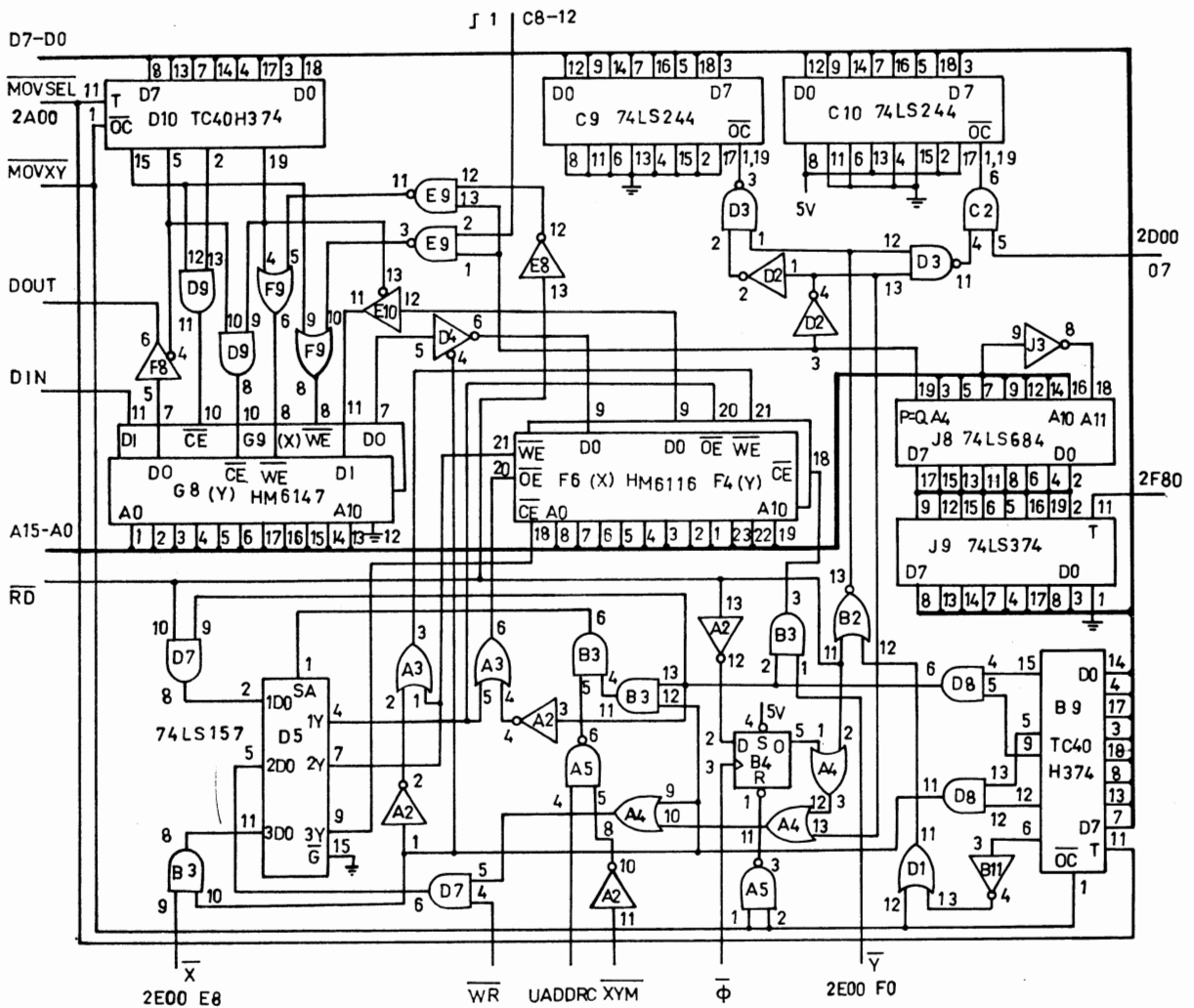


图2 数据块传送逻辑电路

电器 X 的数据输入端，由于 B9-5 为 0 电平，经 D8-13、11，B3-12、11、4、6 到 D5-1，使多路转换器输出对应为 1D0-3D0 输入。这时，B9-5 的 0 电平经 D8-13、11，B3-10、8，D5-11、9 到 F6-18，即输入继电器 X 的 \overline{CE} 有效。又因 Z80CPU 执行空操作，其读信号 \overline{RD} 经 A2-13、12 反相到 B4-2，即为 RD。时钟将 RD 送入 B4，如图 3 所示。由于地址译码 D10-1、B9-1 \overline{OC} 有效，即 MOVXY 为 0 电平，所以 B4-1 为 1 电平，直接复位无效。根据 Z80CPU 机器周期时序图， \overline{RD} 信号由时钟 T1 的负跳变产生，所以 T1 的负跳变在 \overline{RD} 的负跳变前（由虚线所示）。同样， ϕ 的正跳变在 RD 的正跳变之前，所以 B4-5 的输出，在 T1 负跳变，即 ϕ 的正跳变接收的 RD 为 0 电

平。当 T2 负跳变，即 ϕ 正跳变时，B4 接收 1 电平，当 T3 负跳变，即 ϕ 正跳变时，B4 又接收 0 电平，B4-5 输出到 A4-1 与 A4-2 的 \overline{RD} 信号相或，在 A4-3 输出负脉冲。在 J8-19， $\overline{P=Q}$ 为 1 电平时，即数据块未传送完时，A4-13 为 0 电平。A4-3 输出的负脉冲，经 A4-12、11、10、8（因 B9-5 的 0 电平送到 A4-9），D7-5、6（因 Z80 CPU 取指周期 \overline{WR} 为 1 电平），D5-5、7 到 F6-21，即为输入继电器 X 写信号 \overline{WE} 有效。这样，从输入映象存储器 X 读出并送到输入继电器 X，即 F6-9 的数据 D0 存入输入继电器 X。

至于数据块的传送，在数据传送前，Z80CPU 由端口 2F80H 将数据块长度参数 80H 送锁存器 J9，并

输送到长度比较器 J8。由于 J8 比较的变量是地址 A11A10-A4, A3-A0 不比较, 所以当 Z80CPU 寻址为 7800H 时, 此数为 000H, 当寻址为 7FFFH 时, 此数为 7FFH, 这是 2K 寻址空间, 也是数据块的长度, 由于长度参数为 80H, 所以在地址 7800-7FFFH 区间, 比较器 J8 两边数据不等, $J8-19P=Q$ 输出为 1 电平, 这就使 A4-3 的负脉冲能够送到 F6-21, 成为 \overline{WE} 有效, 当地址为 8000H 时, A11A10-A4 成为 80H, 比较器 J8 两边数据相等, $P=Q$ 输出 0 电平, A4-13 为 1 电平, F6-21 成为 1 电平, 无写入作用。

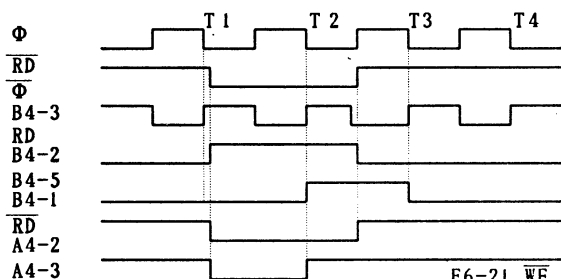


图3 数据块传送时产生 F6- \overline{WE} 定时图

又由于在地址 7800-7FFFH 时, 译码输出 MOVXY 为 0 电平, 送到 D1-12, 又因 B9-6 为 1 电平, 经 B11-3、4 反相, 0 电平到 D1-13, D1-11 输出 0 电平到 B2-12, Z80CPU 取指周期时, \overline{RD} 到达 B2-11, B2-13 输出 RD, 并送到 D3-1, 又因 $P=Q$ 的 1 电平经 D2-3、4、1、2 到 D3-2, D3-3 输出 \overline{RD} 到 C9-1、19, 使 C9 输出数据为 00H, 这就是 Z80CPU 在地址 7800-7FFFH 空间读取空操作指令的原因。当地址为 8000H 时, 由于 $P=Q$ 输出 0 电平, 使 D2-2 为 0 电平, D3-3 输出 1 电平, 使 C9-1、19 \overline{OC} 无效。又由于在地址 8000H 时 MOVXY 为 1 电平, 经 D1-12、11 到 B2-12, 使 B2-13 为 0 电平, D3-11 为 1 电平。但由于在数据传送前, 在端口地址 2D00H 写入 07H, 当寻址到 8000 时, C2-5 输入 0 电平, 这就使 C2-6 输出 0 电平, C10-1、19 \overline{OC} 有效, C10 输出数据为 C9H, 这就是 Z80CPU 的返回指令 RET, 表示数据块传送子程序结束。如果所传送的数据块不到 2K, 即长度参数 < 80H, 这时因 MOVXY 为 0 电平, 又 B9-6 为 1 电平, D1-13、11 为 0 电平并送到 B2-12, \overline{RD} 经 B2-11、13 输出正脉冲 RD 到 D3-12。又当数据块传送完成后, 比较器输出 $P=Q$ 为 0 电平, D3-13 为 1 电平。这样, RD 经 D3-12、11 反相, \overline{RD} 经 C2-4、6, 使 C10-1、19 \overline{OC} 有效, Z80CPU 从 C10 读出返回指令 RET (C9), 数据块传送结束。

2、数据从输出继电器 Y 读出, 并存入输出映像存储器 Y。当控制字 FE 由端口地址 2A00H 写入 D10、B9 后, 在地址 7800-7FFFH 时, \overline{MOVXY} 使 D10、B9 的 \overline{OC} 有效。由于输出为 FE, 所以 D10-19、B9-15 为 0 电平, 其余输出为 1 电平。B9-15 的 0 电平, 同样使 D5-1 为 0 电平, 同时使 F4-18 \overline{CE} 有效, 又经 D7-9、8, D5-2、4 到 F4-20, 使 \overline{OE} 有效。这样, 数据从输出继电器 Y 读出, 从 F4-9 读出 D0 送到 E10-12。又由于 D10-19 为 0 电平, 即 E10-13 为 0 电平, 数据经 E10-12、11 到输出映像存储器 Y, 即 G8-11 DI 端。同时, D10-19 的 0 电平送到 F9-4。又 J8-19P=Q 端, 在地址 7800-7FFFH 时输出为 1 电平, \overline{RD} 信号经 E8-13、12, E9-12、11, F9-5、6 到 G8-8, 使 \overline{WE} 有效, 还因 D10-19 为 0 电平, 使 G8-10 \overline{CE} 有效。这样, 来自输出继电器 Y 的数据, 由 G8-11 DI 存入输出映像存储器 Y (G8)。至于数据块的传送过程与控制字为 FD 时完全一样。

