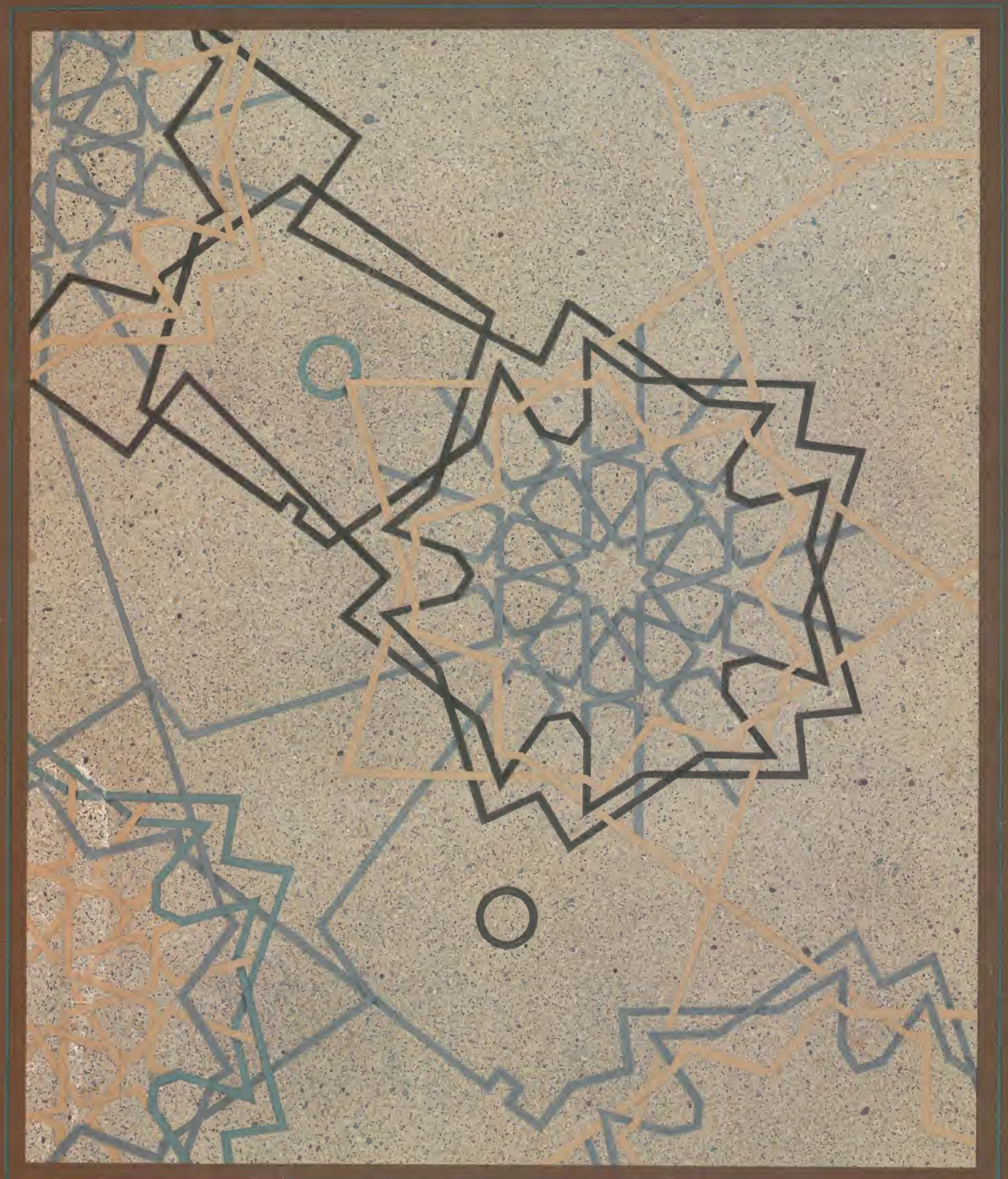
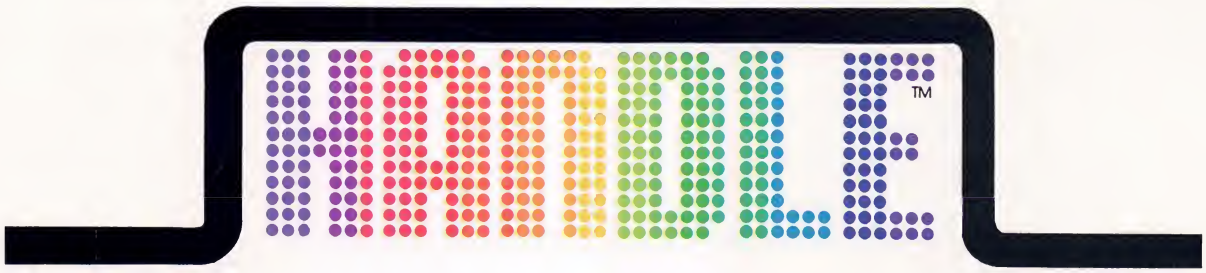
UNIX™ REVIEW

THE PUBLICATION FOR THE UNIX™ COMMUNITY

AUGUST 1985 $3.95



DATABASE INTRICACIES



Modular.
Integrated.
Now.

Handle Writer/Spell™
Word processing with integrated spelling correction and verification.

Handle Calc™
Spreadsheet with up to 32,000 rows and columns. Conditional and iterative recalculation.

The **Handle Office-Automation Series** is a powerful set of modular, integrated software tools developed for today's multiuser office environment. **Handle** application modules can be used stand-alone or combined into a fully integrated system.

The **Handle Office-Automation Series** modules offer:

- Ease of Use and Learning
- Insulation from **UNIX**
- Data Sharing Between Multiple Users
- Data Integration Between Modules
- Data Sharing with Other Software Products
- Sophisticated Document Security System

Handle Technologies, Inc.

Corporate Office
6300 Richmond
3rd Floor
Houston, TX 77057
(713) 266-1415

Sales and Product Information
850 North Lake Tahoe Blvd.
P.O. Box 1913
Tahoe City, CA 95730
(916) 583-7283

TM—HANDLE, HANDLE HOST, HANDLE WRITER, HANDLE SPELL, HANDLE WRITER/SPELL, and HANDLE CALC ARE TRADEMARKS OF HANDLE TECHNOLOGIES, INC.
TM—UNIX IS A TRADEMARK OF AT&T BELL LABORATORIES.

file manager. And C-ISAM,™ the de facto standard ISAM for UNIX. It's built into all our products, but you can buy it separately.

And when you choose RDS, you'll be in the company of some other good companies. Computer manufacturers including AT&T, Northern Telecom, Altos and over 60 others. And major corporations like Anheuser Busch, First National Bank of Chicago and Pacific Bell.

Which makes sense. After all, only RDS offers a family of products that work so well together. As well as with so many industry standards.

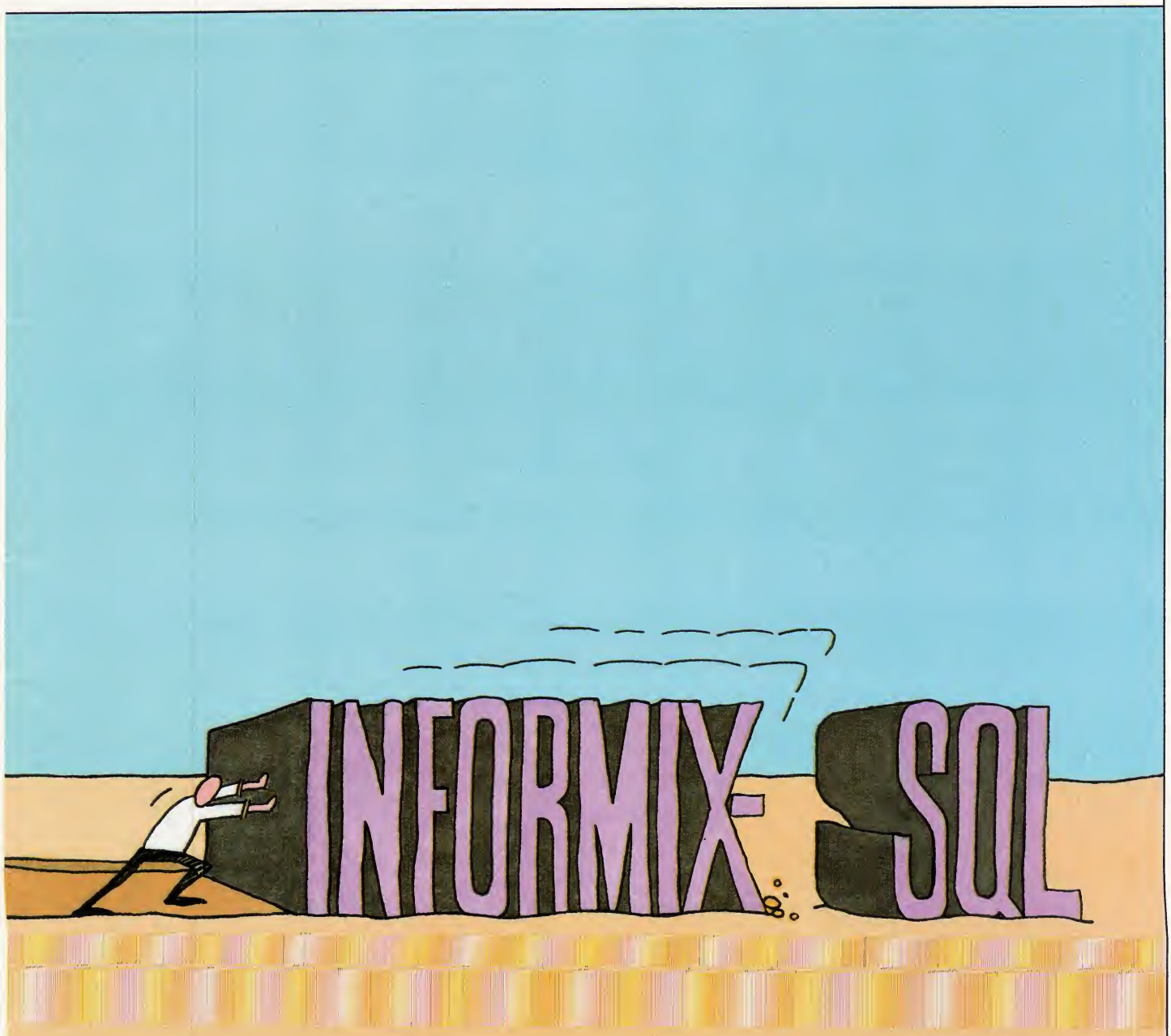
So call us for a demo, a manual and a copy of our Independent Software Vendor Catalog. Software vendors be sure to ask about our new "Hooks" software integration program. Our number: 415/424-1300.

Or write RDS, 2471 East Bayshore Road, Palo Alto, CA 94303.

And we'll show you how we took a good idea and made it better.

RDS™

RELATIONAL DATABASE SYSTEMS, INC.



UNIX™ REVIEW
THE PUBLICATION FOR THE UNIX COMMUNITY

Volume 3,
Number 8
August 1985

DEPARTMENTS:

6 **Viewpoint**

8 **The Monthly Report**
By Roger Strukhoff

16 **The Human Factor**
By Richard Morin

66 **Rules of the Game**
By Glenn Groenewold

70 **Industry Insider**
By Mark G. Sobell

76 **Devil's Advocate**
By Stan Kelly-Bootle

80 **C Advisor**
By Bill Tuthill

87 **The UNIX Glossary**
By Steve Rosenthal

92 **Recent Releases**

104 **Calendar**

108 **Advertiser's Index**

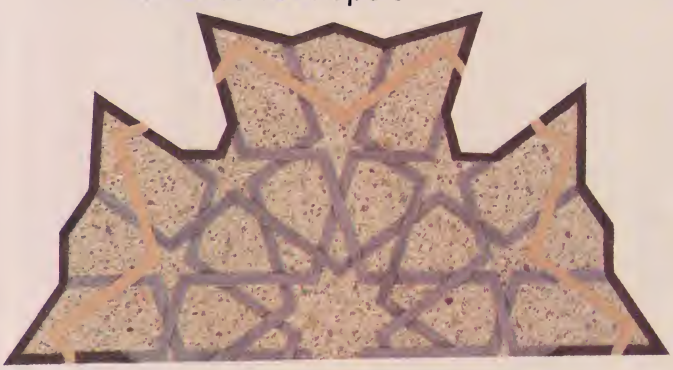
Cover art by Ivy Nichols

FEATURES:

24 **DATA MANAGEMENT OVERVIEW**

By Eric Allman

A survey of the challenges, possible approaches, and goals facing database developers.



34 **THE DBMS-UNIX MATCH**

By Roger Sippl

It may not be a match made in heaven, but it is workable—and critical to the commercial success of UNIX.

DATABASE INTRICACIES

42 INTERVIEW WITH PETER WEINBERGER

By Ned Peirce

One of Bell Labs' foremost database gurus calls them as he sees them.

51 BACKEND DATABASE MACHINES

By Paula Hawthorn

The whys and wherefores of database work on dedicated machines.

58 TRANSACTION PROCESSING

By Kathryn Anderson

Not only is TP possible under UNIX—it can actually be done without kernel modifications.

UNIX REVIEW (ISSN-0742-3136) is published monthly by REVIEW Publications Co. It is a publication dedicated exclusively to the needs of the UNIX community. Second class postage paid at Renton, WA 98055 and at additional mailing offices. POSTMASTER: Please send Form 3579 to UNIX REVIEW, 500 Howard Street, San Francisco, CA 94105.

Entire contents copyright 1985. All rights reserved and nothing may be reproduced in whole or in part without prior written permission from UNIX REVIEW.

Subscriptions to UNIX REVIEW are available at the following annual rates (12 issues): US$28 in the US; US$35 in Canada; US$48 in all other countries/surface mail; US$85 in all other countries/airmail. Correspondence regarding editorial (press releases, product announcements) and circulation (subscriptions, fulfillment, change of address) should be sent to 500 Howard Street, San Francisco, CA 94105. Telephone 415/397-1881. Correspondence regarding dealer sales should be sent to 901 South 3rd Street, Renton, WA 98055. Telephone 206/271-9605.

Letters to UNIX REVIEW or its editors become the property of the magazine and are assumed intended for publication and may so be used. They should include the writer's full name, address and home telephone number. Letters may be edited for the purpose of clarity or space. Opinions expressed by the authors are not necessarily those of UNIX REVIEW.

UNIX is a trademark of Bell Laboratories, Inc. UNIX REVIEW is not affiliated with Bell Laboratories.

PUBLISHER:
Pamela J. McKee
ASSOCIATE PUBLISHERS:
Ken Roberts, Scott Robin
EDITORIAL DIRECTOR:
Stephen J. Schneiderman
EDITOR:
Mark Compton
ASSOCIATE EDITORS:
Roger Strukhoff
David Chandler
EDITORIAL ADVISOR:
Dr. Stephen R. Bourne, Consulting Software Engineer, Digital Equipment Corporation.
EDITORIAL REVIEW BOARD:
Dr. Greg Chesson, Chief Scientist, Silicon Graphics, Inc.
Larry Crume, Director, AT&T UNIX Systems Far East
Ted Dolotta, Senior Vice President of Technology, Interactive Systems Corporation
Gene Dronek, Director of Software, Aim Technology
Ian Johnstone, Project Manager, Operating Software, Sequent Computer Systems
Bob Marsh, Chairman, Plexus Computers
John Mashey, Manager, Operating Systems, MIPS Computer Systems
Robert Mitze, Department Head, UNIX Computing System Development, AT&T Bell Labs
Deborah Scherrer, Computer Scientist, Mt. Xinu
Jeff Schriebman, President, UniSoft Systems
Rob Warnock, Consultant
Otis Wilson, Manager, Software Sales and Marketing, AT&T Information Systems
HARDWARE REVIEW BOARD:
Gene Dronek, Director of Software, Aim Technology
Doug Merritt, Technical Staff, International Technical Seminars, Inc.
Richard Morin, Consultant, Canta Forda Computer Laboratory
Mark G. Sobell, Consultant
SOFTWARE REVIEW BOARD:
Ken Arnold, Consultant, UC Berkeley
Jordan Mattson, Programmer, UC Santa Cruz
Dr. Kirk McKusick, Research Computer Scientist, UC Berkeley
Doug Merritt, Technical Staff, International Technical Seminars, Inc.
Mark G. Sobell, Consultant
CONTRIBUTING EDITOR:
Ned Peirce, Systems Analyst, AT&T Information Systems
PRODUCTION DIRECTOR:
Nancy Jorgensen
PRODUCTION STAFF:
Cynthia Grant, Tamara V. Heimarck, Florence O'Brien, Barbara Perry, Denise Wertzler
BUSINESS MANAGER:
Ron King
CIRCULATION DIRECTOR:
Wini D. Ragus
CIRCULATION MANAGER:
Jerry M. Okabe
MARKETING MANAGER:
Donald A. Pazour
OFFICE MANAGER:
Tracey J. McKee
TRAFFIC:
James A. O'Brien, Manager
Tom Burrill, Dan McKee, Corey Nelson
NATIONAL SALES OFFICES:
500 Howard St.
San Francisco, CA 94105
(415) 397-1881
Regional Sales Manager:
Collen M. Y. Rodgers
Sales/Marketing Assistant:
Anmarie Achacoso

370 Lexington Ave.
New York, NY 10017
(212) 683-9294
Regional Sales Manager:
Katie A. McGoldrick

BPA membership applied for in March, 1985.

VIEWPOINT

Getting down to business

People who endorse UNIX as a business solution have learned to expect snickers and raised eyebrows for their trouble. It's not difficult to see why.

Despite a surge in applications development over the past 18 months, UNIX is still commonly perceived as the exclusive playground of the technically sophisticated. Databases somehow do not fit into that picture.

The fact of the matter, though, is that database management is possible in the UNIX environment. A large body of software—over 50 UNIX DBMS products by a recent count—testifies to that. Nevertheless, reservations persist for good reason.

First, there is the hierarchical file structure of UNIX. Despite its elegance and overall functionality, it fails to meet the specific needs of relational database software. To compound matters, UNIX lacks standardized file and record locking.

All is not lost, however. Clever software developers have managed to work around the operating system's limitations to produce DBMS software that suffers neither for lack of functionality nor performance.

This issue of UNIX REVIEW explores some of the strategies employed by those developers. Eric Allman lays out the challenge in the introductory article, which he begins with a discussion of the evolution of data management and concludes with a survey of desirable database features.

Roger Sippl, president of Relational Database Systems Inc., follows with a piece that details how UNIX succeeds and fails in meeting the needs of data management software. A few modest proposals are interspersed.

One of the strategies Roger introduces is the use of a backend machine. Paula Hawthorn expands on this notion in the article that follows. Using Britton Lee's Intelligent Database Machine as an example, she discusses how the use of a separate dedicated processor for database work can improve DBMS performance, even while it frees up CPU cycles for other tasks.

Another piece, authored by Kathryn Anderson, probes strategies that can be used to facilitate transaction processing. Although TP is purportedly a bugaboo topic in UNIXland, Kathryn tells how System V can support robust transaction applications—without resorting to kernel modifications. She focuses largely on the topics of control and tunability.

Ned Peirce closes out the theme with an interview of Peter Weinberger, the head of Computer Science Research at AT&T Bell Labs. Peter's database work under UNIX has become well known in development circles and his comments on the suitability of UNIX for database work carry the tone of authority.

As you may have guessed, all voices in the issue support UNIX as a database environment, although it's acknowledged that some adjustments are necessary. This should be heartening to those who wish to see UNIX succeed in the business community, because database software will be the foundation upon which much of tomorrow's business applications will be built.

Mark Compton

How to go from UNIX to DOS without compromising your standards.

It's easy. Just get an industry standard file access method that works on both.

C-ISAM™ from RDS.

It's been the UNIX™ standard for years (used in more UNIX languages and programs than any other access method), and it's fast becoming the standard for DOS.

Why?

Because of the way it works. Its B+ Tree indexing structure offers unlimited indexes. There's also automatic or manual record locking and optional transaction audit trails. Plus index compression to save disk space and cut access times.

How can we be so sure C-ISAM works so well? We use it ourselves. It's a part of INFORMIX,® INFORMIX-SQL and File-it!,™ our best selling database management programs.

For an information packet, call (415) 424-1300. Or write RDS, 2471 East Bayshore Road, Palo Alto, CA 94303.

You'll see why anything less than C-ISAM is just a compromise.

RELATIONAL DATABASE SYSTEMS, INC.

© 1985, Relational Database Systems, Inc. UNIX is a trademark of AT&T Bell Laboratories. INFORMIX is a registered trademark and RDS, C-ISAM and File-It! are trademarks of Relational Database Systems, Inc.

How we improved Structured Query Language.

Actually, we didn't change a thing.

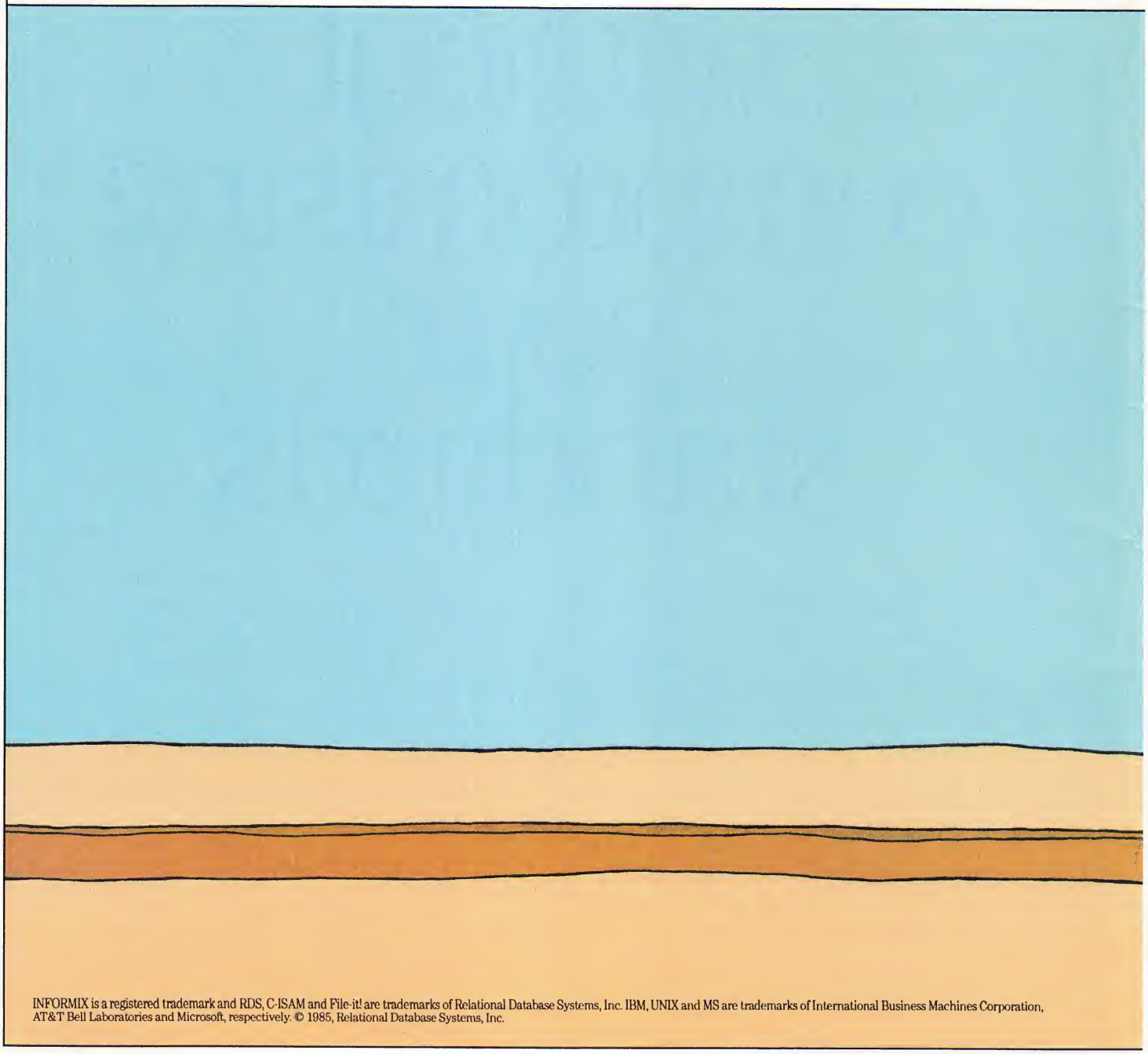
We just combined it with the best relational database management system.

Introducing INFORMIX®-SQL.

It runs on either UNIX™ or MS™-DOS operating systems. And now with IBM's SQL as part of the program, you can ask more of your database. Using the emerging industry-standard query language.

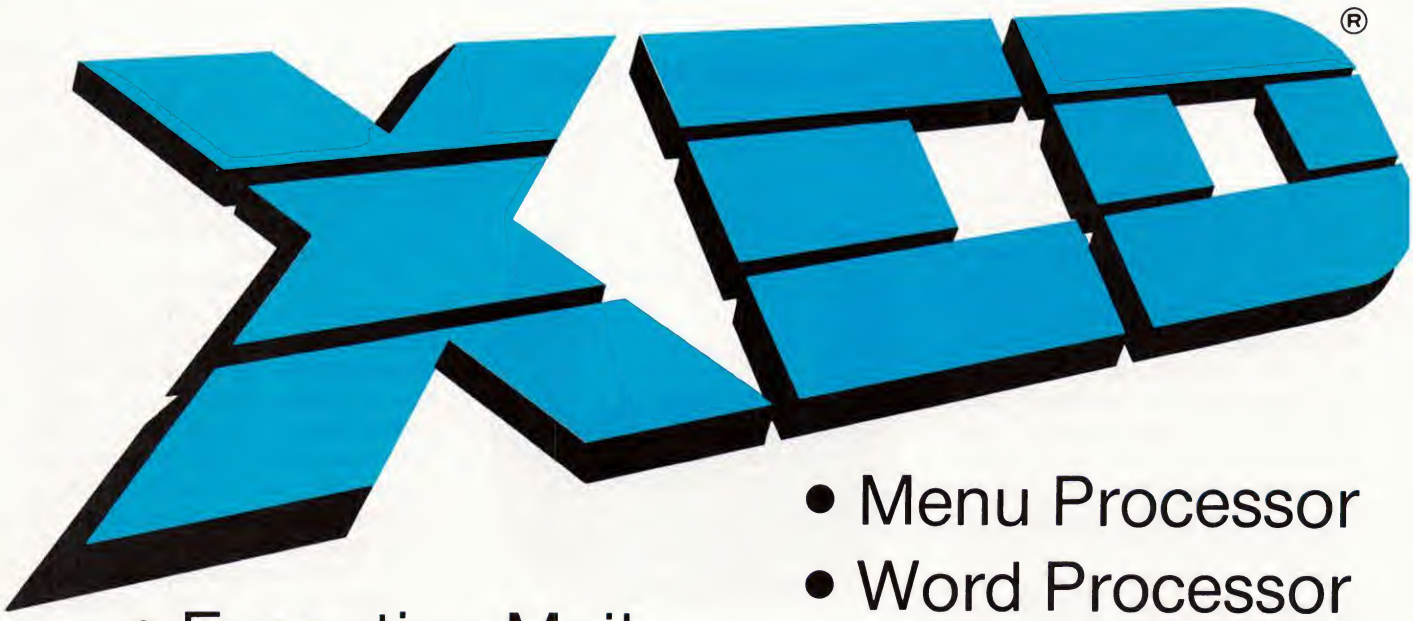
To make your job easier, INFORMIX-SQL comes with the most complete set of application building tools. Including a full report writer and screen generator. Plus a family of companion products that all work together.

Like our embedded SQLs for C and COBOL. So you can easily link your programs with ours. File-it!™ our easy-to-use



INFORMIX is a registered trademark and RDS, C-ISAM and File-it! are trademarks of Relational Database Systems, Inc. IBM, UNIX and MS are trademarks of International Business Machines Corporation, AT&T Bell Laboratories and Microsoft, respectively. © 1985, Relational Database Systems, Inc.

The First Name In Integrated Office Automation Software



- Executive Mail
- Telephone Directory
- Menu Processor
- Word Processor
- Forms/Data Base
- Spreadsheet

Certified and Deliverable Since 1981

XED was the first independent software company to introduce a Unix WP package and achieved early success by selling to the government and international market (XED is the only Unix WP package to meet government specifications). Worldwide sales of XED rank Computer Methods first in both sales and units installed in 1984.

XED® INTEGRATED OFFICE SOFTWARE

Box 3938 • Chatsworth, CA 91313 U.S.A. • (818) 884-2000
FAX (818) 884-3870 • Intl. TLX 292 662 XED UR

XED is a registered trademark of CCL Datentechnik AG
UNIX is a trademark of AT & T Bell Laboratories, Inc.

Circle No. 60 on Inquiry Card

THE MONTHLY REPORT

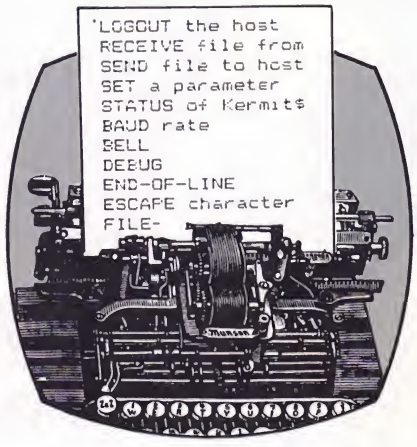
Pyramid power

by Roger Strukhoff

One of the more impressive machines making its debut at the National Computer Conference in Chicago July 16 was Pyramid Technology Corp.'s 98x, a superminicomputer built with the idea of equal opportunity for equal microprocessors. Pyramid calls the 98x an "isoprocessor" system; there are two 32-bit RISC (reduced instruction set computer) processors working as equal partners. If one processor fails, the system continues to run at about 60 percent of capacity.

The 98x supports up to 98 users, comes with as much as 16 MB of main memory, and offers disk drives with nearly a gigabyte of capacity ("only" 940 MB actually). The machine uses Pyramid's OSx operating system, a dual port (or "dualPort", in Pyramid parlance) of UNIX, incorporating both 4.2BSD and System V.

Charles Krahling, Pyramid's director of marketing, says his company is fully aware of the traditionally weak I/O of boxes based on UNIX, and so has developed a new I/O subsystem for the 98x. "I/O (in any system) can always use improvement," Krahling noted. "Plus, with the powerful disks we see coming out in the near future, we needed to develop a stronger I/O system for our machines." The subsystem has what Pyramid calls an Intelligent I/O Processor (IOP), with an aggregate throughput capability (PTAL bandwidth) of 11 MB per second. The IOP also provides disk rotational sensing and overlapped seeks.



The idea behind isoprocessing comes from Purdue's dual processor implementation, Krahling said. CPU balance is achieved through the use of a proprietary semaphore system that protects critical sequences of code and controls simultaneous access to kernel data structures. The semaphore is designed to complement standard UNIX process synchronization concepts, allowing symmetric multiprocessor support without the need to implement major structural changes to the UNIX kernel. Normal UNIX organization is maintained, but both CPUs share a single copy of the OSx kernel and data structures, and have equal access to all shared resources.

The isoprocessors are implemented in a fast Shottky TTL, and have a cycle time of only 100 nanoseconds. Each isoprocessor has an instruction unit (I-unit), an execution unit (E-unit), a 32K byte data cache, and a microcode sequencer. The I-units prefetch instructions from a 4K byte instruction cache and take operands from register stacks or immediate fields. The pipelining architecture of the 98x overlaps I-unit and E-unit activity.

RISC technology is, of course, key to the system's performance. Each CPU uses 528 registers of 32 bits each, implemented in stack form with 16 levels of 32 registers, plus 16 global registers. The register stack allows parameters to be passed between stack levels without data being moved.

The 98x will be shipped in October, according to Krahling. Prices vary between $260,000 and $500,000. Field upgrades for 90x users will cost about $90,000. Pyramid also has an extended system in the works, the 98xE, which will add an I/O expansion bay to support a maximum of 256 users.

Pyramid began shipping computer systems in October 1983. It had gross sales of $12 million in 1984, and is anticipating about

COHERENT™ IS SUPERIOR TO UNIX* AND IT'S AVAILABLE TODAY ON THE IBM PC.

Mark Williams Company hasn't just taken a mini-computer operating system, like UNIX, and ported it to the PC. We wrote COHERENT ourselves. We were able to bring UNIX capability to the PC with the PC in mind, making it the most efficient personal computer work station available at an unbelievable price.

For the first time you get a multi-user, multitasking operating system on your IBM PC. Because COHERENT is UNIX-compatible, UNIX software will run on the PC under COHERENT.

The software system includes a C-compiler and over 100 utilities, all for $500. Similar environments cost thousands more.

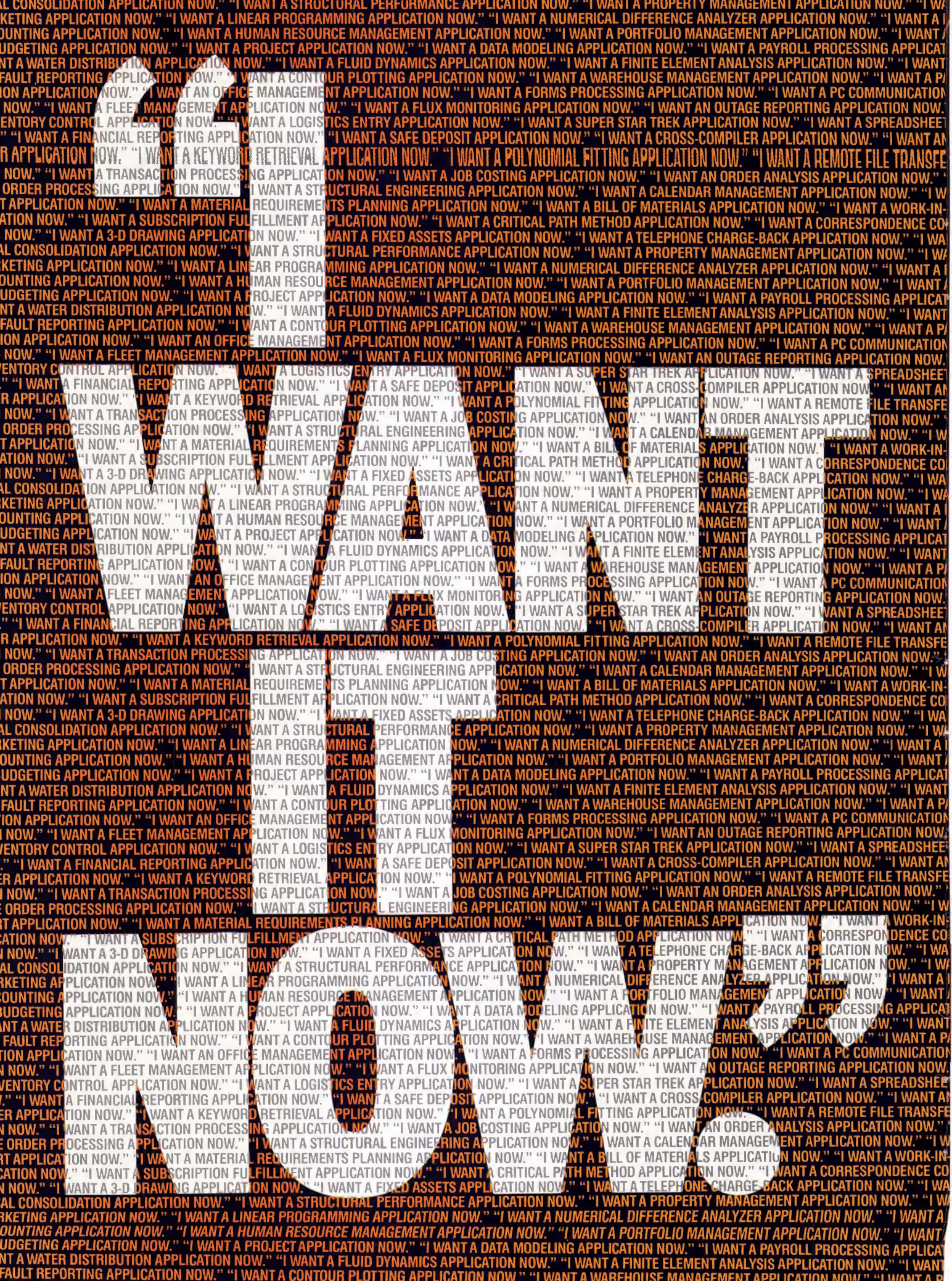
COHERENT on the IBM PC requires a hard disk and 256K memory. It's available on the IBM XT, and Tecmar, Davong and Corvus hard disks.

Available now. For additional information, call or write,

Mark Williams Company
1430 West Wrightwood, Chicago, Illinois 60614
312/472-6659

COHERENT is a trademark of Mark Williams Company.
*UNIX is as trademark of Bell Laboratories.