由上述分析可以看出, 数据块传送, 将数据从输入映像存储器 X, 即 G9 传送到输入继电器 X (F6), 或者数据从输出继电器 Y (F4) 传送到输出映像存储器 Y (G8), 都是利用 Z80CPU 在地址 7800-7FFFH 范围内执行 C9 的空操作指令来实现, 利用 C10 的返回指令使数据块传送结束, 而所传送的数据通道和读写存储器各不相同。所以, 可以用相同的指令控制不同的数据通道和存储器, 同时传送数据, 也就是将 FD、FE 两个控制字合并起来, 用 FC 作控制字, 这样, 可以实现两路数据同时传送。

上述数据块传送时, 系统程序中执行的某些程序段和数据如下所示, 以供分析理解。

```

0D44  LD A, 07
      LD (22F6), A ; 端口标志
      LD (2D00), A ; 为 8000H 时读 C9 作好准备
0D62  LD HL, 7800 ; 数据块起始地址
0D6A  LD A, FC
      LD (2A00), A ; 置数据块传送控制字
      LD A, 80 ; 数据块长度参数
      CALL 1C17
1C17  LD (2F80), A ; 置数据块长度参数到比较器
      LD (20EB), HL ; 置数据块起始地址
      JP 20EA ; 调用 7800-7FFF 及 8000 子程序
20EA  CD 00 78 ; CALL 7800
7800  00 00 00..... ; NOP
7FFF  00 ; NOP
8000  C9 ; RET

```

注: [1]《一个大型可编程控制器的电路分析》电脑 1993 年第二期

[2]《可编程控制器中的位处理器》电脑 1993 年第三期

陕西省计算机(集团)股份有限公司

陕西省计算机(集团)股份有限公司(下称陕西计算机公司)是国家计算机定点生产单位,是西北地区最大的微型计算机、电子应用产品厂家和计算机技术服务的综合性企业。目前公司资产总值已达6000多万元,年产值达14677万,年销售收入12480万元。

目前公司拥有6000M²的生产大楼和3000M²的技术服务大楼。建有两条具有世界先进水平的微型计算机生产线和两条电子应用产品生产线。生产线配置了先进的:波峰焊机、清洗机、光板测试仪、在线测试仪、软硬盘测试仪、计算机开发系统、逻辑分析仪、集成电路测试设备、元器件检测和规整设备等完整的生产设备。可年产20万台套微型计算机和其它电子应用产品。公司还设有西北地区计算机及电子应用产品维修中心和计算机技术培训中心,配备了先进的检测设备和现代化的教学设备。

陕西省计算机公司技术力量雄厚,工程技术人员占公司职工总数的52%,其中高级工程师10名,工程师48名,近年毕业的硕士研究生、大学毕业生及助理工程师120名。近年来,该公司研制开发了长安-033E微型计算机、长安PC、长安-286、长安-386等中高档微型计算机及CEC-I、CEC-E、CEC-G、CEC-PC型中华学习机及工业控制机、工业平缝机电脑控制器等产品。还研制开发了多种工具、系统、教学和网络软件。公司产品中,长安-033E微型计算机一九八七年被评为国家优选系列机,并获陕西省优质产品称号,CEC-I中华学习机获机械电子部科技进步二等奖,在一九八八年全国中华学习机行业评比中获得总分第一名和陕西省优质产品称号。1990年在全国电子产品成果展览会上荣获“飞跃杯”优秀产品奖。长安-PC微机

人1992年一上市就以它良好的性能价格比赢得用户青睐,年产量达数千台,并获“飞跃”成果奖。长安-PC微机从1990年一上市就以它良好的性能价格比赢得用户青睐,年产量达数千台,并获“飞跃杯”成果奖。产品畅销全国二十多个省市自治区,深受用户好评。

陕西省计算机公司下设有陕西省计算机厂,陕西省计算机技术服务分公司、陕西省计算机公司应用研究所、新产品开发中心。分别从事计算机软硬件研制、开发、生产、销售和技术服务。公司还与香港计算机厂商合资在西安建立了“西安电子企业有限公司”,生产计算机键盘、显示器等,产品出口加拿大等欧美各国,在深圳建立了“深圳长安电子企业有限公司”承担微机的加工出口任务。形成了内外结合,相互补充的计算机生产经营体系。公司和日本、美国、新加坡、香港、马来西亚等国外四十多个公司有着密切的使用和广泛的业务往来,是美国HP公司和香港金山计算机公司在西北地区的总代理。

陕西省是全国知识分子密集的省份,陕西省共有大专院校41所。陕西省计算机公司和西安交通大学、西安电子科技大学、西北工业大学等大专院校密切合作,拟建陕西省计算机及微电子开发中心,引进多套CAD系统,研制高档微型计算机及其ASIC电路和外部设备。公司准备与美国、新加坡、日本及港台等厂商联合建立软件开发中心。承接国内外用户的各种软件开发。与国外厂商联合建立计算机通讯设备工厂,专门研制生产各种通讯设备。公司还大力开展和东欧的技术合作,与俄罗斯等国家签定了电话机生产线的包建合同。

陕西省计算机公司将加强国内、国际合作和技术交流,不断开拓计算机新领域。

158

如何提高FOXBASE+的统计运算速度

四川师范大学附中 冯建军

目前的数据库系统都存在一个突出的问题，这就是运行速度的问题。FOXBASE+比DBASE的运行速度提高了许多。但是，它的运算速度，尤其是数据统计运算速度，仍然是不令人满意的。针对这一问题，笔者提出了一个切实可行的方案。该方案原理易懂，操作简单，可极大地提高FOXBASE+的统计运算速度。

一、BUFFERS 设置和 PCACHE 设置与运行速度的关系实验

同行知道，要提高数据库系统的运行速度，就要充分利用系统仅有的内存。为此，DBASEⅢ和FOXBASE+都在DOS的BUFFERS设置基础上增加了在自身的环境设置文件CONFIG.DB和CONFIG.FX中设置BUFFERS和PCACHE参数功能。

通常的FOXBASE+资料中称“FOXBASE+与DBASEⅢ相比的一个主要改进是内存的使用方式。在执行时，它能动态调整内存缓冲程序的存贮和其它内容，使得在较小的内存条件也能运行得很快。”相应地，“它在CONFIG.FX文件中不需要设置BUFFERS、PCACHE参数，从而使得使用过程大大简化。”

为了验证这一点，笔者做了两个实验，情况见表1。

表1 BUFFERS 设置与运行速度的关系

DOS 中 BUFFER 值设置	进入 FOX 后所余空间(用 CHKDSK 命令检查)	完成相同内容所花时间
BUFFERS=2	220592B	147 秒
BUFFERS=50	195248B	148 秒
BUFFERS=90	174128B	147 秒

注：1、BUFFERS参数用来设置内存缓冲区的大小，影

响 I/O 速度；

2. CONFIG.FX中的BUFFERS设置对运行速度不产生影响，而且不导致内存的减少，故不再列出；
3. 被统计数据库文件长度为40KB，均被一次性调入内存。

表2 PCACHE 设置与运行速度的关系

P CACHE	被统计数据库长度	进入 FOX 后所余空间		完成相同内容所花时间
		CHKDSK 检查	SYS(12) 检查	
8	40K	215K	283K	147 秒
64		215K	283K	147 秒
8	217K	215K	283K	514 秒
64		215K	283K	514 秒

注：1、PCACHE参数，用来设置分配给执行期间缓存编译好程序的内存总量，以1K字节为单位，取值范围是8-64，约定值是64。

2. 被统计数据库文件均被一次性调入内存，命令文件相同。

二、如何提高较大数据库的统计速度

数据库统计运算的速度问题，笔者曾有过两次深刻的体验。第一次是处理一个学生调查的数据统计，使用FOXBASE+2.1在PC机上整整运行了23小时多；另一次是12000余个记录的成绩统计，使用FOXBASE+2.1在386机上运行了6.5小时多，才完成一个项目的统计。这是让人难以忍受的。

有了这两次深刻体会，便开始认真思考提高数据库统计速度的问题。大家知道，在进行读写操作时，软盘的速度最慢，只及硬盘的十分之一，然而内存的读写是纯粹电的操作，速度是硬盘的百倍以上。如果每统计一个数据项目都要介入磁盘的读写，这样，统计出成千上万个数据来，单花在磁盘的读写上的时间就显得很大了。请看表3：

表3 统计过程有无磁盘读写介入对运行速度的影响

库长	有无磁盘读写介入	完成相同内容所花时间	时间比
217K	有	83016 秒(23.06 小时)	100%
217K	无	16387 秒(4.55 小时)	19.7%

注: 1、两次实验仅内存大小不同, 其余条件均相同;
2、命令文件长度为755B。

从上表可见, 被统计数据库不能被一次性调入内存而介入频繁的磁盘读写是影响统计运算速度的主要原因。

目前国内使用的几乎都为汉化 FOXBASE+, 在双软 PC 机上通常让一级汉字驻留内存, 这时可一次性调入内存的数据库长度不超过 78K (从表一可见, BUFFERS 值的设置情况对这个值有影响), 在实际工作中这只能算一个小型数据库。对长度大于 78K 的数据库应怎样处理呢? 这是实际工作为我们提出的一个课题。

针对这一问题, 容易想到不让汉字库驻留这 640K 内存, 这样可以将这个极限提高到 180K。但是, 对于国内拥有量极大的 IBM PC/XT 及其兼容机由于汉字库驻留磁盘 (内存不到 1M), 就不能在命令文件和数据库文件中使用汉字, 否则仍存在频繁的磁盘读写问题。——这仍不是最满意的方法。

其实, 汉化 FOXBASE+ 是由西文汉化而来, 因此它与西文状态下运行是完全兼容的。那么, 我们为什么不能在汉字状态下设计并调试好程序, 而退出汉字状态来运行程序呢? 实践证明, 这是可行

的。并且对含有汉字的命令文件和数据库文件可以不作任何修改。使用这一方法, 可以将仅 640K 内存的 PC 机的一次性调入内存的数据库长度由 78K 提高到 224K。见表 4:

根据这一原理, 对于拥有 1M 内存的 286 机, 可利用扩充内存建容量为 600K 的虚盘, 不用来装字库, 而作为使用盘。这样可以使长度超过 500K 的数据库的统计过程全在内存中进行。同样方法, 对内存达 4M 的 386 机型, 则可以实现长度达 3M 的数据库的统计运算过程不介入频繁的磁盘读写, 这会比传统方法的速度提高百倍以上。

这样, 一般意义的“大库”的统计问题就可以迎刃而解, 不再需要烦人的等待了。对于更大的数据库, 可以根据上述原理对数据库进行拆分。即根据统计要求进行横分或纵分 (分记录或分字段), 达到能一次性调内存的目的, 统计后再将结果合并, 以求效率最佳。

当然, 要提高速度, 在命令文件中是忌讳使用 SET TALK ON 的, 这不仅是显示慢, 同时, 进行 SET TALK ON/OFF 转换需转换内存的中系统覆盖程序, 介入磁盘的读操作。至于统计结果, 一般放于另一库中。从表四及第一部分的讨论结果可看出, DOS 中的 BUFFERS 设置也宜小些好, 这样可提高一次性调入内存的数据库的长度。

该过程使用的软件是 PC-DOS3.3、CCDOS4.0、FOXBASE+2.1。除特别说明外使用机型均为 IBM PC/XT 兼容机, 双软驱, 640K 内存。

表4 各种设置和状态对一次性调入内存的数据库长度的影响

项目	是否汉字状态	进入 FOX 系统后所余空间		能一次性调入内存的数据库最大长度	文件长度比
		CHKDSK 检查	SYS(12)检查		
BUFFERS=2	是	32K	99K	78K	100%
BUFFERS=99	否	165K	233K	202K	259%
BUFFERS=2	否	215K	283K	224K	287%
CONFIG.FX 中 BUFFERS 值设置				无影响	——
CONFIG.FX 中 PCACHE 的设置				无影响	——

注: 测试用命令文件长度为 755B。

159

IBM PC 机VGA显示拍摄技术

机电部三十所 叶 宾

目前对 IBM-PC 机的显示系统编程的文章已发表了不少,但是在不同程度的运用上都受到了一定限制。比如:对位平面结构显示方式,线性内存结构显示方式,对应的内存就不一样,许多文章都一直认为把内存中的显示缓冲区内容保存下来,再恢复出来就把当前屏上内容保存和复原了,其实这看法很不全面。事实上,各种显示方式其编程都不同,为正确地编程,充分发挥各种显示器的特色,本文以一个 VGA 的两种显示方图像存取为例,加以说明。

随着 PC 机的发展,显示系统的分辨率经历了由 CGA、EGA 到 VGA 的演变,显示分辨率从 640×200 点阵 CGA、 640×350 点阵 (EGA) 发展到 640×480 点阵的 VGA,并向更高点阵发展。色彩也从单色发展到了较丰富的 256 种颜色。显示速度并没有因为分辨率的提高,扫描象素点多而使显示速度慢下来。

VGA 是在 CGA 和 EGA 的基础上发展起来的,其控制电路主要包括以下几部分:CKTC、GDC、TS、ATC、DAC、CRTS……。CKTC 是完成显示器控制,确定频率,以及确定主机 CPU、显示缓冲区、CRT 显示三者之间的地址接口等。GDC 是图形显示控制器,主要完成 CPU 对显示缓冲区的数据操作。必须注意的是 GDC 只能支持平面式内存结构,而 VGA 的 640×480 分辨率显示方式则为平面式内存结构。TC 时序发生器产生 CRT 及动态存储所需的时序信号,它也用来解决主机处理器和 VGA 的图形控制器访问显示缓冲区的时序冲突,它允许 CPU 在动态显示时可访问显示缓冲区。属性控制器 ATC,主要用来在字符和图形的视频显示时,作高速视频移位和属性处理。DAC 数字模拟信号转换器,主要完成把 VGA 卡的八位二进制颜色信息转换成模拟信号,并输出驱动 CRT 显示,内部结构类似于调色板。VGA 允许主机处理器通过对 DAC 的 I/O 操作访问调色板寄存器,另外还有一个主时钟选择机制,不同的显示模

式需不同的主时钟频率。

VGA 的显示缓冲区基本配置是 256K,当支持 16 种以下的颜色时,使用存储位平面结构 (PLANE,此时 256K 的显示缓冲区分个 64K 的存储位平面,屏上的每点 (象素点) 由 4 个存储位平面的各一位组合后表示,最多可表示 $2^4 = 16$ 种颜色。当支持 256 种颜色时,要使用线性内存结构,4 个存储平面链接,形成 256KB 的线性内存,屏上每象素由一个字节来表示。颜色最多 $2^8 = 256$ 种。VGA 一般都有自己专用 BIOS 支持,此 BIOS 是视频控制程序,固化于 ROM 中,成了 VGA 卡的一部份。一般 VGA 视频 ROM, BIOS 入口地址为 C000H-CFFFH 之间。

VGA 的特点:支持最高分辨率为 640×480 ($16/256$ 颜色),有的甚至还能支持更高的分辨率如 800×600 点阵、 $1.24K \times 768$ 点阵。支持 IBM 提出 VGA 标准显示模式,显示模式号为 0-13H。支持存储位屏面 (PIANE) 结构和线性内存 (LINEAK) 结构。基本内存 256K,可扩为 512K。图形模式有 320×240 四色、 640×200 二色、 640×480 十六色、 320×200 二百五十六色...等。目前最为运用广泛的图形显示方式是 640×480 与 320×200 这两种。 640×480 十六色显示方式很好地体现了 VGA 的图形特色,它的分辨率高,色彩较丰富,在许多运用软件的图形方式中采用之。特别在 C 语言中直接提供了这种图形模式的建立方式,运用十分方便。而 320×200 二百五十六种颜色方式则以分辨率为代价,而换得了色彩极大的丰富,它显示出包括生活中我们可见的几乎全部颜色。因此不少的彩色图像摄入器采集的画面以此方式显示出来,图像画面栩栩如生。

现在动画软件爱好者越来越多,但是优美而丰富的图案因算法难度高,不易掌握。为什么不可以将现有的某些软件的图案为我所用呢?在此,结合前面介绍的 VGA 知识,给大家提供一个实用 (已调试通过的) 屏幕图形保存程序。与一般方法不同

的是：通常大家使用把内存缓冲区内容存入文件，但这种方法需要根据不同显示器、不同显示方式而变换内存地址，存入方式很繁琐，为什么不可以象摄像机一样扫描成像呢？本程序的思想是通过显示方式进行探测，在确定显示方式之后，对屏上每点逐一扫描，读出其色彩存入文件中。这样我们就以数据的方式完整地保存了一幅画。并且这样处理而得到的数据几乎可以原封不动地发向打印机，从而得到屏幕硬拷贝的图形。因为打印机的图像打印方式实际上也是按像素点扫描逐点打印的。程序 1、2 是一个驻留内存的 C 程序，它是修改了屏幕打印中断服务程序，以按 PRINT-SCREEN 键来激活。

在此程序的主程序中，没有给出恢复 INT5 对应于屏幕打印服务部分。在屏幕的拍摄中，先给出了一段判断，建立图形显示方式的程序，再对 DAC 调色板逐个地测试，读出其状态保存之。因为在实际运用中，不少的图形对 DAC 进行了编程，所以不对调色板寄存器编程的话就会出现面目全非的情况。恢复图像的程序与之对应如程序 3、4：

本程序的不足之处是，由于多次地与磁盘进行数据交换（读和存），因此减慢了速度。但是，从驻留内存程序来考虑，它减少了驻留内存的程序量，因而减少了与原软件的内存冲突机会，这也是值得的。何况它思路简单、可行呢？

程序1

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

```
void interrupt pt();
char busy=0;
union REGS r;
char t[20];
```

main()

```
{struct address { char far *p}; 指向屏幕打印中断
 struct address far *addr=(struct address far *)20;
 addr->p=(char far *)pt;指向驻留内存的屏幕拍摄程序
 printf(" Please input your filename:");
 gets(t);给你拍下的图形文件取一个文件名
 printf("\nPlease hit print-screen
        key to save picture.\n");
 pr(20000);为驻留内存部分开销20000字节的空间
 }
 pr(size)
```

```
long unsigned size;
```

```
{r.h.ah=49;
 r.h.al=0;
 r.x.dx=size;
 int86(0x21,&r,&r);
 }
```

```
void interrupt pt()
```

```
{if(! busy){
        busy=! busy;
        cop(); 真正的屏幕拍摄部分
        busy=! busy;
    }
}
```

```
cop() 真正的屏幕拍摄部分
```

```
{FILE *fp;
 int i,j;
 unsigned s;
 fp=fopen(t,"w+");
 for(i=0;i<256;i++){
    r.x.bx=i;
    r.h.al=0x15;
    r.h.ah=0x10;
    int86(0x10,&r,&r);
    fprintf(fp,"%x",r.h.dh);
    fprintf(fp,"%x",r.h.ch);
    fprintf(fp,"%x",r.h.cl);
```

```
for(i=0;i<320;i++) 逐点扫描
```

```
for(j=0;j<200;j++){
    r.x.dx=i;
    r.x.cx=j;
    r.x.dl=320*j+i;
    outporth(0x3cd,0x40); 从内部输出某点的色彩值
    s=peekb(0xa000,r.x.di);
    s=s& 0x00ff;
    fprintf(fp,"%x",s);
};
fclose(fp);
}
```

程序2

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

```
void interrupt pt();
char busy=0;
char t[10];
```

main()

```
{int gmode,gdriver=DETECT;
```

```

struct address {char *p;};
struct address far *addr=(struct address far *)20;
addr->p=(char far *)pt;
printf(" Please input you filename:");
gets(t);
registerbgidriver(EGAVGA-driver);
initgraph(&gdriver, &gmode, "d:\\tc\\");
pr(20000)
closegraph();
}
pr(size)
long unsigned size;
{r.h.ah=49;
 r.h.al=0;
 r.x.dx=size;
 int86(0x21, &r, &r);
}
void interrup pt()
{if(! busy){
    busy=! busy;
    cop();
    busy=! busy;
}
}
cop()
{FILE *fp;
 int i, j, s;
 fp=fopen(t, "w+");
 s=getbkcolor();
 fprintf(fp, "%d", s);
 for(i=0; i<640; i++)
    for(j=0; j<480; j++){
        s=getpixel(i, j);
        fprintf(fp, "%d", s);
    };
 fclose(fp);
}