Circle No. 16 on Inquiry Card


"I WANT A STRUCTURAL PERFORMANCE APPLICATION NOW." "I WANT A PROPERTY MANAGEMENT APPLICATION NOW." "I WANT A LINEAR PROGRAMMING APPLICATION NOW." "I WANT A NUMERICAL DIFFERENCE ANALYZER APPLICATION NOW." "I WANT A HUMAN RESOURCE MANAGEMENT APPLICATION NOW." "I WANT A PORTFOLIO MANAGEMENT APPLICATION NOW." "I WANT A PROJECT APPLICATION NOW." "I WANT A DATA MODELING APPLICATION NOW." "I WANT A PAYROLL PROCESSING APPLICATION NOW." "I WANT A WATER DISTRIBUTION APPLICATION NOW." "I WANT A FLUID DYNAMICS APPLICATION NOW." "I WANT A FINITE ELEMENT ANALYSIS APPLICATION NOW." "I WANT A CONTOUR PLOTTING APPLICATION NOW." "I WANT A WAREHOUSE MANAGEMENT APPLICATION NOW." "I WANT AN OFFICE MANAGEMENT APPLICATION NOW." "I WANT A FORMS PROCESSING APPLICATION NOW." "I WANT A PC COMMUNICATION NOW." "I WANT A FLEET MANAGEMENT APPLICATION NOW." "I WANT A FLUX MONITORING APPLICATION NOW." "I WANT AN OUTAGE REPORTING APPLICATION NOW." "I WANT A LOGISTICS ENTRY APPLICATION NOW." "I WANT A SUPER STAR TREK APPLICATION NOW." "I WANT A SPREADSHEET" "I WANT A FINANCIAL REPORTING APPLICATION NOW." "I WANT A SAFE DEPOSIT APPLICATION NOW." "I WANT A CROSS-COMPILER APPLICATION NOW." "I WANT A KEYWORD RETRIEVAL APPLICATION NOW." "I WANT A POLYNOMIAL FITTING APPLICATION NOW." "I WANT A REMOTE FILE TRANSFER" "I WANT A TRANSACTION PROCESSING APPLICATION NOW." "I WANT A JOB COSTING APPLICATION NOW." "I WANT AN ORDER ANALYSIS APPLICATION NOW." "I WANT A STRUCTURAL ENGINEERING APPLICATION NOW." "I WANT A CALENDAR MANAGEMENT APPLICATION NOW." "I WANT A MATERIAL REQUIREMENTS PLANNING APPLICATION NOW." "I WANT A BILL OF MATERIALS APPLICATION NOW." "I WANT A WORK-IN-" "I WANT A SUBSCRIPTION FULFILLMENT APPLICATION NOW." "I WANT A CRITICAL PATH METHOD APPLICATION NOW." "I WANT A CORRESPONDENCE" "I WANT A 3-D DRAWING APPLICATION NOW." "I WANT A FIXED ASSETS APPLICATION NOW." "I WANT A TELEPHONE CHARGE-BACK APPLICATION NOW." "I WANT A CONSOLIDATION APPLICATION NOW." "

VMS. WHEN THE ONLY THING YOUR USERS WANT IS EVERYTHING, NOW.

If "I want it now" seems like the only thing you ever hear from your users – and if your applications backlog is telling you that "now" is going to be a long way off – you need to know about our VMS™ Virtual Memory System software.

Our VMS software, which was designed exclusively with VAX™ computer systems, is far more than a mere operating system. It is a complete operating *environment*. One that can encompass all the ways you use computers. To get it done, now.

The VMS operating environment provides you with the industry's most complete set of utilities and software products for program development and system management. Interactive, realtime, even background batch applications can be developed with ease and speed.

Everything you need for corporate-level applications is included. Advanced office automation software. Proven transaction processing capabilities. Comprehensive CODASYL-compliant and relational database managers. User-friendly query tools and an integrated data dictionary. Plus the industry's most complete set of local and wide area networking facilities, allowing you to exchange data files and cooperate with systems made by IBM – via SNA gateway – as well as many other vendors.

VMS PUTS YOUR PROGRAMMERS IN THE FAST LANE.

You've heard a lot of promises about increasing programmer productivity over the years. After all, the application backlog in most companies has reached 18 months to 2 years. And the demand for new applications is increasing by some 50 percent a year. So there's hardly a more important subject.

Digital's VMS software does more to increase programmer productivity than any other system can. Because it combines everything you need for application design, development and maintenance into a single, integrated operating environment.

You'll have the industry's best symbolic debugger to work with. A command language so flexible it lets you stop application execution and enter new commands at any point, and even develop your own commands through its macro facility. A common runtime environment that lets you reuse code instead of rewriting it. And more advanced languages than any other system – some 16 in all, including Ada©, C, COBOL, FORTRAN, and Lisp – all of which you can combine in a single program through the common calling standard.

This gives you a distinct advantage in team development projects. Each developer can use the language best suited to his or her talents and to the task. Large, complex projects such as transaction processing applications can be designed, developed, maintained and managed easily.

The end result – better code in less time. And satisfied, productive users at every level, from the factory floor to the executive suite.

The point is, many other systems promise you ease of use, up to a certain point. The VMS environment simplifies development not only for small, ad hoc projects – but also when your applications reach highly sophisticated, complex levels.

WITH VMS, FLEXIBILITY IS BUILT IN.

You get another big advantage when you develop your applications with the VMS operating environment: the range of hardware you can run them on. Namely, our VAX computer family, the industry standard for 32-bit computing.

The VMS environment spans the entire VAX systems family, from the smallest MicroVAX™ system to the largest VAX 8600™ and multiprocessor VAXcluster™ systems. This is the biggest software-compatible growth path in the world. And with the data security mechanisms built into VMS software, increased access doesn't mean compromised security.

If you need UNIX© software capabilities, the VMS operating environment can readily provide them through the VNX™ option. And if you need the convenience of applications packages, you'll have over 2,000 to choose from – created for VMS software by Digital and independent vendors. Your choices are never limited when you start with the VMS environment.

BEST ENGINEERED MEANS ENGINEERED TO A PLAN.

Digital's VMS operating environment, like all Digital hardware and software products, is engineered to conform to an overall computing strategy. This means that our systems are engineered to work together easily and expand economically. Only Digital provides you with a single, integrated computing strategy direct from desktop to data center.

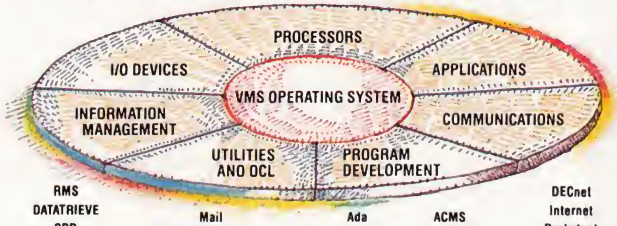
For more information about how our VMS operating environment can help you cut your applications backlog, contact your local Digital sales representative. Or call 1-800-DIGITAL, ext. 219.

THE BEST ENGINEERED COMPUTERS IN THE WORLD.

digital™

Circle No. 59 on Inquiry Card

THE VMS OPERATING ENVIRONMENT



RMS
DATATRIEVE
CDD
DBMS
Rdb/VMS
TDMS
FMS
DECslide
DECgraph
DEC/CMS
DEC/MMS
ACMS
ALL-IN-1
DECalc
DECmail
DECheath
DECspell
VTX

Mail
Text Editors
Librarian
Linker
Symbolic Debugger
Analyze/Media
Monitor
Sort/Merge
Commons Run-Time Library
Help
Set/Show
File Differences
File Transfer
Remote Terminal Capability
Back/Restore
Error Logging and Print

Ada
APL
BASIC
BLISS
C
COBOL
CORAL 66
DIBOL
DSM
FORTRAN
Lisp
Macro
PASCAL
PL/I
RPG II

ACMS
Rdb/VMS
VAX-11 RSX
GKS
RGL
DEC/SHELL
DEC/TEST MANAGER
LANGUAGE SENSITIVE EDITOR
PERFORMANCE AND COVERAGE ANALYZER
DEC/CMS
DEC/MMS

DECnet
Internet
Packetnet
Ethernet
DECnet/SNA Gateway
DATATRIEVE
DX
LAT

© Digital Equipment Corporation 1985. Digital, the Digital logo, VMS, VAX, VNX, MicroVAX, VAX 8600 and VAXcluster are trademarks of Digital Equipment Corporation.
IBM is a trademark of International Business Machines Corporation. Ada© is a trademark of the U.S. Government Department of Defense. UNIX© is a trademark of AT&T Bell Laboratories.

The 98x supermini from Pyramid Technology Corporation.

$35 million in 1985, predicated on the success of the 98x. Pyramid's first system was the 90x, a 32-bit supermini that Krahling claimed was the first RISC machine. The 90x has 2.5 times the power of a VAX 780; a multiprocessor system, the 90Mx, reportedly provides 4.2 times VAX 780 power (and doesn't need to be stored in silos). But the 90Mx has a master-slave multiprocessor environment.

Krahling says there are 5000 VAXen on order, with about 20 percent of that total (1000) running UNIX. Pyramid thinks of itself as the number two supplier in superminicomputers, and will certainly try harder to gain as much of that 1000-system DEC market as it can. It plans to sell 80 percent through its own sales force, according to Krahling.

AT&T CONTINUES SYSTEM V PUSH

AT&T continued its aggressive efforts to promote System V as the "UNIX system for business" by showing off its Remote File System (RFS) at the Usenix Conference in Portland. Demonstrated on 3B2 hardware, the system was described as "*Streams*-based", and capable of providing "protocol and media independence" within System V.

The Remote File System provides transparent access to directories, files, special devices, and named pipes. An administrator may select directories in the local file tree to *advertise* to remote machines (by using the **adv** command). Other users can then use the **mount** command to mount advertised directories at any place in their local file tree (see Figure 1). Its name server features machine-independent resource naming using a hierarchical *domain*-naming convention. Its recovery mechanism ensures that no single machine failure will bring down the network.

Administrative support will include: selective resource sharing that allows administrators to advertise any directory for read/write or read-only access, and to provide lists of machines authorized to mount specific directories; machine authentication, allowing administrators to determine passwords to aid in identification of systems requesting remote mounts; user and group ID mapping (providing such options as the ability to map all remote IDs to a single remote ID, thus maintaining identical IDs across a group of machines—with selected exclusions—and the use of explicit mapping tables); and a monitor that allows report generation to separate remote from local activity.

The Remote File System provides transparent access to directories, files, special devices, and named pipes.

The rumor is that AT&T will release the Remote File System on or before the Anaheim UniForum show in February 1986.

A company already in the distributed file system arena, Sun Microsystems, is undaunted by the recent AT&T developments. Bill Keating, marketing manager for Sun's Network File System (NFS), first pointed out that "this is not the first time (AT&T has) talked about (their remote file system). Who knows when it will really be available?" Keating pointed out that Sun's NFS can be implemented with non-UNIX systems as well. "Our intent is to provide a system that allows interconnection among heterogeneous computers and operating systems. We're after DEC, and we're after the IBM PC, just to name a couple." He referred to NFS as "a superset" of distributed file systems because of its ability to be implemented across the computer spectrum.

But design flexibility is not as important as third-party support these days, particularly when the competition comes from a company the size of AT&T. Keating conceded that, saying that Sun "will become compatible with whatever AT&T offers," but he also said Sun "is trying to make NFS a standard (in its own right) by getting other companies to adopt it." He said Celerity Computing has been added to the list of NFS supporters—a list that already included Pyramid, Gould, and Mt. XINU. The latter announced a few months ago that it would implement NFS on VAXen running 4.2BSD.

For its part, AT&T claims not to be in direct competition with Sun. Spokesperson Lawrence Brown said, "(NFS and the Remote File System) are really targeted at

NAME THE MOST WIDELY USED INTEGRATED OFFICE AUTOMATION SOFTWARE FOR UNIX™ SYSTEMS.

"UNIPLEX II"™

YOU'VE GOT IT!

User satisfaction is the primary reason no other product can make this claim. Already in its second generation, UNIPLEX II offers features designed to meet the requirements of the most demanding user.

The beauty of UNIPLEX II is its simplicity. One personality and one command structure throughout the program provide an ease of use never before experienced with UNIX application software.

UNIPLEX II integrates sophisticated word processing, spreadsheet, and relational database applications into a powerful one-product solution.

UNIPLEX II uses termcap, so it can run on virtually any computer terminal. "Softkeys" allow the user to define function keys which are displayed on the 25th line of most terminals to provide versatility and ease of use.

All this at a price you'd normally pay for a single application software package.

UNIPLEX II is available immediately from UniPress Software, the company that's been at the forefront of quality UNIX software products longer than anyone else.

Call today! Once you've got it, you'll see why UNIPLEX II is the most widely used integrated office automation software for UNIX-based systems.

OEM terms available. Mastercard and Visa accepted!

Write to: UniPress Software, 2025 Lincoln Hwy., Edison, NJ 08817 or call: 1-800-222-0550 (outside NJ) or 201-985-8000 (in NJ); Telex: 709418. European Distributor: Modulator SA, Switzerland 41 31 59 22 22, Telex: 911859.

UNIX is a trademark of AT&T Bell Laboratories. Uniplex II is a trademark of Uniplex Integration Systems.

NOW AVAILABLE ON THE AT&T UNIX PC 7300 & 3B SERIES!

UniPress Software

Your Leading Source for UNIX™ Software.

Circle No. 61 on Inquiry Card

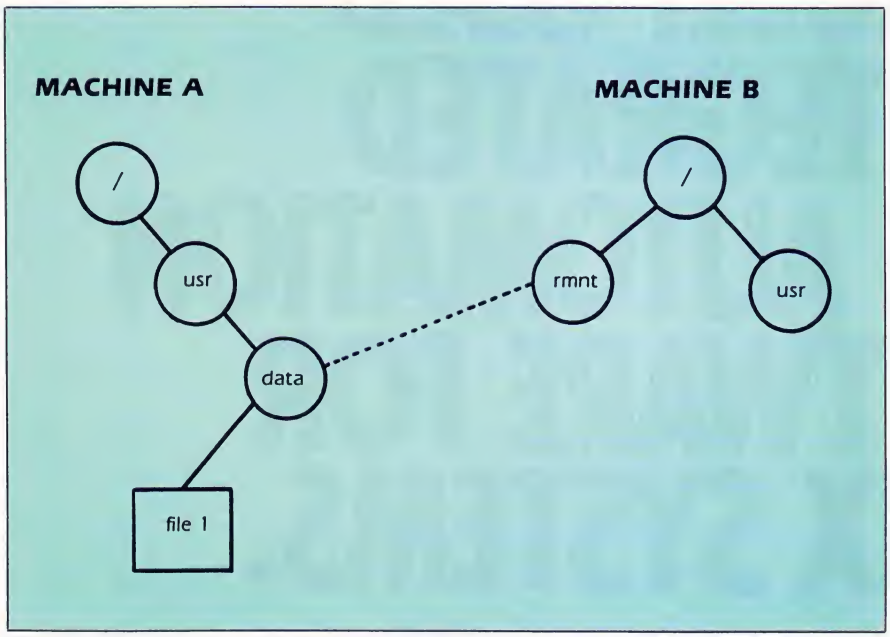


Figure 1 — *Diagram representing the transparent access capability of AT&T's Remote File System.*

different markets." Brown thinks Sun is more interested in "the engineering workstation" than the multiuser business office system stressed by AT&T—a natural assumption given Sun's commitment to 4.2BSD. Brown praised NFS as a "well-designed system". But he pointed out that the Remote File System will not "sacrifice UNIX functionality," as is the case with NFS.

THE REAL WORLD LOOMS

There was a scene in the movie *Woodstock* where well-known Hog Farmer and Berkeley-San Francisco cultural icon Wavy Gravy implored those who "don't think capitalism is too weird" to buy some hot dogs from a poor soul whose tube steak stand had burned to the ground. Certainly, a similar feeling was in the air as hackers mingled in Portland at what is becoming an increasingly commercial Usenix Conference.

The dreaded "C-words", capitalism and commercialism, are inexorably wending their way into the UNIX community. One needed look no further than the AT&T army swarming about its dominant booth in the center of the exhibition. Even some of the technical sessions were more like "the gospel according to (enter company name)" than objective discourses. To be sure, some of the sessions were still of the old school. There was substance and humor enough for those who were persistent enough to ferret it out.

The dreaded "C-words", capitalism and commercialism, are inexorably wending their way into the UNIX community.

But there's big bucks to be had in UNIX systems, and like Wavy Gravy, there's more than a couple of folks who think that latching onto a few of those bucks might not be so weird after all.

CRAY-2 RUNS SYSTEM V

Having $17.6 million doesn't make you as rich as it used to, but it's enough to make you the proud owner of a Cray-2 supercomputer, the new UNIX-based monster from Cray Research, Inc. The Cray-2 is designed to deliver as much as 12 times the performance of the Cray 1 and is the first (and only) system of this size to run System V. Previously, Cray ran its own operating system, COS, on its systems. (The company continues to support COS, but it's also reportedly planning to offer UNIX on its XMP computer sometime next year.)

The Cray-2 can consume just about anything for breakfast. Its clock cycle is 4.1 nanoseconds, and main memory takes in 256 million 64-bit words, or about 1.6 gigabytes. Main memory megabytes are divided into four quadrants with 128 interleaved banks. The CPU has one foreground and four background processors. The four background processors, each said to be more powerful than the Cray-1 CPU, perform scalar and vector calculations, and can operate either independently or jointly. The CPU has 320 plug-in modules, each holding 750 integrated circuit packages. There are about 240,000 chips in each module.

To keep all this horsepower cool, a liquid-immersion, fluorocarbon-based cooling system is used. The inert fluid is circulated throughout the CPU cabinet, where it comes into direct contact with the integrated circuit packages.

Roger Strukhoff is the Associate Editor of UNIX REVIEW. ■

NEW RELEASE

UNIPRESS EMACS™

VERSION 2

Another in a series of productivity notes on UNIX™ software from UniPress.

Subject: Multi-window, full screen editor.

Multi-window, full screen editor provides extraordinary text editing. Several files can be edited simultaneously, giving far greater programming productivity than vi. The built-in MLISP programming language provides great extensibility to the editor.

New Features:

- *EMACS is now smaller and faster.*
- *Sun windows with fonts and mouse control are now provided.*
- *Extensive on-line help for all commands.*
- *Overstrike mode option to complement insert mode.*
- *New arithmetic functions and user definable variables.*
- *New manual set, both tutorial and MLISP guide.*
- *Better terminal support, including the option of not using unneeded terminal drivers.*
- *EMACS automatically uses terminal's function and arrow keys from termcap and now handles terminals which use xon/xoff control.*
- *More emulation – TOPS20 for compatibility with other EMACS versions, EDT and simple WordStar™ emulation.*

Features:

- *Multi-window, full screen editor for a wide range of UNIX, VMS™ and MS-DOS™ machines.*
- *"Shell windows" are supported, allowing command execution at anytime during an edit session.*
- *MLISP™ programming language offers extensibility for making custom editor commands! Keyboard and named macros, too.*
- *"Key bindings" give full freedom for defining keys.*
- *Programming aids for C, Pascal and MLISP: EMACS checks for balanced parenthesis and braces, automatically indents and reformats code as needed. C mode produces template of control flow, in three different C styles.*
- *Available for the VAX™ (UNIX and VMS), a wide range of 68000 machines, IBM-PC,™ Rainbow™ 100+, and many more.*

Price:

	Binary	Source
VAX/UNIX		$995
VAX/VMS	$2500	7000
68000/UNIX	395	995
MS-DOS	475	*

*Call for terms

For more information on these and other UNIX software products, call or write:
UniPress Software, Inc.,
2025 Lincoln Hwy.,
Edison, NJ 08817.
Telephone: (201) 985-8000.
Order Desk: (800) 222-0550.
(Outside NJ). Telex: 709418.
Japanese Distributor:
Softec 0480 (85) 6565.
European Distributor:
Modulator SA (031) 59 22 22.

OEM terms available.
Mastercard/Visa accepted.

Trademarks of UniPress EMACS & MLISP, UniPress Software, Inc., UNIX & AT&T 3B Series, AT&T Bell Laboratories; VAX/VMS & Rainbow 100+ Digital Equipment Corp., MS DOS, Microsoft Corp., WordStar, MicroPro

UniPress Software

Your Leading Source for UNIX Software.

Circle No. 10 on Inquiry Card

THE HUMAN FACTOR

Prototyping a memorandum database

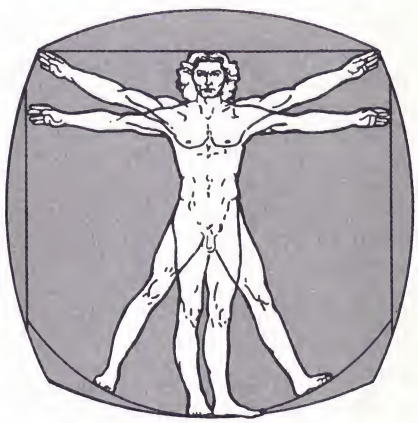
by Richard Morin

Paper is miserable stuff. It piles up everywhere, is impossible to organize well, and makes one irritable by its mere presence. The real problem is not with standard-sized pieces of paper, though. Filing cabinets, binders, and other containers hold these reasonably well. It's the little scraps of paper that are the real enemy—particularly those that can't be handled, filed, or thrown away.

This column previously has offered an overview of UNIX tools suitable for text handling (August, 1984). It also has touted the use of prototyping techniques under UNIX (May, 1985). Finally, it has speculated on the uses of daemons and hashed tables (July, 1985). The current column follows in the same vein, describing the prototyping of a UNIX tool for handling memoranda, and speculating about future work.

MEMO

The first version (see Figures 1a and 1b) of **memo** is more a proof of concept than a usable tool. Still, it is a working program, able to copy text from standard input (normally taken from the keyboard and terminated with a CTRL-D) to a file named *.memo* in the user's home directory. It even adds a time stamp and some formatting text to the message. Since the command can be invoked from anywhere in the file system, it can be used to make small notes without changing directories. The collected memos can then be perused, annotated, or even (gasp) acted upon at the user's leisure.



The principal limitation is that everything ends up in the same file, *$HOME/.memo*. It would be much handier to have **memo** file text under specified topics, as:

```
% memo thf
```

This would add text to a file by the name of *$HOME/memos/thf*. Memos of a general nature could default to the topic *misc*, stored in *$HOME/memos/misc*. A second version of **memo**, shown in Figure 2, implements the new option.

If the user puts a link in the *memos* directory, data will be stored in the file specified by the link. This means that the user can use links in the *memos* directory to funnel text to files in arbitrary locations. This is an interesting, useful, and completely serendipitous side effect. Part of UNIX's fertility lies in the number of such "freebies" it gives us.

The tool itself still lacks a bit of polish and bulletproofing, however. It ignores extra arguments, and crashes if a user tries to invoke it without first creating the *memos* directory. Although a README file might alert the user to these problems, a better answer is to make the program more robust and self-sufficient.

A third version, shown in Figure 3, automatically creates a *memos* directory for the user. Being defensively programmed, it first checks for the existence of a file named *memos*. It also performs a simple argument check, requiring either zero or one arguments.

Now that we have a version that can be safely handed to a naive user, we could easily quit. The code is small, relatively simple, and does its job well. Unfortunately, creeping featurism once again rears its ugly head. Why not use the UNIX hierarchical file system to handle sub-topics?

A fourth version, which is shown in Figures 4a and 4b, does this, building a tree of directories

```
: memo.v1 - file memo

m=$HOME/.memo
u= ============================

date      >> $m    # date stamp
echo $u   >> $m    # underline
echo      >> $m    # blank line
cat       >> $m    # copy stdin
echo      >> $m    # blank line
```

Figure 1a — *A rudimentary* ***memo*** *shell script.*

```
Thu Jan 3 13:10:31 PST 1985
============================

This is a test...

Thu Jan 3 13:12:21 PST 1985
============================

This is another test...
With a second line...
```

Figure 1b — *Output from the script in Figure 1a.*

```
: memo.v2 - file memo under topic

topic=$HOME/memos/${1-misc}
u= ============================

date          >> $topic
echo $u       >> $topic
echo          >> $topic
cat           >> $topic
echo          >> $topic
```

Figure 2 — *An enhanced version of* ***memo*** *capable of filing entries by topic.*

Another in a series of productivity notes on UNIX™ software from UniPress.

COMPILERS

AMSTERDAM COMPILER KIT

Subject: A complete Kit of compilers, cross compilers and assemblers.

The Amsterdam Compiler Kit is the only C and Pascal UNIX package which includes a wide range of native and cross tools. The Kit is also easily modifiable to support custom targets.

Features:

- *C and Pascal compilers (native and cross) for UNIX machines.*
- *Host and target machines include VAX™, 4.1/4.2 BSD, PDP™-11/V7, MC68000™ and 8086.™ Cross assemblers provided for 8080,™ Z80,™ Z8000,™ 8086,™ 6800,™ 6809,™ 68000,™ 6502 and PDP-11.*
- *The Kit contains complete sources* of all programs, plus comprehensive internals documentation on how to make modifications needed to add a new program language or new target machine.*

**A source UNIX or C license is required from AT&T.*

Price:

Full Source System	*$9950*
Educational Institutions	*995*

Selected binaries are available – contact us with your machine type.

For more information on these and other UNIX software products, call or write: UniPress Software, Inc., 2025 Lincoln Hwy., Edison, NJ 08817. Telephone: (201) 985-8000. Order Desk: (800) 222-0550 (Outside NJ). Telex: 709418. Japanese Distributor: SofTec 0480 (85) 6565. European Distributor: Modulator SA (031) 59 22 22

OEM terms available. Mastercard/Visa accepted.

UNIX is a trademark of AT&T Bell Laboratories. VAX & PDP 11 are trademarks of Digital Equipment Corp. MC68000, 6803 & 6809 are trademarks of Motorola Corp. 8080 & 8086 are trademarks of Intel Corp. Z80 & Z8000 are trademarks of Zilog, Inc.

UniPress Software

Your Leading Source for UNIX™ Software.

Circle No. 17 on Inquiry Card

to handle topics and subtopics specified by the user. It also initializes and generalizes topics automatically, detecting and handling conflicts in naming. Assume that the user has typed:

```
% memo thf/cols
```

and has entered the appropriate memorandum. The program will append the output text to *$HOME/memos/thf/cols*.

The main script (Figure 4a) will perform the functions of previous iterations of **memo**, adding only some syntax checking, a call to a subsidiary script (see Figure 4b), and a test for whether the topic has been generalized. The subsidiary script will create any needed directories, and rename conflicting topic files, creating files such as *$HOME/memos/thf/misc*. It will also tell the main script which directory to use for the text.

> It's the little scraps of paper that are the real enemy—particularly those that can't be handled, filed, or thrown away.

As you can see, the code has grown a bit, from 10 to almost 100 lines. Still, it does quite a bit, and it is fairly bulletproof and convenient. So much for the problem of storage. Now, what about retrieval?

RETRIEVAL

The Summer 1985 proceedings of the Usenix Conference and Exposition contains, along with other interesting items, a paper entitled: "UNIX tools for a Personal Database" by Michael J. Hawley of Lucasfilm, Ltd. The paper describes a set of tools for finding files according to the keywords contained therein. The tools use a hashed inverted index, linking each keyword to the most

NEW UNIX SYSTEMS UTILITY SOFTWARE AVAILABLE NOW!

For more information, call us at (703) 734-9844.

SPOOLING & QUEUE MANAGEMENT

SPR SUPER SPOOLER:™ A full-feature print spooling and general purpose queuing system featuring: multiple device support, multiple queues, manual or automatic device assignments, complete tailorability, and background job scheduling.

TELEX COMMUNICATIONS MANAGEMENT

S-TELEX:™ A full-feature integrated hardware and software telex communication management system. Use as a stand-alone system or integrate with your word processing system. Features: automatic dialing, answer verification, message exchange, and recording of all status. Concurrent message transmission and reception. Conversational.

APPLICATION DEVELOPMENT

SSL:™ A powerful terminal-independent screen manager and application development system with many automatic functions such as menu formatting, screen and table handling, data base calls, and more.

EDITORS

SSE:™ A full screen editor for UNIX developers and non-technical end users. Features type-it-as-you-wish-to-see-it text entry, typewriter-like margin settings, and a full set of text editing functions. Easy to learn.

These products are available now for most UNIX or UNIX-derivative operating systems, including System V, 4.2 BSD, 4.1 BSD, Xenix, Version 7, System III, Uniplus, and others.

*UNIX is a trademark of AT&T Bell Laboratories.

UNITECH SOFTWARE INC 8330 OLD COURTHOUSE RD. SUITE 800 VIENNA, VIRGINIA 22180

Circle No. 18 on Inquiry Card

```
: memo.v3 - file memo under topic

if test $# -gt 1; then # too many args?
   echo "Usage: memo [topic]"; exit 1
fi

if test -f $HOME/memos; then # file conflict?
   echo "A file named $HOME/memos already"
   echo "exists.  memo will not create"
   echo "the $HOME/memos directory until"
   echo "the file is removed or renamed."
   exit 2
fi

if test ! -d $HOME/memos; then # missing dir?
   mkdir $HOME/memos
fi

topic=$HOME/memos/${1-misc}
u="============================"

date          >> $topic
echo $u       >> $topic
echo ""       >> $topic
cat           >> $topic
echo ""       >> $topic
```

Figure 3 — *A version of* ***memo*** *for automatically creating a "memos" directory for output.*

recent 500 (or so) references found.

The principal tool is a printing utility, **p**, which searches the index for logical combinations of keywords. It has options that allow the printing of file data, names, and so forth. Another tool, **grok**, expands the scope of the search by using multiple indices, adding files to the current index, and performing several other functions. Finally, a number of low level tools assist in managing the system and building new applications.

Automated keyword indexing is a nifty facility that handles many searching problems cleanly. The only problem, a lack of an easy means for users to impose structure on their databases, can be handled by **memo**. Truly adventurous types may wish to include all sorts of text files in the indexing game. If the indexing takes too much time, let a daemon do it at low priority.

```
: memo.v4 - file memo under {sub-}topic
# Usage:        memo [topic[/subtopic...]]
#
# The entered text is stored in a file named
# topic(/...) in $HOME/memos/. The default
# (sub-)topic is misc. Topic generalization
# is handled automatically.
#
# By appropriate use of symbolic and/or hard
# links, actual file locations may be elsewhere.
#
usage="Usage: memo [topic[/subtopic...]]"
if test $# -gt 1; then  # too many arguments?
   echo $usage; exit 1
fi

dir=$HOME/memos
if test -f $dir; then   # conflicting file?
   echo "$dir already exists as a file, so "memo""
   echo "cannot create it as a directory."; exit 2
fi

if test ! -d $dir; then  # missing directory?
   mkdir $dir
fi

topic=${1-misc}         # handle topic details
if test -n "`echo $topic | sed '
   s@$@@;               # tack on a trailing &
   s@[^/][^/]*/@@g;     # kill off .../s
   s@^[^/][^/]*@@'`"    # kill off ...&s
then echo $usage; exit 3; fi    # syntax error

cd `memo.v4.s $topic`

u="============================"
topic=`echo $topic | sed '
   s@^.*/@@'`           # kill off .../s
if test -d $topic; then # topic is general?
   topic=$topic/misc
fi

date          >> $topic
echo $u       >> $topic
echo ""       >> $topic
cat           >> $topic
echo ""       >> $topic
```

Figure 4a — *A version of* ***memos*** *that shows what comes of creeping featurism.*

CEEGEN-GKS GRAPHICS SOFTWARE in C for UNIX

- Full implementation of Level 2B GKS.
- Outputs, Inputs, Segments, Metafile.
- Full Simulation for Linetypes, Linewidths, Fill Areas, Hatching.
- Circles and Arcs, Ellipses and Elliptic Arcs, Bezier Curves.
- Ports Available on all Versions of UNIX.
- **CEEGEN-GKS** is Ported to Gould, Masscomp, Plexus, Honeywell, Cadmus, Heurikon, Codata, NBI, NEC APCIII, IBM-AT, Silicon Graphics, Pyramid, Tadpole Technology, Apollo, AT&T 3B2, AT&T 6300, DEC VAX 11/750, 11/780 (4.2, 5.2), NCR Tower.
- **CEEGEN-GMS** GRAPHIC MODELING SYSTEM: An Interactive Object-Oriented Modeling Product for Developers of GKS Applications. **CEEGEN-GMS** and **GKS** Provide the Richest Development Environment Available on UNIX Systems.
- Extensive List of Peripheral Device Drivers Including Tektronix 4010, 4014, 4105, 4109, HPGL Plotters, Houston Instruments, Digitizers, Dot Matrix Printers and Graphics CRT Controllers.
- **END USER, OEM, DISTRIBUTOR DISCOUNTS AVAILABLE.**

CEEGEN

CEEGEN CORPORATION
20 S. Santa Cruz Avenue, Suite 102
Los Gatos, CA 95030
(408) 354-8841
TLX 287561 mlbx ur

EAST COAST:
John Redding & Associates
(617) 263-8206
UNITED KINGDOM:
Tadpole Technology PLC
044 (0223) 861112

UNIX is a trademark of Bell Labs.
CEEGEN-GKS is a trademark of Ceegen Corp.

Circle No. 19 on Inquiry Card


```
: memo.v4.s - handle general topics

next=`echo $1 |
   sed 's@[^/]*$@@;      # kill off ^...$
   s@/[^/]*$@@           # kill off /...$

base=''
slash=''
md=$HOME/memos
cd $md
while test -n "$next"; do

   this=`echo $next | sed '
   s@/.*$@@              # kill off /...$
   newbase=$base$slash$this
   next=`echo $next | sed '
   s@^[^/]*@@;           # kill off ^...
   s@/@@                 # kill off 1st /

   if test -f $this     # conflicting file?
   then
   mv $this memotmp
   mkdir $this
   mv memotmp $this/misc
   fi

   if test ! -d $this    # no directory?
   then
   mkdir $this
   fi

   cd $this
   base=$newbase
   slash='/'
   done

   echo $md$slash$base   # for memo.v4's cd
```

Figure 4b — *A subsidiary script for* ***memo*** *capable of filing memoranda according to topic and subtopic.*

Prototyping has a way of becoming addictive. Each new facility suggests others, and the prototyper may never be able to say that an application is really "done". Still, any working snapshot of a prototyped system can be used or even shipped, and the users need not know all of the prototyper's dreams.

Mail for Mr. Morin can be sent to the Canta Forda Computer Lab, PO Box 1488, Pacifica, CA 94044.

Richard Morin is an independent computer consultant specializing in the design, development, and documentation of software for engineering, scientific, and operating systems applications. He operates the Canta Forda Computer Lab in Pacifica, CA. ■

YOU CHOOSE:

	MLINK	CU/UUCP
Terminal Emulation Mode		
Menu-driven Interface	Yes	
Expert/brief Command Mode	Yes	Yes
Extensive Help Facility	Yes	
Directory-based Autodialing	Yes	
Automatic Logon	Yes	Yes
Programmable Function Keys	Yes	
Multiple Modem Support	Yes	Yes
File Transfer Mode		
Error Checking Protocol	Yes	Yes
Wildcard File Transfers	Yes	Yes
File Transfer Lists	Yes	Yes
XMODEM Protocol Support	Yes	
Compatible with Non-Unix Systems	Yes	
Command Language		
Conditional Instructions	Yes	
User Variables	Yes	
Labels	Yes	
Fast Interpreted Object Code	Yes	
Program Run	Yes	
Subroutines	Yes	
Arithmetic and String Instructions	Yes	
Debugger	Yes	
Miscellaneous		
Electronic Mail	Yes	Yes
Unattended Scheduling	Yes	Yes
Expandable Interface	Yes	
CP/M, MS/DOS Versions Available	Yes	

MLINK™

The choice is easy. Our MLINK Data Communications System is the most powerful and flexible telecommunications software you can buy for your Unix™ system. And it's easy to use. MLINK comes complete with all of the features listed above, a clear and comprehensive 275-page manual, and 21 applications scripts which show you how our unique script language satisfies the most demanding requirements.

Unix System V	BSD 4.2	MS-DOS
Unix System III	Xenix	CP/M
Unix Version 7	VM/CMS	and more...

Choose the best. Choose MLINK.

Altos	Data General	IBM
Arrete	DEC	Onyx
AT&T	Kaypro	Plexus
Compaq	Honeywell	and more...

MLINK is ideal for VARs and application builders. Please call or write for information.

Corporate Microsystems, Inc. P.O. Box 277, Etna, NH 03750 (603) 448-5193

MLINK is a trademark of Corporate Microsystems, Inc. Unix is a trademark of AT&T Bell Laboratories. IBM is a registered trademark of IBM Corp. MS-DOS and Xenix are trademarks of Microsoft Corp. CP/M is a registered trademark of Digital Research.

Circle No. 57 on Inquiry Card

4.2BSD

FROM NOW ON, CONSIDER IT SUPPORTED.

When it comes to Unix™ systems, "standard" isn't always good enough.

Experts agree that the most powerful and most technically advanced **Unix** system is the Berkeley version. That's why 4.2BSD from Berkeley is the operating system of choice for software development, networking, engineering, CAD/CAM and demanding scientific applications. Other **Unix** systems don't have the features advanced users require.

But 4BSD was developed at a university, so it has never had real-world support. User assistance, bug fixes, updates and enhancements have not been provided.

Now that's changed.
MT XINU, the 4BSD specialist, supplies:

- Fully supported 4.2BSD-based binary licenses (MORE/bsd) for VAX™ computers.
- 4.2BSD source support and source updates for current 4.2BSD source licensees.
- Enhanced 4.2BSD-based source software for new sites, with or without redistribution rights.
- Full support for a wide variety of DEC™ and non-DEC peripherals.
- Assistance for OEM's and hardware manufacturers developing 4.2BSD-based products.

MT XINU personnel have been involved with 4BSD development from the beginning. Now we are producing 4BSD performance enhancements, advanced networking, other **Unix** system extensions, and support for new peripherals and architectures. As a service, we distribute 4BSD bug reports and proposed bug fixes to the community. Our years of experience can speed and improve your 4BSD implementations.

4.2BSD. It's always been better than just "standard." Now, with MT XINU, consider it supported.

"We know UNIX™ Backwards and Forwards"

MT XINU™

UNIX™ SUPPORT FROM BERKELEY

Circle No. 7 on Inquiry Card

739 Allston Way, Berkeley, CA 94710 ■ 415/644-0146 ■ ucbvax!mtxinu!mtxinu

MORE/bsd and MT XINU are trademarks of Mt Xinu Inc., DEC and VAX are trademarks of Digital Equipment Corp., UNIX is a trademark of Bell Laboratories.

ANNOUNCING:
Verdix Ada Development System
for the Sun Workstation® and VAX/ULTRIX®

VERDIX ADA® DEVELOPMENT SYSTEM

Why the DoD mandated Ada. When the Department of Defense mandated Ada for embedded and mission-critical systems development, there was good reason. This reusable, high-order language can put an end to the Software Crisis. Ada decreases skyrocketing software costs, improves management and control, reduces life cycle costs, boosts productivity, dramatically reduces errors and cuts training costs. Ada is the language of the day, and Verdix speaks it. Louder and clearer than anyone.

Why others are mandating Verdix. We have the first highly portable production-quality Ada development system. The Verdix Ada Development System (VADS™) is the first production-quality Ada compiler system to meet DoD's stringent requirements for Ada language mission-critical systems development. It is now available on the DEC VAX™ series computer systems and includes:

- High performance, rehostable/retargetable Ada compiler under UNIX™ 4.2 BSD and ULTRIX™ (soon under VMS) with excellent diagnostics;
- Symbolic Debugger;
- Library Management Utilities;
- Run Time System.

Designed for large scale and embedded systems development, VADS speeds programming faster than any other Ada compiler.

VADS also helps programmers quickly learn the Ada language. The unexcelled diagnostics speed correction time and shorten development time. The Symbolic Debugger lets you watch your program execute, Ada line by Ada line or machine instruction by machine instruction, even for remote embedded systems. *And it's highly portable:* VADS will be available on a wide range of computer systems.

Fully DoD validated; available now. VADS is validated and ready to be put to work. It's not a promise. It's available now (as in "here and now") and already in use by major DoD contractors.

To find out for yourself how the Verdix Ada Development System can work for you, write, or call (703) 448-1980 and talk to Howard Nevin, Vice President of Product Planning and Corporate Development for more information.

IT'S VALIDATED. USE PROVEN. AVAILABLE NOW!

VERDIX
Ada Development System

Verdix Corporation
14130 Sullyfield Circle
Chantilly, Virginia 22021
Tel: (703) 378-7600

© 1985, Verdix Corporation. Ada is a registered trademark of the US Government (Ada Joint Program Office). UNIX is a trademark of Bell Laboratories. Verdix and VADS are trademarks of Verdix Corporation. ULTRIX is a trademark of Digital Equipment Corporation. Sun Workstation is a registered trademark of Sun Microsystems, Inc.

Circle No. 4 on Inquiry Card

PATTERNS IN DATA MANAGEMENT

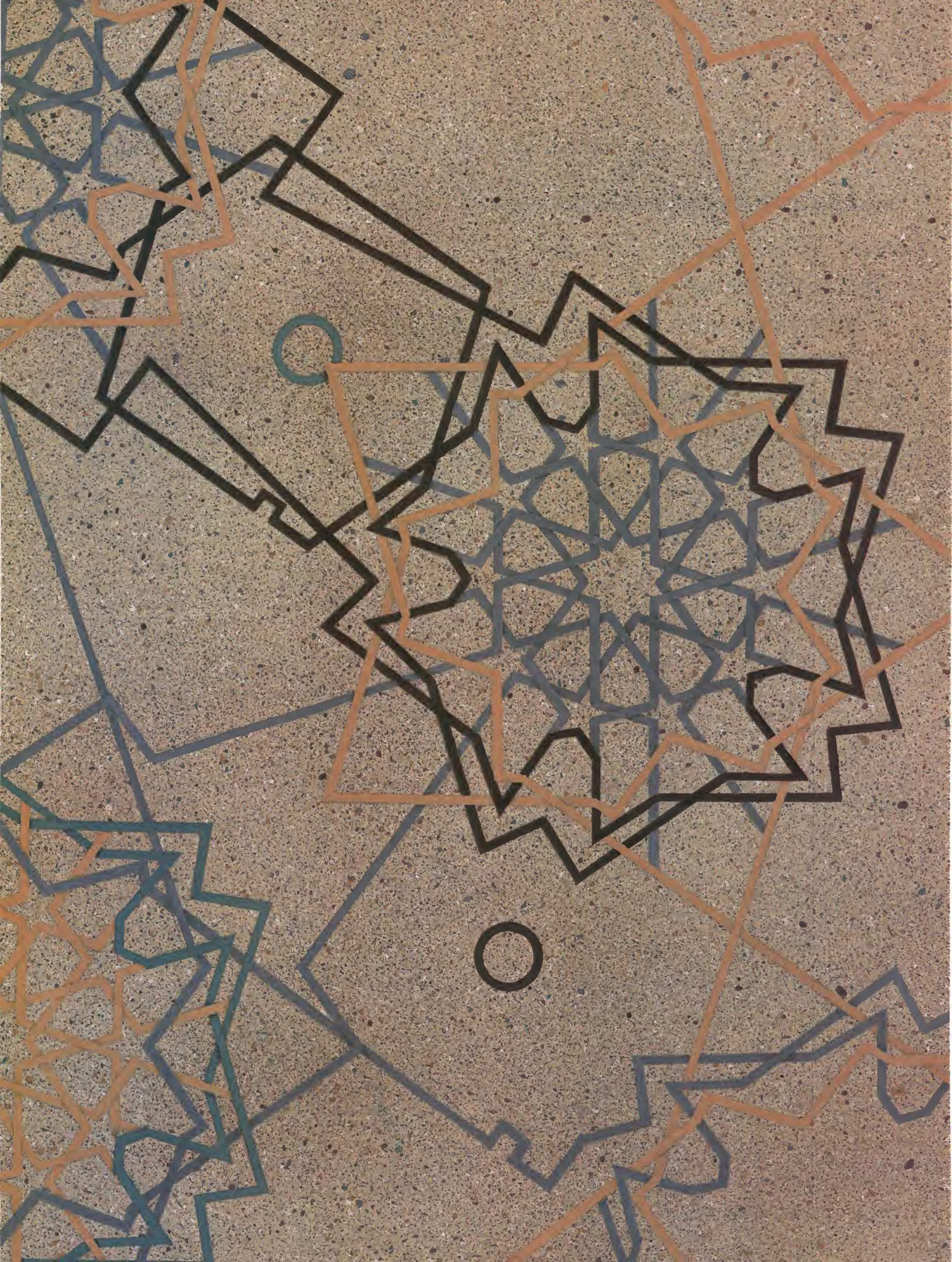
The intricacies of database systems

by Eric Allman

ne of the earliest applications of computer equipment was for storing, organizing, and retrieving information. By its very nature, the computer is adept at accurately performing tedious jobs. A task such as a search through documents for a particular piece of information is the very sort of thing that humans do poorly. Unlike the computer, the human mind drifts; digits can be transposed and sheets of paper tend to get missorted.

Early on, IBM proved particularly astute in exploiting the power of the computer for database activities. Up to that time, database management was often machine-*assisted* (using equipment such as card sorters and collators), but processors themselves tended to be limited to scientific tasks requiring substantial mathematical computation. Then came the IBM 1401, which differed radically from previous computers: I/O was emphasized, even more so than computational ability; most instructions operated on sequences of bytes rather than on words; and operations were included in the hardware to perform such functions as comma insertion and leading blank suppression.

Database technology has grown in three directions since those days. First, the underlying hardware has become faster, smaller, and cheaper. In some cases this has been used to provide dramatic "brute force" systems such as "processor per track" technologies that scan all the data read in from disk. Second, the techniques for organizing data have become more sophisticated. In particular, the assumption of a linear media such as cards or tape is now the exception rather than the rule.



New generations of tools are eliminating the need to consult a programmer in order to access data.

Third, new generations of tools are eliminating the need to consult a programmer in order to access data.

Today, the database market tends to be divided into three camps. The "big guns" camp usually requires huge machines, facilities for storing databases measured in gigabytes, and large staffs of programmers. This camp includes credit card companies, the IRS, and airline reservation systems. Their databases are typically very traditional, leaning toward numbers and fixed-size character fields. They can often be accessed by tens of thousands of people at once. Unsurprisingly, investments in such systems can easily run into tens of millions of dollars.

At the opposite extreme are the many small database systems currently popular on micros and minis. These databases are small, measured in kilobytes, and usually can fit on a single diskette. Programming is often done using a "fill in the blanks" style of interface that can be used with minimal training. In many cases, non-traditional data types such as text and graphics can be accommodated. However, small systems are usually single-user, slow, and limited to small databases. The investment seldom exceeds a few thousand dollars.

Between these two extremes, a new class of supermini-scale systems has arisen. They have many of the features of large systems, such as multiuser access, good performance, and ample programming language access. Many also include some of the nicer features of the small systems, such as applications generators and non-traditional data types. Such databases are typically in the 1 to 100 MB range and cost $10,000 to $100,000.

DATA MODELS

Database management systems typically have a preferred way of organizing data. This is called the *data model*. For example, a file cabinet lends itself naturally to a particular way of organizing data. There are many data models, but three are both popular enough and different enough to be interesting: the hierarchical model, the network model, and the relational model.

The *hierarchical model* most closely approximates a tree. For example, the UNIX file system is a flexible form of hierarchy: every node in the file system has a unique parent and may have some number of children. (It departs from a strict hierarchical model, though, when links are considered since files can be created that exist in more than one directory.)

Hierarchies have obvious physical correspondences, making the transition from manual to automatic systems convenient (for example, a file cabinet is a hierarchy—a cabinet contains drawers, a drawer contains files, a file contains pages, and so on; moreover, a file can be in at most one drawer, a page in at most one file, and so forth). In the early days of computing, hierarchies were convenient since they could be represented easily on linear mediums such as punched cards and magnetic tapes.

However, hierarchies are rife with problems. Finding information that has been classified under a different heading than you have can require an expensive sequential scan of the entire database. Since a piece of information can only exist in one place, data must be duplicated if it logically appears in more than one location. This complicates updates: when a datum is updated, all copies must be found and updated.

The *network model* represents an attempt to fix the data duplication problem inherent in the hierarchical model by allowing arbitrary pointers. This has the extremely desirable property of allowing data to appear under more than one heading. For example, the description of the parts making up an assembly could reasonably appear in three places: in the file for the assembly, once in each of the files for the parts themselves, and once in each of the files for the manufacturers supplying the parts.

The network model—like the hierarchical model—has update problems. When a datum is deleted, all pointers to it must be found and deleted. UNIX solves this problem by deleting *references* rather than the data itself; only when the last reference is deleted is the data actually deleted. In database terms, this is called an "update anomaly", since the delete action functions differently depending on the state of the database. Update anomalies can occasionally be useful, as in this example, but they normally are harmful and should be avoided.

Functioning network database systems such as CODASYL include reference counts and back pointers so that deletion of a datum can also delete all references to that datum. The cost of this is

DEMO UNIFY: THE MOST POPULAR DBMS IN ITS CLASS.

Order the UNIFY Demo Kit, and learn why UNIFY is the DBMS chosen by more applications developers and UNIX-based computer vendors.

You can interactively demo UNIFY's speed advantages, made possible by its automatic selection of four access methods.

You can see the simplicity of UNIFY's menu-based design and Query-By-Forms capability. Witness the power of its SQL query language and RPT™ report writer. Companion documentation details UNIFY's easy forms design, multiple security features and unmatched host language interface—and proves why this is the one DBMS that can keep pace with your needs.

Test it yourself. The Demo Kit includes disk or tape, demo handbook, plus comprehensive manuals that show how to build virtually any application—all for $150.

Contact UNIFY, 4000 Kruse Way Place, Lake Oswego, OR 97034, 503/635-6265.

Optimum

What is a clustered workstation?

Get more than you bargained for.

A powerful high-performance, VME based graphics workstation that's easy to upgrade to the MC68020.

Upgrades to the new Motorola 68020 are as easy as a single board swap in the Optimum V Series. And when you're ready to add on even more processing power, you have a wide range of sophisticated Integrated Solutions' products to choose from. Peripherals. Packaging alternatives. Performance features. Applications. All at low incremental cost.

High-performance graphics for the technical professional.

The Optimum V offers impressive features. Like a high-resolution (1280 x 1024), bit-mapped graphics display in monochrome or color. 32-bit VME architecture and AMD 29116 graphics display processor. Together with 11.2 MHz 68010, dual ported high-speed memory and UNIX 4.2 BSD, the Optimum V is designed to provide you, the technical professional, with state-of-the-art technical processing.

And Integrated Solutions makes processing even easier with a unique user interface. Multiple windows, icons and pop-up menus mean convenient access to programs, files and directories.

Support and more.

Integrated Solutions is behind you all the way with on-site hardware maintenance and local software support. Call us now for more information.

Integrated Solutions
An NBI Company
2240 Lundy Avenue
San Jose, CA 95131
800-538-8157, ext. 823
In California, 800-672-3470, ext. 823

Circle No. 2 on Inquiry Card

Call Integrated Solutions.

Database management systems typically have a preferred way of organizing data.

potentially enormous. In general, network database systems are so gargantuan that they either require huge processors or require that the user handle "strange cases".

Hierarchies and networks share one particularly annoying property: the physical representation of data (that is, what I can get to quickly from where I am now) is intertwined with the data's logical organization. For small applications or applications that are well understood in advance, this is not a problem, but if the organization of the data (the *schema* in database parlance) changes, all programs accessing that data may need to be rewritten. The property we are looking for is called *data structure independence*. For example, UNIX routines like *getpwnam* permit relatively trivial insertion of hashed password files since they hide the physical structure of the data, whereas a change in the directory format requires changes to many programs that know the physical format of directories.

A relatively recent development is the *relational model*. It was originally considered little more than a mathematical curiosity, since it was "obviously" too inefficient to actually implement—much as tree-structured file systems, device independence, and dynamic processes were "obviously" too inefficient. As a result of this genesis, a large amount of the language surrounding the relational model is mathematical rather than intuitive.

In the relational model, data is structured as tables. These tables are physically disjoint from each other, although they may be logically related. For example, a database describing the parts making up an assembly might have one *relation* ("table") containing the list of parts making up each assembly, another relation containing the list of suppliers that supply each part, and so on. Connections are made using logical links: to find the list of suppliers that make parts for a given assembly, find the set of parts that will be required, then find the list of suppliers making those parts.

There are several important points to this example. First, the data language used to access the database is normally non-procedural (that is, it describes *what* data is wanted rather than *how* it is obtained), and set-oriented rather than datum-oriented. Second, the relational model depends on the existence of efficient search structures. Third, key data is duplicated; the part number is listed both in the "assembly" relation and the "supplied-by" relation. Fourth, the data structures can be changed transparently, since the users never say "follow that pointer" in their programs.

The relational model of data has become popular because of its flexibility and simplicity. Almost all the database products available on UNIX today are relational systems; to reflect this, the remainder of this article will focus on this type of system.

First, though, there are several terms that bear description:

A *relation* is a collection of semantically related data. For example, the */etc/passwd* file is an example of a relation matching a login name (the unique *key*) to information about a user. Relations are sometimes called *tables* by analogy to the convenient printed representation of a relation.

A single entry in a relation is called a *tuple*, short for "n-tuple", taken from mathematical usage. It is sometimes called a *record* (from the obvious data processing analogy) or a *row* (from the "table" analogy).

Each of the individual pieces of data in a tuple is called an *attribute*, another word borrowed from the mathematical model. In database-land there is nothing smaller than an attribute, since if you could subdivide an attribute, you would be creating a hierarchy. Attributes are sometimes called *fields*; the "table" analogy would call them *columns*. For example, the */etc/passwd* "relation" has seven attributes: *user name*, *password*, *user id*, *group id*, *gcos*, *home directory*, and *shell*.

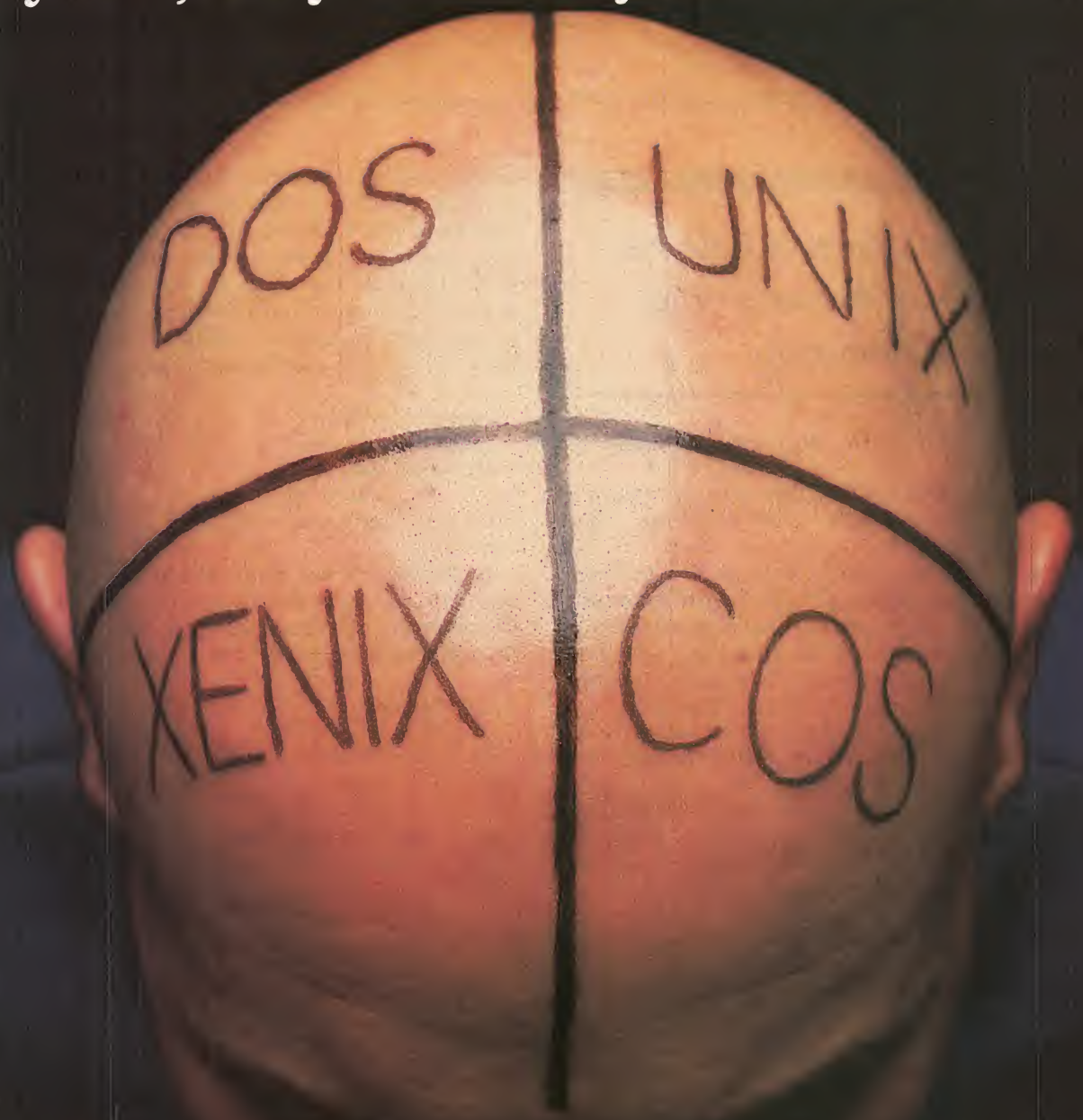
BASIC OPERATIONS

All database systems provide a few basic operations. Without these, the system would be unusable. However, different systems express the operations in different ways. In some cases these operations may be so hidden by the user interface that they are not obvious at all.

Of course, the most obvious operation is to *retrieve* data (this is also called *select*, *get*, or *query* in some systems). Conditions can normally be applied to limit the amount of data that is returned. For example, a retrieval might ask the system to "show the employees who work in department 23" or "retrieve all employees who earn more than their manager".

Data can be added to a relation using the *append* (also called *insert* or *add*) operation. For example,

Learn this integrated office program on one system, and you'll already know three more.



R Office ties a powerful word processor together with the file management, table spreadsheet, report generation and desktop management chores common to nearly every modern office.

Incredibly, the UNIX version is identical to the DOS version. And the RM/COS. And the XENIX.

Now you can switch from a stand-alone PC to multi-user computer systems without missing a beat. You take all the functions, all the commands, all your learning with you.

Yates Laboratories gave R Office highest scores in its recent Competitive Edge report, praising its "straightforward and efficient user interface which products designed by committee typically lack."

It's the only office automation program many companies need to own. (And the only one their suppliers need to support.)

We designed R Office to make the most out of very little memory, too. After the first 320K bytes of RAM for installation, each additional terminal requires only 32K. (Compare that with other, so-called multi-user software.)

R Office is already helping small companies and major corporations improve their office productivity. And unlike hopeful imitators, R Office is available, today, to help yours.

Write for information to R Systems, Inc., 11450 Pagemill Road, Dallas, Texas. Or phone toll-free, (800) 527-7610. In Texas, call collect, (214) 343-9188.

R Office™
BY R SYSTEMS, INC.

UNIX AND COS ARE TRADEMARKS OF AT&T LABS AND RYAN-McFARLAND CORP., RESPECTIVELY. XENIX IS A REGISTERED TRADEMARK OF MICROSOFT, INC.

Circle No. 11 on Inquiry Card

The relational model of data has become popular because of its flexibility and simplicity.

"add Eric Allman with an initial salary of $200,000".

The *delete* operation can be used to remove existing data. For example, "fire everyone with salaries over $20,000".

Data can be changed using the *replace* operation (also known as *modify* or *update*). A good example might be "give all programmers a 40 percent raise". The *replace* operation can be simulated using a *delete* followed by an *append*, but most systems supply this function as a primitive operation.

In addition, some database systems support more complex operations, although normally they are not given keywords in the language.

The *projection* operation is used to select certain fields from the records. For example, "give me names and salaries (but discard the rest of the information)".

Restriction limits the number of tuples to be retrieved. For example, "just give me the information about the people working in software".

Join matches information from one relation against another relation. For example, "match employee information against department information" (this is normally used in conjunction with restriction, so that a real query might be something like: "give me employees who work in departments with sales over $1 million").

Aggregation is used to summarize data. For example, "give me a count of employees in software" (as opposed to a list of those employees) or "what is the average salary in my company?"

INTERFACES

Many kinds of user interfaces to database management systems exist. Most sophisticated database systems have many different user interface modules, varying from very powerful modules that require a great deal of user sophistication to modules that can be used with a minimum of training but have correspondingly less power. For example, a system may offer a programmer interface, an end user query facility, and an applications generator.

Ad Hoc Query Language. An ad hoc query language allows a user to enter queries in a database language such as SQL or IDL. These languages require a fair amount of training to use. Popular languages today are non-procedural, which is to say they describe the data to be accessed without describing how to find it. For example, to find the names and salaries of all employees of the toy department, one might enter in IDL:

```
range of e is employee
range of d is department
retrieve (e.ename, e.salary)
            where e.dno = d.dno
            and d.dname = "toy";
```

Embedded Programming Language. In order to build up more powerful programs, it is often popular to embed the database sublanguage into a general programming language. For example:

```
showsalaries()
        {
$       char name[50];
$       int salary;

$       range of e is employee;
$       retrieve ($name = e.name, $salary = e.salary)
$       {
                printf("name=%s, salary=%d\n", name, salary);
$       }
        }
```

These interfaces require even more training than ad hoc query languages.

Query by Example. This popular interface asks the user to fill out an example of what the desired output should look like. For example, if the user draws a box on the screen with columns headed "ename" and "salary" and puts "Eric Allman" in the first column and a question mark in the second column, QBE will assume that this means "give me Eric Allman's salary". Fairly minimal training is required, but complex queries are almost impossible to express.

Browsers. Browsers display a single record at a time. A user can then update the values on the screen and ask the database system to change the tuple accordingly. Browsers are extremely useful for a number of common applications.

Some browsers require that a semi-sophisticated user set up the screen format in advance, after which naive users can access the data. More clever browsers will set up screens themselves, so that they can be used immediately by naive users.

Application Generators. Many applications

have a number of common features that cannot be adequately handled by a browser. In these instances, an application generator provides a framework in which programs can be written. They vary from extremely simple packages to forms-based programming environments. In most cases, a medium-sophisticated user can use an applications generator. The applications that result can generally be used by very naive users.

Report Writers. Businesses lust for reports, so naturally a separate class of interfaces is entirely devoted to producing nicely formatted reports—including columns of figures, page headers and footers, subtotals, duplicate value suppression, and whatever else the latest rage might be on Wall Street. Most report writers provide default formats that naive users can use to produce reasonably pleasing reports, with lots of hooks to provide fine control over format elements that are intelligible only to the initiated.

Special Purpose Interfaces. The world is filled with special-purpose interfaces. These can vary from extremely simple ones (such as Automatic Teller Machines, usable by a wholly untrained public) to extremely complex ones (like the control program for a "factory of the future").

FEATURES AND TRADEOFFS

A wide variety of features are available in database systems currently on the market. These features can be very important if you need them, but in virtually every case they come with an associated cost.

Handling Large Databases. The **grep** program is fine for small databases (less than a few thousand "records") that have moderate performance requirements. For larger databases or databases requiring fast access, more powerful access methods will be required. For example, the **look** program uses a binary search algorithm on the dictionary and the **dbm** routines use hashed indices.

Arithmetic Capability. The need to do simple arithmetic inside the database management system is common. For example, you might need to computer "age = 1985 - birthyear" or "metres = feet * 0.3048". The **awk** program exemplifies this capability.

Aggregation. Many applications need a summary of data rather than a slew of raw values. For example, the **wc** program produces a summary of its input data. Aggregates can be simple, such as average salary or maximum age, or they can return a set of values, such as total population by country (returning one value for each country in the database).

The **awk** program includes the ability to compute these aggregates using a procedural interface. High-level languages provide these as primitives, such as "avg(emp.salary)" to find the average employee salary.

Data Structure Independence. Contrary to popular belief, computer professionals are not omniscient. They often fail to properly anticipate actual reference patterns. Data structure independence affords the ability to change the "fast access paths" without changing existing programs. For example, if */etc/passwd* were hashed on login name, but it later became clear that it would be better to produce a B-tree on user id instead, it would be nice if the change could be made without all the old programs being affected in the process. This feature is common on relational systems, but rare on other types of systems.

Multi-File Capability. It is often necessary to correlate data between files. This requires a more complex query language capable of expressing the appropriate queries. Processing, of course, gets somewhat more complicated along the way. The **join** command in UNIX is an example of such a program.

Concurrent Access. In commercial settings, it is common for many people to access the same database at the same time. Some control must be provided to make sure these people do not destroy each others' work. This is usually provided with some sort of locking mechanism. With locking, though, comes the potential for deadlocks, so part of the cost of this feature includes deadlock detection and resolution algorithms.

Crash Resilience. If your data is very valuable, it is important that it be left in a consistent state in the event of a system crash. The usual definition of "consistent" is that "the update I was executing either should be completed or backed out altogether". To provide this, the database system must have a notion of the *commit* operation—that is, it must be able to atomically specify that an update is to be finished rather than backed out. Since all the appropriate data must be on disk, this implies a *sync* operation as well. Finally, all changes must be logged for the duration of the query so that they can be backed out if necessary.

Transactions (or Atomic Multiple Commands). Often an operation that should be considered atomic must actually be implemented using several smaller operations. For example, to transfer money from one account to another, each account must be updated. In the middle of the transfer, there is a brief moment when the money either disappears entirely

Continued to Page 96

MAKING A MATCH

DBMS on the UNIX trail

by Roger J. Sippl

On its way to becoming a standard commercial operating system, UNIX has sometimes been criticized for not being particularly "commercial". Some say that since UNIX was developed in a research lab, perhaps it was meant to stay there. To these people, it seems clear that UNIX was never designed to be run on computers used for keeping records, ledgers, and other "large DP type" information. For them, programming productivity and engineering document preparation seem more in keeping with the system's flavor. But are they? In the years since UNIX moved into spheres outside of the research lab, what has it shown itself to be truly good at?

It's an interesting question with many sides to consider. For the moment, let's focus on the ways in which UNIX has become commercial. Certainly the ability to support data processing applications and to make large corporate information resources available to a large body of users are important criteria for commercial success. The best way to see if UNIX meets the test is to evaluate whether database management systems can be built to run—and run well—on it. If so, one can be assured that UNIX is legitimately commercial, since all commercial applications will ultimately be built using DBMS tools.

In the early days of UNIX, there was some question as to whether it could meet this test. Shortcomings existed—some of which still have not been completely solved. The story to tell here, though, is of how a non-commercial operating system has been commercialized, mostly through standards born of necessity.

We are nearing the end of the changes UNIX must undergo as part of the process, so this is a good time to recap some of the issues that have been addressed over the last five years. At times it will be necessary to discuss specific implementation issues and how they have been addressed. Since my own company's products are best known to me, I will occasionally mention how these issues have affected the architecture of the Informix DBMS and the C-ISAM indexed file system subroutine library. By no means,

Illustration by Mark Macleod

UNIX
DBMS

though, should it be thought that these are issues limited to these products alone.

The major database issues that have been addressed during the commercialization of UNIX have included:

1) *Record locking.* This is probably the best known.

2) *Languages.* In the corporate and government data processing world, COBOL and DBMS go hand in hand. The original definition of DBMS taken from the mainframe environments of the 1960s and '70s was, in fact, one comprised of subroutine interfaces to COBOL. Nevertheless, UNIX has a heavy C bias that often shows.

3) *The file system.* The UNIX operating system actually used to limit the size of a file to 1 MB so that users wouldn't get carried away. To get database packages to run, some system administrators actually had to turn off this protection against "runaway" programs. Early UNIX implementations also sometimes imposed "indirection" overhead whenever they encountered big files.

4) *Multiprocessing architecture and shared memory.* DBMS code is usually pretty big, and it likes to buffer lots of disk data. Can UNIX handle the challenge?

RECORD LOCKING

One of the strengths of UNIX is that it offers multiuser capabilities. Thus, it is imperative that any database system designed to run under UNIX be able to operate properly in a multiuser environment.

What does it mean to "operate properly"? The answer is not obvious. It turns out that a fundamental problem arises when several users change the same body of information at the same time. Fortunately, there are ways to solve this, but note that I say "ways" rather than "way" because not all versions of UNIX agree on the same solution. To illustrate the problem and a typical solution, let's look at an example.

Each new machine can often pose an adventure in record locking.

Consider a database system with a screen-oriented data entry and inquiry module (most commercial UNIX database products contain such a package). In our example, the user has designed a database file to contain records that list a customer name and a corresponding receivable, like so:

Customer Name	Amount Owed
John Smith	20.00
Paul Stevens	230.00
Jane Doe	750.00
Mary Q. Public	570.00

Now consider a data entry and maintenance screen that allows a clerk to find a record, update it, and write it back to the database. The actual sequence of operation follows:

1) The user presses the "Q" key, which represents the screen package's "query" command. The user is presented with a screen that allows part or all of the customer's name to be entered into a customer name field. With this information, the screen package software can then search the database for that customer's record and return it to the screen.

2) The clerk then strikes the "U" key (for "update") and has the option of changing any information on the screen.

3) By striking the ESC key, the operator can tell the screen package to write the updated record to the database stored on disk.

Let's look a bit closer at what is actually going on inside the computer system over the course of these three steps. Assume the record of Paul Stevens is the one to be changed. Prior to step 1, it resides on disk, having no reason to be read into main memory.

When the search is performed as part of the query in step 1, though, Mr. Stevens' record is found and read into main memory. At this point, one copy of the record exists in main memory, while another remains on disk.

In step 2, the clerk changes the record on the screen. This modifies only the copy of the record contained in main memory. Only after step 3 is executed can the main memory copy of the record be written back to disk. Thus, only after the operation is complete will the main memory copy and the disk copy agree. (In most UNIX systems, a "buffer flushing" issue must also be considered as part of this scenario, but let's neglect this subtlety for the time being since it does not affect our fundamental concern.)

Note that no problem arises from the fact that the main memory and disk copies differ for a short period of time. Even if the clerk should go on a coffee break without first telling the screen package that an update was complete, two copies of the same record could differ for a fairly lengthy period without trouble arising.

But what if we now introduce a second clerk? Let's say both clerk A and clerk B are updating the same record concurrently. Kindly note that "concurrent" does not mean "simultaneous". A computer's CPU can only do one thing at a time. To say that two events are taking place concurrently means that the second event begins before the first event finishes. Since UNIX offers time-slice multiuser and multitasking functionality, it can run programs concurrently. The switching between processes is fast enough to be transparent, making it appear as though processes occur simultaneously. If this weren't the case, UNIX would have gained precious little acceptance as a multiuser system.

Returning to our two clerks, let's say clerk A reads the Paul Stevens record from disk into main memory and changes the $230.00 to $1000.00 since a new invoice has been sent for $770.00. However, before clerk A gets the opportunity to write the record to disk by striking the ESC key, clerk B reads another copy of the Paul Stevens record into main memory. This leaves us with three copies of the record: one on disk, one in clerk A's main memory "data area", and a third in clerk B's data area.

At this point, clerk B changes the balance from $230.00 to $235.00 to reflect a new $5.00 bill for Paul Stevens.

Unknown to the clerks, they have become participants in a race. If clerk A writes the Paul Stevens record to disk first, clerk B wins since the $235.00 balance will overwrite the $1000.00 one already stored on disk. If, on the other hand, clerk B writes the record first, clerk A's $1000.00 balance will be the amount stored on disk. The true balance, of course, should be $1005.00.

Almost every manufacturer with a locking problem is actively working on it.

The fundamental problem this scenario illustrates is known as "concurrency control". There are several places in a multiuser operating system where concurrency control can become a problem. Besides the problem with concurrent updates, several users might choose to send jobs to the same printer concurrently. Most multiuser operating systems provide solutions for this fundamental problem by using "locks".

Locks do not apply when something is merely read into main memory. But once an update is initiated, screen package software should lock the record. Locking a record usually doesn't mean that other users can't read it, but it does mean that other users can't lock the same record. Once an update is complete, though, the record will be written back to disk, and the screen software will unlock the record so that other users can modify it.

To see this in practice, let's take one more look at the two-user scenario, using the assumption that the system offers the ability to lock files. Clerk A reads the record and begins to update it. The Paul Stevens record thus is locked. Clerk B then reads the Paul Stevens record and tries to update it. Since the record is already locked, the screen software acknowledges this by printing a message on the clerk's terminal explaining that the record is temporarily unavailable, and that it would be best to try the update again shortly.

Notice that the means for affixing a lock, detecting a lock, and removing a lock really must reside in the operating system itself, as personified by subroutine calls to the various programming languages running on the system. The locking job falls to the operating system because all the programs running on the computer need to coordinate their activities through "lock tables". It would be unreasonable, probably impossible, for all the programs running on a computer at the same time to make inquiries of all other programs on the system before locking a record.

This may make it seem obvious now that record locking is a job for the operating system, but it took a long, hard battle to get such facilities included in commercial versions of the UNIX operating system. Not until System V Release 2 was it that an official description of a locking call was included.

Nevertheless, some database management systems available on UNIX have been providing multiuser database functions for five years. How have they done this without a locking call? The truth is: they haven't. The void left by official releases of UNIX actually has been filled by a number of vying alternatives. The locking calls for the Fortune, Onyx, and Plexus machines are all the same. But the locking call for Xenix differs from these, and the Zilog locking call offers yet another variation. All work fine; they are just different. Since Informix and C-ISAM run on over 80 different machines, though, we have found that each new machine can often pose an adventure in record locking. The System V Release 2 standard is helping in this regard, however.

The source code we use to do locking has *ifdefs* for code

CONCENTRIC ASSOCIATES, INC.

WE DON'T PRODUCE TRAINING.

We Produce Shell Programmers, C Programmers, Ada Programmers, System Administrators, Kernel Hackers, Doc Preppies, and Project Managers.

- We will work with you to find out what your people need to know.
- At no charge, we will propose a curriculum tailored so that your people are immediately productive.
- Our instructors will deliver the courses or you can license the courses and we'll teach your teachers.

Circle No. 12 on Inquiry Card

WE ARE ALSO COMMITTED TO BRING TO MARKET A LINE OF SOFTWARE TOOLS TARGETED AT PROGRAMMER PRODUCTIVITY.

The first of these products is:

shacc—the **sh**ell **acc**elerator—is a compiler for the Bourne shell. It translates Bourne shell programs into C and then invokes the C compiler to produce an "a.out" file. The C code that is generated is well-structured and very readable, so it can be further optimized by hand if you like.

shacc allows you to write production code in the Bourne shell: Do the fast prototyping in shell and then **shacc** it and ship it.

Call us for information about our on-line demonstration.

shacc
By Paul Ruel
Concentric Associates

SHACC UP WITH CONCENTRIC ASSOCIATES, INC.

For further information on our Educational Services or shacc, call or write:
Linda Cranston/ Concentric Associates, Inc/ One Harmon Plaza/ Secaucus, NJ 07094
201-866-2880

Circle No. 13 on Inquiry Card

to handle dozens of different locking calls and "non-operating system" locking call schemes. Schemes that exist outside the operating system use some more esoteric tricks to work around the UNIX kernels that lack locking calls. These schemes are not needed as much anymore, but they have been used in the past, and some of the better workarounds are still in use.