```

程序3

```

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

```

union REGS r;

main()

```

{FILE *fp;
 char b,a,l[20];
 int a,i,j,t,o;
 printf("Please input picture's
 name which you want see.\n");
 printf("Name is:");
 gets(l);

```

```

r.h.ah=0x0;
r.h.al=0x13;
int86(0x10, &r, &r);
fp=fopen(l, "r");
for(i=0; i<256; i++){
    r.x.bx=i;
    fscanf(fp, "%x", &s);
    r.h.dh=s;
    fscanf(fp, "%x", &s);
    r.h.ch=s;
    fscanf(fp, "%x", &s);
    r.h.cl=s;
    r.h.al=0x10;
    r.h.ah=0x10;
    int86(0x10, &r, &r);
    for(i=0; i<320; i++){
        for(j=0; j<200; j++){
            fscanf(fp, "%x", &s);
            s=s&0x00ff;
            r.h.bl=s;
            r.x.dx=i;
            r.x.cx=j;
            r.x.dl=320*j+i;
            outportb(0x3cd, 0x40);
            pokeb(0xa000, r.x.di, r.h.bl);
        }
    }
    fclose(fp);
}

```

程序4

```

#include <stdio.h>
#include <alloc.h>
#include <graphics.h>

```

main()

```

{FILE *fp;
 int gmode, gdriver=DETECT;
 int i, s, t;
 char l[20];
 printf("Please input you filename:");
 gets(l);
 registerbgidriver(EGAVGA-driver);
 initgraph(&gdriver, &gmode, "D:\\tc\\");
 fp=fopen(l, "r");
 fscanf(fp, "%d", &s);
 setbkcolor(s);
 for(i=0; i<640; i++){
    fscanf(fp, "%d", &s);
    putpixel(i, j, s);
}
fclose(fp);
closegraph();
}

```

西文状态下特殊字符的打印

广后设计院

黄真康

在中文状态下，很多制表字符及其他图形字符都可以通过其驱动程序输出到打印机；但在西文状态下若用其驱动程序或用 PRINT 打印是行不通的，打出来的只是不期望的普通字符（ASCII 码值小于 128 的字符）。事实上，打印机此时处于字符打印状态，它所接收的只能是普通字符。要能在西文状态下打印特殊字符：首先，根据打印机针数，构造出特殊字符的点阵数据，一字节数据代表 8 个点。如 24 针打印机，其字节数据与打印针的对应关系如图 1 所示。其次，把打印机设置为点图打印方式，设置方法随打印机类型不同而不同，然后将所构造的点阵数据送至打印机，如用汇编语言的话，可通过 DOS 功能调用的 05H 功能或 BIOS 中的 17H 中断。如下例程是打印如图 2 所示的图形，本程序用 MASM5.0 编译连接，再用 EXE2BIN 转换为 COM 文件便可运行。此程序在 DOS3.3、286 及其兼容机、M1724 或 M2024 打印机上调试通过。

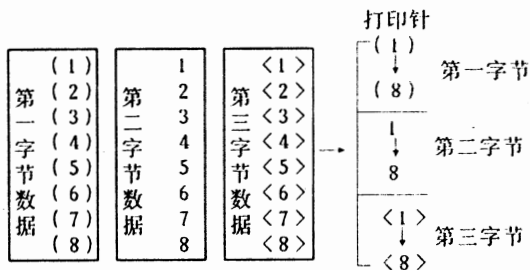


图1

例程:

```

code    segment public
        assume cs:code, ds:code
        org 100h

start:  jmp     begin
        int17   macro
        mov     ah,00h
        int     17h
        endm

prt     db 1bh,'4','0',16
        db 1, 2, 28, 3, 2, 28, 7, 2, 28
        db 15, 2, 28, 31, 2, 28, 63, 2, 28
        db 127, 2, 28, 255, 2, 28, 255, 2, 28
        db 127, 2, 28, 63, 2, 28, 31, 2, 28
        db 15, 2, 28, 7, 2, 28, 3, 2, 28
        db 1, 2, 28

begin:  mov     ax,cs
        mov     ds,ax
        mov     dx,0
        mov     si,32

conti:  mov     cx,52
        mov     bx,offset prt

next:   mov     al,[bx]
        int17
        inc     bx
        loop    next
        dec     si
        cmp     si,0
        jne     conti
        mov     al,0dh
        int 17
        mov al,0ah
        int 17
        mov     ah,4ch
        int 21h

code    ends
        end     start
    
```

161

图2

《一个自动输出表格的多功能 dBASE 程序》的修改

晋城市委党校 阎家富

本刊 1992 年第一期第 11 页介绍了陆瑞光同志的文章《一个自动输出表格的多功能 dBASE 程序》，笔者按程序试了一下，确实可行，临时输出表格该程序是个好方法，但有两点不足之处：

一是数据库的字段值为零时，表中不是空白的，而有一小数点（原文已介绍过），不能令人满意。

二是数据库的最后字段不输出时，表格右端不成其为表格，如图 1 所示。

三是有条件输出时，如结束记录不满足条件，表格的最后一条线用不同的线重复打印两次，如图 2 所示，不符合表格要求。

现将程序加以修改，以上两缺点都可克服。修改后的程序清单如下（其中打“△”号的为笔者加的，打“▲”号的为笔者改的，另外还删除了一部分程序）：

```
set talk off
clear
p=space(8)
```

```
nn=2
lpn=1
do while .t.
@ 1,10 say "*****"
@ 3,20 say "自动制表输出软件"
@ 4,20 say "*****"
@ 7,20 say "请输入数据库名:" get p
read
if file ("&P .dbf")
exit
else
wait "文件名有错或不存,按N退出,按任意键继续" to xy
if upper(xy)="N"
return
endif
endif
enddo
close data
c="-----"
cl="-----"
n11=1
mm="C"
pn=60
bt=space(50)
sele 1
use &p
go bott
nc=recno()
copy stru exte to as
sele 2
use as
coun to n2
sum to n3 field-len
n31=n3+n2*2+2
@ 8,10 say "输入标题:" get bt
@ 9,10 say "输入标题字体:(A,B,C,D)" get mm
@ 10,10 say "输入每页行数:" get pn
read
bt=trim(bt)
n32=len(bt)
p1="┌"
p2="└"
p3="├"
p4="┤"
clear
xy="N"
```

```

NB=1
tg=space(30)
@ 1,5 say "使用的数据库:" +p
@ 1,30 say "已有记录:" +str(nc,4)
@ 2,5 say "设定的标题为:" +bt
@ 3,5 say "原数据库有字段:" +str(n2,3)
@ 3,30 say "设定每页的行数为:" +str(pn,2)
@ 4,5 say "原数据库记录宽:" +str(n3,3) + "列, 加表格线后为:" +
str(n31,3) + "列"
@ 5,5 say "选择起始记录:" get nb
@ 5,30 say "结束记录:" get nc
@ 6,5 say "选择输出条件:" get tg
read
tg=trim(tg)
@ 7,10 say "修改字段名或字段宽吗? " get xy pict "!"
read
ll=2
i="01"
go 1
do while .not. eof()
store 0 to sp1,sp2
x=field-name
y=field-len
z& i=x
if xy="Y"
@ 7,8 say "第" +str(val(i),2) + "字段"
@ 8,20 say "字段名:" get x
@ 9,20 say "字段宽:" get y
read
if y=0
n2=n2-1
skip
loop
endif
endif
x=trim(x)
y=int((y+1)/2)*2
Yx=len("&x")
if Y<yx
y=yx
else
sp1=int((Y-yx)/2)
sp2=y-yx-sp1
endif
ll=ll+y+2
.& i=ll
i=subs(str(val(i)+101),9)
p4=p4+space(sp1) + "& X" +space(sp2)
p1=p1+subs(c1,1,Y)
p2=p2+subs(c,1,y)
p3=p3+subs(c1,1,y)
if xy="Y"

```

```

if y=0
dele next 1
else
repl field-len with y
endif
endif
skip
if .not. eof()
p1=p1+"_"
p2=p2+"+"
p3=p3+"_"
p4=p4+"|"
else
p1=p1+"|"
p2=p2+"|"
p3=p3+"|"
p4=p4+"|"
endif
enddo
△ p5=len(p1)
△ if substr(p1,p5-1,2)="_"
△ p1=substr(p1,1,p5-2)+"_"
△ p2=substr(p2,1,p5-2)+"_"
△ p3=substr(p3,1,p5-2)+"_"
△ p4=substr(p4,1,p5-2)+"|"
△ endif
if (upper(mm)="B").or.(upper(mm)="D")
n32=n32+2
nn=4
endif
n33=(ll-n32)/nn
clear
set devi to print
▲ @ 0,1 say " \@&mm"
@ 1,n33 say bt
▲ @ 1,40 say " \@A&17"
@ 2,0 say p1
@ 3,0 say p4
@ 4,0 say p2
sele 1
go nb
u=5
do while recno()<=nc
if len(tg)>0
▲ set filter to &tg
endif
i="01"
nn=2
@ u,0 say "|"
do while val(i)<=n2
△ sele 2
△ go val(i)

```

如何在DR-DOS 6.0下

使用2.13H

上海铁路局电力试验所 徐洲痕

```

△ b=field-len
△ d=field-type
△ sele 1
z=z&i
△ if d="N"
△ if &Z=0
△ @ u,nn say space(b)
△ else
△ @ u,nn say &z
△ endif
△ else
▲ @ u,nn say &z
△ endif
nn=l&i
if val(i)=n2
@ u,nn-2 say " | "
else
@ u,nn-2 say " "
endif
i=subs(str(val(i)+101),9)
enddo
skip
u=u+2
if recno()<=nc
if u>pn
@ u-1,0 say p3
@ u,ll/2 say str(lpn,2)
lpn=lpn+1
@ u+1,0 say " "
? chr(7)+chr(7)
wait "请换纸! 准备好后按任意键继续."
▲ @ 0,1 say " @&mm "
@ 1,n33 say bt
▲ @ 1,40 say " @A "
@ 2,0 say pl
@ 3,0 say p4
u=5
endif
@ u-1,0 say p2
else
@ u-1,0 say p3
@ pn,ll/2 say str(lpn,2)
@ u+1,0 say " "
exit
endif
enddo
set devi to scre
set print off
return