For example, UNIX 4.1 and 4.2BSD versions do not have record locking calls per se, but 4.2 does allow programmers to lock a file. A clever programmer can develop lock table management code to secure records or anything that needs to be locked quickly. When System V came out prior to Release 2, it did not have a locking call, but it did provide for semaphore calls. This has allowed clever programmers to lock the lock tables. However, with both the 4.2 *flock* call and System V semaphores, the lock tables have had to be kept in a "public place", such as a disk, and performance of updating has been affected (although not badly because of the least-recently-used disk buffering scheme utilized by UNIX).

For all the improvements, some of the machines currently on the market still lack a locking call. Those that use early versions of UNIX typically have no locking facilities whatsoever unless the manufacturer has added them independently. It should be said, though, that almost every manufacturer with a locking problem is actively working on it. Most DBMS software in the UNIX market thus is safe, though some packages still do not provide record-level locking. Those that lack record locking must change because even though it is easier to lock an entire file, users cannot obtain as much freedom and "concurrent throughput" that way.

LANGUAGES

First, there was C under UNIX. But for business users, COBOL has always ranked number one. Happily, COBOL has been successfully ported to UNIX by several third-party software companies. Better yet, the interactive debuggers and "workbench" type environments available for CO-

EASIER THAN 1-2-3...

BUT DESIGNED FOR LARGER SYSTEMS

It's simple, C-CALC from DSD Corporation is more flexible, has more functions, and is easier to use than the best selling spreadsheet. We made it that way for a very simple reason, you'll get more work done and make better decisions in less time. That's what makes you successful whether you are planning for the future, forecasting trends, or analyzing profits.

The most popular spreadsheets require a great deal of time to get up and running. When we created C-CALC we kept in mind that time is your most important resource. Our On-Line Help facilities, prompts and menus allow even someone with minimal experience to see meaningful results in very little time. Our built-in training procedures let you pace your own learning with tutorial topics that range from basic to advanced. As you become more experienced, C-CALC allows you to bypass prompts and menus to save even more time.

So call DSD Corporation at (206) 822-2252. C-CALC is currently available for: **UNIX,** VMS, RSTS, RSX, IAS, P/OS, AOS, AOS/VS (Data General), IBM CSOS.

DSD CORPORATION
P.O. BOX 2669
KIRKLAND, WA 98033-0712

EFFECTIVE SOFTWARE FOR BUSINESS

C-CALC is a registered trademark of DSD Corporation. **UNIX** is a registered trademark of Bell Labs. P/OS, RSTS and RSX are registered trademarks of Digital Equipment Corporation. AOS and AOS/VS are registered trademarks of Data General Corporation.

Circle No. 25 on Inquiry Card

BOL programmers under UNIX are among the best offered anywhere.

However, the interfaces between COBOL and DBMS products have not been quite so illustrious. COBOL has a file type ISAM (Indexed Sequential Access Method) that allows programmers to declare "search keys" on selected fields of a data file. The ISAM file type facility allows programmers to quickly search for records by the contents of fields rather than by record number.

This ISAM file type is typically supplied by the operating system, largely because it can thus be tied in with record-locking mechanisms. But since UNIX had no record locking when it first became commercially available, it certainly didn't offer ISAM. Indeed, UNIX programmers could not even read records by number, which was considered a standard capability by most other operating systems at the time. Instead, records had to be read by byte position.

The same suppliers that provided COBOL under UNIX solved these problems by implementing fixed-length records on top of the byte-pointer style of interface offered by UNIX. Some COBOL implementations used C-ISAM since they were often written in C.

This use of a standard ISAM for UNIX has supplied an interface between DBMS tools and COBOL (as well as other languages using the same ISAM). The COBOL CALL verb has also been used effectively as a subroutine style interface (by both Informix and Unify). These interfaces almost always have worked better when they have come from C rather than COBOL, demonstrating the UNIX prejudice. But this is now changing in response to commercial demands.

The best technology for interfacing a language with a DBMS is not by way of a subroutine at all, but rather by using an "embedded query language" approach. Until recently, this was used only by Ingres (embedded Quel), but now it is also available in the Informix-SQL product line for C and COBOL (embedded SQL).

DBMS software often looks like part of the operating system. At times, it actually is.

The embedded approach allows programmers to interleave statements of a query language with statements written in a high-level language, such as C or COBOL. The term "query" language is a bit of a misnomer since languages such as SQL or Quel really have statements for insert, delete, update, and structure-changing operations in addition to sophisticated data retrieval statements.

These embedded languages, even when used on non-UNIX machines, are usually compiled by first translating the database language into the syntax and subroutine calls of the host language. This technique, known as "running a pre-processor", is old hat in the UNIX community.

C has always had a pre-processor, and many of the software development tools and program text editors on UNIX cope with such compile-time architectures well. Error messages that come from the compiler of the underlying language contain error messages referencing line numbers from the original source listing—if the product is set up to do this and if it follows the rules of UNIX pre-processors. In short, since language processing is something UNIX has always been good at, these types of programming productivity tools fit well into the UNIX environment. Considering that pre-processors offer the best interfaces to database systems, it could be said that UNIX enjoys a significant advantage over competing "commercial" operating system environments.

THE FILE SYSTEM

UNIX was written to operate on machines that had meager processing resources by today's standards. Accordingly, it could not impose much overhead. It's hardly surprising then that the designers produced a system built around the assumption of small files. This was reflected in design tradeoffs that assessed a performance penalty whenever files were large.

How did this penalty work? In essence, by wreaking havoc with the inodes of large files. The inode for each file is a block of disk that explains who created the file, when it was last modified, what the permissions are, where to find the file on disk, and how big it is. The information about size and location needs to be fairly concise to fit into a single disk block. If the file is too big for an inode to explain its location fully, the information in the inode block can act simply as a "pointer" to a group of other blocks containing the full story. This additional step means, though, that the operating system is spending more time reading and writing large files, because it first has to read and study pointers and indirect blocks.

The amount of overhead imposed by this "indirection" has been much debated. Some have estimated that retrievals on systems with "old" architectures can be slowed by as much as 20 percent. Discussions of this

question typically have been more theoretical than practical, though. And the question is becoming more academic still—for a number of reasons.

First of all, the block size on most UNIX machines is changing from 512 bytes to 1K, 2K, or even 4K bytes. The bigger the machine, the bigger the block size—and, in more cases than not, the bigger the database files. Also, the bigger the block size, the bigger the file can be without "overflowing" its inode. As block sizes continue to grow, it becomes less likely that overflow will occur at all.

Also, the 4.2BSD implementation contains a "fast" file system that has been reorganized to diminish or eliminate this problem. Although 4.2BSD is not widespread commercially, many of its concepts for building a file system capable of handling large files have been picked up by commercial implementors.

Another reason for diminishing concerns over inode overflow is the falling price of main memory. This may not seem relevant, but because of the elegant buffering scheme of UNIX, it is. Even where indirection is necessary, indirect blocks can easily be located in main memory to facilitate frequent use. Users with big database files who must work with antiquated file system architectures and small machines can get around the overhead by increasing the number of operating system file buffers whenever performance problems become noticeable. The default number of buffers is usually very low—often 30 (each only 512 bytes long, giving the system a *total* of 16K bytes for file system buffering). This could be enlarged to 50, 75, or even 100 buffers, and still be conservative. Increasing the number helps ensure that the indirect pages, if any, remain in main memory.

Throughout the history of DBMS products on UNIX, the advent of standards has always helped.

Some DBMS vendors have developed file systems able to "work around" UNIX in order to address the indirection problem. These solutions usually require that the database be placed on what is called a "raw disk" logical or physical device. The data is then written by the DBMS software to an "unmounted device", obviating the need for formatting it for the UNIX file system.

The disadvantage to this approach is that UNIX doesn't know how to cope with these "raw disk" chunks, and thus many UNIX utilities—such as **ls**, **tar**, **df**, **icheck**, **dcheck**, **fsck**, and the like—are of no use. Also, other programs cannot share space on the disk allocated to a DBMS file system. This means that any space set aside for future DBMS growth translates by necessity into wasted system space for the present. Nevertheless, those using the "work-around" approach are best advised to reserve plenty of space at the outset since major problems are sure to arise if the space should ever fill. In the event the space does become saturated, the entire device will have to be unloaded to tape, reconfigured as a larger device (using some of the more esoteric UNIX utilities), and then reloaded. This process also usually means that some other logical disk device will have to be borrowed from to obtain the necessary space. This means other data will have to be unloaded as well. Databases that span disks are more complicated yet, since UNIX pathnames are not available for non-UNIX file structures and thus the process cannot be made transparent.

MULTIPROCESSING ARCHITECTURE AND SHARED MEMORY

"Commercial" data processing has traditionally allotted special privileges to DBMS code. In fact, DBMS software often looks like part of the operating system. At times, it actually is.

Why did this come to be? Mainframe DBMS code generally has been very big (.5 to 2 MB, depending on the product), and typically has had to be able to handle several programs at the same time. It also has traditionally wanted to buffer lots of data, just like an operating system's file system.

Because of these design issues, database systems usually have been implemented with front-end/backend architectures. The "backends" have typically contained DBMS software machines capable of performing fast indexed retrievals, storing new data, and maintaining indices. The front-ends either have offered user programs that were linked to DBMS software through a programming language interface library, or collections of DBMS tools like interactive query languages, screen packages, or report writers.

Is the front-end/backend architecture essential? Can UNIX provide it? The answers depend chiefly on the size of the machine. Small machines usually don't have many users and, as a result, don't need the two-process architecture as much as larger machines. In fact, the overhead of

Continued to Page 98

DATABASE DILEMMAS

An interview with Peter Weinberger

Within AT&T Bell Laboratories, Peter Weinberger is known for many things. One is his knowledge of how databases work under UNIX.

As head of Computer Systems Research at the Labs, Weinberger is concerned with more than just data management. But that nevertheless is where he made his mark nearly 10 years ago when he joined with Alfred Aho and Brian Kernighan to develop **awk**, *still one of the most powerful data manipulation tools available under UNIX. (The command name itself stands for "Aho, Weinberger, and Kernighan"—a classic example of just how obtuse UNIX mnemonics can be.) Since those days, Weinberger's database interests have branched into new directions, as evidenced by a paper he published in the November, 1982, issue of the* Bell System Technical Journal, *"Making UNIX Operating Systems Safe for Databases".*

He claims his focus is now set elsewhere, but the insights that lace his responses to the questions that follow would indicate otherwise. Contributing Editor Ned Peirce, a consulting systems analyst, delivered the queries.

REVIEW: *How well does the UNIX system support databases?*

WEINBERGER: There are two issues that come to mind. First of all, there is the commercial issue. I do not really understand the commercial issues and the ground rules for discussing what you need in various types of products. That has a lot to do with marketing, and I explicitly disclaim any understanding of marketing.

The other question is: what would it take to make UNIX suitable for various kinds of databases—small databases, big databases, United Airlines reservation systems, and all the rest of it? This is the question that interests me. Does UNIX require big changes to handle database systems? Here, "database systems" roughly means conventional transaction processing.

sary, since it could almost be replaced by **cat**.

REVIEW: *I am told that you have some interesting insights into the problems database system designers faced during the development of the United Airlines reservation system.*

WEINBERGER: No special insight. What I heard was that though they always bought the biggest and fastest computers, they still wanted more. They weren't so worried about reliability, though.

REVIEW: *They had a capacity problem?*

WEINBERGER: A real capacity problem. If you think of transaction rates measured in hundreds per second, which they were, and allow yourself a little growth while you do your research, you find yourself considering transaction rates of thousands per second. This starts to get pretty tricky. If I want to go to disk, I know it used to take 30 msec to get a block off the disk. But if I assume there's been improvement, and that the time it takes to get a block off the disk might go down to 10 msec...

REVIEW: *It won't leave much time to do anything.*

WEINBERGER: That's right. You need lots of disks so that you can get all these things done in time.

REVIEW: *Or run the risk of losing data?*

WEINBERGER: Right. It's a real problem. Now that memory is getting cheaper you might be able to keep much of the database in memory.

REVIEW: *But won't you run into reliability problems?*

WEINBERGER: You have to decide what you're going to do about logging, and what you're going to do about backups, and what you're going to do about memory card failures. I'm not sure you have to actually do anything about memory card failures, but it is one of the questions that arises. After having decided all that, you then get to decide what you're going to do about access structures on the disk. You can use B-trees or some kind of clever hashing scheme that matches the fact that you copy in disk blocks as the fundamental unit of access. But, that's not what's going on in memory, right? Memory can get at individual words equally fast, so you can try to change the access structures or you can forget about it; it may not make any difference. I don't think it's all that hard. It is an intriguing way of getting more performance, though. Just put a few hundred megabytes of memory on your machine and move large parts of the database into memory and see what happens.

> "As your view of the world changes—and as your customers' view of the world changes—your database will get weirder and weirder."

REVIEW: *Wouldn't you need some scheme for reading the data out to a less volatile medium?*

WEINBERGER: I don't think you should leave it to good fortune. You would need to have a piece of the algorithm write log records. I think you could write whatever you would have normally written had the data been on disk.

You also need ways to handle the kinds of high transaction rates you can expect when you get lots of people typing and submitting transactions at the same time. In that situation, the system won't respond to transactions right away when it processes them. It won't respond until it has batched up all the little transaction records, written the log, and written the whole thing out to disk. Only then will it respond to all those people. If you're doing 1000 transactions a second, you can write very large log records because you can do them 100 at a time. You can do that because you only have to write 10 large blocks a second, which is easier than 100 or 1000. A delay of a tenth of a second in a response is barely perceptible. So the scheme should work okay. There are a lot of little questions and details, though. It sure wouldn't look much like a timesharing system.

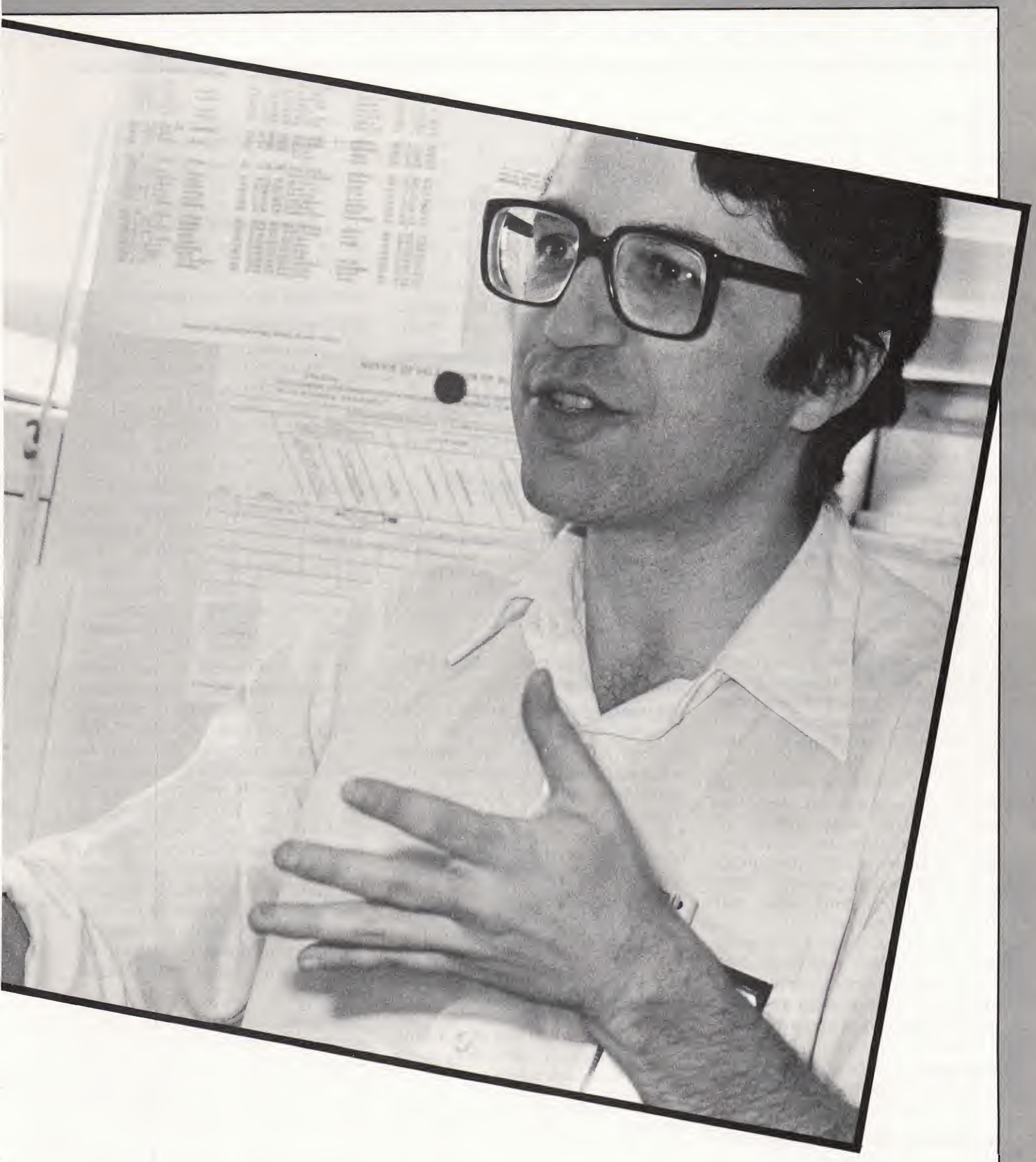
REVIEW: *Would it look like a UNIX system?*

WEINBERGER: It could look *like* a UNIX system, but you would never actually see the UNIX system itself. Let's say you turn on your machine for the first time and then load up all this software that eats up all your memory. That would not encourage people to do a lot of software development or whatever on the system.

That's to say that for all that work, you will not have solved any of your problems except for speed. All the problems of independent transactions, safety, and deadlock still have to be dealt with. Logging and all that stuff is still there.

REVIEW: *And all that requires hacking in the kernel?*

Continued to Page 100



Photos by George Peirce/Studio 84

REVIEW: *How would you go about putting a database system together under the UNIX system?*

WEINBERGER: There seem to be three approaches—or some number like three. People who build database systems under UNIX can build them without using the UNIX file system; that is, they can take a raw disk and use some scheme like shared memory to make buffering to the disk more efficient and easily controlled by the database management system. This means, though, that they have to implement their own concurrency control mechanisms to make sure transactions don't stomp on each other in unacceptable ways. That's been done in a fair number of products sent out by Bell Labs to operating phone companies. The people who keep track of cable repair work have database systems of this sort. These are transaction processing systems that are at least partly run on UNIX. They just use a raw disk scheme to take control without making big changes in the kernel.

Another approach is to go whole hog and get out the database literature, read through all the concurrent control stuff, and say, "Well, we've just gotta put all that stuff in the kernel." That would be fairly remarkable, but it's sort of possible.

The question is: might there be an intermediate approach that could give you control where you absolutely need it without distorting the underlying image of UNIX too much?

REVIEW: *And in that way see to it that the kernel remains useful to other people?*

WEINBERGER: That's right. You don't want to distort the kernel too much. It's fairly plastic, but if you stretch it a lot, it gets sort of rigid. You want to avoid that. But solutions are still more or less a research question. My view is that putting a database system together under UNIX is possible, but that a couple of problems will have to be solved along the way. Some of those problems are hard and some are easy.

The easy part is the control of stuff in the system—making sure that disk buffers have gone onto disk, so that if the system crashes, you'll know what the state was—that sort of thing. That's all fairly trivial.

> **"I explicitly disclaim any understanding of marketing."**

REVIEW: *Ordered rights, that type of thing?*

WEINBERGER: Yes, and if necessary, a special system call that flushes out all the disk buffers associated with a specific file. The conventional image of a database system is that a single process is in charge that somehow keeps track of everything. The classic UNIX system approach to it would be that roughly each terminal, each person, each transaction would have a process, and then somehow you would have to get the processes to communicate among themselves to resolve deadlocks. Apparently, one of the most efficient ways of controlling concurrency in databases is to lock shared resources.

REVIEW: *If something's locked, do you just keep retrying until it's unlocked?*

WEINBERGER: You have different choices there. The real question, though, is: what happens when you get a lock on something and somebody else wants a lock on it as well. They can wait for a small amount of time and try again, or they can somehow register a request that says, "Let me know when the lock becomes free."

REVIEW: *And then* **sleep** *on it?*

WEINBERGER: Lots of commercial systems have these things that allow you to enqueue on a lock and then somehow get serviced later. There are two parts to this. One is that if you have a database system in which there are lots of lock conflicts, you probably have an inefficient system. Presumably, though, you have a system in which you get very few of these conflicts. If so, you probably hate to have to go to the operating system each time you want to get a lock because that takes 500 microseconds or whatever. One good place to put locks in order to minimize such objections is in shared memory. That way, when there's no conflict, you can use some kind of atomic memory operator to grab things and lock them. That works perfectly well when there's no conflict. When there is a conflict, you have two independent processes that have to talk to each other somehow, and that's through the operating system. The question is: how little can you put in the kernel to support locking and deadlock detection? It's probably some fairly small amount.

REVIEW: *We're talking about changes above and beyond shared memory and record locking. Are you saying that those alone are, to some people's minds, insufficient?*

WEINBERGER: They *are* insufficient, precisely because of this problem. Say I'm going to move money from my account to your

account, and you're going to move money from your account to her account, and she's going to move money from her account to mine. Let's say we all start at once. We each lock our accounts because we're about to change them. Then we all go out and try to grab the other accounts because we're going to change them too. Now we're deadlocked. Somebody's got to discover this. And, when they do, we have to have some way of dealing with it.

REVIEW: *Don't you have a security issue if every process can detect deadlocks?*

WEINBERGER: That's why you want the kernel involved. That's where all the coordination comes from. There are other ways of doing it, though.

REVIEW: *By creating a large process that can keep track of all the locks?*

WEINBERGER: You can have one process that's in charge of locking, lock resolution, and some kind of interprocess communication. The trouble with that is that bad things can happen. If I lock my stuff and ask to lock you, and then I divide by zero and die; there's no point in ever trying to talk to me again. The process in charge of locks needs a way of deciding the process has gone away, but some forms of interprocess communication don't provide that. Some forms might, but some forms don't. The kernel, though, always knows.

REVIEW: *It also runs at a very high priority.*

WEINBERGER: That's not as important as knowing it all. The hope is that these funny deadlocks are quite rare.

REVIEW: *You mean you hope things occur promptly enough?*

WEINBERGER: The hypothesis is that locks don't conflict very often, and that it requires three conflicts, roughly, to get a deadlock. That means deadlocks should be quite rare.

REVIEW: *It's hard to prepare for those instances since they depend on the size of the database and the way it is used.*

WEINBERGER: Yes. That's one

of the dilemmas the poor database administrator is supposed to keep up with. You know how it is—any time you don't know how to deal with something automatically, you say, "Oh well, it's up to the database administrator to handle that." The database administrator is supposed to know all about how the database works, and if it's not working very well, then of course the administrator gets stuck with fixing it.

Before saying how, we should digress slightly to talk about an argument concerning this business of detecting deadlocks. If you look for deadlocks too often, you'll end up spending a lot of time looking. But if you look for deadlocks infrequently, things might stay deadlocked for long time. This gives rise to some kind of cost/benefit curve. If you plot the two crossing curves, you'll find an optimal point in the middle. Jim Gray was the one who made this observation originally.

There's a lot of detail known about low-level implementations of database systems. You just have to decide how much of that kind of stuff to use. Implementing a flexible transaction processing database system nowadays is a lot of work. You have to get at the low-level stuff—which I find interesting—and then you have to get all the user interface, validity checking, bulk loading, forms package, and terminal handling stuff together.

REVIEW: *I suppose this sort of work is being done over and over again, even within the Labs.*

WEINBERGER: Yes. Whenever I have had a project, I've never felt I was building a *database* system—I've felt as if I was working on a project. It might be that the project needed a database system because perhaps there was none available that was quite suited to my needs. There are a lot of these tradeoffs to make, and I want to make the ones that favor my world. That means that when the next person needs a database system, they might discover that my design is a little too special-purpose to suit their needs. So they'll just have to do another one.

REVIEW: *It's also true that those other people might never have heard of your database system.*

WEINBERGER: That's also quite possible. But there are relatively few systems that have the sort of transaction processing and reliability you might want in the event of a crash.

REVIEW: *Does the limited number of systems relate to difficul-*

ties with UNIX?

WEINBERGER: Not really. It's more of a reflection of the fact that it takes a lot of work to build one of these things. If you don't need it, why bother?

REVIEW: *Is it appropriate for the UNIX system to provide database facilities as an option?*

WEINBERGER: The question becomes: are you going to provide the whole thing as a single unit, or are you going to provide pieces only.

REVIEW: *The UNIX system is like a toolkit now.*

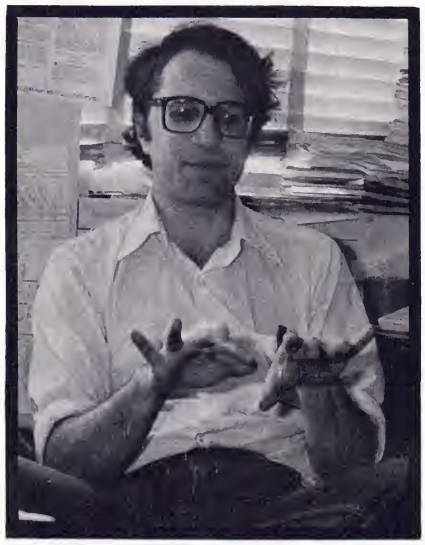
WEINBERGER: That's my feeling. Separate the two things—the pieces that you would definitely want to put in the system and the stuff that's sort of high level, like the real database systems.

REVIEW: *Would that mean you could use whatever forms package you wanted?*

WEINBERGER: Well, you might provide one of those also, but you would be able to separate everything into the two types of offerings.

REVIEW: *If you're trying to design a series of database tools that would be generic to UNIX, would you try to adhere to some particular query language, like IDL, or do you think that whole issue needs to be re-thought?*

WEINBERGER: I do not know enough to answer that question. There are some query languages that seem appealing, either because they're widely used, or because they're supposed to be very easy to use. Query By Example and these other sorts of languages for filling out forms are often talked about. Query languages and transaction processing are really two quite different things, though.



"The kernel is fairly plastic, but if you stretch it a lot, it gets sort of rigid."

REVIEW: *What kinds of database things can be done on the UNIX system as it stands?*

WEINBERGER: The UNIX system, as opposed to a lot of the systems on which these things were first built, already comes with a bunch of commands that you can use to search text files.

REVIEW: *Such as* **awk**?

WEINBERGER: Sure, **awk**, **grep**, or whatever.

REVIEW: *Did you design* **awk** *to be easy to use?*

WEINBERGER: As I remember, when we designed it, the idea was to make a lot of the syntax like C so that other people in the center wouldn't have to learn much new stuff. It wasn't designed to be particularly easy to use.

A project called POPLAR was underway at Xerox PARC at roughly the same time. Their goals were moderately similar to ours, but in many ways more ambitious. It is not a very fair comparison. The Xerox guys were producing a functional language that was fancy and had a couple of very cute ideas in it. It was also designed to be easy to use. But the end result was that lots of people use **awk** and nobody uses POPLAR.

REVIEW: *Strange.*

WEINBERGER: Life is strange. I think it had more to do with the environment the programs were run in. It's easier to use stuff in UNIX—it fits together better. The stuff in their environment at the time was harder to use. There was no easy way, for instance, of joining independent programs together.

REVIEW: *Do you think tools like* **sort** *and* **awk** *became popular because the UNIX system became so popular?*

WEINBERGER: Yes, but I think it goes both ways. The fact that UNIX came with a bunch of useful little tools is one of the reasons it got popular.

REVIEW: *What are the limitations of using UNIX tools like* **grep**, **awk**, **sort**, *and the like?*

WEINBERGER: First of all, they work on text files only. It is possible to build databases solely out of text files since you can keep the data in text files and have special-purpose index trees access the data. You can also have tools that generate the indexes.

Many years ago, when there was a big debate about how hard it was to build relational database systems, I built a toy relational database system as a demonstration. It was made up of ordinary text files and it built its indexes so that programs, when given an

UNIX™ SYSTEM V. SOFTWARE THAT STACKS UP.

High-quality application software written for **UNIX** System V is available now. And new business programs are continually being developed.

That means increased market opportunities for VARs and ISVs. And end users can be certain their investment in a **UNIX** System V-based computing system is a smart, sound one.

Quantity and quality

The market for products that are based on **UNIX** System V has grown significantly over the last two years. There are now hundreds of packages written for **UNIX** System V. And this growth will continue.

Quality applications—general business packages and development tools—are now available for the AT&T **UNIX** PC. And AT&T is putting together even more hardworking, industry-specific software to run on AT&T 3B2 and 3B5 Computers.

With so many packages available, VARs can offer more turn key systems based on **UNIX** System V to an ever-growing customer base. ISVs can sell software to even more customers. And end users can invest in a **UNIX** System confident there is plenty of quality software available.

Software you can bank on

UNIX System V has gained acceptance as a powerful, versatile computing standard. More hardware vendors like NCR, Altos, Motorola, Perkin-Elmer and Sperry are joining AT&T in offering products based on **UNIX** System V.

And we've introduced the System V Interface Definition. Software written under the Interface Definition can run on current and future releases of AT&T's **UNIX** System V, as well as various System V derivatives offered by AT&T licensees. So there will be an even larger, more comprehensive base of portable software from which to choose.

Our comprehensive **UNIX** System V Software Catalog lists a full range of packages that run under **UNIX** System V. For end users it's a reference guide to the programs available. And for developers it's a smart way to ensure packages will have even greater exposure to the growing **UNIX** System V market.

To learn more about **UNIX** System V market opportunities, order the **UNIX** System V Software Catalog—at $19.95 plus tax. Call 1-800-432-6600 and ask for Operator 397.

UNIX System V. From AT&T. Consider it standard.

AT&T
The right choice.

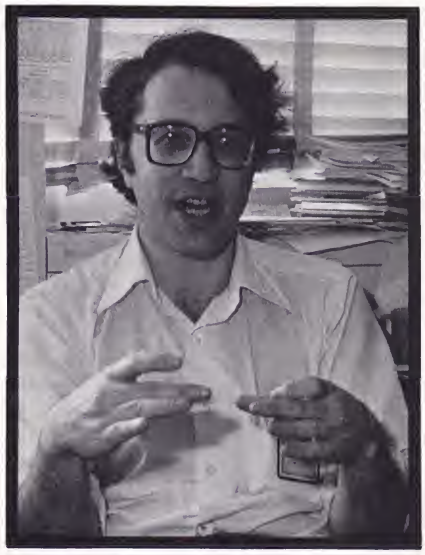
© 1985 AT&T Information Systems.

index, could produce all their records in index order. Because of that, you could run **grep** or **sort** on that data and then put it back in the database. It wasn't a wonderful system, but it worked.

REVIEW: *How absolute are the limitations of text files? If you treat some of the text as numbers, what do you really lose? Is conversion going to slow you down a great deal?*

WEINBERGER: If you keep your database in text files that are indexed by separate files, you have to be careful that people don't change the text files without changing the index files. Whereas, a database system that takes your data and squirrels it away itself gives you no choice but to use database programs to get at the data. The other problem is: if I do a lot of updates on my text file, it's likely that I will keep adding stuff at the end. That means that if I'm scanning the text file, I'm going to detect a lot of records that aren't there anymore. That's going to make work a little inconvenient, so I'm going to have to do something about that. With a bit of discipline, though, there's no particular reason why this sort of scheme shouldn't work fine. It doesn't quite seem to fit into good retrievals, though.

There is another interesting question concerning efficiency where I'm on both sides. If I have a database I want to deal with fairly efficiently that contains things like numbers, is it better to store the numbers in ASCII as a text file, or should I store the numbers in binary so I don't have to convert them? A group at Whippany [the Whippany, NJ, installation of Bell Laboratories] decided that they had been forced to convert their project once too often. They said, "This is silly. Why not give up 15 percent of our performance and just keep the



stuff in ASCII?" The performance hit was only that big because of the work their databases did. If their typical retrieval had been, "What is the average salary at Bell Labs?", switching over to ASCII would have been expensive, but that wasn't the case.

REVIEW: *They were not doing conversions during searches?*

WEINBERGER: That's right. A lot of the work in a transaction database system goes into finding records and saving logs. A lot of CPU cycles go into that, but converting numbers isn't all that slow. That's one argument in favor of keeping your database in a sort of "texty" format.

REVIEW: *I imagine it's also easier to fiddle with the data in text format.*

WEINBERGER: You want to avoid fiddling with it, though, because you don't know how many access paths exist or how many ways there are for looking things up. If I build a static index of some sort to keep track of data, and then you go and fiddle with the data, you're probably going to mess up the index, which would be quite embarrassing.

REVIEW: *What are the real advantages of going to text? What made the Whippany people want to stay with it?*

WEINBERGER: It is so much more portable.

REVIEW: *They didn't have to be concerned about byte ordering?*

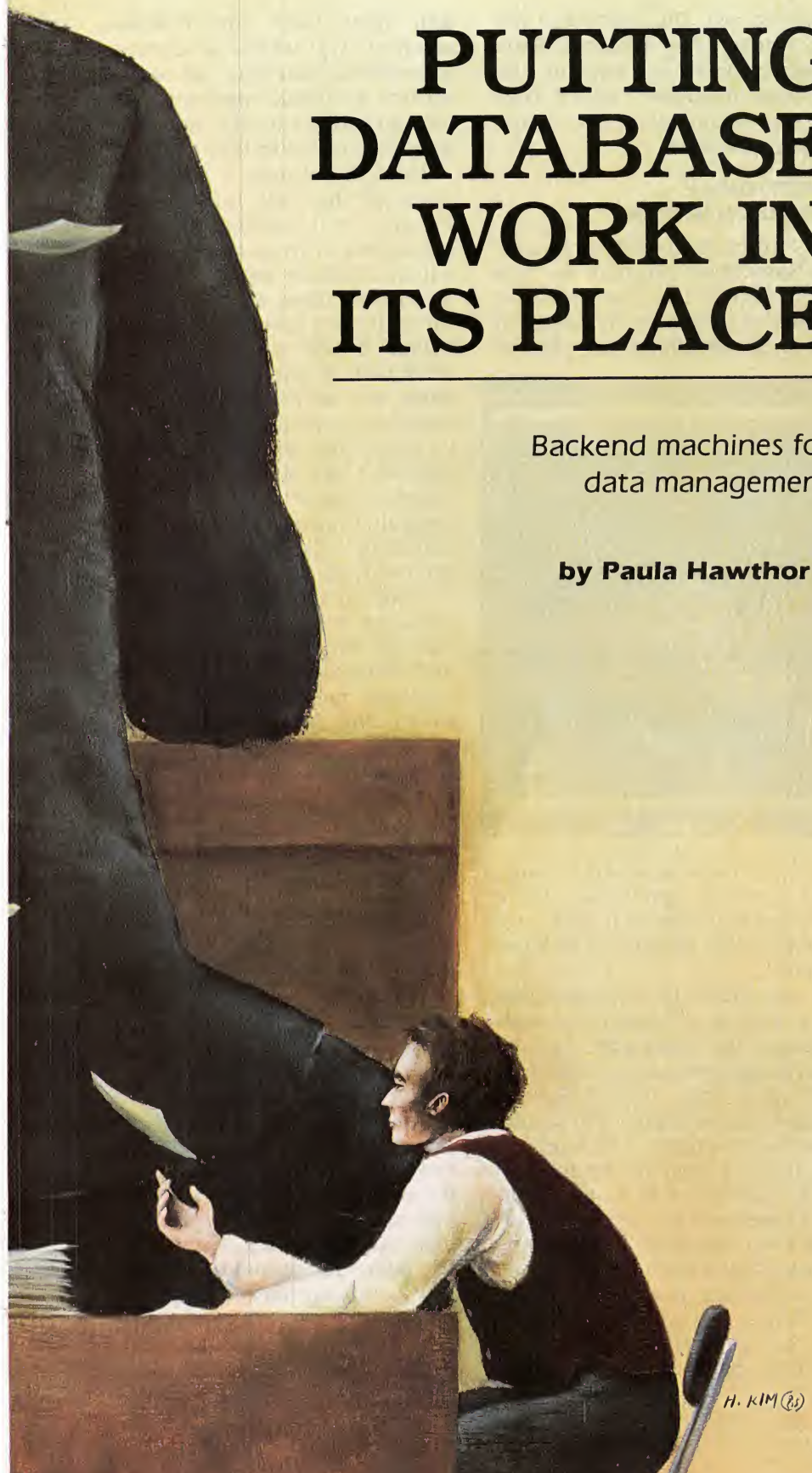
WEINBERGER: That was their problem. They got into this either because they were converting their systems or because a database that was being built overnight by an operating company on an IBM machine was going to be moved to some UNIX machines in the morning. There was just no question about the portability of text on the whole. So long as you kept your character set under some sort of control, most of the translation was well understood.

REVIEW: *There are, in fact, UNIX tools available for typical conversions.*

WEINBERGER: Right. The bit-fiddling stuff may be worthwhile on a big project, but writing your own conversion programs for a small project only adds complexity. If I want to replace some 11/70s with 3B20s in a project that stores binary data in files, I can't just slide one machine out—I have to slide them all out.

One of the things that's not very modern sounding, insofar as type checking is modern, is the fact that the common intermediate file format in the UNIX system is basically a text format. Pipes connect text files and make it all wonderful.

There are a few things that aren't stored in text files, probably for reasons of efficiency. Even in a couple of those cases, though, you could probably store things in text files just as easily. Take directories, for instance; they probably don't need any binary. It would make **ls** unneces-



PUTTING DATABASE WORK IN ITS PLACE

Backend machines for data management

by Paula Hawthorn

A database machine is a special-purpose computing system dedicated to the management of data. When used as a backend processor, it can provide significant performance advantages. This article explores how.

The generic term "backend machine" refers to a computing system designed to work in tandem with a "front-end" computer. While the front-end handles general computing, including the management of the user interface, the backend processes those commands sent to it by the front-end. The machine's general-purpose processor thus is freed to perform other work. The two major commercial incarnations of backend machines are database machines and array processors.

Two Rules. Backend machines are developed when: *Rule 1*) a task that consistently consumes significant general-purpose computer resources comes to be regarded as a major source of performance problems, and *Rule 2*) the task can be executed independently with a minimum of data and control structure communication. Both conditions are satisfied by array processors and database machines.

Array Processors. An array processor is a backend machine that accepts data (such as a matrix) and performs, independently of the front-end, some function on the data (such as an inversion). The changed data can then be returned to the front-end. Array processors are typically attached to computer systems where matrix manipulation functions consume a heavy percentage of CPU time and take a long time to complete. An array processor can benefit a front-end computer by offloading it, thus making the CPU available for other tasks. In addition, an array processor can run faster than a front-end computer because it contains special hardware and software for array manipulation.

Database Machines. The database management systems that are functionally complete—those

Illustration by Hyon Kim

providing multi-table commands, transaction management, and protection—also require a significant amount of the processing power afforded by any computing system. A functionally complete database management system (DBMS) can contain more lines of code and be more complex than many operating systems because it must not only manage resources but also attempt to optimize high-level user commands. Several users running concurrent, complex queries on a single computer can cause all other users to scream with frustration because the tricks that the DBMS can play to optimize database response time (such as locking buffers in memory, monopolizing disks with long I/O exchanges, and more) are just the sort of actions that can drag down response time for other users. A DBMS, then, qualifies for a backend machine of its own if only because it satisfies the first rule by placing an onerous burden on the general-purpose front-end processor.

The second rule, which requires that the backend act independently of the front-end in order to keep the two from spending all their time exchanging data and control information, is satisfied only if the relational database model is used. Commands in the relational model are at a high level where information and data are exchanged in *sets* of values.

An additional reason that relational data management systems are good candidates for executing in a backend machine is that they make it possible to pose and enter very complex, resource-eating commands instantly. In a navigational-oriented data model, these commands might take weeks to program. The relational model thus greatly benefits the person who needs to run a query, but distresses other users who attempt to use the computer for, say, editing. The solution? Send these complex queries to the backend machine, where they can be run independently of the front-end system.

FUNCTIONAL DIFFERENTIATIONS

A "functionally differentiated" multiprocessor system is one where each of the processors are limited to permanently assigned tasks. By assigning the processors their tasks at system design time, control functions (those instructions that tell the processors what to do) can be minimized.

> It seems counterproductive to weigh down general-purpose operating systems with a large group of specialized commands.

Functional differentiation does work well in a backend system because the front-end can be used for general purpose processing, while only those functions the backend is designed to handle need be sent there. The backend can then be designed exclusively for a narrow set of tasks. Database machines and array processors both exemplify functionally differentiated machines.

To further discuss database machines, I'm going to use Britton Lee's Intelligent Database Machine (IDM). I could just as well have used the Teradata machine [1], which is another commercial machine, or one of several academic machines (see bibliography), but the one I am most familiar with is the IDM.

Host Functions. A block diagram of the IDM is shown in Figure 1. In it, multiple users are connected to front-end systems, which could be single-user personal computers or more powerful multiuser systems. From the DBMS user's perspective, the chief task of the front-end is to parse the user's database command and send it to the IDM. This is performed in the "host" or front-end box, where we find the "host interface". Here resides the program that translates the commands to the parse trees that are then sent on to the IDM.

Thus, in the functional differentiation model, we see that the host is committed to receiving and translating commands. The database machine is committed to handling all transaction management, recovery, security, and protection issues surrounding the commands, as well as executing them.

The interface to the database machine could have been different: in an early, important research project (the Relational Associative Processor [2]), partial queries were sent across to the database machine. In concept, one could also send direct user input to the machine, but by waiting until full queries can be syntactically verified, user mistakes can be screened out. There are other advantages to splitting the multiprocessors: one computing system (the front-end) is left to concentrate on the screen management and query definition to the point that standard database commands are formed. Then the standard commands can be acted on by a machine that has been explicitly designed for that pur-

pose. This split (parsing in the front-end, execution in the backend) was first used by DeWitt on the Wisconsin database machine, DIRECT [3].

Channel Functions. Figure 1 shows that data commands enter the backend system via a channel. This represents another opportunity for functional differentiation: each channel has a dedicated processor for handling communication between the outside world and the database machine. Services for buffering, protocols, and other functions can thus be offloaded from the database machine. The channel simply notifies the main processor after it receives a command that there is something new to do.

Processor Functions. The IDM has two processors dedicated to command execution: a general-purpose processor (a Z8000), and a Britton Lee-designed processor named the Database Accelerator (DAC) [4]. The DAC is an 8 mips reduced instruction set machine focused on data management. It can be called as a co-processor by the general-purpose processor via subroutine calls.

The IDM has two processors for general command execution because it was decided that to handle up to 16 disks and still stay within bus/shared memory bandwidth limitations of the architecture, two would be enough. This particular architecture was designed around a target system, a target customer base, and a target timeframe for a deliverable, reliable system (see *Design Decisions*, [5]). A number of other approaches might be equally well suited to different market particulars. For instance, the Teradata machine contains more processors to support a greater aggregate processing power. It offers a large number of processors arranged in a tree-interconnect pattern and is correspondingly more complex than the IDM.

The functional differentiation of the processors within a backend machine comes from two sources: the hardware (as in the DAC), and the specialized operating system selected to serve a dedicated environment.

Operating System Functions. In a dedicated machine, optimizations can be utilized in a way that would be difficult in a general-purpose environment. For instance, in the IDM, the only function of the backend machine is data management. Therefore, at every possible point, the data management operating system is cognizant of the special environment in which it is acting. For instance, the buffers are managed according to a scheme that recognizes *types* of buffers, and thus knows which ones need to stay in memory longer; the process manager shows a special sensitivity to data management by taking care not to schedule out processes before they are done with their buffers; and the disk allocation algorithm allocates space in a manner appropriate to databases.

Much has been written about the problems that general-purpose operating systems give data management systems [6, 7, 8], but the point is not that general-purpose operating systems are so bad, but that functional differentiation is so good. In the IDM, the operating system and the data-management system were written as a single entity. There are occasional attempts to include more specialized operations in general operating systems so as to make data-management systems run more efficiently. But it seems counterproductive to weigh down general-purpose operating systems with a large group of specialized commands. My view is that data management should be carried out on a machine designed especially for it—and that general-purpose operating systems servicing general-purpose machines should be kept relatively simple.

Controller and Disk Functions. The IDM also includes controllers, each with its own processor. Ordinary, commercially available disks are used to store the data that the IDM controls. This distinguishes it from the British database machine, CAFS

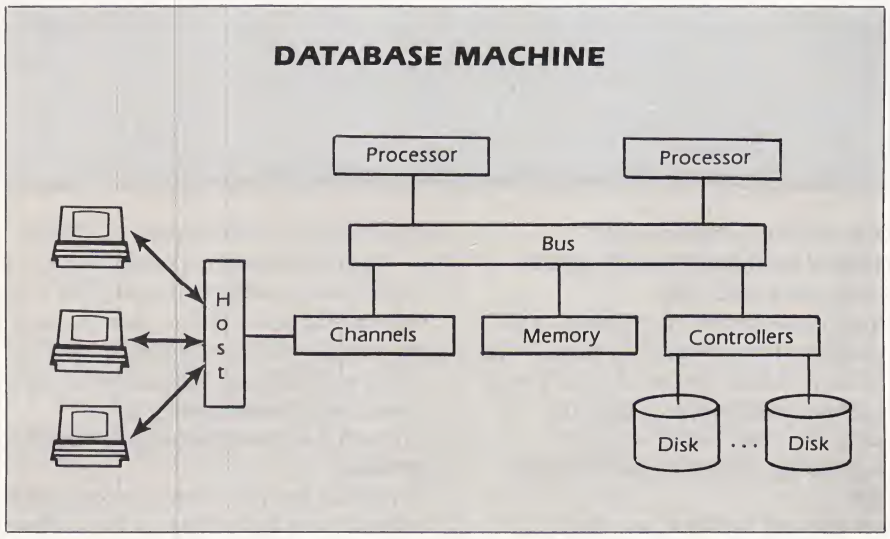
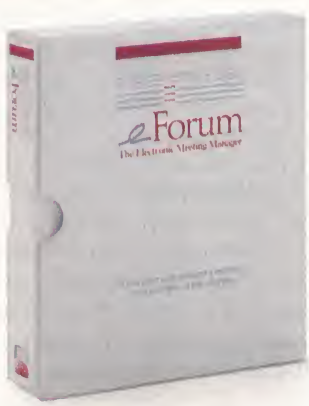


Figure 1 — *Block diagram of the Britton Lee Intelligent Database Machine (IDM), exhibiting functional differentiation in a backend system.*

fo · rum, n. (pl. FORUMS)
1. A public meeting place for open discussion. 2. A medium (as a newspaper) of open discussion or expression of ideas. 3. A public meeting or lecture involving audience discussion. 4. A program involving discussion of a problem by several authorities.



*eForum designed by Marcus Watts, Copyright 1984, Network Technologies International, Inc. (NETI).

Electronic meetings continue the automation of knowledge transfer which started with electronic mail.

Electronic meetings are an extension of the communications revolution which started with electronic mail. It takes seconds to send a letter using electronic mail instead of days via regular mail. Certainly e-mail is a giant step in automating correspondence between two people.

eForum goes yet further to provide immediate communications automation. But for groups. It creates electronic meetings which allow attendees to participate in discussions using the dynamic ebb and flow of points, counterpoints, comments and conclusions just like in-person meetings.

From an economics point-of-view, eForum is the most cost effective method for bringing together the best minds in your company to meet on key issues—without the price of a single plane trip, the aggravation of schedule conflict or time-consuming delay.

eForum is a communications breakthrough product.

eForum lets you create electronic meetings with attendee lists as large as the company staff or as small as a three-person design team.

Not only can eForum handle hundreds of meetings for your company, but, at the same time, limits each participant to only attending meetings to which he belongs.

eForum, n. 1. Low cost electronic meeting system (as in needing no scheduling or travel to attend). v. 1. Automatically organizes, indexes, files and leaves a complete written record of entire meeting. 2. Allows adding more attendees than normal at no extra cost. 3. Gives plenty of time to think before responding. adj. 1. Keeps everyone up-to-date. 2. Doesn't let geographic or time zones determine who can attend the meeting.

eForum

The Electronic Meeting Manager

If you have ever attended a meeting, you know how to use eForum.

Simply attend eForum meetings any time convenient for you. Review new discussion materials. eForum keeps track of what you've seen. Enter your comments or new discussion points. Instantaneously, your ideas are available to every member of your eForum group regardless of geographic location. That's productivity.

eForum has the flexibility to fit your communications needs.

- eForum 4000 - a national communications network available with a local phone call from most locations.
- eForum 2000 - UNIX™ based central host software for supermicro and minicomputers.
- eForum WS - software for the IBM PC and compatibles to interact with eForum central host software.

Call 1-800-638-4832 or in Michigan call (313) 994-4030 collect for information on:

- Automating your company's meetings by using General Electric Information Service, the world's largest communications network, to tie together your microcomputers and terminals.
- Creating your own meeting network for your department or company. Software, hardware and leasing available.
- Establishing OEM and VAR agreements to enhance the value of your software or hardware, with the communications power of eForum.

Circle No. 46 on Inquiry Card

Network Technologies International, Inc.
The Arbor Atrium Building
315 West Huron
Ann Arbor, Michigan 48103

™UNIX is a trademark of AT&T Bell Laboratories
™eForum is a trademark of Network Technologies International, Inc. (NETI)

(made by ICL), in that CAFS uses specially designed multiple-readout disks [9]. A multiple-readout disk transfers data from all its disk heads at once and therefore sustains a much larger transfer rate than a conventional disk, which transfers data from one head at a time. CAFS has used the design to enable the fast scans of databases that are needed in applications where there are no useful indices. The absence of useful indices means you must read the whole file to get an answer to a query. The CAFS system speeds up that process because the disks themselves are functionally differentiated.

This is not necessarily desirable in all instances, however. In the case of Britton Lee, it was decided not to develop and manufacture such disks because it was thought that the performance benefits of increased bandwidth would not justify the added cost and complexity. Decisions such as this riddle system design. Indeed, one of the harder parts of designing a dedicated system is deciding where to use general-purpose systems, and where to build specially tailored ones. In this particular case, it was decided that the performance advantage of multiple-readout disks was significant only in those situations where users have no idea of what they should use as indices—an occurrence that's rare in practice.

IS IT WORTH IT?

Making a functionally differentiated system is not easy. Special-purpose hardware and software must be developed and maintained, and a means of working with general-purpose hardware and software must be devised. The question arises as to whether it's all really necessary. If, for instance, you are having response time difficulties running a suite of matrix manipulation programs, why not just buy a faster processor? If you want to increase the number of people using a database management system, why not just buy a faster general-purpose machine? The reason is simple. The cost/performance ratio is much better for functionally differentiated systems: fast array processors are cheaper than equally fast general-purpose machines. Likewise, fast database machines are cheaper than equally fast general-purpose computer systems packaged with DBMS software. It's easy to understand the cost/performance differences simply by doing a parts count: general-purpose systems require more parts (compilers, interface requirements, compatibility requirements, as well as a variety of devices). General-purpose systems also cost more to build and maintain than single-purpose systems—costs that you can be sure are passed along to end users.

One of the harder parts of designing a dedicated system is deciding where to use general-purpose systems, and where to build specially tailored ones.

There are several factors that cause people to want to use backend database machines, as detailed in "Why Database Machines?" [10]. The opportunity for heterogenous hosts to share a centralized database, and the ability of a database machine to act as a network server are among the most important factors. These could be accomplished without the use of a functionally differentiated system; a distributed database system would work equally well. But two other factors—reduced cost and faster database access—are powerful arguments favoring dedicated database machines.

CONCLUSION

If we take array processors as an example, we can see how the product evolution of database machines is likely to proceed. Array processors first surfaced after a period of academic gestation. Ideas then became reality as a few small companies began manufacturing array processors.

The next logical step would be for major computer vendors to start up production. But this has not been true for either array processors or database machines. Why? Pretend you are a general-purpose computer maker. My speculation is that it does not make sense for you to offer low-cost alternatives to the high-priced high-end systems you already offer. To commit to manufacturing array processors or database machines, you'd first have to overcome three objections. First, why dampen sales of the general-purpose system you're already offering? Second, why build competing product lines when you don't have to? And, third, why take on additional development time? It takes a long time and a sizable commitment of resources to get a new product to market; you must decide if you are willing to make that commitment to a general-purpose product with a potentially wide customer base, or to a special-purpose machine with a necessarily narrower cus-

tomer base. If you already make general-purpose machines, the decision must be obvious. For this reason, backend machines remain the domain of small, specialized companies.

It has taken a while for array processor companies to earn customer confidence in hardware not built by large general-purpose computer houses, but the cost-performance benefits offered by backend array processors, together with good customer experiences with the companies that manufacture them has led to the the presence of several strong manufacturers and a relatively large number of array processors. The same pattern is emerging in the backend database machine realm.

With a concurrent proliferation of fast, cheap processors and fast communication networks, an increasing number of proposals, designs, and products have come to include functionally differentiated multiprocessor systems. Some of these, such as graphics display devices, operate as specialized front-end machines. Some, such as signal processing machines, act as semi-standalone systems. And some, such as array processors and database machines, act as backend systems. The benefits of creating specialized devices for handling complex functions have become apparent. Further advances in the use of dedicated machines surely won't be long in coming.

BIBLIOGRAPHY

1. J. Shemer and P. Neches, "The Genesis of a Database Computer", *IEEE Computer*, November, 1984, pp. 42-56.

2. E.A. Ozkarahan, S.A. Schuster, and K.C. Smith, "RAP—Associative Processor for Database Management", in *AFIPS Conference Proceedings*, vol 44, 1975, pp. 370-388.

3. D.J. DeWitt, "DIRECT—A Multiprocessor Organization for Supporting Relational Database Systems", *IEEE Trans. Computers*, June, 1979, pp. 395-406.

4. M. Ubell, "The Intelligent Database Machine (IDM)", in *Query Processing in Database Systems*, Won Kim, editor, Springer-Verlag, 1985.

5. R. Epstein and P. Hawthorn, "Design Decisions for the Intelligent Database Machine", *Proceedings of the 1980 National Computer Conference*, pp. 237-241.

6. J. Gray, "Notes on Database Operating Systems", in *Operating Systems: an Advanced Course*, R. Bayer, R.M. Graham, and G. Seegmuller (ed.), Springer-Verlag, 1979.

7. P. Hawthorn, "Evaluation and Enhancement of the Performance of Relational Database Management Systems", ERL Memo M79-70, Univ. of California, Berkeley, 1979.

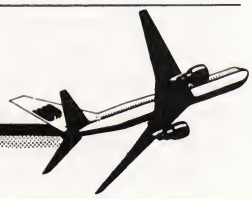
8. M.R. Stonebraker, "Operating System Support for Database Management", *Communications of the ACM*, Vol. 24, No. 7, July 1981, pp. 412-418.

9. E. Babb, "Implementing a Relational Database by Means of Specialized Hardware", *ACM Trans. on Database Systems*, Vol. 4, No. 1, March 1979.

10. R. Epstein, "Why Database Machines?", *Datamation*, July, 1983, pp. 139-144.

Paula Hawthorn is the Director of Product Development for Britton Lee Inc., where she has participated in the design of the Intelligent Database Machine. While working toward her Ph.D in EECS at UC Berkeley, she was a member of the INGRES project. Dr. Hawthorn has also worked at Lawrence Berkeley Lab and Hewlett-Packard. She is currently a candidate for the Vice Chairperson slot on the ACM's Special Interest Group on the Management of Data. ■

UNITED AIRLINES



UNITED,
UNIX*
& "C"

United is aggressively developing one of the world's most sophisticated distributed application environments. It's an exciting time at United, and you can become a significant part of it at United's Corporate Headquarters Complex in Chicago's Northwest suburbs.

UNIX MS-DOS SOFTWARE ENGINEERS & PROGRAMMER/ANALYSTS

Responsibilities will include working with system architects and other software and hardware engineers to develop microcomputer systems interconnected by local and wide area networks. Minimum of 3-4 years "C" or Assembly language design and/or programming experience in a UNIX or MS-DOS environment are required. Degree in Electrical Engineering or Computer Science along with demonstrated experience in one or more of the following areas is preferred:

- **Distributed or cooperative processing**
- **SNA**
- **LAN (Baseband, broadband)**
- **Intelligent workstations**
- **Computer graphics**
- **Real-time applications**
- **On-line transaction processing**
- **Network gateways**
- **Application and database servers**

Find out more about United's dual career paths into management and technical areas, top salaries, relocation assistance, and pass/reduced fare air travel privileges. Send your resume, including salary history, to: **Professional Employment/ EXOPX-UR, United Airlines, P.O. Box 66100, Chicago, IL 60666.** *Equal Opportunity Employer.*

**Unix is a trademark of AT&T Bell Laboratories.*

Circle No. 24 on Inquiry Card

BUCKET BRIGADE COMPUTING

Use of the standard UNIX system kernel for transaction processing

by Kathryn J. Anderson

There is no trick to distinguishing transaction processing from the sort of data processing typically found on UNIX systems. The two differ in both their transaction administration needs and their database processing needs, though the latter requirements are much better understood in the UNIX system community than the former.

Databases used in transaction processing, of course, must pay more attention to performance, concurrency, integrity, and atomicity issues than do databases used for other purposes. Happily, a number of commercial DBMS products based on the UNIX system already have begun to solve some of these classic problems. On another front, though, the transaction administration issues of control and tunability have been wholly unaddressed by the body of generally available database packages. What's more, little attention has been paid to either process architecture efficiencies or performance-oriented forms handling.

This should not be taken to suggest, however, that transaction processing within the UNIX system operating environment is impossible. In fact, transaction processing has been available for years inside AT&T, using internally developed UNIX system tools. One such application runs 300 bisynchronous terminals entering a total of 3500 transactions per hour, all on a DEC PDP 11/70. With the availability of shared memory and interprocess communication in System V, transaction processing has become a particularly realizable goal since no kernel modifications beyond the addition of communication device drivers are necessary.

TRANSACTION PROCESSING VERSUS TIMESHARING

Transaction processing applications tend to be closely tied to activities that generate revenue. They can perform much of

the drudge work necessary to maintain accounts receivable, for instance. In the course of these activities, large, complex databases are acted on by a high volume of transactions. Most of these transactions are of the "bread-and-butter" variety, meaning that they involve only short interactions with the database (less than 10 reads and/or writes), arrive in predictable patterns, and are usually manipulated by clerical people stationed at CRTs. Some of the data used in transaction processing must be available to many users simultaneously, both for reading and updating.

Transaction processing also involves some batch work, including both report and update programs. Processing of this sort is often scheduled to run during off-peak intervals.

Because transaction processing is used so heavily in revenue-related activities, it generally is less cost-sensitive than other applications. Companies are typically willing to spend more both for operating expenses (in the form of well-paid system administration staffs) and development and maintenance expenditures (the price paid for data processing expertise) when they can see a direct link to revenue generation. The willingness to spend is buoyed by the fact that there aren't many options. For the present at least, end users simply cannot develop their own transaction systems using fourth-generation languages. The applications are much too complex and performance is altogether too crucial.

To do successful transaction processing requires that a system administrator be able to both control and tune the running application. "Control" requires that a mechanism exist for giving preference to important jobs, while the ability to "tune" allows the system administrator to dynamically change the priority of jobs as needs change from day to day and hour to hour.

Attention to control and tuning provides a useful basis for contrasting transaction processing with the UNIX system's traditional timesharing mindset.

First, a timesharing workload is far less predictable than a transaction processing workload. A timesharing administrator might know that 10 to 11 am

Illustration by Victor von Beck

is the busiest part of an average day, but this offers no insight into the sorts of jobs that will be run during any one particular period. Without a clear notion of workload or schedule, the system administrator cannot choose in advance the jobs that should have highest priority. Control is thus lost.

Secondly, even if the more important jobs could be identified, the UNIX system offers no mechanism beyond superuser **nice** intervention for favoring them. Using **nice** alone is not sufficient. For instance, if a long running job, say the formatting of a large document using **nroff**, was temporarily of utmost importance, **nice** could not by itself ensure that it would keep running. That's because the system offers no systematic capability for tuning.

Unfortunately, the natural bias of UNIX system users toward timesharing and away from control and tuning is apparent in the commercial DBMS products currently available.

TRANSACTION PROCESSING WITHIN AT&T

One of the most widely deployed UNIX-based transaction processing systems is known as the Automated Repair Service Bureau (*Bell System Technical Journal*, July-August 1982). ARSB was developed by Bell Laboratories to verify, record, and track customer phone service troubles for the Regional Bell Operating Companies. It is a distributed system capable of running 3500 transactions per hour on each of several PDP 11/70s connected via a local area network. The first UNIX system version of the ARSB was deployed in 1980, and today over 200 ARSB installations are active across the country. Originally developed on UNIX Release 4.0 with an early version of shared memory and interprocess communication, ARSB was recently ported to standard System V.

The ARSB is just one of several dozen System V-based transactional application developments that have been developed at AT&T for internal use and external sale. Many of these systems use tools that evolved during the ARSB development.

TRANSACTION ADMINISTRATION AND PROCESS ARCHITECTURE

Key to transaction processing efficiency in the UNIX operating system is process architecture. Typical commercial DBMS products use two processes per terminal, one for the terminal handler and one for database processing. These processes are often connected via a pipe. As might be guessed, processes multiply quickly under this approach. Had ARSB employed a similar scheme, for instance, it would have had to handle well over 1000 concurrent processes (quite impossible on an 11/70!).

For transaction processing, the process-per-terminal approach is not much better than one once taken in AT&T by a group of System V rookies. These programmers, well experienced in IBM's IMS, were told to build their first UNIX system-based transactional application. In their initial design, the process-per-terminal that handled user input would **fork**/**exec** each different application as needed. The **exec**'d load module contained both application code and DBMS code. (See Figure 1 for a representation.) Fortunately, the system prototype only had to handle 12 users who each entered a maximum of 30 transactions per hour. More fortunate yet was the fact that an expert from ARSB was called in *before* the system actually went into production.

The process architecture inefficiencies were obvious to the expert. First, the overhead of the **fork**/**exec** cost the system about 200 milliseconds per transaction and often led to additional disk accesses. Second, the application process opened the database with each transaction, adding an extra

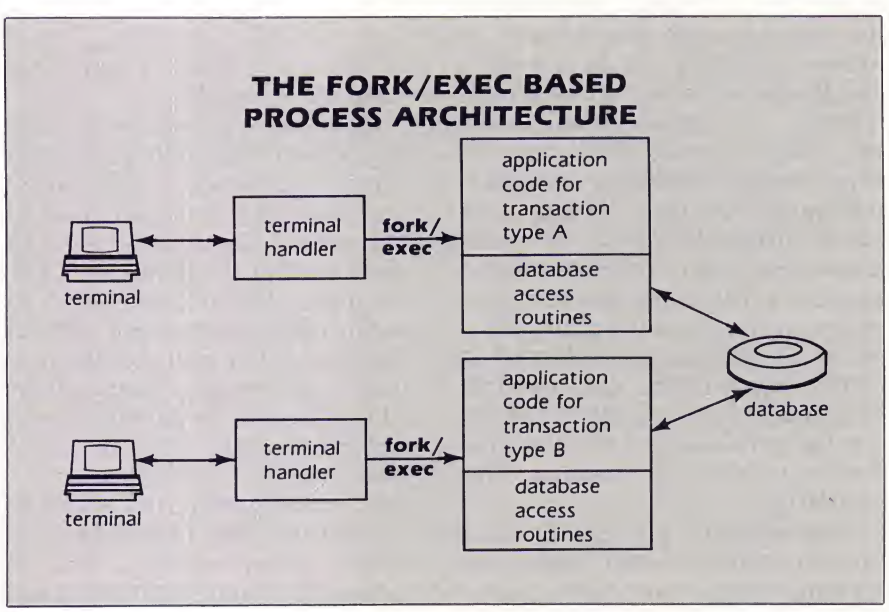


Figure 1

250 milliseconds per instance. Third, memory was used inefficiently since each transaction was encapsulated in a different load module. This was the most subtle of the problems. If five different terminals were involved in five different transactions, the application processes could not share text—a particularly wasteful problem since 80 percent of the code, the DBMS part, was the same from application to application.

In both the rookies' approach and the typical commercial DBMS approach, there is no central point through which transaction requests can be funneled. As a result, there is no means of transaction administration—no way to control or tune transaction execution. Suppose 10 out of 50 users were to start up long-running database scans. Since the typical DBMS product uses an application process per terminal, the requests can't be throttled by single-threading them. Even if they could, how would the system administrator be able to see to it that some long-running transactions enjoyed a high priority? For example, a bill printing transaction is almost always more important than the printing of a marketing summary report, but it would be extremely difficult to tune for temporary favoritism in process-per-terminal schemes.

The ARSB system allows for transaction administration by taking a "bucket brigade" approach to process architecture. A set of about 100 cooperating processes, known as the Group, is started at system boot time by a module known as the Group Manager. Each process runs continually as a daemon. Input is first processed by the terminal handler, which is a single user-level process capable of handling all 300 terminals. The input is then passed via System V messages to a parser/validator, where it in turn is passed to an appropriate application process running back and forth to one of two identical DBMS processes. One transaction visits an average of five processes, while three or four transactions reside together at any one time in a "brigade". (See Figure 2 for a representation.)

This implementation removes the process architecture inefficiencies of more conventional DBMS approaches. Daemons that are running continually cause only one **fork/exec** to be executed per day, just as database processes only need to open appropriate databases once a day. Duplicate modules, in the meantime, can share text. Control and administration, however, are implicitly achieved. The Group is pre-tuned for the ARSB application and hardware configuration, both in terms of the means by which transactions proceed from process to process, and the number and type of processes that are started.

All of these advantages also can be achieved using other approaches. In the case of the rookies' application, a "name server" strategy was used to solve their process architecture problem. As in the ARSB, System V messages were used, and there were far fewer processes than users. However, the architecture was not implicit in a Group Manager—instead, a requesting process used a name server to see where it needed to go to get the

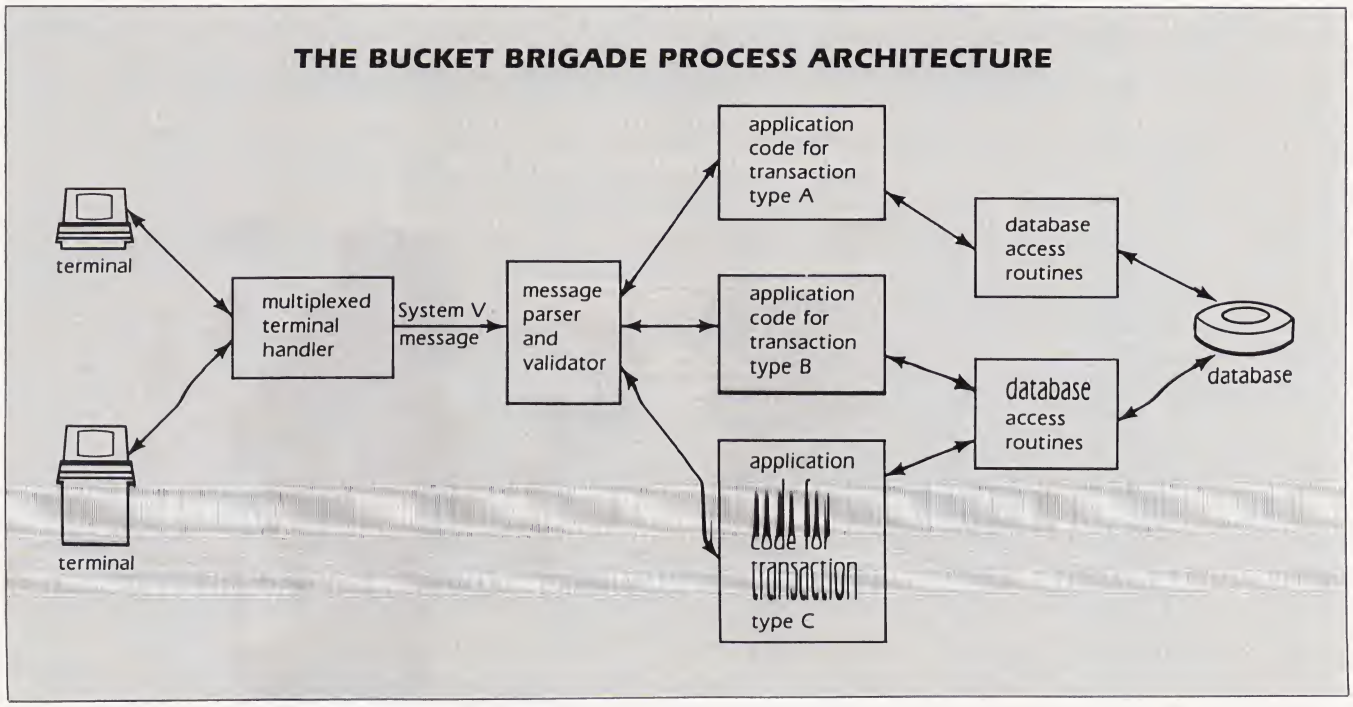
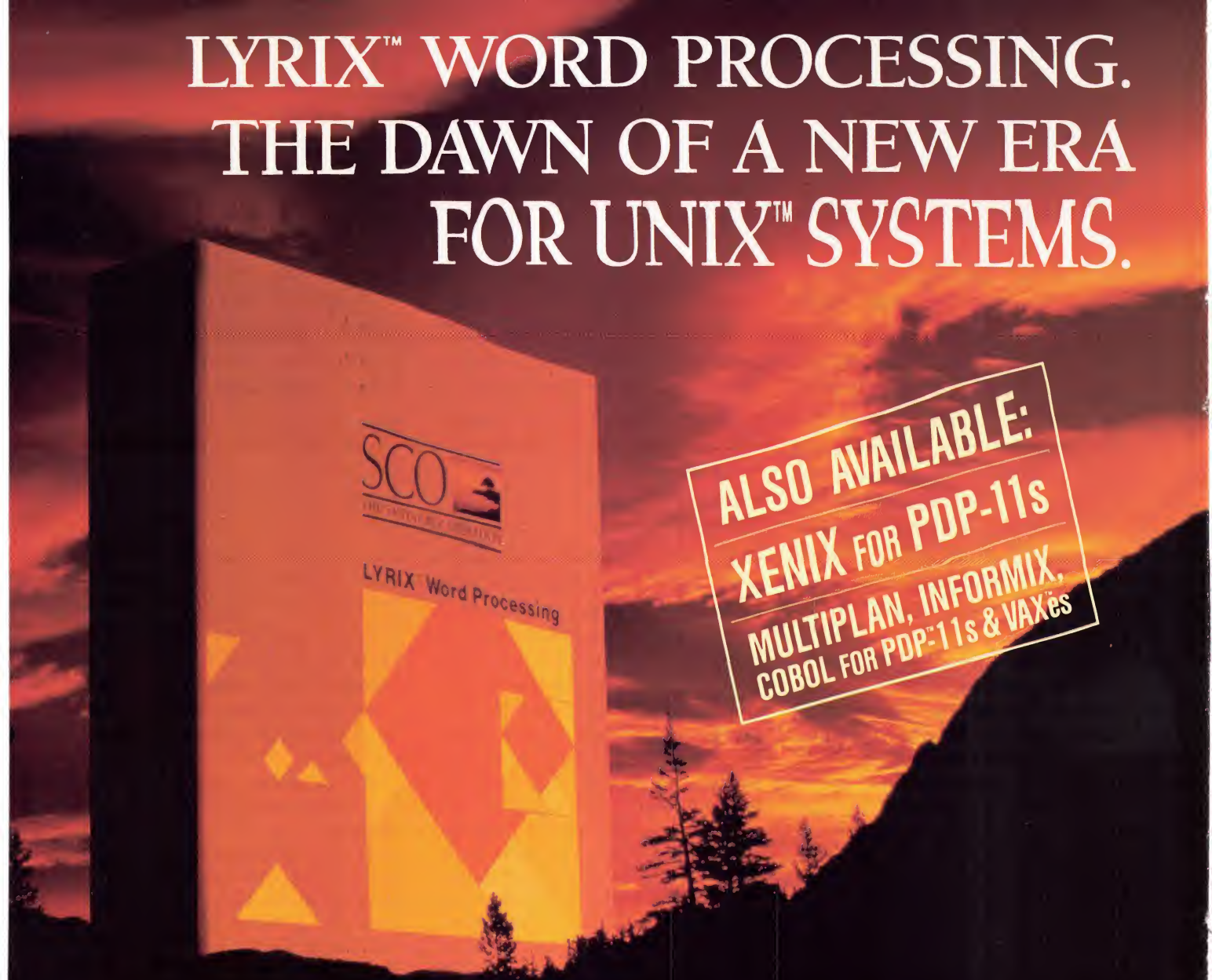


Figure 2

LYRIX™ WORD PROCESSING. THE DAWN OF A NEW ERA FOR UNIX™ SYSTEMS.



State-of-the-art word processing developed specifically to harness the power of UNIX-based systems. A breakthrough in business productivity and a cornerstone of the multi-user automated office.

Adaptable to any language, it's available worldwide for more UNIX systems – minis, micros and personal computers – than any other word processor.

Come up to Lyrix. Great word processing from the people who know UNIX and your office automation needs best – SCO.

For the UNIX systems user, it's a brand new day.

SCO
THE SANTA CRUZ OPERATION

(408) 425-7222
TWX: 910-598-4510 SCO SACZ

UNIX business software available from SCO includes the XENIX® Operating System, Lyrix Word Processing System, Multiplan® Electronic Worksheet, Informix® Relational DBMS, LEVEL II COBOL™ and uniPATH™ 3270 Mainframe Communications.

©1984 The Santa Cruz Operation, Inc. • The Santa Cruz Operation, Inc., 500 Chestnut Street, P.O. Box 1900, Santa Cruz, CA 95061 • (408) 425-7222

Lyrix is a trademark of The Santa Cruz Operation, Inc. • UNIX is a trademark of AT&T Bell Laboratories
XENIX and Multiplan are registered trademarks of Microsoft Corporation • Informix is a registered trademark of Relational Database Systems, Inc.
LEVEL II COBOL is a trademark of Micro Focus, Ltd. • uniPATH is a trademark of Pathway Design, Inc.
PDP and VAX are trademarks of Digital Equipment Corporation

Circle No. 52 on Inquiry Card

next step done. This solution extended the ARSB approach, allowing for more flexible control and more direct tuning, and thus now is evolving into a standard transaction processing architecture within AT&T.

The name server does provide a means of control by allowing administrators to de-queue requests in priority order. The information determining priority might include expected runtime, relative importance of transactions to business needs, and percentage of original request completed by previously-visited processes. Specific long-running database scans can be throttled by having only one daemon, represented by a single entry in the routing table. The most important long runners may have two or more entries.

The name server also provides tuning capabilities. Transaction priority information can change over time; for example, queries could be more critical than updates most of the time, but be far less important in rare instances, such as at the end of the month. The system administrator can respond to this by adding and deleting daemons on the name server's tables.

Further, name servers provide for the real-world occurrence of system overload. By making note of a systemwide upper threshold of work, it can tell a requesting process, "System busy, please try again later."

A degree of recoverability can also be provided this way. In at least one case I'm aware of, redundant daemons were installed in the trial site for a new application. When one crashed, the other picked up all incoming work without delay.