```

162

DR-DOS6.0 是 DIGITAL RESEARCH 公司 1991 年 8 月推出的一个非常成功的操作系统, 它比 MICROSOFT 的 MS-DOS5.0 大大地提高了一步。它运用了磁盘压缩 (DISK COMPRESSION) 技术, 能使可用的磁盘空间增大近一倍; 能够更有效的利用内存; 实现了双机通信; 提供了多任务切换功能和对系统及文件的保护功能, 并且具有良好的用户介面, 可随时显示帮助信息。而 CC-BIOS2.13H 汉字系统是目前国内最流行的微机汉字操作系统之一, 它拥有众多的用户。但是, 2.13H 在 DR-DOS6.0 下却无法运行, 原因是 2.13H 需要用于键盘和显示器的设备驱动程序 ANSI.SYS 支持, 并且必须将 DEVICE=C:\ANSI.SYS 放在 CONFIG.SYS 的第一行。ANSI.SYS 替换 IBMBIOS 中的标准 CON 设备驱动程序, 仔细检查显示数据以查找 ESC (1BH) 字符, 如果没有找到, 数据就如同在标准 CON 驱动程序下显示一样, 如果找到了 ESC, 随后的几个字符使 ANSI.SYS 执行某个特殊功能。

DR-DOS6.0 由于其本身的特点, 它的 ANSI.SYS 不支持 2.13H, 使得 2.13H 一执行就死机。笔者通过将 MS-DOS3.30A 的 ANSI.SYS 替换 DR-DOS6.0 的 ANSI.SYS, 在 CONFIG.SYS 的第一行加入 HIDEVICE=C:\DRDOS\ANSI.SYS, 重新启动机器, 即可在 DR-DOS6.0 下使用 2.13 使用 2.13H, 这样, 同时拥有了这两者的卓越性能, 使微机的性能成倍提高。

163

解除CCED的日期限制

广东省物价局 刘 川

在 CCED.EXE 和辅助程序 DBST.EXE 中(Ver 02-24-1989,文件长度分别为 86930 和 41884 字节),均设置有系统日期检测过程作为程序使用寿命的限制,当系统日期年份 < 1992 年时可以进入正常的操作;当系统日期年份 > 1992 年时,则只是显示标题和“CCED 使用期限已到,需要更新版本”的信息之后退出,使程序无法继续使用。

程序获取系统日期是由 DOS INT 21H 中断的 2AH 号子功能完成的,调用的汇编语句为 MOV AH,2A INT21,程序代码为 B42A CD21,返回结果年份在 CX。由于程序只作年份值检查而不检查月日值,故要解除 CCED 的日期限制,只需取消程序读系统日期的操作,代之以设置 CX 为 CCED 认可的年份值(例如 1991 年),使程序认为“读”到的年份为合法值而可以进入程序的正常操作,从而可解除 CCED 对日期的限制。使用 PCTools 可以非常方便地完成这项修改。

首先用 PCTools 的 Find 功能查出代码 B42ACD21 在 CCED.EXE 中的偏移位置,显示结果为:

```
search argument found in relative sector 00149
offset 0044
```

```
press "E" to view / edit the sector or
"G" to continue searching
```

相当于在程序的第 76332 字节处。此时按 E 键可以直接进入 FILE View / Edit 状态,屏幕显示 CCED.EXE 的代码内容,并且光标停在相对扇区 149 偏移 44 字处。可见到下面几个字节:

```
50 51 52 56 BB 0A 00 E8 C1 06 53 52 B4 2A CD 21
```

将它们修改成:

```
50 51 52 56 BB 0A 00 E8 C1 06 53 52 B9 C7 07 90
```

在光标所在位置起可以看到代码 B42ACD21,如图中的下划线部分。这时可以将这四个字节修改为 B9C70790(汇编语句为 MOV CX,07C7,设置 CX = 1991 年,剩下一个字节用空操作码 90H 填充),按 F5 键将修改结果存盘,然后结束修改,退出 PCTools,完成对程序的修改工作。这时 CCED.EXE 已解除了对日期的限制,无论系统日期为哪一年份都可以正常使用。

对 CCED 的辅助程序 DBST.EXE 的修改方法与对 CCED.EXE 的修改方法完全相似。用 PCTools 的 Find 功能可以查出代码 B42AD21 在程序中的偏移位置为相对扇区 0063 偏移 281 字节,相当于在程序的第 32537 字节处。用同样的方法将这四个字节修改为 B9C70790 然后存盘退出即可完成对辅助程序 DBST.EXE 的修改。

由于 DBST.EXE 的默认打表日期是取系统日期,并且与检测程序使用期限调用同一过程来获取系统日期,而此时程序中原来读系统日期过程已被修改因而无法返回正确的系统日期参数,返回的结果只是一个固定的年份值 CX = 07C7H,在 DX 中的月、日值未知,已没有意义。因此,在使用修改后的 DBST.EXE 时需要注意,建立参数表文件 DBST 时必须指定打表日期的年、月、日而不能使用默认参数——使用系统日期,出就是在 DBST 文件中必须有 /Y = XXXX / M = XX / D = XX 参数指定年月日。

164

在MS-DOS 5.0下使用高端内存的方法

长江水利委员会水文局 傅运林

内存是计算机系统最重要的资源之一。所有程序必须装载到内存中后才能运行。因此,计算机上配置大量的内存才能运行很大程序。随着计算机技术的不断发展,应用程序变得越来越复杂,对内存的需求量也不断增大,但 MS-DOS 及其它大多数应用程序所占用的地址范围为 000000H~09FFFFH,这 640K RAM 称为常规内存 (Conventional Memory)。由于 MS-DOS 本身要占据一定的内存,再加上一些设备驱动程序和内存驻留程序,当运行较大的应用程序时,这 640K 常规内存变得非常拥挤,甚至使用得某些程序由于内存不够而不能运行。目前 286 以上的计算机都具有一兆以上的内存,那么,怎样使用 640K 常规内存以外的内存呢?

紧接着 640K 常规内存的 384K 内存空间称为高端内存 (Upper Memory Area)。在大多数的系统中,高端内存没有使用。如果将一些设备驱动程序和内存驻留程序从常规内存移到高端内存运行,那么就可以空出更多的常规内存供应用程序使用。

MS-DOS5.0 提供了一系列的设备驱动程序,使得设备驱动程序和内存驻留程序甚至 DOS 本身都能在高端内存中运行。下面介绍这一技术的实现方法:

一、准备工作:

某些程序不能在高端内存中正常运行,而确定某一程序能否在高端内存中运行的唯一方法是试错法。将一个不能在高端内存中正常运行的程序装入高端内存运行时,往往导致死机,从而系统不能重新启动,因此,要将一个程序移到高端内存运行前必须做一个系统软盘:

C>FORMAT/S A:

并将 CONFIG.SYS 和 AUTOEXEC.BAT 文件考到系统软盘上。

二、设置 CONFIG.SYS 文件

为使 DOS 能访问高端内存,需要设置 CONFIG.SYS 文件。MS-DOS 需要 HIMEM 和 EMM386 内存管理程序才能使用高端内存,因需要在 CONFIG.SYS 文件中增加如下几条命令来安装这两个设备驱动程序:

1、用下面命令安装 HIMEM 设备驱动程序:

DEVICE=C:\DOS\HIMEM.SYS

HIMEM 是一个扩充内存管理程序,它提供了对扩充内存访问的方法,并保证在同一时刻没有两个程序使用扩充内存的同一区域。

2、增加 DOS=HIGH, UMB 命令:

DOS=HIGH 使得 MS-DOS 在高端内存中运行。DOS=UMB 规定 MS-DOS 将维护常规内存和高端内存的连接。

3、为 EMM386 程序增加一条 DEVICE 命令:

DEVICE=C:\DOS\EMM386.EXE NOEMS

EMM386 内存管理程序提供了访问高端内存中未使用区域的方法,它使得设备驱动程序和其它程序能在高端内存中运行。

4、HIMEM 和 EMM386 的 DEVICE 命令必须在其它的 DEVICE 命令之前,且 HIMEM 的 DEVICE 必须在 EMM386 的 DEVICE 命令之前。

5、用 CTRL+ALT+DEL 键重新启动计算机,此时系统已作好了在高端内存中运行程序的准备了。

三、获取高端内存的信息

设置好了 CONFIG.SYS 文件之后,就可显示高端内存的信息了。在将程序移到高端内存运行之前,需要了解高端内存的可用空间和移到高端内存运行的程序所需的内存量。用下列命令即可显示高端内存的信息:

C>MEM/C| MORE

此命令的输出见后面的实例。

四、将设备驱动程序和内存驻留程序从常规内存移入高端内存。

为使程序在高端内存中运行,必须有足够的可用高端内存块,因此必须按下列步骤将程序移入高端内存:

1、用 MEM/C 命令获取内存的信息。

2、在 Largest available upper memory block (最大可用高端内存块) 栏中记下最大可用高端内存块数。

3、在 Conventional memory 栏中找出需要移动的程序占用的内存量,它必须小于或等于最大可用高端内

存块数。

4、确定了那些程序要移到高端内存中运行之后，修改那个程序的启动命令，使得系统在下次启动时，程序能装入高端内存（注意：MS-DOS 的数据、HIMEM 和 EMM386 不能移到高端内存中）。

对设备驱动程序，修改 CONFIG.SYS 文件，将那个设备驱动程序的 DEVICE 命令改为 DEVICEHIGH 命令。例如 ANSI 在 CONFIG.SYS 中的 DEVICE 命令为：

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

将其改为：

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

对内存驻留程序，将 LOADHIGH 命令插入启动这个程序的命令之前。例如 DOSKEY 程序用下列命令启动：

```
C>DOSKEY
```

将其改为：

```
C>LOADHIGH DOSKEY 或 LH DOSKEY
```

大多数内存驻留程序都是在 AUTOEXEC.BAT 中启动，因此需要修改这个文件。

5、用 CTRL+ALT+DEL 键重新启动系统，再用 MEM / C 命令观察程序是否在高端内存中运行。如果移到高端内存的程序仍在常规内存中运行，那么很可能是因为没有足够的可用高端内存块装载这一程序（某些程序装载时所需的内存量比运行时所需内存量大）。

下面给出一个在高端内存中运行 CPAV 的 VSAFE、DOSKEY 和 2.13H 的实例。

未使用高端内存时的 CONFIG.SYS 和 AUTOEXEC.BAT 文件如下：

```
C>TYPE CONFIG.SYS
```

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

```
DEVICE=C:\DOS\EMM386.EXE NOEMS
```

```
DEVICE=C:\213\ANSI.SYS
```

```
C>TYPE AUTOEXEC.BAT
```

```
PATH C:\DOS
```

```
C:\CPAV\VSAFE /1+/2-/3-/4+/5+/6+/7-/8-
```

```
C:\DOS\DOSKEY.COM
```

```
CD\213
```

```
FILE0A 82
```

```
CCCC
```

```
CV26
```

```
INT10F
```

```
YX1
```

```
FILE16B
```

```
FILE24A 1SFHK
```

```
FILE40A 1SFHK
```

```
ZF24 3
```

```
KWB
```

```
WBZX
```

内存的使用情况如下：

```
C>MEM / C
```

Conventional Memory:

Name	Size in decimal	Size in Hex
MSDOS	57312	(56.0K) DFE0
HIMEM	3200	(3.1K) C80
EMM386	9424	(9.2K) 24D0
ANSI	1552	(1.5K) 610
COMMAND	4704	(4.6K) 1260
VSAFE	24608	(24.0K) 6020
FILE0A	16464	(16.1K) 4050
DOSKEY	4128	(4.0K) 1020
CCCC	43584	(42.6K) AA40
INT10F	3216	(3.1K) C90
YX1	768	(0.8K) 300
FILE16B	2992	(2.9K) BB0
FILE24A	9728	(9.5K) 2600
KWB	1200	(1.2K) 4B0
WBZX	65408	(63.9K) FF80
FREE	64	(0.1K) 40
FREE	406496	(397.0K) 633E0
Total FREE:	406560	(397.0K)

Total bytes available to programs: 406560 (397.0K)

Largest executable program size: 406496 (397.0K)

3145728 bytes total contiguous extended memory

0 bytes available contiguous extended memory

3091456 bytes available XMS memory

64Kb High Memory Area available

从 Conventional memory 栏中可知，VSAFE、FILE0A、DOSKEY、CCCC、INT10F、YX1、FILE16B、FILE24A、KWB 和 WBZX 九个程序共占内存 158.6K，可将它们移到高端内存中运行。

使用高端内存时的 CONFIG.SYS 和 AUTOEXEC.BAT 文件如下：

```
C>TYPE CONFIG.SYS
```

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

```
DEVICE=C:\DOS\EMM386.EXE NOEMS
```

```
DOS=HIGH,UMB ;使 MS-DOS 在高端内存中运行
```

```
DEVICE=C:\213\ANSI.SYS
```

在 AUTOEXEC.BAT 中，将 VSAFE 等九个程序的启动命令前插入 LOADHIGH 命令，使它们在高端内存中运行。

```
C>TYPE AUTOEXEC.BAT
```

```
LH C:\CPAV\VSAFE /1+/2+/3-/4+/5+/6+/7-/8-;
```

装入高端内存

```
LH C:\DOS\DOSKEY\COM ; 装入高端内存
```

```
CD\213
```

```
LH FILE0A 82 ; 装入高端内存
```

```
LH CCCC ; 装入高端内存
```

```
CV26
```

```
LH INT10F ; 装入高端内存
```

```
LH YX1 ; 装入高端内存
```

```
LH FILE16B ; 装入高端内存
```

再

谈漂亮实用的 动态标题程序

广州航务工程学校 王坚城

FILE24A ISFHK
FILE40A ISFHK
ZF243
LH KWB ; 装入高端内存
LH WBZX ; 装入高端内存

内存的使用情况如下:

Conventional Memory:

Name	Size in Decimal	Size in Hex
MSDOS	12992	(12.7K)
HIMEM	1184	(1.2K)
EMM386	9424	(9.2K)
ANSI	1552	(1.5K)
COMMAND	2624	(2.6K)
FILE24A	9728	(9.5K)
FREE	64	(0.1K)
FTEE	617536	(603.1K)
Total FREE:	617600	(603.1K)

Upper Memory

Name	Size in Decimal	Size in Hex
SYSTEM	163840	(160.0K)
VSAFE	24608	(24.0K)
FILE0A	16464	(16.1K)
DOSKEY	4128	(4.0K)
CCCC	43584	(42.6K)
INT10F	3216	(3.1K)
YX1	768	(0.8K)
FILE16B	2992	(2.9K)
KWB	1200	(1.2K)
WBZX	65408	(63.9K)
FREE	1168	(1.1K)
Total FREE:	1168	(1.1K)

Total bytes available to programs (Conventional+Upper):
618768 (604.3K)

Largest executable program size: 617536 (603.1K)

Largest available upper memory block: 1168 (1.1K)

3145728 bytes total contiguous extended memory

0 bytes available contiguous extended memory

3091456 bytes available XMS memory

MS-DOS resident in High Memory Area

从 Upper memory 栏中可知, MS-DOS 和 VSAFE 等九个程序已在高端内存中运行。没使用高端内存时可用的常规内存量 (Total bytes available to programs) 为 397.0K。使用了高端内存后可用的常规内存量为 603.1K。由此可见, 使用高端内存后, 可大大增加常规内存, 在此例中, 增加了 206.1K。

166

笔者读了 92 年第 5 期《电脑》杂志《漂亮实用动态标题程序》一文后, 想就这个问题继续探讨一下。不管汉字从屏幕的上方一个一个地掉下来, 还是从屏幕的下方一个一个飞上去, 它的原理是将标题逐字取出, 从上 (或下) 往下 (或上) 显示, 在显示下一行的同时清除掉当前行的显示, 给人视觉上产生一种动态感。

笔者介绍一程序, 使标题汉字一个一个从屏幕的右端往左移动, 它的原理是将标题逐字取出, 并在其后连接一个空字符, 从屏幕的最右边逐列往左显示。因为一个汉字相当于两个字符, 往左一列显示时, 原汉字的左边被刚显示的汉字覆盖, 右边被空字符覆盖。(如果要设计一程序使标题汉字一个一个从屏幕左端往右移动, 空字符必须连接在取出汉字的前面)

本程序用 dBASEIII 编写, 可以在 dBASEIII 和 FoxBASE 系统下运行。程序清单如下:

```
set talk off
i=1
xm='广州航务工程学校'
h=len(xm)
do while i<=h/2
    xml=subs(xm,i*2-1,2)+' '
    x=0
    do while x<=58-i*4
        set colo to g+
        @ 10,78-x say xml
        x=x+1
    endd
    set colo to w
    i=i+1
endd
retu
```

166

微机屏幕图象的压缩存贮与恢复

北京丰台总后医专电教中心 陈惠生

随着计算机技术的飞速发展,微机显示器的分辨率也愈来愈高。目前几乎绝大多数微机都配备了 VGA 或 TVGA 图形卡,分辨率高达 1024×768 点。由于分辨率的提高和颜色值的增加,屏幕图象占用的显示存贮空间也随之增加。VGA 或 TVGA 卡在 12 模式下工作时,屏幕分辨率为 640×480 点。显示存贮空间为 $38400 \times 4 = 153600$ Byte,如以普通方法将图象存盘,一张 1.2 兆的高密磁盘也只能存放 7 幅屏幕图象。为了减少图象文件占用的存贮空间,本文以 VGA 图形卡为例,介绍由 Turbo C 2.0 编写的屏幕图象压缩存贮与恢复的方法。

一、图形方式下 VGA 卡显示存贮结构与组织

在 VGA 卡上显示存贮器由 64-256KB RAM 组成(有的达 512KB 甚至更高)。在 VGA 卡图形方式的 D、E、12 模式下,图形象素采用彩色位平面方式,这些存贮器被分为四个彩色位平面,每个位平面 64KB,按字节编址,四个位平面在 CPU 所控制的存贮空间内占有相同的地址,显示存贮地址从 A000:0000 开始,每个象素由 4 个二进位表示,分别占用 4 个位平面中的一位,屏幕上的象素位置与显存地址成线性关系。由于 VGA 内部是 32 位数据通路,因此 4 个位平面可同时读/写。当 CPU 向 VGA 显存写一个字节时,这个字节可同时向 4 个位平面写入,或向 4 个位平面中的一个写入,这由操作定序器中的位屏蔽寄存器控制;当 CPU 从显存读出一个字节时,4 个位平面同时读出,然后由图形控制器中的位平面选择寄存器选择其中一个字节送到 CPU 中去。在 VGA 图象存贮与恢复时要用到以下四个寄存器:

操作定序索引寄存器,地址为 3C4H,向显存写数据时向该寄存器写入索引号 2。

操作定序数据寄存器,地址为 3C5H,位 0 置 1 时选择位平面 0,位 1 置 1 时选择位平面 1,...

图形控制索引寄存器,地址为 3CEH,从显存读数据时向该寄存器写入索引号 4。

图形控制数据寄存器,地址为 3CFH,向该寄存器写入 0、1、2、3 来选择位平面 0、1、2、3。

二、图象的压缩存贮

通过对 VGA 显存的结构及组织分析可知,图象数据在显存中是以连续地址存贮的,因此利用改造的游程编码法设计图象压缩存贮程序比较方便。

在此方法中,为了提高速度,牺牲一点压缩率,对游程编码法进行适当改造。改造后的压缩方法是:对于重复字节和不重复字节均采用三个字节记录,即前两个字节是重复或不重复字节的长度,第三个字节是重复或不重复字节的内容。由于每个位平面中占用的最大显存 38400 字节,两位无符号数最大可表示 65535 字节,因此两个字节足以表示重复字节的长度。

存贮图象时,首先向图形索引寄存器 3CEH 写入索引号 4,选择位平面读数据,然后向图形控制数据寄存器 3CFH 写入位面号,接着从 A000:0000 地址开始读数据进行重复计数,存贮压缩数据。经压缩后的图象一般占用 20 多 K 存贮空间,只有不压缩存贮的 1/5。大大提高了外存设备的利用率。程序见 save_picture()函数。