The name server concept has been implemented by various AT&T projects as device drivers or as user-level code. Some applications use a binary request model, where responses are always sent directly back to originating processes. Some other applications, though, use table-driven process routing. Results from one server are routed dynamically to either the next appropriate server or back to the original client.

Systems providing quality transaction processing service *have* been built on top of standard System V without any kernel modifications whatsoever.

INPUT HANDLERS

Experience shows that transaction processing involves an entire family of input handlers. Clerks use fixed forms to enter bread-and-butter transactions every minute or so from CRTs. System administrators and more sophisticated end users interact with the applications on a more ad hoc basis. And, there's often a non-CRT form of input that must also be read and processed.

Terminal handlers designed for clerical users generally do not bother with pop-up windows, full-screen friendly error messages, or fancy help. The object, after all, is efficiency, and these users are so well trained for specific interactions that fancier features are superfluous. Handlers designed with clerical users in mind thus should refrain from expensive field-by-field validation communications with the DBMS since errors are not common. To avoid the process-per-terminal syndrome, a software multiplexor is commonly preferred.

For the system administrator or sophisticated end user, though, fancier forms are appropriate. Since clerks presumably outnumber such users, efficiency is not as critical. Thus, for them, a process-per-terminal architecture is often acceptable. The needs of the sophisticated user are less predictable than those of the clerk, so multiple windows, mouse controls, and elaborate help also become important.

Even these non-terminal input handlers and fancy form handlers must be easily integrated into the process architecture, though. Despite their small numbers, sophisticated users could generate as much transaction load as clerks requesting monthly summaries. The controls that apply to bread-and-butter transactions thus must also rule ad hoc work.

A caching mechanism for form definitions is also valuable if one is to avoid going to disk for each form request. One terminal handling system I'm aware of provides a shared-memory buffer that can be pre-loaded with popular form definitions. The segment of the buffer not taken by preloaded forms is managed by the system in a least-recently-used fashion.

A FEW DATABASE ISSUES

Among the absolute necessities for DBMS applications used in transaction processing are high performance, atomic transactions, recovery, and concurrency control. These are topics that are covered extensively in other literature, including several of the other articles in this magazine, so I won't belabor them

here. But experience suggests that there are also other needs that are not quite so obvious.

The various files involved in a complex database are not homogeneous. Thus, it is necessary to vary many parameters file-by-file, including block size, hash function, logging strategy, and buffer-pool management algorithms. For example, one network-model DBMS developed and used widely in AT&T provides several buffering strategies, selectable per file at database generation time, including:

- a memory-only (i.e. non-disk) strategy used for temporary data needed only during the life of a single transaction.
- a permanent in-memory strategy useful for small files that are frequently accessed.
- a least-recently-used strategy for traditional files.

Further, the performance that characterizes complex databases makes a two-tiered interface to the DBMS necessary for transaction processing. One tier should offer a navigational, record-at-a-time interface while the other should provide a non-procedural interface.

The navigational interface, implemented via a programming language, is the one most appropriate for bread-and-butter transactions. Granted, it takes a week or two for a programmer using such an interface to develop a transaction that does inserts, modifies, retrieves, and deletes on a single record, while the same transaction could be developed in just a few hours using one of the forms-oriented application generators provided with many commercial DBMS products. But trades of bread-and-butter transaction runtime efficiency for reduced development time are not cost-effective. Since these transactions can be executed thousands of times an hour, the overhead of interrogating the data dictionary at the time of each request accumulates quickly.

Application generators and related ad hoc query systems *are* useful in transaction processing, though. End user report development and batch program development in the DP shop would be quite difficult without them. For reports that are only generated infrequently, it makes no sense to trade development time for runtime speed. Application generators and ad hoc query systems are also useful in rapid prototyping.

CONCLUSIONS

To do transaction processing requires two things: a bias toward performance in data management and forms handling, and a process architecture amenable to system administration.

There is much discussion now in the UNIX system community about the need to modify the kernel for transaction processing. File system and scheduler enhancements are themes that are especially hot. There's no question but that these features would be useful. But, while we wait for them to surface, note that systems providing quality transaction processing service *have* been built on top of standard System V without any kernel modifications whatsoever.

Kathy Anderson was a System V rookie, well experienced in IBM's IMS, when she was asked to write her first UNIX system-based application. She has been with AT&T Bell Laboratories for 12 years, the last two of which have been spent supervising the development and support of a System V transaction monitor and database management system for use within AT&T. She currently supervises the Transaction Systems Group at Bell Labs. ■

adax

X.25 FOR UNIX* Communications System

- Efficient, error-free data transmission to multiple hosts via international standard X.25, the only fully certified error-free public networking system used world-wide.
- User utilities
  - Remote user login
  - Remote mail service
  - Remote file transfer
- Compatible with widest number of host computers.
- Hardware available for VME, Multibus and others.
- Previously certified on TELENET, TYMNET and UNINET networks.
- Lowest cost per node.

Adax, Inc.

737 Dwight Way
Berkeley, CA 94710
(415) 548-7047

* UNIX is a trademark of Bell Laboratories.

Circle No. 23 on Inquiry Card

$4399.00*

Multi-user UNIX™ System V including: full operating system + 26MB Winchester + 1MB floppy + 1MB RAM + 2 RS232 ports.

EXPANDABLE!

COMPACT!

POWERFUL!
MC68000based

LOW PRICE!

LIGHT WEIGHT!

BUILT-IN SCSI INTERFACE!

* Price per unit. Volume discounts available. Dealer and Representative inquiries invited.

NEED MORE? An expansion board allows up to 2MB of RAM and eight additional users. Or choose the 45MB hard disk option and add more storage such as streaming tape drive, another Winchester or floppy—up to four devices attach to the built-in SCSI interface. Team the Model 7 with your favorite terminals and printers.

STANDARD SOFTWARE AVAILABLE: C, Cobol 74, Fortran 77, Pascal, Basic, Assembler, as well as spreadsheets, word processing, DBMS, etc.

All this and MORE in a light weight, compact (17.5″w × 16.2″d × 4″h) PORTABLE package from

MEGADATA

One of "America's Hot Growth Companies" (Business Week—May 27, 1985)

UNIX is a registered trademark of AT&T Bell Laboratories.

Call, write or use coupon.

MEGADATA
CORPORATION
35 Orville Drive, Bohemia, New York 11716
Tel. 516-589-6800 • Telex 14-4659

YES! I'm interested!

- ☐ Please send more information
- ☐ I'm in a hurry—please call
- ☐ Please send information about your 16-user UNIX™ SYSTEM

Name/Title _____

Company _____

Address _____

City/State/Zip _____

Telephone _____

Circle No. 53 on Inquiry Card

RULES OF THE GAME

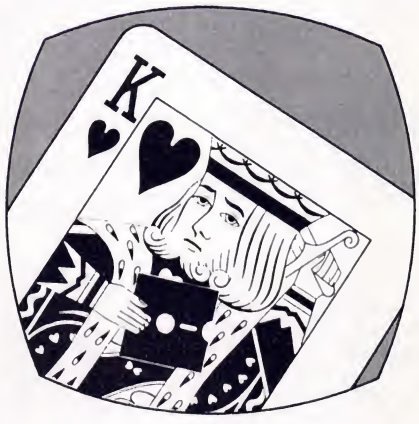
What's in a name?

by Glenn Groenewold

When I was a child, my father worked for an oil company. One day he happened to mention that since his employer had no refinery in our area, the gasoline we were buying really came from a competitor, though it was marketed under his company's name. I was astonished by this revelation. Hadn't I been hearing and seeing advertising in which his company's product was pronounced superior to all competing ones?

Patiently, my father explained that gasoline, basically, is gasoline, and that the equivalent products from other major companies were quite interchangeable. They were differentiated chiefly by what I have now learned to recognize as such things as brand names, logos, and slogans. These devices are at the core of modern marketing.

Even after we learn this truth, many of us continue to permit our choices to be governed to some extent by marketing factors irrelevant to the product itself. We do this each time we select the can with the familiar label from among the stacks of canned tomatoes in the supermarket. Perhaps it's habit, or maybe we resort to familiar products simply because it saves us time and mental effort.



But what has this to do with computing? Plenty, some people say. They point out that the typical software user usually doesn't care *how* the programs competing for his or her attention achieve their ends. Oh, a sophisticated few may take the trouble to investigate such things as how much memory each program requires, but these discriminating consumers are a distinct minority. Most users, it's claimed, will base their selection on the same sorts of factors that determine their choices of gasoline or canned tomatoes.

If this view is correct, product superiority shrinks in importance as a determinant of success in the software marketplace, while such items as trade names and logos correspondingly loom larger. Let's take a look at how the law views these marketing devices.

Federal and state laws protect several categories of proprietary marketing aids, and some of these overlap. From our standpoint, one of the most important is *trademarks*.

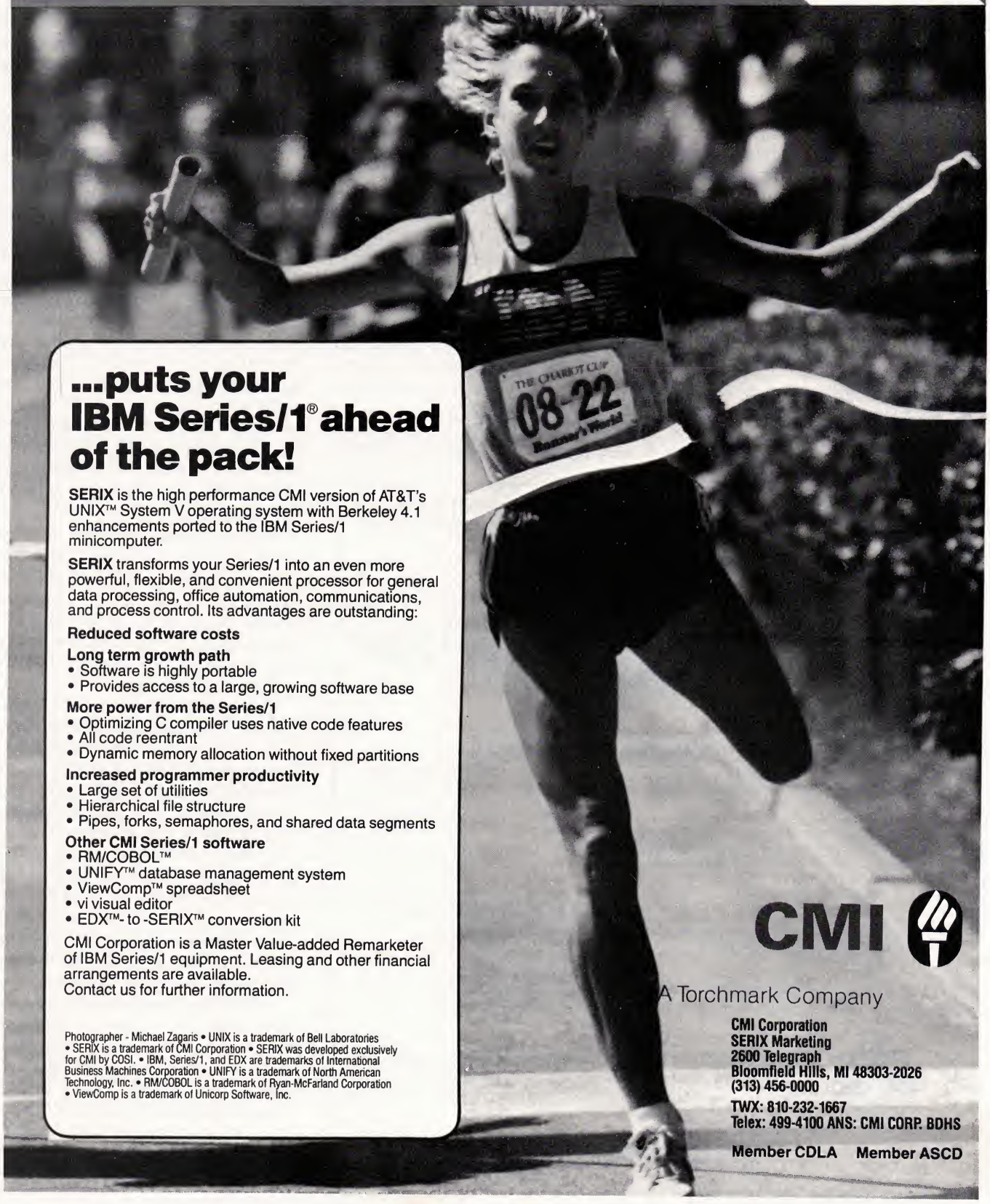
A trademark relates to *goods*, and computer software is considered such. Broadly speaking, a trademark is something that a manufacturer or merchant uses to distinguish his or her goods from those of others. This can be a name or words, or it can be something like a symbol or picture. Logos and corporate symbols therefore can constitute trademarks.

An important thing to remember about trademarks is that they cannot exist independently of a product. This means you can't set aside a trademark for use with some product you intend to develop in the future. The trademark must be placed on the goods themselves, or on their containers or the "displays connected with them", or on labels affixed to them. You can't establish a trademark just by using it in advertising, on letterheads, or on invoices, for example.

Another category of names, words, symbols, and the like that can be protected under the law is *service marks*. These are almost exactly like trademarks, except that they relate to *services* in-

SERIX

ANNOUNCING SYSTEM V UNIX™



...puts your IBM Series/1® ahead of the pack!

SERIX is the high performance CMI version of AT&T's UNIX™ System V operating system with Berkeley 4.1 enhancements ported to the IBM Series/1 minicomputer.

SERIX transforms your Series/1 into an even more powerful, flexible, and convenient processor for general data processing, office automation, communications, and process control. Its advantages are outstanding:

Reduced software costs

Long term growth path
- Software is highly portable
- Provides access to a large, growing software base

More power from the Series/1
- Optimizing C compiler uses native code features
- All code reentrant
- Dynamic memory allocation without fixed partitions

Increased programmer productivity
- Large set of utilities
- Hierarchical file structure
- Pipes, forks, semaphores, and shared data segments

Other CMI Series/1 software
- RM/COBOL™
- UNIFY™ database management system
- ViewComp™ spreadsheet
- vi visual editor
- EDX™- to -SERIX™ conversion kit

CMI Corporation is a Master Value-added Remarketer of IBM Series/1 equipment. Leasing and other financial arrangements are available.
Contact us for further information.

Photographer - Michael Zagaris • UNIX is a trademark of Bell Laboratories • SERIX is a trademark of CMI Corporation • SERIX was developed exclusively for CMI by COSI. • IBM, Series/1, and EDX are trademarks of International Business Machines Corporation • UNIFY is a trademark of North American Technology, Inc. • RM/COBOL is a trademark of Ryan-McFarland Corporation • ViewComp is a trademark of Unicorp Software, Inc.

CMI

A Torchmark Company

CMI Corporation
SERIX Marketing
2600 Telegraph
Bloomfield Hills, MI 48303-2026
(313) 456-0000

TWX: 810-232-1667
Telex: 499-4100 ANS: CMI CORP. BDHS

Member CDLA **Member ASCD**

stead of goods. In the computing field, service marks could be important to someone contracting to perform systems management, for example.

The third category of interest is *trade names*. This simply has to do with the names under which people or companies do business. Trade names do not necessarily attach to particular goods or services, although they may. Sometimes a company's trade name and its trademark are the same; two obvious examples are *Exxon* and *Xerox*.

A MIXED BAG

The laws governing protection of trademarks, service marks, and trade names are based on common law principles. (Common law was discussed in this column last December. In brief, it encompasses the body of "judge made" legal rules that originated in England several centuries ago.) Since the laws passed by Congress don't cover everything in this area, it is often necessary to look to the laws of the various states—most of them have some sort of trademark legislation—or even to common law itself for the answers to questions that arise.

Federal law, and the laws of many states as well, provides for registration of trademarks and service marks under specified circumstances. However, federal statutes do not provide for registration of trade names.

Registration of a trademark or service mark is advantageous because it creates a legal presumption that the mark is valid. This can be quite useful if it becomes necessary to go to court to stop someone from infringing your mark.

Under federal law, a trademark must take on what is known as a *secondary meaning* in order to be registrable. It's difficult to define this term; among the trademarks that have successfully passed the test are *Chap Stick* and *Holiday Inn*. A trademark can't merely be descriptive of the product, nor can it be recognizable only as the surname of the manufacturer or seller. (However, an individual's name *can* be registered as a *service* mark, where it has become associated by the public with a particular type of service; one example would be entertainer Johnny Carson.)

Once a trademark or service mark has been accepted for registration under the federal statute, the owner of the mark is entitled to use the symbol ® in conjunction with it. It's against the law to make use of this symbol in any other situation.

A number of well-known trademarks and service marks have not been registered, sometimes for esoteric legal reasons. In an attempt to protect these unregistered marks, the informal practice of using the letters "TM" to identify a trademark and "SM" to identify a service mark has evolved. Though these abbreviations have no official standing, they do serve notice on persons who see the marks that an ownership right is being asserted.

UNIX is among the trademarks that have not been registered. Since it's particularly important that the owner of an unregistered mark be vigilant against the appropriation of the mark by others, it's not surprising that AT&T's lawyers have been quite busy

UNIX/XENIX Communications Available NOW!

Put your computers on speaking terms.

$295.00

Introducing TERM. Communications Software

Everyone from the beginning computer user to the expert finds TERM easy to learn and powerful to use. Just plug it in and go! In a few keystrokes you can access a remote database or send a group of files to another system.

TERM allows your computer to perform efficient, error-free exchange of binary or text files, over phone lines or hard-wired circuits at speeds of up to 9600 baud. Available options allow you to include or exclude a group of files for transfer in a single command.

TERM's "data capture" feature allows saving transcripts of sessions with remote mainframe and minicomputers to disk for later editing or printout, if desired.

- Pre-installed and ready to run
- Automatic error checking and re-transmission
- Wildcard (*.*) file send/receive, capability
- Xon/Xoff, Etx/Ack, Ascii protocols for communications with non-TERM systems
- Full/half duplex emulation mode for remote systems
- Modem7 protocol for remote bulletin boards
- Auto-dial/Answer and Hangup supported on Hayes Smartmodem 300/1200 and compatibles
- Programmable batch file capability
- Unattended file transfer/auto logon
- Translation tables for input and output
- Remote maintenance capability

Term is available NOW on the Altos 586, IBM AT, Tandy Model 16, AT&T 362 and IBM PC/XT, MSDOS and many others. Find out how easy it is to get your UNIX, Xenix, and MSDOS machines all talking together.

CENTURY SOFTWARE
We make it easy for you.

9558 South Pinedale Circle
Sandy, Utah 84092
(801) 943-8386

Circle No. 21 on Inquiry Card

writing letters telling people to watch their use of the UNIX trademark. The reason that the company's licensing agreements contain specific restrictions on licensees' uses of the trademark should also be apparent.

SELECTING—AND PROTECTING—A NAME

Registration will be denied any trademark or service mark that appears to be in use by someone else. So it's important that this be determined *before* you launch your product or service. There are search companies that will provide this investigation—for a fee, of course—and there are legal publications that contain alphabetical lists of trademarks that have been registered or have been upheld in court decisions. It's a good idea to consult your local law librarian for particulars.

Once you have selected a mark and have begun to use it, your work has only begun. It's vital that you take action any time you get wind of someone else using the same name or symbol to identify another product or service, *or* whenever you hear of someone else using your mark without indicating so. In either case, you should write a letter telling the offenders to "cease and desist" whatever it is they're doing. And if they don't? Well, then it's time to see your lawyer.

THE GENERIC PITFALL

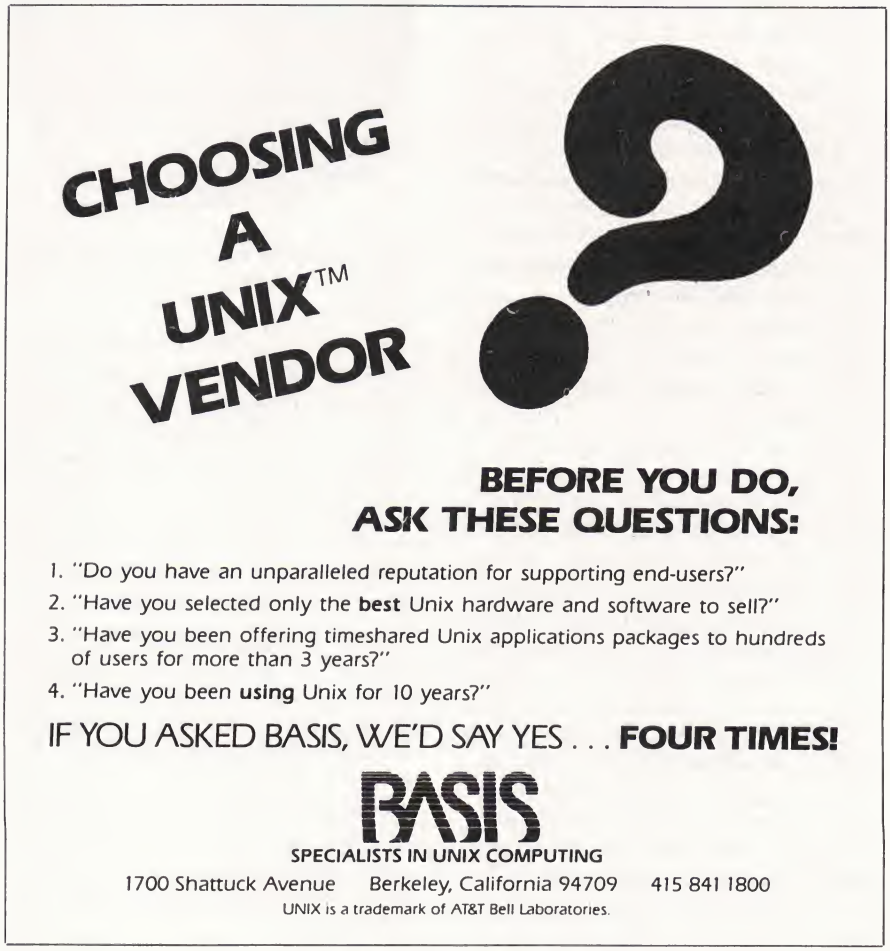
In American law, there often are penalties for succeeding too well, and this area is no exception. If you promote your mark so efficiently that it becomes a *generic* identification of your particular type of good or service, you stand to lose it.

There have been some celebrated debacles of this sort—one of the earliest casualties was *aspirin*—and there also have been a number of close calls. Kodak and Coca-Cola (for "Coke") have had scares at one time or another, and many of us will remember Xerox Corporation's institutional advertisements reminding us that its trademark is *not* a verb. Just recently the makers of Toll House cookies have seen their trademark pass into the public domain as just another part of the language.

Perhaps what will serve to save some creators of software from becoming the victims of their own success in promoting trademarks is simply the ephemeral nature of the product. By the time a program has become well-enough known to the public that its trademark is in danger of becoming a generic term, it may well have been rendered obsolete.

Short-lived or not, trademarks and service marks appear destined to be a major factor in determining marketplace success in the computer industry. While they do not in any way protect the product itself, they may be decisive in dictating how many people purchase it. That just might make the marks more important than all the copyrights, patents, and trade secrets put together.

Glenn Groenewold is a California attorney who devotes his time to computer law. He has served as an administrative law judge, has been active in trial and appellate work, and has argued cases before the state Supreme Court. ■



Circle No. 20 on Inquiry Card

INDUSTRY INSIDER

Why the RISC route?

by Mark G. Sobell

"RISC" is a term one hears often in conversations these days. The reason is simple: RISC describes one of the most exciting directions in current technology. The acronym stands for Reduced Instruction Set Computer. An instruction set in a RISC architecture may consist of 50 or so simple instructions that can play the same role previously assigned to *hundreds* of complex instructions. Most computers available today, whether mainframe or micro, rely on the larger instruction sets. Because the RISC instruction set is simpler, a machine built around it can run significantly faster than comparable units using conventional architectures.

One of the design goals for a RISC is to reduce the number of side effects instructions generally have. No longer is it necessary to have auto-incrementing, fancy type addressing, or instructions that "move a character string terminated by a null byte from here to there". Speed of execution has been increased as a result because simpler instructions run faster. Simpler instructions also mean cleaner exception processing: there is less to deal with when an interrupt occurs. The RISC approach appears likely to avoid one of the major problems confronting chip manufacturers today: the chore of getting exception processing on new, complex chips to work correctly in all cases.

Another feature most RISCs offer is the ability to clean microcode off processor chips. Microcode is CPU chip software that breaks down complex machine instructions into a series of simpler instructions executable by the chip. Under RISC architecture, each instruction is simple enough to go directly to the chip. To compensate, the smarts that were previously embedded as microcode can now be moved to the software (compiler/assembler) level. Because it is at a lower level, RISC object code tends to be about 150 percent of the size of conventional code, but this bulk is more than compensated for by increased processor throughput.

RISC and VLSI (Very Large Scale Integration, a technology used in constructing microprocessors) seem like technologies meant for each other. The most complex, expensive, and time-consuming part of VLSI chip design is the layout and debugging of chip logic (the part of the chip that executes instructions). The easiest part is the design of the data area of the chip since data areas are regular and can be replicated many times. A RISC implemented in VLSI technology offers the advantage of minimizing chip logic work while emphasizing data area design.

DIFFERENT STYLES OF RISC

RISC is a generic term that people have attached many different meanings to. Many of the distinctions between different RISCs come from the information stored in the data portion of the VLSI chip.

The IBM/Stanford style RISC uses the data portion of the VLSI chip for general-purpose registers. An optimizing compiler must be used to take maximum advantage of these registers and increase processor throughput.

The Berkeley style RISC uses the data portion of the chip for register windows. These windows essentially provide for fast access by keeping the top of the stack, or several different stacks, on

ANNOUNCING THE 2.7 MIPS, 68020-BASED UNIVERSE 32.

BUY BEFORE OCT. 31, GET A 140 MB DISK FREE!

In 1982, we leapfrogged the 16-bit minicomputers when we introduced the first 68000-based supermicro with a true 32-bit architecture. Now, in 1985, that means we can take full advantage of the remarkable performance of the Motorola 68020 microprocessor simply by plugging it into a 32-bit architecture that's already proven in 2,000 Universe installations.

The result is the new Universe 32: a 2.7 MIPS powerhouse that we can deliver **now.**

To make sure we deliver **more 68020-based systems in 1985 than any other company**, and to plant the seeds for long-term relationships, we're making an exceptional offer.

You pay for the basic Universe 32 Model UV32/35T (1 Mb RAM, 35 Mb disk, 45 Mb streaming tape, four serial ports; price $24,900).

You get the Universe 32 with a FREE upgrade to 140 Mb (114 Mb formatted) disk; FREE upgrade to 4 Mb RAM; FREE upgrade to 12 serial ports; and FREE UN/System V Operating System (derived from UNIX System V under license from AT&T). Orders must be placed by October 31, 1985, for delivery by December 31, 1985. After October 31, this same system will cost you $43,700. No quantity limit. No additional discounts apply. Offer available only in the United States.

For full details call **(617) 626-1000** or write Charles River Data Systems, 983 Concord Street, Framingham, MA 01701, Telex 681-7373 CRDS UW.

GRABBIT!

CHARLES RIVER DATA SYSTEMS

Universe is a trademark of Charles River Data Systems. UNIX is a trademark of AT&T/Bell Laboratories.

Circle No. 50 on Inquiry Card

the chip. The RISC design employed by Pyramid Technology Corp. uses register windows while simultaneously implementing instructions by way of microcode.

It is also possible to use the data area of the VLSI chip for cache memory (high-speed, immediately available memory), but current technology does not allow you to store enough data on the chip to make this approach practical. Finally, data areas also can be used for memory management functions.

All of the big manufacturers are already at work on RISCs: HP's next generation of machines, codenamed "Spectrum", will be RISCs. Rumor has it that DEC West is working on two RISC-based products. IBM is expected to announce a RISC workstation later this year. And AT&T is reported to be working on a RISC project targeted at running UNIX efficiently.

MIPS COMPUTER SYSTEMS

The acronym *mips* stands for Millions of Instructions Per Second. It is a measure of raw computer power: how many instructions can a processor process in a single second? A VAX 11/780 is rated around 1.0 mips. A Motorola 68010 comes in at between 0.5 and 0.6 mips while a 68020 can do 1.0 to 1.5 mips (or slightly faster in some cases).

MIPS is also the name of a Silicon Valley startup that just received $9 million in venture financing to produce a RISC. According to John Mashey, one of the Bell Labs PWB/UNIX team members who now serves as Manager of Operating Systems at MIPS, boards using the first pass of the MIPS chip will run at between 3.0 and 5.0 mips, depending on memory type. That would assume that the proprietary chip was running at 8 MHz. Mashey expects the second iteration of the chip to double that speed and nearly double the mips rating. Imagine, if you can, a single-board, desktop computer offering the power of eight VAX 780s.

Why else would venture capitalists invest so heavily in a company going up against IBM, AT&T, DEC, and HP? Probably because, in addition to the technology, they looked at the technical accomplishments of the players on MIPS' side: John Hennessy, former project leader of the MIPS RISC effort at Stanford University that produced a RISC-based microprocessor that outperformed commercial microprocessors by a factor of five. John Moussouris, the former IBM liaison to Stanford and Manager of VLSI System Integration at IBM's Thomas J. Watson Research Center, where he designed the logic for a very high performance 32-bit RISC-based VLSI processor. Edward Stritter, chief architect of the Motorola 68000. Todd Basche, architect of the Apollo DN660 workstation. Les Crudele, architect of Motorola's 68010 and 68020 processor family. And, of course, John Mashey, a major contributor to the Programmer's Workbench version of UNIX at Bell Labs. The list continues, including more talent from IBM, Intel, Zilog, and Data General.

The smarts that were previously embedded as microcode can now be moved to the software (compiler/assembler) level.

OVERVIEW OF A RISC SYSTEM

One of the issues facing anyone developing a RISC is the question of just what instructions to put on the chip. "When you minimize what is on the chip, you'd better make sure you've picked out the right stuff," Mashey said. "We will, of course, run UNIX. We set up the chip with UNIX in mind, especially tuning the areas of memory management and exception (interrupt) handling."

Aside from instruction set considerations, RISC compilers are critical to performance. "MIPS licensed the technology developed at the Stanford MIPS project and is using it as a base for further compiler and optimizer development," Mashey explained. "We developed backends for optimization, code generation, and assembly. We developed front-ends for Fortran, Pascal, and C. An optimizing compiler is a good fit with MIPS-style RISC. It helps minimize code size, and only a good global optimizer can take advantage of the large number of registers."

The theoretical goal of a RISC is to execute one instruction per basic machine cycle. To help achieve this goal, the computer overlaps instruction execution in what is called an *instruction pipeline*. The computer is always starting work on new instructions before completing previous ones. This technique works well—until the machine runs across an instruction that's dependent on one that precedes it. There are several solutions to this problem, including hardware-implemented *pipeline interlocks* that can stall the computer until all the necessary instructions have run to completion. The following source code and the result-

ing assembler code demonstrate the problem. The examples are not written in any particular language; they are simply conceptual representations:

```
Problem source code:
    A = B + C
    D = E

Assembler code generated by problem:

      load word    register_1, B
      load word    register_2, C
*     add          register_1, register_2
      store word   register_1, A
      load word    register_3, E
*     store word   register_3, D
```

The instructions marked with asterisks depend on previous loads being complete by the time they start execution. A simple software solution would be to have the assembler insert do-nothing instructions (*nops*) before each of the instructions marked with asterisks. Although this solution works, it is inefficient because it wastes the computer's time.

Most computers use hardware *nops*, or interlocks, to avoid the software *nops* that would otherwise be required. MIPS, however, has come up with another, more efficient software solution called the *Pipeline Reorganizer*. (As a matter of fact, another thing MIPS stands for is "Microprocessor without Interlocked Pipeline Stages".) The Reorganizer takes the assembler code that compilers or programmers generate and moves it around so that it will run most of the time without the need for *nops*. As an example, it might generate the following code based on the preceding example:

```
Assembler code modified by
the MIPS Pipeline Reorganizer

      load word    register_1, B
      load word    register_2, C
*     load word    register_3, E
      add          register_1, register_2
      store word   register_1, A
      store word   register_3, D
```

In this example, the line with an asterisk is the one that the Reorganizer moved. By taking the instruction that loads *register_3* and putting it where it otherwise would have needed a *nop*, the

WE CAN TEACH YOU IN A FEW DAYS WHAT WE'VE BEEN LEARNING FOR OVER SIXTEEN YEARS.

As the developers of the UNIX™ System, we at AT&T offer comprehensive training that's also practical and useful for your business. Whatever your level of expertise, we can teach you the specific skills that will have you using the UNIX System to organize and expand your computing system for maximum efficiency.

At AT&T each student gets the use of an individual terminal, teachers that can stay late at night, and a choice of training centers. You can even center your training right around your own office.

And because we are continually expanding our courses to incorporate the developments of UNIX System V, you're assured of always getting the most up-to-date information.

Discover the power of UNIX System V at an AT&T training course. And develop your UNIX System skills with the people who develop the UNIX System. **Call us today to reserve your seat or for a free catalog.**

1-800-221-1647, Ext. 335

AT&T
The right choice.

©1985 AT&T Information Systems.

Reorganizer got rid of the need for two *nops* (the one it replaced with the load instruction, and the one that would have otherwise been required between between the *load register_3* and *store register_3* instructions.

The MIPS Reorganizer puts useful code in 80 to 90 percent of the delay slots that would otherwise be filled with *nops*. What's more, 20 percent of the instructions ultimately executed typically will be placed in these delay slots. That means the Reorganizer provides 20 percent more throughput than schemes that insert *nops* in delay slots.

This example only hints at how MIPS is moving some of the intelligence of its computer from the hardware/microcode arena to the software side of things. In addition to allowing the hardware to be much simpler, this approach also makes it much easier to modify software (since, as you might guess, microcode embedded on a microprocessor chip is *not* easy to modify).

SUMMARY OF RISC

Mashey summed up the major RISC issues by saying, "Complexity in a computer is like garbage. You can't ignore it, but you can choose where to put it.

"Prior to RISCs, the trend in microprocessor architecture was toward putting as much intelligence on the microprocessor chip as possible, as in complex instructions decoded by microcode. With the advent of RISCs, more of the intelligence is moving into the software, where it is cheaper and easier to implement, debug, and modify.

"And of course the ability to implement a RISC using VLSI technology gives you a better price/performance ratio than was previously possible. The price/performance ratio worsens significantly when you go from a single chip implementation of a RISC to an on-board implementation to a multiple-board implementation."

Unfortunately, you cannot run out and buy a MIPS computer just yet. MIPS Computer Systems does not plan to start shipping for another year. And, when it does, it will be selling to OEMs who will then build products around the boards.

For more information on RISC and VLSI technologies, you might wish to refer to: "VLSI Processor Architecture", *IEEE Transactions on Computers*, vol. c-33, no. 12, Dec. 1984, and "Reduced Instruction Set Computers", *Communications of the ACM*, vol. 28, no. 1, Jan. 1985.

If you have an item appropriate for this column, you can contact Mr. Sobell at 333 Cobalt Way, Suite 106, Sunnyvale, CA 94086.

Mark G. Sobell is the author of the bestselling book, "A Practical Guide to the UNIX System" (Benjamin/Cummings, 1984) and the new "A Practical Guide to UNIX System V" (Benjamin/Cummings, 1985). He has been working with UNIX for over five years and specializes in documentation consulting, database programming, and ***troff*** *typesetting. Mr. Sobell also writes, lectures, and offers classes in Advanced Shell Programming and* ***awk****.* ■

UNIX POWER FROM DYNACOMP

100,000 software developers can't be wrong.*

UNIX is the chosen operating system for more than 100,000 software developers because it has the power they need. But developers aren't the only people who need computing power. Any business that wants multi-users to access the same files at the same time or wants to simultaneously run multi-task operations . . . needs UNIX. At Dynacomp, we offer UniPlus + ® System V by UniSoft Corp. For $1495. U.S. dollars you can run UNIX on the CompuPro® 816/E™ . . . a powerful 68K S-100 bus computer system that maximizes its memory for multi-user/multi-task operations.

UniPlus + includes all the standard UNIX System V features PLUS performance enhancements found only in UniPlus + . These features increase the portability, flexibility, and performance of UNIX, allowing an affordable operating system for program development, text preparation, and general office use.

If it's time for you to upgrade to UNIX, call your local Full Service CompuPro System Center in the United States or call Dynacomp in Canada for complete details.

210 W. Broadway
Vancouver, B.C.
V5Y 3W2
(604) 872-7737

DYNACOMP COMPUTER SYSTEMS LTD.

46-6535 Mill Creek Dr.
Mississauga, Ont.
L5N 2M2
(416) 826-8002

*AT&T estimates that there are more than 100,000 people currently developing software under UNIX. Dynacomp serves all of Canada and parts of Asia and the Pacific Rim. Call us for details and information on our full product line including Plexus. • UNIX is a trademark of Bell Laboratories, Inc. CompuPro is a registered trademark and System 816/E is a trademark of Viasyn Corp. UniPlus + is a registered trademark of Unisoft Corp. AT&T is a registered trademark of AT&T Information Systems.

Circle No. 28 on Inquiry Card

BREAK THROUGH THE SOFTWARE BOTTLENECK

UNIX™ APPLICATION DEVELOPMENT

TODAY is far more than the awkward collection of tricks and tools that are often labelled "4GL". TODAY provides a **COMPLETE application development environment** that will revolutionize the way you develop and maintain applications. **No UNIX* systems knowledge is necessary.**

Let's put it frankly: developing an application is a costly proposition. You'll need a highly skilled team of designers, analysts and programmers, and several man-years to get things off the ground. And that's not to mention the on-going costs of documentation, customization and maintenance!