三、压缩图象的恢复

压缩图象的恢复与压缩过程相反,在向操作定序索引寄存器 3C4H 写入索引号 2 后,向操作定序数据寄存器 3C5H 写入位面控制字 N (N=1、2、4、8,即对应着位平面 0、1、2、3)。接着从压缩的图象文件中读取数据,进行解压缩后写入该位平面中的相应地址。压缩图象恢复程序见 load_picture()函数。

save_picture()和 load_picture()函数由 Turbo C 2.0 编写,在 TVGA 卡上调试成功,稍加改动,即可适应于 CGA、EGA 等显示卡。

```
/* compact save picture-file */
int save_picture(char *file-name)
{
    FILE *fp;
    char ch,*bt="ANIPIC";
```

```

int addr, i;
struct recl {
    unsigned int clen;
    char ch;
} picrec;
if ((fp=fopen(file-name, "wb"))==NULL) {
    printf("%s file not open! ", file-name);
    return 0;
}
fwrite(bt, 6, 1, fp); /* write compact flag */
outportb(0x3ce, 4);
for (i=0; i<4; i++) {
    outportb(0x3cf, i);
    picrec.ch='';
    picrec.clen=0;
    addr=0;
    do {
        ch=peekb(0xA000, addr);
        if (picrec.clen==0) {
            picrec.ch=ch;
            picrec.clen++;
        }
        else if (ch==picrec.ch)
            picrec.clen++;
        else {
            fwrite(&picrec, 3, 1, fp);
            picrec.clen=1;
            picrec.ch=ch;
        }
        addr++;
    } while (addr<=0x9600);
    fwrite(&picrec, 3, 1, fp);
}
fclose(fp);
return 1;
}

```

```

/* uncompact load picture */
int load_picture(char *file-name)
{
    FILE *fp;
    int addr, i, k;
    char ch, *bt, bt1="ANIPIC";
    struct recl {
        unsigned int clen;
        char ch;
    } picrec;
    if ((fp=fopen(file-name, "rb"))==NULL) {
        printf("%s file not open! ", file-name);
        return 0;
    }
    fread(bt, 6, 1, fp);
    if (! strcmp(bt, bt1)) {

```

```

        printf("%s not anipic file! ", file-name);
        return 0;
    }
    outportb(0x3c4, 2);
    for (i=1; i<9; i+=2) {
        outportb(0x3c5, i);
        addr=0;
        do {
            fread(&picrec, 3, 1, fp);
            for (k=1; k<=picrec.clen; k++) {
                pokeb(0xA000, addr, picrec.ch);
                addr++;
            }
        } while (addr<=0x9600 || !feof(fp));
    }
    fclose(fp);
    return 1;
}

```

参考文献:

- (1) IBM PC 的原理与应用(续二) 张福炎等编
- (2) TVGA 图形卡用户手册 (167)

利用 PC-CACHE

提高 WPS 的运行速度

杭州电子工业学院计算机系 黄 林

CCDOS5.1 是香港金山公司和北京大学联合开发的最新超级汉字系统。它提供了集文字编辑和图文编排于一体的 WPS 与 SPT, 具有很强的文字排版功能, 在国内有着广大用户。但是对于使用纯软件(不带汉卡)的用户, 由于在运行过程中不断要从盘上读入汉字点阵信息, 速度比较慢。在实际应用中, 我们发现如果在装入 CCDOS 前先运行 PCTOOLS 工具箱系列中实用软件 PC-CACHE, 开辟扩展内存给 CCDOS, 可以使 CCDOS 汉字系统下的操作(如运行 WPS、SPT 或其它中文软件)速度比不运行 PC-CACHE 快近 2 倍左右, 大大提高了工作效率。具体使用如下:

SPLIB

PC-CACHE / SEZEXT = 256K

SPDOS

<需装入的汉字输入法>

参数 SIZEXT = 256K 说明利用的扩展内存为 256K, 此值可根据机器的实际配置和需要设置。用 /? 参数可获得详细用法。

(168)

《正确用C语言读取数据》的补充

湖南省衡阳市建设银行科技处

肖晓斌

读完本刊1993年第1期黄宏杰同志介绍的《正确用C语言读取数据库》后,受益非浅,特别是该文详细地介绍了dBASE数据库文件的结构,为dBASEⅢ数据直接转换成其它数据库文本文件提供了捷径。但是该方法是用Turbo C语言编写的,只适用于DOS系统,有一定的局限性。由于UNIX/XENIX多任务多用户操作系统的普及,人们普遍倾向于用C语言与informix等关系型数据库系统结合编程,因为informix数据管理系统具有功能齐全,使用方便,易于扩充等优点,在使用UNIX/XENIX操作系统的小型机或高档微机上得到广泛应用。

为了避免重复劳动,充分利用dBASEⅢ数据,需要将其数据进行转换。一般转换方法是:

dBASE数据库DBF→dBASE文本文件TXT
→informix数据库的ASCII码文本文件→informix
库监控程序DBSTATUS (load ascii 文件名 from
“文本文件名”)

该方法的缺点是受dBASEⅢ数据库字段内容的影响,如果字段内包含有空格,逗号,竖线等字符时,转换就不能成功,而且不便于编写通用转换程序(即用程序直接实现DBF→DAT)。

我设计了一种简便方法,能够克服上述方法的缺点,并减少了生成dBASEⅢ文本文件中间步骤:

DBF文件→informix的ASCII码文本文件→
informix的dbstatus转换(下面是实现的程序)

```
$ cat zhdb.c
#include <stdio.h>
#define REC-BEGIN 81
#define REC-NUM 41
#define REC-SIZE 0x1
#define FIELD-BEGIN 0x201
char magin-number[] = {0x03,0x0};
main()
{
    int j,l,n,ch,flag,begin-rec,end-rec,reco-num,rec-num,rec-
    -size;
```

```
char field-rec[0x20],*p,rec[1024],fm[4],name[20],namel
[20],s[40];
int field-size[200];
long rec-begin, pos;
FILE *fp1,*fp2,*fopen();
printf("请输入 FOXBASE/DBASE 数据库文件名:");
gets(name);
printf("\n");
if (! (fp1=fopen(name,"r")))
{
    printf("数据文件 %s 不能打开\n",name);
    exit(1);
}
printf("请输入 informix 文本文件名:");
gets(namel);
printf("\n");
if (! (fp2=fopen(namel,"w")))
{
    printf("目标文件 %s 不能打开\n", namel);
    exit(1);
}
fread(fm,1,1,fp1);
if(strncmp(fm,magin-number,1) != 0)
{
    printf("%s 不是一个FOXBASE/DBASE 数据库\n",name);
    exit(1);
}
fseek(fp1,REC-NUM,0); /* 移动指针 */
fgets(s,9L,fp1); /* 读取有关信息 */
rec-begin = s[4]+256*s[5]; /* 计算结构部分长度 */
rec-num = s[0]+256*s[1]; /* 计算记录数 */
rec-size = s[6]+256*s[7]; /* 计算结构长度 */
printf("该数据库文件的记录总数为: %d\n",rec-num);
pos = FIELD-BEGIN;
l = 0;
while(pos+0x20<rec-begin)
{
    fseek(fp1,pos,0);
    fread(field-rec,0x20,1,fp1);
    field-size[l] = (short) (field-rec[0x10] & 0xff);
    pos+=0x20;
    l++;
}
```

```

while (1)
{
printf("请输入起始记录号: ");
scanf("%d",&begin-rec);
if (begin-rec>rec-num)
{
printf("起始记录大于数据库记录总数\n");
exit(1);
}
fseek(fp1,(begin-rec-1)*rec-size+rec-begin,0);
printf("请输入终止记录号: ");
scanf("%d",&end-rec);
reco-num = begin-rec;
while(reco-num<=end-rec)
{
if(fread(rec,rec-size,1,fp1)!=1) break;
if(rec[0]!='*') continue;
p = rec+1;
for(j=0;j<l;j++)
{
flag = 0;
for(n = 0;n<field-size[j];n++)
{
if(flag!=0||p[n]!=' ')
{
flag = 1;
fputc(p[n],fp2);
}
}
fputc('|',fp2);
p += field-size[j];
}
fputc('\n',fp2);
reco-num++;
}
printf("是否继续转换? (Y/n)");
while ((ch = getchar()) != 'Y' && ch != 'n');
if (ch == 'n') break;
}
fclose(fp1);
fclose(fp2);
exit(0);
}
$ cc -o zhdb zhdb.c

```

该程序能够多次任意地读取 DBF 文件中某一段记录组合成一个 ASCII 文本文件。上述方法在 IBM PC386 机及兼容机和 XENIX V2.3.2 操作系统上均已调试通过。

169

ADM 1.0 Password 的彻底消除

北京邮电学院数据所八室 叶武

ADM 有很强的管理硬盘能力,但如果你将 Password 遗忘,那么即使你是超级用户也不能再修改硬盘分区。除非,你将硬盘低级格式化后,再用 ADM 重新分区。笔者通过分析 ADM.EXE 及 ADM.SYS,发现可用 debug 将它们修改使 Password 失去作用。

ADM.SYS 这一块设备驱动程序用于管理 ADM 的各分区,它在初始化部分询问用户口令。而 ADM.EXE 则在程序执行的开始部分近调用一过程询问超级用户口令。如果将两程序中相关部分跳过,原来所设的 Password 即失去了作用。用 debug 修改这两程序很简单,具体步骤如下 (ADM 为 V1.00,1987 年版):

具体用 debug 修改方法,见下面。

将所有 ADM 文件拷入一系统盘中,在 CONFIG.SYS 中加入一行 device=ADM.SYS。以后,从 A 驱启动即可使用任何你不知道 Password 的 ADM 管理的硬盘,可进行重分区,设超级用户口令等操作。(注意,我们使用的是 ADM V1.00,1987 年版)。

(1)ADM.SYS 的修改

```

A>DEBUG ADM.SYS
-u 964,964
1550:0964 741c jz 0982
;注意此部分代码(可用 S 命令搜索)
-a 964
1550:0964 jmp 982
1550:0966
-u 991,991
1550:0991 803EA40700 CMP BYTE PTR [07A4],00
;注意此部分代码
-a 991
1550:0991 jmp 99d
1550:0993
-w
Writing 00CE8 bytes
-q

```

(2) ADM.EXE 的修改

```

A>REN ADM.EXE ADM
A>DEBUG ADM
-u 45d,45d
1512:045D E82A02 CALL 068A;注意此部分代码
-a 45d
1512:045D nop
1512:045E nop
1512:045F nop
1512:0460
-w
writing 0AB30 bytes
-q
A>REN ADM ADM.EXE

```

170

PRG 文件结构打印一例

上海铁路局上饶材料厂 许立明

只要对这个程序稍稍瞄上几眼，它的结构层次乃至程序的功能，也就了解的八九不离十了。它不仅按照每层嵌套锯齿形整齐地排列，而且各嵌套的关系又用线段勾画出来。然而，这一切都是该程序本身一手制造的。本人不想对程序作什么解释，因为，程序中的变量名都尽可能使用了汉字，这些变量本身就是程序的一种说明。使用的语句和函数，教材上都已介绍得很清楚。在此本人只想补充说明两点：

一、本程序只能在 FOXBASE 中运行，若想在 DBASE 3 下使用，则要对几个函数做一下变通处理，如 REPL() 函数可以用 DO 循环替代，将 LTRIM() 函数去除。

二、数据库 (YJK.DBF) 中语句 (YJ) 的字段宽度设置为 80，是因为一般程序编制时受屏幕显示宽度的限制，对较长的语句都已用“；”号进行分段处理。如果您所需打印的程序中仍有较长的语句，解决的办法有两个：一是先对您的程序做分段处理；二是修改数据库中的字段宽。否则将造成这些语句的不完整的现象。

.LIST STRU TO PRINT

数据库结构 — 数据库 : C:YJK.DBF

数据库中的数据记录个数 : 0

数据库的最后更新日期 : 08/25/92

字段 字段名 类型 宽度 小数

1 YJ 字符型 80

** 总计 ** 81

C:DJ.PRG 程序结构

文件名=SPAC(20)

CLEA

@ 5,20 SAY "请输入文件名: " GET 文件名

READ

文件名=UPPE(TRIM(LTRIM(文件名)))

IF FILE("&文件名.PRG")

| USE YJK

| ZAP

| APPEND FROM &文件名..PRG SDF

ELSE

| @ 8,20 SAY "没有找到 "+文件名+".PRG"

| QUIT

ENDIF

GO TOP

嵌套层=0

SET DEVI TO PRINT

@ PROW(),12 SAY 文件名+".PRG 程序结构"

DO WHILE ! EOF()

| 语句=UPPE(TRIM(LTRIM(YJ)))

DO CASE

| CASE 语句="DO CASE".OR. 语句="DO WHILE".OR. 语句="IF "

| | @ PROW()+1,0 SAY REPL(" "+SPAC(4),嵌套层)+ " "+语句

| | 嵌套层=嵌套层+1

| CASE 语句="ELSE".OR. 语句="CASE ".OR. 语句="OTHE"

| | @ PROW()+1,0 SAY REPL(" "+SPAC(4),嵌套层-1)+ " "+语句

| CASE 语句="ENDI".OR. 语句="ENDD".OR. 语句="ENDC"

| | 嵌套层=嵌套层-1

| | @ PROW()+1,0 SAY REPL(" "+SPAC(4),嵌套层)+ " "+语句

| OTHE

| IF 嵌套层=0

| | @ PROW()+1,2 SAY 语句

| ELSE

| | @ PROW()+1,0 SAY REPL(" "+SPAC(4),嵌套层)+语句

| ENDDIF

ENDCASE

SKIP

ENDDO

@ 0,0 SAY "

SET DEVI TO SCRE

ZAP

retu

171

广告索引

封面: 珠海经济特区科达电源设备厂

封二: 广州市万利科技广场

封三: 广利电脑设备厂

封底: 广州市大恒科技经营部

3. 广州电子设备公司

4. 星晨电脑

5. 广州八一通讯集团、广州星晨企业发展总公司

6. 广州市海谊电子仪器实业公司

7. 广州市新一代科技发展公司

8. 赛宝星河

9. 天河电子

10. 三联电脑贸易部

11. 特强 (广州) 电子有限公司

12. 广州科星电子技术公司

13. 广州袖珍计算机技术服务中心经营部

14. 清华大学科学馆

15. 华力科技开发公司

16. 艾西显示设备有限公司

17. 华粤电子系统公司

172

动画图形设计中的位图像操作与掩模技术

安徽财贸学院计算中心 鄂大伟

【摘要】当动画目标在背景中移动时,一般采用 XOR 异或方式恢复被运动物体所遮盖的背景图形,本文针对位图像 XOR 后所产生的颜色失真与畸变问题,采用双掩模混合技术较好地实现了动画目标的正常显示、背景屏幕的保存与恢复等过程。

在所有的动画显示中,尽管模拟运动的方式有多种,但其基本原理都是相同的,大部分的动画目标(Object)一般都有以下运动周期:在一位置上画出这个目标,然后擦掉它并将其重画在相邻位置上,并重复以上过程。使其静止的物体图形看起来似乎在移动,于是便产生了动画效果。由于这个周期依赖于所生成图像的方法,故实现过程和编程技术也是多样化的。

动画显示中最基本和常用的技术便是位图像操作(Bit image Block 或称位块),它可把屏幕上一个矩形区域的图形复制到 Buffer(缓冲区)中,也可以从 Buffer 中将位块取出重现在屏幕上,而且特别重要的是可以通过位图像逻辑运算来更新像素,以产生所需要的效果。到目前为止,目标图像的擦除与重现的最快和最简单的方法是采用 XOR(异或)方式,即将目标位块中的每个像素点和屏幕上相应的像素点作 XOR,使得目标变得可见,为了擦掉这一目标,只需简单地再将它跟屏幕 XOR 一次,背景中被目标所遮盖的部分又可以还原成原来状态。利用 XOR 运算的这一特性,我们可以使目标在同一屏幕位置连续作两次 XOR,而不必担心背景屏幕的储存与还原问题。然而令人遗憾的是,XOR 操作的最终像素的颜色要由目标和背景颜色二者决定,有时并不能获得令人满意的效果,尤其是当目标与网格背景或复杂背景相 XOR 时,运算后其像素值可能会发生不可预料的变化,从而使目标的颜色发生改变,轮廓变得模糊不清,大大影响了显示效果。当然,目标在与像素值为 0 的背景 XOR 时,目标将保留原像素值,颜色不会产生畸变。

幸运的是,关于以上问题的出现,我们还可以采

用另一种巧妙的方法加以解决,就是利用位图像的 AND 和 XOR 运算,对被移动的目标事先制作好两个特殊的掩模(mask),我们称之为 AND 掩模和 XOR 掩模,这两个掩模的图形制作方式是不同的,AND 掩模要求目标轮廓内位图像各点像素值为 0,而 XOR 掩模应按目标的实际颜色去作,并将这两个掩模图形保存到 Buffer 中去。当一个目标在背景中移动时,先将目标的 AND 掩模与屏幕相“与”,其结果在背景当前位置产生一个像素值为 0 的目标轮廓空洞,然后再用目标的 XOR 掩模与 AND 掩模相“异或”,这两个掩模经混合后,目标以其正常颜色出现在屏幕上。这样,通过采用目标位图像的双掩模混合技术,当移动目标经过背景时,将不会出现目标颜色发生变化的情况。

在掩模动画技术的实际编程中,为了清除目标和恢复背景,必须把目标所占据的背景屏幕事先保存起来,在目标位置变化过程中,物体实际上是通过所存贮的屏幕内容复盖的方法被清除的,然后将目标的下一个位置的屏幕内容存入 Buffer 中,并把目标显示在这个新的位置上。

诚然,采用掩模技术实现的动画过程比较复杂,由于要处理多种屏幕图像,所以处理时间上也较之 XOR 方式要费时,尤其是当移动目标较大时会降低动画效果,在实际应用中目标的位图像区域应限制在一定的范围内。作为示例,下面给出一个用 Turbo C 实现的 mask 动画演示程序,程序运行后,在屏幕中出现繁星闪烁的星际空间,由经纬线组成的蔚蓝色地球镶嵌在太空中,一颗卫星缓缓从屏幕中飞过,整个画面生动美观,同时显示出掩模技术在动画设计中的魅力。

```

/* FILENAME: MASKDEMO.C */

#include <graphics.h>
#include <alloc.h>
#include <stdlib.h>

#define STEP 5
#define DELAY 150
#define SIZE 25
#define YELLOW 14
#define RED 4

void *xormask, *andmask, *red;
void Get-mask(int, int, int, int);
void PutStar(void);

main()
{
    int gmode=VGAHI, gdriver=VGA, midx, i, cx, cy;
    initgraph(&gdriver, &gmode, "\\tc\\bgi");
    cy=getmaxy()/2;
    covered=malloc(imagesize(cx, cy, cx+SIZE, cy+SIZE));
    xormask=malloc(imagesize(cx, cy, cx+SIZE, cy+SIZE));
    andmask=malloc(imagesize(cx, cy, cx+SIZE, cy+SIZE));

    getimage(cx, cy, cx+SIZE, cy+SIZE, covered);
    setfillstyle(SOLID-FILL, getmaxcolor());
    bar(cx, cy, cx+SIZE, cy+SIZE);
    Get-mask(cx, cy+15, 0, 0);
    getimage(cx, cy, cx+SIZE, cy+SIZE, andmask);

    putimage(cx, cy, covered, COPY-PUT);
    Get-mask(cx, cy+15, YELLOW, RED);
    getimage(cx, cy, cx+SIZE, cy+SIZE, xormask);
    putimage(cx, cy, xormask, XOR-PUT);

    Draw-Earth();
    PutStar();
    for (i=0; i<600; i+=STEP) {
        putimage(cx+i, cy, covered, COPY-PUT);
        getimage(cx+STEP+i, cy, cx+SIZE+STEP+i, cy+SIZE, covered);
    };
    putimage(cx+STEP+i, cy, andmask, AND-PUT);
    putimage(cx+STEP+i, cy, xormask, XOR-PUT);
    delay(DELAY);
    closegraph();
    return (0);
}

void Get-mask(x, y, color1, color2)
{
    moveto(x+10, y);
    setcolor(color1);
    linerel(-3*3, 2*4);
    moveto(x+10, y);
    linerel(-3*3, -2*4);
    moveto(x+25, y);
    linerel(-5*5, 0);
    setfillstyle(1, color2);
    fillellipse(x+13, y, 8, 8);
}

void PutStar(void)
{
    int seed = 1993;
    int i, dotx, doty, h, w, color, maxcolor;
    maxcolor=getmaxcolor();
    w=getmaxx();
    h=getmaxy();
    srand( seed );
    for( i=0 ; i<5000 ; ++i ){
        dotx = 1 + random( w - 1 );
        doty = 1 + random( h - 1 );
        color = random( maxcolor );
        putpixel( dotx, doty, color );
        srand( seed );
    }
}

Draw-Earth()
{
    int midx, midy, i;
    midx=getmaxx()/2;
    midy=getmaxy()/2;
    setbkcolor(BLACK);
    setcolor(CYAN);
    for (i=0; i<=13; i++) {
        ellipse(midx, midy, 0, 360, 100, 100-8*i);
        ellipse(midx, midy, 0, 360, 100-8*i, 100);
        setcolor(RED);
        setlinestyle(SOLID-LINE, 0, THICK-WIDTH);
        ellipse(midx, midy, 130, 50, 160, 30);
    }
}

```