TODAY tackles these problems through a new methodology with high performance architecture and a comprehensive range of features. It's so quick and easy to use that TODAY developers can do the whole job—design, analysis, development and documentation.

TODAY provides a comprehensive range of features that keep application building easy while optimizing development resources:

- **Powerful recursive logic and Decision Tables**
- **Synonyms, Menus, Prompts, Helps and Defaults for streamlined definitions**
- **Screen Painter**
- **A Report Generator which includes a Painter**
- **Push-button Self-documentation**
- **Audit Trails**
- **Source-code security through run-time only configurations**
- **Developed Applications instantly portable across UNIX* systems**

Because definitions are **Dictionary-based**, any changes are easily made in one central location. A key feature, **"tailoring"** lets you alter an application — perhaps to customize it for a particular site or user — without affecting the original version. If required, applications can be set up as Models (Prototypes) and later enhanced to grow and change with the business. Tailoring versions is the perfect solution for quickly generating multiple applications based on one Model.

TODAY runs under UNIX* or UNIX*-compatible operating systems on **super-mini down to micro business computers** using any of a range of databases. And if that's not enough, TODAY is backed by 14 man-years of research and development and the confidence of users who are breaking time zones in software development. See us at **Interex, Washington DC, September 8-13, *Booth 714,*** and **UNIX Systems Expo, New York City, September 18-20, *Booth 1303.***

bbj Computer Services, Inc.
2946 Scott Blvd.
Santa Clara, CA 95054
Telephone: (408) 727-4464

Circle No. 3 on Inquiry Card

*Unix is a trademark of A T & T Bell Laboratories Inc. TODAY Copyright © bbj Computer Services Pty Ltd. Melbourne, Australia, July 1983

DEVIL'S ADVOCATE

Look for that silver lining

by Stan Kelly-Bootle

Many of you doubtless are aware of recent hiccups in the erstwhile exponentially expanding Computer Industry. You'll be cheered to know I refuse to join those gloomy commentators who scream "major recession" and "Armageddon" every time Silicon Valley shuts down for longish weekends ranging from five to 15 days. God knows those of us in the business have earned a break: the company parking lot holds an adequate inventory of workstations, my own UNIX integrated accounting package (code named "Larghissimo Ma Non Troppo", which is the last remaining musical idiom unregistered as a software product) is but a few tweaks from perfection, and the Giants have a homestand coming up against the Dodgers.

When reading of plant closures and staff layoffs, it is tempting to proclaim that the Last Days of Tribulation are at hand: "And whosoever was not found written in the book of life was cast into the lake of fire" (Revelation 20:15), where "book of life" is interpreted as a company annual report containing a healthy bottom line.

I, though, prefer the word "hiccup" to "shakeout". This reflects my own calm view that what we are suffering is *not* the final judgment, nor even a chronic sickness. Rather, I maintain that

we are being hypochondriac over the spurious spikes and annoying discontinuities starting to appear on sales and profits graphs. Most of these graphic anomalies, I say, are entirely due to the unendearing quirks of the Macintosh Image Writer. If you step back far enough (to the rear of the Welfare line, for example) and half-close your eyes, the trend curves become smoother and less ominous, the fuzzy pie-charts assume a more edible disposition, and, hopefully, those ghastly Macfont legends disappear altogether.

It is good to see two of the leading mainframe manufacturers, Sperry and Burroughs, responding to this by closing in on a merger that might reverse their ailing fortunes. It will not be easy, though, to forge a unified product line from two such disparate ranges. Indeed, outside of Apple, it would be difficult to find a pair of systems of such daunting incompatibility.

In the widely debated case of the *Rise and Fall of Home Computing*, there is no doubt that even Invincible Business Machines has been disappointed. A closer inspection reveals that the root of the problem lies in unjustified expectations and crazy forecasting. If you predict a 400 percent growth, and crank up production accordingly, then a 200 percent growth, miraculous by any normal standards, becomes abject failure. The *absolute* number of home computers sold is a monument to marketing ingenuity and human gullability.

I offer one recent snippet to back this view. Having replaced a $300 typewriter with a $2000 Word Processor ("The advertisement said it was 'affordable' so how could I resist?"), the home computerist is next offered a $95 software package that allows direct keyboard-to-printer mode! "Bypass all those time-consuming diskettes! Forget all those funny filenames!"

The home computer peddlers have also overlooked the growing number of homeless persons (excluding those with Ph.Ds in Computer Science) who, more than any other segment of the market, are in need of affordable, system-

Why programming experts are choosing Uniworks productivity tools

Uniworks, Inc., was founded in 1984 to market superior software development tools to professional programmers working on UNIX™ and VAX/VMS™, and to support those products with the industry's best technical staff.

Our CCA EMACS™ editor environment is fast becoming an industry standard, and we expect the same from the Safe C™ family.

Find out how we can help you. Call today or send in one of the coupons on the following pages.

Read on . . .

UNIX, VAX/VMS, CCA EMACS and Safe C are trademarks respectively of Bell Laboratories, Digital Equipment Corporation, Computer Corporation of America, and Catalytix Corporation.

atic, integrated, press-any-key-when-ready general problem-solvers.

Pessimists, especially laid-off pessimists facing eviction and vehicular repossession, may well quibble with my carefree analysis. Unemployment, it seems, breeds a nasty form of intolerant cynicism that inhibits any rational assessment of reality. Indeed, I meet many who blame Reason itself for their plight. "My job was secure until they rationalized production", is a common complaint in Santa Clara County.

Ironically, the very work-free people whose leisure to ponder objectively on the deep structure of the cosmos would be the envy of any Golden Age Athenian,

Unemployment, it seems, breeds a nasty form of intolerant cynicism.

waste their time casting stones at the blameless.

Little did the semiconductor assembly line workers realize that, as predicted by Ezekiel, Marx, and Engels, they had been "tilling their own graves through the seven years of abundance; their lamps were left un-oiled, yea, they trimmed not the wicks thereon." They have literally automated themselves out of a job, and should expect no sympathy from the millions who were earlier victims of automation in other trades. An industry that was founded on the proposition that machines can legally work for less than minimum rate must accept the logic of using computerized computer assembly and the ALO (Automatic LayOff) package, which sends termination notices and W2s by electronic mail.

Nowadays, no self-respecting chip wants to be manhandled into this world—or even photographed alongside germ-ridden human fingernails. Letting the chips procreate and assemble in their own unsullied environment will certainly improve the yield and lower the unit cost. Naturally, there also will be cycles of glut and shortage—but eventually the brighter chips will adjust (they can hardly do worse than the semiconductor industry).

Programmers should be aware of similar suicidal trends in automatic software generation! Rash attempts to simplify **awk** are just the thin end of a dreadful wedge that could lead to major layoffs. An unemployed programmer is a pitiful sight. I have seen a few in San Jose. They gather round Automatic Teller Machines, idly tapping dead keyboards and dreaming of past glories.

Liverpool-born Stan Kelly-Bootle has been computing, on and off, at most levels since the pioneering EDSAC I days in the early 1950s at Cambridge University. After graduating from there in Pure Mathematics, he gained the world's first post-graduate diploma in Computer Science. He has authored "The Devil's DP Dictionary" and co-authored "Lern Yerself Scouse" and "The MC68000 Software Primer". ■

WHEN SERIOUS PROGRAMMING IS YOUR BUSINESS...

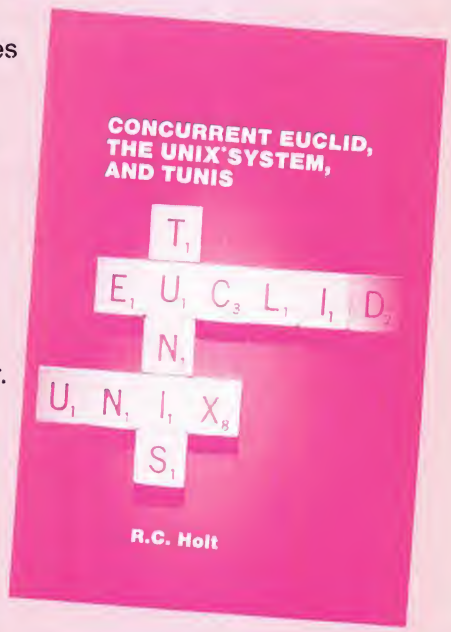
The Concurrent Euclid language for systems programming provides the best in efficiency, portability, reliability, and maintainability

Compilers running on UNIX/VAX, UNIX/11, VMS/VAX, with code generated for MC68000, MC6809, NS32000, 8086/8088 PDP-11, and soon running on IBM-PC

CONCURRENT EUCLID

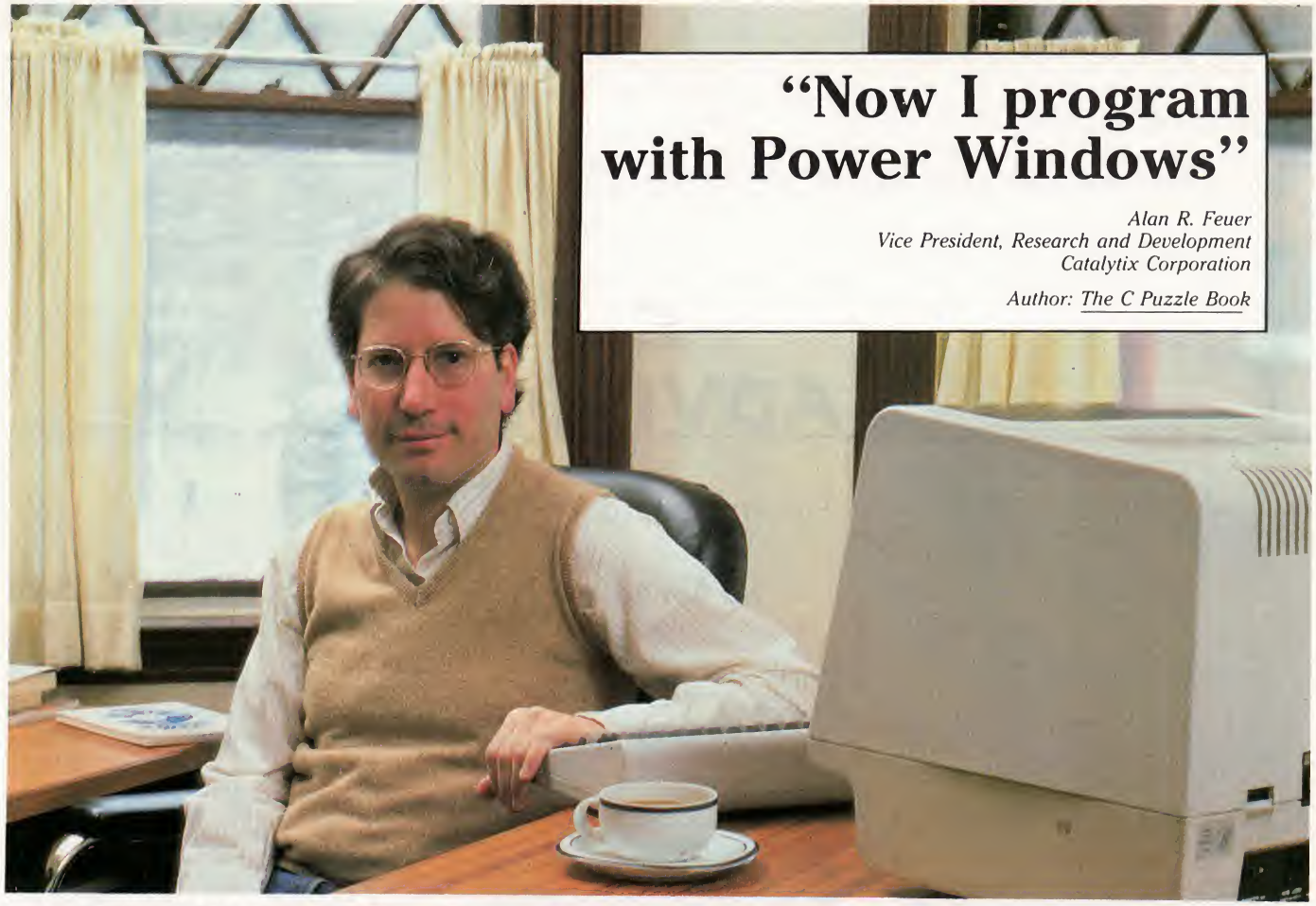
Compiler: CSRI Distribution Mgr.
Sandford Fleming Bldg 2002
10 King's College Road
Toronto, Canada M5S 1A4
Tel: (416) 978-6985

Book:
CONCURRENT EUCLID,
THE UNIX SYSTEM AND TUNIS
Available from:
Addison-Wesley Publishing
Company, Reading, MA. 01867
Tel: (617) 944-3700



CONCURRENT
E U C L I D

Circle No. 22 on Inquiry Card



"Now I program with Power Windows"

Alan R. Feuer
Vice President, Research and Development
Catalytix Corporation

Author: The C Puzzle Book

CCA EMACS™...The Most Powerful Editor Environment Available for UNIX™ and VAX/VMS™

"Programming with CCA EMACS, I can look at two or more files at once in different windows and then move text between them."

"POWER WINDOWS" are only part of the reason so many programmers are using CCA EMACS to make program editing and system development easier and faster.

Unprecedented power, speed, functionality, extensibility, and consistency across systems and on any terminal are others. Nearly 400 built-in commands let you do any job with a few keystrokes. And with our Common Lisp-based extension language, Elisp™, you can customize CCA EMACS to meet all your specific program needs.

CCA EMACS has two extensive recovery facilities, and is supported by a full online documentation package designed for beginners and experts alike.

This complete kit of editing tools runs under Berkeley UNIX (4.1 and 4.2 BSD), Bell UNIX (Sys. III and V), and VAX/VMS. Binary prices range from $380 to $850 for UNIX to $1900 for VMS.

Uniworks, Inc.

A Crowntek Company

Productivity Tools for Programmers

20 William Street • Wellesley, MA 02181

FREE TELEPHONE TRIAL. Call into our system for a tutorial review and actual product trial for CCA EMACS or SAFE C™.

For more information, telephone trial instructions, or to place an order, phone our customer representatives toll-free at:

800-222-0214

in MA 617-235-2600, or mail this form.

VISA and **MASTERCARD** phone orders accepted

Please send me information on:

☐ CCA EMACS
☐ The Safe C Development Tools
☐ AI Development Tools
☐ Your complete line of state-of-the-art programming tools
☐ Tell me about your free Telephone Trial Program
☐ Please send license forms

Name _____

Title _____

Company _____

Address _____

City, State, Zip _____

Phone ( ) _____

Uniworks, Inc. UR885
20 William Street • Wellesley, MA 02181

UNIX, VAX and VMS are trademarks of Bell Laboratories and Digital Equipment Corporation, respectively. Safe C is a trademark of Catalytix Corporation. CCA EMACS and Elisp are trademarks of Computer Corporation of America.

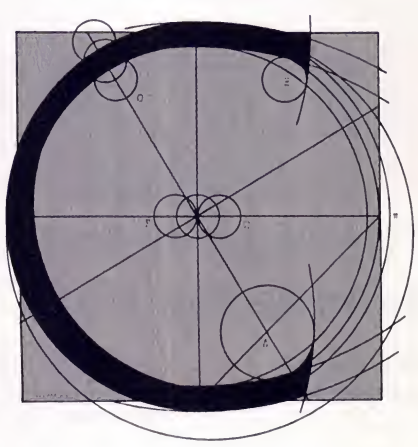
Circle No. 48 on Inquiry Card

C ADVISOR

File and record locking

by Bill Tuthill

As it stands today, UNIX is not the best operating system for running databases. It's true that UNIX itself is reliable enough, the file system is robust enough, and that at least 4.2BSD is fast enough. But VAX/VMS, for one, provides two features that UNIX does not: system-level facilities for indexing files, and standardized file and record locking. Both of these features are critical to database applications.



The lack of the first feature—indexed files—does not pose a huge problem. Because structured access techniques like ISAM (indexed sequential access method) and B-trees (balanced binary trees) are easy to implement on the user level, database packages can have indexed files without system help. It was UNIX, in fact, that proved that operating systems need not impose specific record structures on files. IBM/CMS, an operating system mired in useless notions of file structure, is an example of how it used to be.

So far, so good for UNIX. But then comes the second problem. File and record locking—unlike indexed files—is difficult to implement on the user level. The most primitive locking method is to create a temporary file that acts as a lock. This is inelegant, inefficient, and insecure. Without interprocess communication, the best alternative is to create a lock device driver and configure it into the kernel. As we know, though, UNIX has no standard technique for interprocess communication.

The bottom line is that UNIX offers no standard means for file and record locking. This is not to say that no standard exists: /usr/group, in fact, has had a locking standard for several years. But, of the common versions of UNIX, only 4.2BSD and Xenix provide file locking. Xenix alone provides mandatory record locking. Many independent UNIX vendors who don't deliver Xenix (like Convergent, Fortune, Onyx, Plexus, and Zilog) have implemented the /usr/group standard, but they have sometimes done so in subtly incompatible ways.

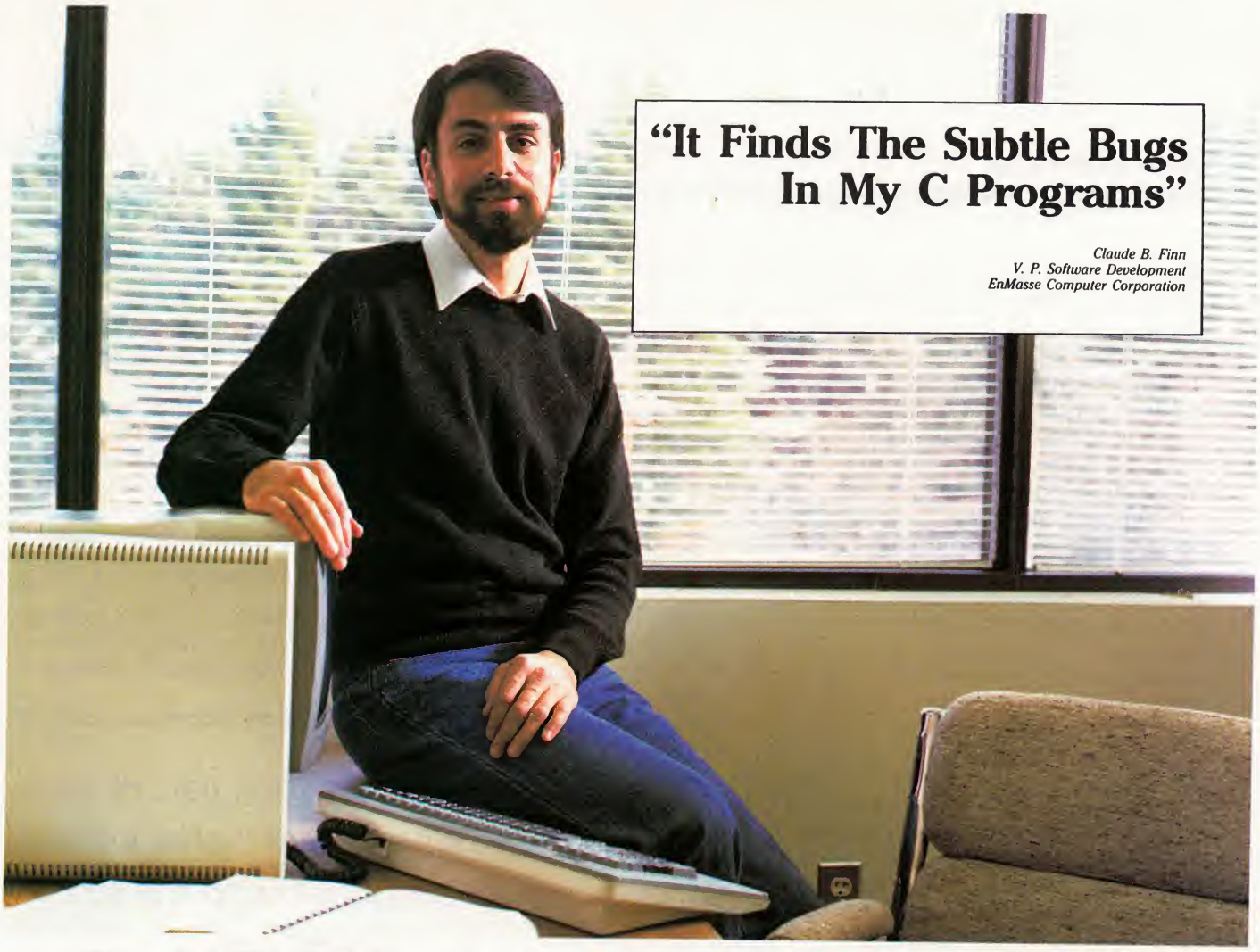
System programmers often recognize the importance of file locking. They know, for instance, that mail spool files should be locked during a mail reading session (so that mail isn't delivered unexpectedly after the spool file is changed). But programmers outside the database community generally don't recognize the importance of record locking. UNIX kernel hackers are often hostile to the very notion.

Consider an airline reservation system. While you are booking a seat on a particular flight, the record for seating on that flight must be locked so that somebody else doesn't book the same seat as you do. Locking the entire database file, though, would be an uneconomical measure, for it would preclude other agents from booking different flights at the same time. Clearly, a multiuser database system must be able to lock records (or regions) within a file.

If UNIX is ever to be successful at running serious database systems, it must have system primitives for file and record locking. This article traces the locking facilities available on various versions of UNIX.

THE /usr/group STANDARD

In the spring of 1981, John Bass published a paper in the Usenix newsletter *:login:*, that detailed the interface and implementation of a *locking()* system call for file and record locking. The proposal



"It Finds The Subtle Bugs In My C Programs"

Claude B. Finn
V. P. Software Development
EnMasse Computer Corporation

The SAFE C™ Family Can Literally Cut Software Development Time In Half. For UNIX™ and VAX/VMS.™

"Evasive bugs that use to eat up days — I'm finding them in minutes. Stray pointers, errant array indexes, parameter mismatches, misuse of string functions...I'm using Safe C automatic error detection to find them all."

Claude Finn is one of the many C programmers who have discovered that the Safe C family of software development tools dramatically enhances programmer productivity and improves software reliability and portability. Most Safe C customers have recouped their investment in these tools within the first month of active use. And with the security of Safe C their programmers are sleeping a lot easier!

The Safe C family includes the Runtime Analyzer, Dynamic Profiler, Standalone Interpreter, English to C Translator and C to English Translator.

FREE TELEPHONE TRIAL. Call into our system for a tutorial review and actual product trial for CCA EMACS™ or SAFE C.

For more information, telephone trial instructions, or to place an order, phone our customer representatives toll-free at:

800-222-0214
in MA 617-235-2600, or mail this form.

VISA and **MASTERCARD** phone orders accepted

Please send me information on:
☐ CCA EMACS ☐ The Safe C Development Tools
☐ AI Development Tools ☐ Your complete line of state-of-the-art programming tools
☐ Tell me about your free Telephone Trial Program
☐ Please send license forms

Name _____
Title _____
Company _____
Address _____
City, State, Zip _____
Phone (____) _____

UR885

Uniworks, Inc.
20 William Street · Wellesley, MA 02181

UNIX is a trademark of Bell Laboratories. VAX and VMS are trademarks of Digital Equipment Corporation. Safe C is a trademark of Catalytix Corporation. CCA EMACS is a trademark of Computer Corporation of America.

Uniworks, Inc.
A Crowntek Company
Productivity Tools for Programmers
20 William Street · Wellesley, MA 02181

Circle No. 47 on Inquiry Card

called for mandatory locks. About a year later, /usr/group published a standard that included a *lockf()* system call that had the same parameters and locking modes. The difference was that the /usr/group standard allowed for both mandatory and advisory locks. Advisory locks may be circumvented by programs not using *lockf()*, while mandatory locks cannot. Files with the *setgid* bit set are subject to mandatory locking under this standard.

Mandatory locks are probably not necessary, though. Both OS/360 and VAX/VMS have survived for years with advisory locks only, and many large databases, including airline reservation systems, have been implemented on these operating systems. Furthermore, mandatory locks are a potential security problem. Some user program could lock */etc/passwd*, for instance, and then go to sleep, causing the entire protection subsystem to hang.

The proposed standard *lockf()* system call allows a process to lock sections of a file. Other processes that attempt to lock that section will either block until the section becomes unlocked or return an error value. All locks on a file are removed once the file is closed, and all locks for a process are removed when the process terminates. The *lockf()* call looks like this:

```
lockf(fd, mode, length)
int fd, mode;
long length;
```

The file descriptor *fd* must come from a successful *open()*, *creat()*, *pipe()*, or *dup()* system call. The mode may be F_LOCK to lock a region for exclusive use, F_TEST to test for other locks, F_TLOCK to both test and lock, and F_ULOCK to unlock a region. Actually F_TLOCK is a non-blocking lock: if a region is locked, it will return an error, rather than sleep. The third parameter, *length*, specifies the number of bytes to lock, measured from the current position in the file. This can, of course, be changed with the *lseek()* system call. Negative values indicate how far back from the current position to lock. Even locks past the end of file are possible if one wishes to protect against appending. If locked regions overlap, they are combined into a single region.

The potential for deadlock occurs if a process controlling a locked resource accesses another resource locked by a different process and is thus put to sleep. Because of this, calls to *lockf()*, *read()*, and *write()* scan for a deadlock before sleeping on a locked resource. An error is returned if sleeping on a locked resource would cause a deadlock. A sleep on a resource can be interrupted with any signal. Thus, the *alarm()* system call may be used to provide a timeout facility if necessary.

Record locking for a simple ISAM database is relatively straightforward: lock the data, and lock the index pointer for the data. But in a B-tree database, record locking is much harder. Modifying a record may require shuffling the leaves of the tree. The safest and easiest thing is to lock the entire B-tree, but this may not be acceptable in highly sophisticated applications.

SYSTEM V CONSIDERED LACKING

System V has no mandatory file or record locking features. System V Release 2 has twice as many—none. To my knowledge, no UNIX system delivered by AT&T has mandatory file or record locking. *[Current VAX and 3B2 releases of System V.2 do have advisory file locking, however. A 3B2 release of V.2 scheduled for later this year will include mandatory record locking.—Editor]*

The most interesting part of the new AT&T standard is that locking can be controlled with the *fcntl()* system call.

The *System V Interface Definition* includes the /usr/group locking standard, for advisory locks only. Mandatory locking may or may not be included in future specifications. The programming usage is the same as in the /usr/group standard, as are the modes F_LOCK, F_TEST, F_TLOCK, and F_ULOCK.

The most interesting part of the new AT&T standard is that locking can be controlled with the *fcntl()* system call. This affords a distinction between read locks and write locks, something not present in the /usr/group standard. The file descriptor passed to *lockf()* must have O_WRONLY or O_RDWR permission in order to establish a lock.

Locks may also be established with the F_SETLK or F_SETLKW command to *fcntl()*; the distinction is that F_SETLKW waits, whereas F_SETLK is non-blocking. Either command takes the arguments F_RDLCK and F_WRLCK to lock, and F_UNLCK to unlock. A read lock (F_RDLCK) prevents other

COSI

T A N G O™

Use Tango to:

- Connect IBM and compatible PC's running DOS to UNIX systems.
- Offload processing to PC's.
- Control data and applications on remote PC's.
- Distribute processing between UNIX and PC's.

Buy Tango for:

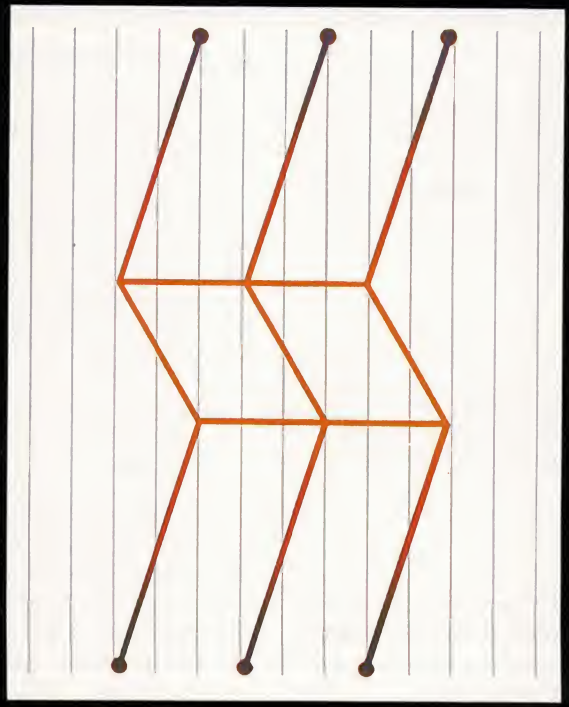
- Execution of DOS programs on the PC under UNIX control.
- Simple elegant file transfer under error correcting protocol.
- DEC, IBM, and Tektronix (graphics) terminal emulation.

Tango utilizes a standard RS-232 serial port on the PC and connects to the UNIX computer via a modem or direct connection.

COSI

313 N. First St.
Ann Arbor, Michigan
48103
(313) 665-8778
Telex: 466568

Tango is a trademark of COSI.
UNIX is a trademark of Bell Laboratories.



The PC-to-UNIX™ Connection

Circle No. 6 on Inquiry Card

processes from write-locking the protected area. More than one read lock may exist for a given region at any given time. The file descriptor in question must have been opened with read access. A write lock (F_WRLCK) prevents any process from read-locking or write-locking the protected area. Only one write lock may exist for a given segment of a file at any one time. The file descriptor in question must have been opened with write access.

In a production database system, it is best to place a read lock on a record during browsing. If the record gets modified, the read lock can be upgraded to a write lock (if no other read locks exist), the record can be quickly updated, and the write lock can be removed. The system must arbitrate race conditions, as when two processes are both waiting for a write lock.

FILE LOCKING ON BERKELEY UNIX

Recent releases of Berkeley UNIX (4.2 and the forthcoming 4.3) contain the *flock()* system call, which allows processes to place advisory locks on files. Since advisory locks are not enforced by the operating system, *flock()* is useful primarily for cooperating processes that have already agreed upon a locking protocol. When a process attempts to lock a file already locked by another process, *flock()* blocks until the first process releases the lock. If called with the LOCK_NB (noblock) option, however, *flock()* will simply return the error EWOULDBLOCK rather than block when it encounters a file that is already locked. Both exclusive and shared locks are available. At any one time, a file may have only one exclusive lock, but multiple shared locks are permitted. This call establishes an exclusive lock, and will block until the lock in effect is released:

```
if (flock(fd, LOCK_EX) < 0)
   perror("fatal error: flock");
```

The following call, on the other hand, establishes a shared lock. It will block if the file has an exclusive lock, but not if there are multiple shared locks:

```
if (flock(fd, LOCK_EX) < 0)
   perror("fatal error: flock");
```

Another option is represented by the following call, which establishes an exclusive, non-blocking lock. It can be used when one can try again later, rather than wait for a lock to be released:

```
if (flock(fd, LOCK_EX|LOCK_NB) < 0)
   perror("try again later: flock");
```

Any of the above locks can be released by this call:

```
(void)flock(fd, LOCK_UN);
```

In all of these examples, the file descriptor *fd* is obtained from a system call such as *open()*. The *flock()* system call returns -1 if the file descriptor is invalid, or if it does not refer to a file.

Some Berkeley UNIX commands that establish locks are **tip** (when writing a log of the call), **dump** (when recording information about incremental dumps), and some versions of **mail** (for locking the spool file during mail browsing).

> John Bass should be commended for his work on file and record locking.

The main problem with the **flock()** facility is that it lacks record locking. Database applications could perform record locking by using a socket-based lock manager, but this would be slow compared to record locking done by the kernel.

FILE AND RECORD LOCKING ON XENIX

Xenix release 2.0 includes the *locking()* system call for performing both file and record locking. This facility is similar to the original 1981 *locking()* proposal by John Bass, except that it provides read locks in addition to read/write locks. As in the original proposal, all locks are mandatory rather than advisory.

The programmer supplies as parameters the file descriptor, the locking request, and the number of bytes to lock:

```
locking(fd, mode, length)
int fd, mode;
long length;
```

The file descriptor is obtained from a successful system call such as *open()*. The number of bytes to lock is specified from the current position in the file, which can of course be changed with the *lseek()* system call. The available modes are LK_LOCK to lock a region, LK_NBLCK to lock a region without blocking, LK_RLCK to read-lock a region, LK_NBRLCK to read-lock a region without blocking, and LK_UNLCK to remove any of the above locks. Both LK_LOCK and LK_RLCK wait until the lock is

The Firebreathers continue on the cutting edge of high performance computers.

The most powerful line of computer systems made. Gould PowerNodes™ and CONCEPT/32s®

Any way you slice it they beat the VAX.™

Our mainframe PN9000 and CONCEPT 32/97 are up to twice as fast as the VAX 8600.

And even though the mid-range PN6000 and CONCEPT 32/67 are 30-50% smaller than the VAX 11/780, they're still up to three times more powerful.

More power for a slice of the price.

Despite their superior power, our mid-range models cost 40% less than the VAX 11/780. Our mainframes cost about 30% less than the new VAX 8600. The bottom line is more power for less money.

Operating environments that are a cut above the rest.

There's also a choice of system software to consider. Gould's unique UTX/32® is the first operating system to combine UNIX* System V with Berkeley BSD 4.2. So it allows you to access virtually any command format you want whenever you want.

And in real-time environments, Gould's MPX/32™ operating system offers performance that's unmatched in the industry, as well.

Delivery that's right on the mark.

Unlike the VAX 8600, that has up to a 12 month wait for delivery, when you order either a Gould PowerNode or a CONCEPT/32 system, they'll be shipped within 90 days ARO.

You can also be sure with Gould you're getting a computer that's backed by years of experience – the kind of experience we used to develop the first 32-bit real-time computer.

If you need more information or just have a few questions, give us a call at 1-800-327-9716.

See for yourself why VAX no longer cuts it. Go with a Gould computer and ax the VAX.

CONCEPT/32 and UTX/32 are registered trademarks and PowerNode and MPX/32 are trademarks of Gould Inc. VAX is a trademark of Digital Equipment Corp. UNIX is a trademark of AT&T Bell Labs.

GOULD
Electronics

Only Gould computers have a big enough edge to ax the VAX.

Circle No. 58 on Inquiry Card

available; LK_NBLCK and LK_NBRLCK return an error instead.

Portions of a file may be locked against both reading and writing, or just against writing. Processes that attempt to read or write a file region locked against reading and writing by another process (using LK_LOCK or LK_NBLCK mode) will sleep until that region has been released. Processes that attempt to write to a file region locked against writing by another process (using LK_RLCK or LK_NBRLCK mode) will sleep until that region has been released.

It was helpful for Microsoft to implement read-locks, but the facility that Xenix provides conforms neither with the /usr/group standard, nor with the System V interface definition. This may change in Xenix 5.0, however.

CONCLUSION

John Bass should be commended for his work on file and record locking. Without him, there would be no standard today. His public-domain locking facility has not only been published as part of the /usr/group standard, but has been included by various far-sighted vendors. Even some people within AT&T have finally seen the light, and have included locking primitives in the *System V Interface Definition*, as well as an intelligent interface with *fcntl()*.

As time goes by, I believe Bass' locking standard will become widely accepted. The 4.2BSD locking facility will die out simply because it does not provide adequate functionality (it has no record locking). Perhaps in a few years, programmers will be able to write database systems with the certain knowledge that file and record locking will be available on all UNIX systems. Until then, we will have to limp along with a standard that is only partially standard.

Bill Tuthill was a leading UNIX and C consultant at UC Berkeley for four years prior to becoming a member of the technical staff at Sun Microsystems. He enjoys a solid reputation in the UNIX community earned as part of the Berkeley team that enhanced Version 7 (4.0, 4.1, and 4.2BSD). ■

UPGRADE YOUR UNIX SYSTEMS

TALK TO YOUR COMPUTER IN PLAIN ENGLISH

The Bell Screen Editor™ uses plain English commands for all your Unix® and MS-DOS® word processing needs. Perfect for programming and Informix®, too. Over 10,000 installed.

NEW! 80 MEGABYTE AT&T UNIX PC 7300® AND AT&T 3B2®

Our B40 System™ and B80 Systems™ upgrade your Unix PC or 3B2 to 40 or 80 Megabyte hard disk capacity. Easy to install in minutes.

UNIFY® USERS REJOICE

The DataView System™ provides a spread-sheet like interface to any UNIFY application. Multiple records, sorted data, and much more.

Bell Technologies
415-792-3646 / PO Box 8323
Fremont, California 94537

Call today for our special demo offer!

Circle No. 30 on Inquiry Card

TOWER™ POWER

Give your Tower more Power!

up to 1.2 gigabytes of disk and tape storage:

- removable disk
- fixed media disk
- disk and tape

SHA Computers, Inc.
RD#3, Tait Rd., Box 51
Saratoga Springs, NY
12866
(518) 587-5886

Personal Secretary™ word processor
***profit•maker*™** integrated accounting system
DIBOLIX™ DIBOL™ for UNIX™

Trademarks: Tower, NCR Corporation, DIBOLIX, C DIBOL Ventures, UNIX, AT&T, DIBOL, Digital Equipment Corporation, Personal Secretary, Finished Software, profit-maker, SHA Computers, Inc.

Circle No. 29 on Inquiry Card

THE UNIX GLOSSARY

Database terminology

by Steve Rosenthal

Note: Only those meanings applicable to databases and UNIX have been included in this listing.

activity ratio—the proportion of a file that has been read, updated, or written during a given period. The optimal method of storing a large amount of data can vary in accordance with the expected activity ratio.

add—to introduce a new record into a database. In some systems, additions are classified as *appends* (at the end of the file) or as *inserts* (within the file). The addition of data to a field in an existing record is known as an *update*.

ad hoc—said of queries to a database management system (DBMS) made directly by a user rather than as part of a program. Most of the more sophisticated DBMS packages for UNIX support an ad hoc query facility that typically uses a format resembling IBM's SQL (Structured Query Language). This feature allows non-programmers to get quick answers to simple questions.

aggregate—to derive a value, such as a count or sum, based on the contents of many records.

applications development system—a software package intended to help with the development of

applications programs, generally by using descriptions of needed inputs, outputs, and their relations. This saves the user from having to specify each step, as would be necessary with procedural languages. Some applications development systems for UNIX function as complete user shells, providing a user interface that substitutes for the normal UNIX environment. Compared to program generators, applications development systems are generally more sophisticated, but usually produce code that requires a more extensive runtime support package. Applications development systems are often called "application generators".

atomic—an indivisible operation, that must either be run to completion or aborted altogether. Transaction updating is a typical atomic operation.

attached processor—a supplementary processing unit used to speed up the processing of specialized types of data. For example, arithmetic calculations or database searches can be facilitated by the use of an attached processor. When the processor is a chip or a board, the term *co-processor* is commonly used.

attribute—an item of information entered into a single field (or a "single cell" in a row-and-column database). In relational databases, columns are often referred to as "attributes".

audit trail—the recording of each update, addition, or deletion of records such that a database can be reconstructed later by referring to logs.

browser—a program or mode allowing the user to look through a database on screen without using a set of procedural display commands. Most browsers also allow users to move through data using simple keystrokes to indicate direction. Many will let users change data by overwriting old data.

B-tree—shortened term for "balanced tree", a way of organizing

pointers to information in databases that allows quick retrieval of any single specified record. So-called "B+" or "B*" trees allow records to be efficiently retrieved in sequential order. Many databases designed for UNIX use B+ trees for their indices, and some arrange their records using this kind of structure in preference to the UNIX file system.

cardinality—in formal descriptions of databases, "cardinality" refers to the number of tuples in a table or set. Translated into everyday language, this means the number of records in a database or the number of rows in a table.

CODASYL—an acronym (pronounced Code-a-sill) for Conference on Data Systems and Languages, a computer industry organization set up by the US Department of Defense in the late 1950s. The most famous product of the group is the language COBOL, but it also developed a set of standards for database structures that have been used with other languages as well.

column—in databases that use a table-like organization, a "column" is a record component that contains similar information in each record (and thus is represented in a single column when the data is displayed as a table). In more traditional database terminology, "field" is an analogous term.

concurrency control—in a distributed system, "concurrency control" is used to ensure that simultaneously input events do not interfere with each other or lead to the processing of incomplete records and files. The usual method for exercising this control is for the system to finish the processing of one transaction before allowing another user to access the same record or file.

database—in the most general sense, "database" refers to any clearly identified collection of data. Some people differentiate between a "data base" (two words), meaning an underlying collection of data in the real world, and a "database" (single word), meaning a coherent collection of data stored in a computer system. When taken as the latter, the word makes particular reference to data organized so that various programs can access and update it.

database management systems (DBMS)—a program or set of programs providing a framework for creating, editing, and maintaining collections of data for use by different programs. DBMS systems serve as an interface between programs and data and may also include a query facility for making individual (ad hoc) requests for information from the database and a reporting facility for producing formatted listings of selected data.

data dictionary—the "data dictionary", in most complex databases, is a list of defined fields and record formats. It is used as a guide or constraint to ensure that all programs using the database treat the data consistently.

decompose—to change a request couched in non-procedural language to a required procedural form. This is one of the principal tasks of an ad hoc query system.

delete—to logically remove information (usually a record) from a database. Often, deleted data is marked but not physically removed until the file is copied or consolidated.

field—in data entry, a "field" is an area in which a certain type of information is to be placed. For example, a database program might reserve a 10-character field for area code and telephone number. Conceptually, fields are similar to the blanks to be filled in on a pre-printed form. When shown on a screen, fields are generally marked by flags indicating a beginning and an end, by temporary fill-in characters (such as periods), or by differences in color or brightness.

file—a group of records treated as an overall unit by the operating system. In some types of databases, each file is made up of structurally identical records, while other types allow variations in a single file.

get—to retrieve records from a database, or at least to mark them for further processing. A *get* operation is often followed by a definition of the desired group of information.

hash—to make a pointer or index by applying a transformation to the characters or values comprising a key or record. Hashing provides a very fast way of indexing large lists or databases, but it requires complex programming to deal with collisions (when the hash function produces identical results for different input values) and to fold long keys into short hash values. Many UNIX utilities use hash functions to create pointers to their internal tables.

hierarchical—a model for organizing data that uses "ownership" as its basic conceptual unit. Each item "belongs" to a higher item, and is accessed through that higher item. This model is used on most CODASYL-type databases, including most of those written in the COBOL language. The UNIX file system can also be thought of as a hierarchical database.

indexed sequential access method—a method of organizing files that is most popular on older magnetic tape-based systems. It

depends on keeping the file in overall order, but creates overflow areas and indices for changes that don't fit in place. Periodically, the entire file must be reorganized. Some of the older UNIX utilities use ISAM, but most use a direct access method better suited to disk storage.

insert—to add records to a file, particularly in a mode that allows additions to the middle of a file (as opposed to an *append* operation, which only allows additions at the end of a file).

ISAM—an acronym (pronounced "eye-sam") for Indexed Sequential Access Method. This arrangement of data was pioneered by IBM when magnetic tape was the principal means of storing data, but it's only used rarely now by small systems that have disks. In effect, ISAM keeps data in a basic order, making an exception list of out-of-order items as data is added, deleted, or changed. Periodically, the file must be cleaned and rewritten to eliminate the exception list.

join—in general, to combine two databases (or when used as a noun, "join" refers to the result of that combination). In particular, as applied to relational databases, "join" means the creation of a new file containing all the records of a second file referred to by yet another file. One example would be a shipping list file produced by joining an orders file with an inventory file.

key—the part of a record that will be used as a identifier when records are indexed or sorted. For example, in the telephone book, the key is each subscriber's name. Some databases allow duplicate keys (where the key is the same for two or more records), but other systems require that each key be unique.

log—a list of transaction records entered since a certain checkpoint. Good practice calls for each transaction to be logged before requesting an update or operation. With a log, data can be reconstructed following a system failure by using records dating from an appropriate checkpoint to resubmit all transactions.

modify—to change the content

New from **Image Network**!

Documenter's Workbench®

for laserprinters and typesetters.

DWB is *troff, eqn, tbl,* and *pic* interfaced to raster printing devices.

Our existing **XROFF** product allows **DWB** to work with the following systems and printers:

- *System V*
- *Berkeley 4.2*
- *VAX/Ultrix*
- *IBM/PC MS/DOS*
- *Eunice*
- *UniPlus+*
- *System III*
- *V 7*
- *VAX/VMS*
- *Amdahl/UTS*
- *Xenix*
- *UNOS*
- *DEC LN01s, LN03*
- *APS–5 typesetter*
- *Compugraphic 8400*
- *Xerox 2700, 3700*
- *Xerox 8700, 9700*

Use **DWB** with a laser printer to make high quality documents or to make proof copies before typesetting.

Call or write to tell us *your* printing requirements!

Image Network, (408)746-3754,
424 Palmetto Drive, Sunnyvale, CA, 94086-6760

*Documenter's Workbench is a trademark of AT&T Bell Laboratories.

This ad was typeset using **DWB**.

Circle No. 33 on Inquiry Card

Q-CALC

A superior spreadsheet on UNIX*

As powerful as Lotus 1-2-3*

- large spreadsheet
- many business functions
- complete GRAPHICS package
- translates 1-2-3 models into Q-CALC
- already ported to: VAX, Callan, Fortune, Nixdorf, Cyb, Plexus, Codata, Cadmus, Masscomp, SUN, etc.

Available since Jan. '84
For more information write/call
Quality Software Products
348 S. Clark Drive
Beverly Hills, CA 90211
213-659-1560

*Lotus 1-2-3 is a trademark of Lotus Development Corp. UNIX is a trademark of Bell Labs.

Circle No. 34 on Inquiry Card

of a database or the structure of a record or file. All databases for UNIX support modification of database content, but only some will allow record structures to be modified once data has been entered.

non-procedural—said of systems and languages where users specify *what* they want done instead of *how* to carry it out. For example, in a database, a non-procedural query might ask for all records with values between certain limits, while the equivalent procedural statements would actually specify how to sort and select the database to find those records.

project—to make a selection of possible fields from database records for further processing. The fields may be selected from all records in the database or from a more restricted set.

QBE—short for "Query By Example", a non-procedural method of specifying record selection. See *query by example* for details.

query by example—to find or select records in a database by specifying acceptable ranges of values in a sample record instead of programming the steps needed to make that selection. This feature, which allows database queries to be made by those who are not expert programmers, has been steadily moving down from large mainframe systems to even modest-sized UNIX database management systems.

query language—a computer language that acts as an interface between user and database to facilitate the retrieval of information. Most query languages are procedural, based on a set of commands (often called verbs) and qualifiers. The trend, however, is toward "natural language" interfaces that allow queries posed in forms more akin to ordinary speech.

record—a collection of entries in a database filed under a single key or identifier. Files are composed of records, which in turn are composed of fields. Each record usually represents a single instance of the type of information collected in the database, such as all information related to a single part number or the wages of a single employee. In the relational database model, a record is equivalent to a single row in a table. UNIX itself treats files as streams of characters, so the database program must handle the task of grouping information into logical records.

record-locking—the exclusion of users from accessing (or, sometimes, just writing to) a record that another user is already updating. This prevents the corruption of data that might occur if each user could make changes without taking stock of changes made to the same data by other users. Record-locking affects only those records in use, allowing other users to access other parts of the file. AT&T has added record-locking to System V Release 2, but previous versions of UNIX lacked a standard record-locking call.

relation—a table (row-and-column structure) with attributes forming the columns, and tuples

FRANZ
THE FIRST NAME IN
LISP

Franz LISP from Franz Inc. is currently available under UNIX and VMS. Now with Flavors and Common LISP compatibility. Franz sets the standard for LISP.

Franz Inc.
1141 Harbor Bay Parkway
Alameda, California 94501
(415) 769-5656

UNIX is a trademark of Bell Labs. VMS is a trademark of Digital Equipment Corporation.

Circle No. 31 on Inquiry Card

UNIX*
JOBS
REGISTRY

National registry of candidates and jobs in the Unix field. Please give us a call; send a resume; or request a free Resume Workbook & Career Planner. We are a professional employment firm managed by graduate engineers.

800-231-5920
P. O. Box 19949, Dept. UR
Houston, TX 77224
713-496-6100

Scientific Placement, Inc.

*Unix is a trademark of Bell Labs

Circle No. 32 on Inquiry Card

(records) forming the rows. Each relation represents a linking of data values with the attributes or fields in which they fall. Note that usage is moving towards calling a combination of two tables a "relation". In formal terminology, this would be known as a "join".

relational—in the strict sense, "relational" refers to databases that are conceptually organized in a row-and-column format, with the data defined as the relation of a part of a record (the row) to a category or field (the column). In recent use, however, the term has been taken to refer to a database that can join or display two or more files based on a shared field or category.

remove—to take values or records out of a database. Often, "removal" implies actual physical erasure or reorganization, while "deletion" may signify that material was only logically eliminated or marked for later removal.

replace—to change values in a database record or field to new ones. Often, *replace* implies the complete change of an entire record or field, while a *modify* or *update* operation may change only a portion.

report—when referring to databases and their use, "report" refers to a collection of data gathered, formatted, and output according to user request. Getting a formatted, focused report out of a database is often much more difficult than collecting the data in the first place.

report writer—a program or subsystem that produces reports from a database. Most early report writers required users to specify selection and formatting steps in extensive detail, but the trend in many recent packages has been to offer a menu-driven interface.

restrict—to select certain records from a database, either for display or further processing.

restructure—to change the organization of a database, particularly the layout or makeup of fields in each record. Only some databases allow restructuring once data has been entered.

retrieve—to get records from a database. Some people use "retrieve" as a synonym for the general operation of reading records back from storage, while for others it implies a selection of certain records.

row—in the relational database model, the values that print out on a horizontal line when a file (relation) is shown in table form. This includes all the values in the table associated with a single key, and is also termed a "tuple" or "record".

select—in a sort or transfer operation, "select" refers to efforts to isolate all the records meeting a specified criteria. A select always implies restriction of the set, and sometimes it also implies retrieval.

SQL—short for Structured Query Language, an approach to extracting information from databases pioneered by IBM.

table—a file or relation logically organized in row-and-column form. Each record (row or tuple) is identical in format, which makes for easier processing and joining with other tables.

transaction—an update, addition or deletion of a record in a database. Transactions are usually treated as atomic (they either must be completed or backed out completely). Systems designed for transactions are optimized to handle many changes.

tuple—the more formal name for a row in a relation (table). It is a shortened form of "n-tuple" (taken from the realm of mathematics) that is used to refer to multiple attributes (fields or columns) associated with each row.

update—to change information in a database without resorting to removing records, making new ones, or creating copies. Updates can be done in real-time (transaction-based), or in batches (batch mode).

Comments, questions, or corrections? Please send them to Rosenthal's UNIX Glossary, Box 9291, Berkeley, CA 94709.

Steve Rosenthal is a lexicographer and writer whose work appears regularly in six personal computer magazines. ■

ARTISAN 2221 Blacksmith Dr., Wheaton, IL (312) 260-1315

SEMINARS AVAILABLE:

UNIX™ Overview & Fundamentals	$400
C Programming	$500
Text Formatting	$300
Shell Programming	$300
Software Development in UNIX	$500
LAN, Multiplan™, dBASE III™	$100 ea.

Public Seminars available in Chicagoland or your area (5 persons minimum).
Customized Seminars also available.

*UNIX is a trademark of Bell Laboratories
Multiplan is a trademark of Microsoft Corp.
dBASE III is a trademark of Ashton-Tate

Circle No. 45 on Inquiry Card

RECENT RELEASES

PLEXUS FILLS OUT LINE WITH P/20

In an effort to offer a low-end microcomputer to system integrators and customers with several end users, Plexus Computers, Inc., has introduced the P/20. Part of the P/15 family, the P/20 is said to offer more networking options than the P/15, and yet offer a lower price and comparable performance to the P/35. This new model will compete with such machines as the SCR Minitaur and AT&T 3B2.

The base package for the P/20 features a 24 MB hard disk, a 1 MB floppy, .5 MB of memory, and accommodates eight users. This package operates under a Plexus port of UNIX System V, Release 2; the basic offering contains the utilities needed to get the system going: an editor, **uucp, cu,** and tape and backup facilities, according to Lynn Macey, Plexus' National Analyst Manager. He also said the P/20 is fully licensed for a complete port if a VAR wishes to install it. The machine incorporates dual MC68010 microprocessors that run at 10 MHz with no wait states, and SCSI and Multibus interfaces. The Multibus permits the addition of a controller board to provide up to 16 serial ports. The base package is priced at $10,950. A fully outfitted P/20, with 152 MB of hard disk and 2 MB of floppy, carries a price of $20,300.

Plexus Computers, Inc., 3833 N. First St., San Jose, CA 95134, 408/943-2248.

Circle No. 36 on Inquiry Card

Plexus Computers adds to its model line with the P/20.

ALTOS: DO I HEAR 20? DO I HEAR 30?

For those who don't live in a Spanish-influenced region of the country, "Altos" translated means "tall", or "the tall ones". The San Jose-based computer firm bearing the name may have had this in mind when plotting its new marketing strategy, for it appears to be setting its sights high. Altos is intensifying its marketing efforts around two new products, the 2086 and much-anticipated 3068.

The Altos 2086 supermicro is a high-end addition to the company's Intel CPU-based product family. Rather than compete with the 586 and 986, which support up to five and nine users respectively, the 2086 is sold by Altos as a complement or upgrade to these machines, as it supports up to 20 users (hence the model name). The price reflects this upgrade ($19,990 for the base configuration, compared to $7990 for the 586 and $11,990 for the 986), but the 2086's specs are worthy of consideration: based on a 16/32-bit Intel 80286 running at 8 MHz, the box comes with 2 MB of RAM, an 80 MB hard disk, a 1.2 MB floppy, a 60 MB streaming tape unit, and an Altos III terminal. Hard disks can be upgraded to 189 MB (formatted) in 63 MB increments. The 2086 runs Xenix 3.0.

The Altos 3068 has the same modular design characteristics and many of the same components as the 2086, but it is a notably distinct machine—for two reasons. First, the 2086 was designed as a dealer product, to be marketed through Altos distributors, though Altos hopes for interest from major accounts as well; the 3068, on the other hand, is designed as an OEM product.

The second distinction is especially noteworthy: the 3068 is a mass-produced supermicro based on the 32-bit MC68020 microprocessor. Announced last March and available as of last month, the 3068 comes with a base package featuring 1 MB of RAM, a 20 MB hard disk, and a 1.2 MB floppy supporting up to 10 users. When properly outfitted, however, this machine will support up to 30 users. The design includes eight board slots with four available for custom configuration. RAM can be expanded to 16 MB in 1, 2, or 4 MB increments; hard disks can be upgraded to 240 MB

(unformatted) in 20, 60, or 80 MB increments.

Other available features include a streaming magnetic tape unit, with up to 60 MB of backup storage; an operating system (the 3068 runs System V) supporting demand-paged virtual memory with 1 K page size; and various software products, including the Altos Office Manager (which includes a windowing package) and a database management package based on Unify.

Altos emphasizes that while the 3068 capably functions as a standalone system, it can be joined with other Altos multiuser systems via the Altos WorkNet local-area network, and with mainframes via 3270 Bisynch, SNA, X.25, and 3780 communications options. With Altos PC Path attached to WorkNet, the 3068 can also act as a file server and communications gateway for personal computer users.

Philon, Inc., of New York City, has been selected as one of the suppliers of compilers for the 3068. BASIC-C, BASIC-M, COBOL, and C compilers are presently available, and Fortran, Pascal, and RPG compilers will be ready later this year.

The base configuration price for the Altos 3068 is approximately $7000 in OEM quantities.

Altos Computer Systems, 2641 Orchard Pkwy., San Jose, CA 95134, 408/946-6700.

Circle No. 35 on Inquiry Card

SWEET TALK THROUGH THE (RCA) MAIL

SofTest, Inc., has signed a contract with RCA Service Corporation for SofTest's new communication product, Sweet Talk. RCA will be using Sweet Talk to access the RCA Mail network and provide access to RCA Mail for its customers.

Sweet Talk is a UNIX-based product that can link the user with on-line database services and remote computers, as well as electronic bulletin boards and mailing services. It claims to bring to UNIX all of the features of various popular MS-DOS products such as CROSSTALK and SmartCom, as well as some other benefits not available to PC users; Sweet Talk is also compatible with these two products. Since it is UNIX-based, Sweet Talk can be shared and used concurrently by several people on the same computer. It is now available on Altos computers and Radio Shack 16Bs

ACUITY® business software is compatible with any budget, and all these systems:

AT&T 3B's	Plexus	Gould
Motorola	Convergent	Sperry
Charles River Data	Cromemco	Momentum
Sun Microsystems	Altos	Dual
All Unix based micros	Harris/VOS	Harris/Unix
All Unix "look-alikes"	VAX/Ultrix	VAX/VMS

Serving general accounting, wholesale, distribution, manufacturing and project/job costing applications on over 30 different machines, ACUITY allows you to select from individual modules to build a fully integrated software system specifically for your needs.

Accounts Payable • Accounts Receivable
General Ledger • Fixed Assets • Payroll
Customer Order Processing • Inventory
Purchasing/Receiving • Project Management
MRP • Master Scheduling • BOMP
Project Scheduling • Labor Projections
Work Breakdown Structure

For more detailed information, call 619/474-6745.

COMPUTER COGNITION

225 West 30th Street, National City, California 92050

Circle No. 37 on Inquiry Card

dyalog APL™

Version 3.0 Available Now!

The Reliable High Performance APL for UNIX* Systems

Dyalog APL is fast!
Version 3.0 is up to 10 times faster than previous versions!

Dyalog APL is functional!
Nested Arrays
Full Screen Editor
Full Screen Data Manager
Event Trapping
Interface to all UNIX* Facilities
Optional Graphics

Dylalog APL is reliable!
Dyalog APL has been in commercial use for over two years and is available NOW for most UNIX* Systems so call or write today.

MIPS Software Development, INC.
31555 W. 14 Mile Rd. #104
Farmington Hills, MI 48018
(313) 855-3552

† Improvements are a function of system and usage
* UNIX is a trademark of AT&T Bell Laboratories

Circle No. 38 on Inquiry Card

and 6000s, and SofTest plans to port Sweet Talk to the AT&T and IBM lines of UNIX machines, among others. Single copies of Sweet Talk sell for $300.

SofTest, Inc., 555 Goffle Rd., Ridgewood, NJ 07450, 201/447-3901.

Circle No. 39 on Inquiry Card

ANALOG RUNNING IN APOLLO'S DOMAIN

Analog Design Tools, Inc., has announced it will develop a version of its analog circuit design software to run on all of Apollo Computer's DOMAIN 32-bit family of engineering workstations. Originally designed for the Sun workstation, Analog's computer-aided engineering software also has been produced in a recently-released version for Daisy Systems, as well as others to be announced later.

Under the terms of a Software Supplier Agreement, Analog will provide Apollo with training in the operation of its Analog Workbench software, along with support and assistance for demonstrating the software on Apollo workstations. Apollo in turn will provide Analog with technical support and will inform Apollo customers of the availability of Analog's software for its systems.

Analog Design Tools, Inc., 800 Menlo Ave., Suite 200, Menlo Park, CA 94025, 415/328-0780.

Circle No. 40 on Inquiry Card

FORTRAN LIBRARY FOR HP 9000

As evidence of the continuing penetration of UNIX into the scientific/engineering market, witness the now-available collection of over 500 Fortran subroutines that can be run on Hewlett-Packard Series 9000 computers. The IMSL Library contains tested programs for a range of mathematical and statistical applications that can be selected by a programmer rather than developed from scratch. The Library has been available for some time on HP 1000 and 3000 Series machines, and is now compatible with the 9000 Series (Models 520, 530, and 540) running a Fortran 77 compiler under HP's version of UNIX, HP-UX.

An annual supported license is priced at $1200 for the initial year, and is renewable at $1000. IMSL also offers a reduced price to educational institutions and discounts on multiple purchases.

IMSL Sales Division, The NBC Building, 7500 Bellaire Blvd., Houston, TX 77036, 1/800/222-IMSL; in Texas, 713/772-1927.

Circle No. 41 on Inquiry Card

ALIS: THROUGH THE UTEK-GLASS

A contractual agreement has been reached between Applix, Inc., and Tektronix, Inc., whereby Tektronix will market Alis, the Applix office software system. Alis will be run under UTEK, the Tektronix version of UNIX, on the Tektronix 6000 family of 32-bit workstations. Applix will port Alis to the Tektronix 6130 processor and provide support for the 6000 family and 4107 terminals.

UTEK is a 4.2BSD-based port of UNIX, combining 4.2 with aspects of System V, Release 2. It also offers Tektronix enhancements, according to Tektronix Product Support Manager Bruce Harris, including a virtual memory system faster than 4.2's, and an added distributed file system. The operating system is supportable on workstations and comes with packaged learning sessions. Though it is "large UNIX", Tektronix claims the advantage of compatibility with both 4.2 and V.2.

Applications of the Alis system include multifont word processing, drawing, spreadsheet, business graphics, electronic mail, and network-based information sharing. The package works at combining graphics-based and integrated PC applications with communications-based office automation systems. Applix has previously announced agreements with Convergent Technologies and Computer Sciences Corporation.

Applix, Inc., 112 Turnpike Rd., Westboro, MA 01581, 617/870-0300.

Circle No. 42 on Inquiry Card

IN-HOUSE PUBLISHING FROM ETP

ETP Systems, Inc., has produced an in-house publishing system providing laser printer typography for UNIX computer users. A turnkey system that includes an Imagen 300 dot/inch, 8 page/minute (12 and 24 page/minute also available), Canon-powered laser printer, is distinguished from other publishing systems by ETP-developed "usr/tools" software. This software package contains an enhanced device independent **troff** (with a focus on menu-driven processing of **troff** commands), several macro formatting packages, a laser printer driver, a choice of up to 33 fonts, and a font magnification program (with a capacity for sizing typefaces from 6 to 72 points).

ETP Systems markets its publishing package through computer manufacturers and OEMs, and provides a product support package that includes descriptive data sheets, sample output, and a manual set. The in-house system, with three fonts, is approximately $13,000; additional fonts are available at $175 per face.

ETP Systems, 10150 SW Nimbus Ave., Suite E-2, Portland, OR 97223, 503/639-4024.

Circle No. 43 on Inquiry Card

Only Sperry can make the following four statements.

Our PC runs the XENIX™ system, as well as MS-DOS™.

Our 4 new microcomputers run the UNIX system.

Our new minicomputer runs the UNIX system.

Our Series 1100 mainframes run the UNIX system.

All of which means there is a great deal we can do for you.

For instance, our family of computers based on UNIX systems has incredible transportability for all your software.

And being able to accommodate from two to hundreds of users, it's impossible to outgrow our hardware.

Of course, this linking of all your computer systems can add measurably to your productivity.

And a fast way to find out more is to get a copy of our Sperry Information kit. For yours, or to arrange a demonstration at one of our Productivity Centers, call **1-800-547-8362.**

*UNIX is a trademark of AT&T Bell Laboratories
XENIX and MS-DOS are trademarks of Microsoft Corporation
©Sperry Corporation 1985.

SPERRY

Introducing an idea that makes obsolescence obsolete.

The UNIX* operating system from PC to mainframe.

Circle No. 8 on Inquiry Card

DATABASE OVERVIEW

Continued from Page 33

or exists simultaneously in two accounts. Transactions can hide this anomaly from other users, and can ensure that if the system crashes at this point, the database will be restored to a consistent state. This requires a more complex *commit* operation, and deadlocks become even more common and more difficult to resolve (since *many* objects—not just one—may have been updated).

Audit Trails. It has been said that lawyers and bookkeepers will inherit the earth. To assuage our future owners, databases containing information of legal significance should include the ability to maintain audit trails—logs of all changes (and possibly accesses) made to a database.

If care is taken in designing the log, audit trails (sometimes also called *transaction logs*) can be used to "roll back" a database to a previous state, that is, to undo changes that have been made. This can be necessary for recovery from system crashes or deadlocks.

Backup/Recovery. Databases should be dumped periodically. A database dump is equivalent to a "level zero" dump on UNIX. In addition, a dump of the transaction log (see above) can be considered equivalent to an "incremental dump" on UNIX; a log can be used to *roll forward* a database and redo the incremental changes made since the last database dump.

An important issue is whether the database can be backed up while it is live (available for users), or if it must be in a quiescent state. This is important because some applications simply cannot afford to be offline for the several hours it can take to back up a large database.

UNIX-style dumps are not normally acceptable. For example, if one record in a 10 million-record file changes, a UNIX incremental dump will save the entire file, while a clever database backup will save only the changed record.

Systems supporting very large databases should allow dumps of subsections of the database.

Protection. Often it is necessary to restrict access to data. For example, "managers can read the salaries of the people who work for them; the personnel department can read all salaries; all other access to salaries is denied". This can be relatively low resolution (perhaps access could be limited on a per-file basis—as on UNIX) or very high resolution (whereby individual fields and/or individual records are protected).

Data Dictionary. If you have a particularly complex database structure, you may want to have a data dictionary available. This feature allows you to ask for information about the data itself. For example, you might need to know the attributes from different relations that can be correlated against one another.

Integrity Constraints. Some systems give you the ability to place additional constraints on the data. For example, one could specify that "salaries must be positive" or "every employee must be in a department".

Non-Traditional Data Types. Typically, database systems have concentrated on fairly ordinary data, such as integers, character strings, and the like. As the use of databases expands, they are being extended to handle new types, such as text, graphics, and "experts" (a time expert, for example, would understand "yesterday", "three weeks from last Tuesday", and other time and date-oriented constructs).

As superminicomputers have steadily become more available, the availability of comparably sized database management systems also has increased. These systems are often *relational*, which is to say they operate on data in simple tables rather than in less flexible data structures, such as hierarchies or networks.

All database management systems offer certain basic operations: retrieval, insertion, deletion, and modification. Larger systems may also provide aggregation and the ability to correlate tables against one another.

User interfaces represent the most obvious differences between database management systems today. Many kinds of interfaces are available, ranging from complex programmer interfaces to simple facilities that can be used by relative novices.

Internally, database management systems may have many important features. Since these are usually less obvious than the differences between interfaces, special care should be taken to evaluate them carefully. Every feature has a cost, but failure to include an important feature can make the difference between a pleasant interaction with data and a sentence to live out a technological nightmare.

Eric Allman has spent many years working on almost all aspects of database management systems. Currently, he works at Britton Lee developing interfaces to database machines. He previously spent several years at UC Berkeley, working on the development of the INGRES system. Between database responsibilities he has worked on a variety of other projects, including text preparation, electronic mail, and computer games. ■

Material presented in this article was first presented at the Spring 1984 European UNIX Users' Group Conference. UNIX REVIEW expresses its gratitude to the EUUG.

SETTING THE STANDARD FOR TOMORROW

RUBIX™ is a high performance database management system (DBMS) for the entire range of computers—from single user microcomputers to large mainframes. It offers the ideal solution to the micro-mainframe compatibility issue. RUBIX is a true relational DBMS which is a fully integrated part of UNIX, not an afterthought. The friendly English-like user language and relational programming language (Q) allow complex applications to be developed in a fraction of the time associated with conventional development tools. And where desired, the developer can turn to the most complete Host Language DBMS Interface now available. Extensions to the relational model, including dated relations and updated views, make RUBIX the most powerful DBMS available today. RUBIX is the only DBMS under UNIX which supports simultaneous access to multiple databases and view definitions across databases. And, on network computers, the databases can even reside on different machines.

Data manipulation may be performed by:

- The RUBIX interactive relation editors
- Shell scripts containing interactive RUBIX and UNIX commands
- Queries processed interpretively by the Q interpreter
- Compiled queries linked with C functions
- C programs invoking RUBIX macros and functions

PREFIX™ was designed to provide nonprogrammers a convenient and easy-to-use interface to RUBIX. Through PREFIX, even nonprogrammers can generate applications software for online transaction processing environments.

PREFIX features include:

- Interactive full-screen forms generation facility
- User-friendly menu interface
- Online context sensitive help
- Comprehensive data entry, editing and validation
- Automatic retrieval from other relations and databases
- Multiple search facilities
- C language interface

RUBIX/PREFIX offers an efficient method for creating applications tailored to specific needs. The resulting systems are quickly learned and reliably operated by clerical personnel with little or no computer background.

When productivity is the key, the choice is RUBIX.

INFOSYSTEMS TECHNOLOGY, INC.
6301 Ivy Lane / Greenbelt, MD 20770 / (301) 345-7800

Circle No. 5 on Inquiry Card

Use the Power of Your Computer

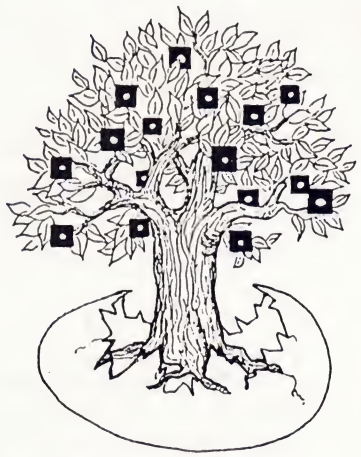
. . . to automatically look up city, state and county information based on zip code. Table of 48,000 zips allows significant savings on data entry, error corrections and file maintenance. This set of floppy disks, including easy instructions, is just $149. Most popular 5¼" and 8" formats are available. Hard disk required. Call or write for free information.

DCC Data Service
1990 M Street, N.W., Suite 610
Washington, D.C. 20036
202-452-1419

Circle No. 27 on Inquiry Card

Tree Shell

A Graphic Visual Shell for Unix/ Xenix End-Users and Experts Alike!



Dealer inquiries welcomed.

COGITATE

"A Higher Form of Software"
24000 Telegraph Road
Southfield, MI 48034
(313) 352-2345
TELEX: 386581 COGITATE USA

Circle No. 26 on Inquiry Card

MAKING A MATCH

Continued from Page 41

piping data between front-end and backend may argue against the two-process architecture on small machines. Computers that accommodate many users, though, can benefit in several ways from the front-end/backend approach.

First of all, UNIX runs all its programs as re-entrant code, which is to say that even when two or more users are running the same program, the code segment is still only loaded once. However, each user gets a private data segment. Thus, when all users on a system run the same query language, the code only occupies physical memory once. However, if each user runs different database programs, code segments for each program will have to be loaded. Chances are, though, that 80 percent of the code is redundant across all the segments since each contains the same DBMS retrieval code. Thus, if a single backend process were handling all calls from user processes, the code for these different programs could be shared to a large extent.

A problem with the two-process architecture, though, is that it makes it impossible for a process to talk to several other processes. Strictly speaking, each of the backends looks like a separate process to UNIX; they all simply see to it that the system only loads code once. Several different data segments still exist for a backend process, which really only wants one—meaning that buffered data needs to be stored redundantly. This is bad both because it is inefficient and because it can cause "concurrency control" problems of its own.

The secret to turning this problem into an advantage is the use of shared memory (standardized in System V). Shared memory allows a backend process to look like several processes to the UNIX process table, but to still "share" data areas with other invocations of itself. In essence, the backend process becomes one beast with several concurrent pipes to requesting processes. This is the architecture required for DBMS efficiency. The more users there are, the more important this architecture becomes. Pipe overhead thus is more than paid for.

CONCLUSION

Throughout the history of DBMS products on UNIX, the advent of standards has always helped. This has been particularly true in the case of locking and shared memory. Other interesting standardization efforts are also going to have a major impact on UNIX. For example, European UNIX suppliers have developed a group called X/OPEN to standardize on a subset of the *System V Interface Definition* and make a number of "commercial extensions", such as C-ISAM indexed file manipulation calls.

It is interesting that the commercial extensions have focused on typical data processing needs. This trend places UNIX and DBMS directly at the heart of commercial data processing. UNIX clearly has gone through a tough evolution, but it has adapted well and finally come of age.

Roger J. Sippl is President of Relational Database Systems, Inc. (soon to move from Palo Alto to Menlo Park, CA). As such, he has helped pioneer the UNIX DBMS market with such products as Informix and the recently released Informix-SQL, which includes embedded SQL interfaces for C and COBOL. Mr. Sippl holds a degree in Computer Science from UC Berkeley, and is a founder and former board member of /usr/group. ■

CLEO is your SNA or BSC Gateway

Connect your IBM, Apple, Tandy, Zenith, A.T.&T., Hewlett-Packard, Televideo, NCR, IMS, SUN, or other DOS or UNIX-based system to another micro or to your mainframe with CLEO Software.

Now you can connect your PC LAN, too!

For details call: 1(800) 233-CLEO
In Illinois 1(815) 397-8110

CLEO™

CLEO Software
a division of Phone 1, Inc.
1639 North Alpine Road
Rockford, IL 61107
TELEX 703639

Circle No. 55 on Inquiry Card

CLEO and 3780Plus are registered trademarks of CLEO Software.
IBM is a registered trademark of International Business Machines Corporation; Apple is a registered trademark of Apple Computer; UNIX is a registered trademark of A.T.&T. Technologies, Inc.

WEINBERGER

Continued from Page 49

WEINBERGER: Either in the kernel or with interprocess communication, depending on how fast you want to go. There's more to the United Airlines system than that. The United Airlines system has, oh, 10,000 terminals or so. A lot of them are what are called *multidrop* terminals. They're all connected by way of Bisync, or HDLC, or something, and they look a bit like 3270s.

Even if we had a database management system we were happy with, one of the things I don't have is the knowledge needed to deal with these communication networks. That's a really important part of a database system. It's partly a reliability thing. You don't know that your transaction is complete until you get a message saying so. That means you have to log the messages. You have to control everything. There is no point in doing something or, more precisely, failing to do something if it is not recorded. If you enter something, but the line goes away, you may deserve to know whether or not that transaction was completed.

REVIEW: *It may even be worth money.*

WEINBERGER: That's conceivable.

REVIEW: *Do you think that's a communications issue?*

WEINBERGER: No, in principle it has to be handled wherever the database is. I have two separate objects that have to be backed up—to be logged. I have the database itself, and I have communications messages. To get both halves of that consistent, there's got to be some controlling entity that handles both. So the logical control for both has to be in one place. It doesn't have to be centralized in any sense, but logically, there can be only one coordinating entity that handles the sending of messages and the assigning of tasks.

REVIEW: *That sounds hard to do with an add-on package.*

WEINBERGER: It's certainly difficult to do *right* with an add-on package. You could probably come very close. But, yes, it is very hard to do. When my transaction runs, it does two things: it writes a disk block, and it sends a message to the terminal. If you're not careful, you can send them in the wrong order, and then whatever wasn't sent first may not happen if there's a failure.

REVIEW: *Is the logging of communication needed so that if a line is dropped, you can give the person the message again whenever they come back?*

WEINBERGER: Yes. You may desire to know that a message never got through. The other thing you have to watch out for in the database literature is that it deals mostly with transaction processing stuff. At least that's where the interesting theoretical questions typically have been. Another set of questions, though, considers what you can do if you do not have transactions and you're not retrieving little records. These questions surface when you consider relational retrieval systems. What if my objects are programs or pictures?

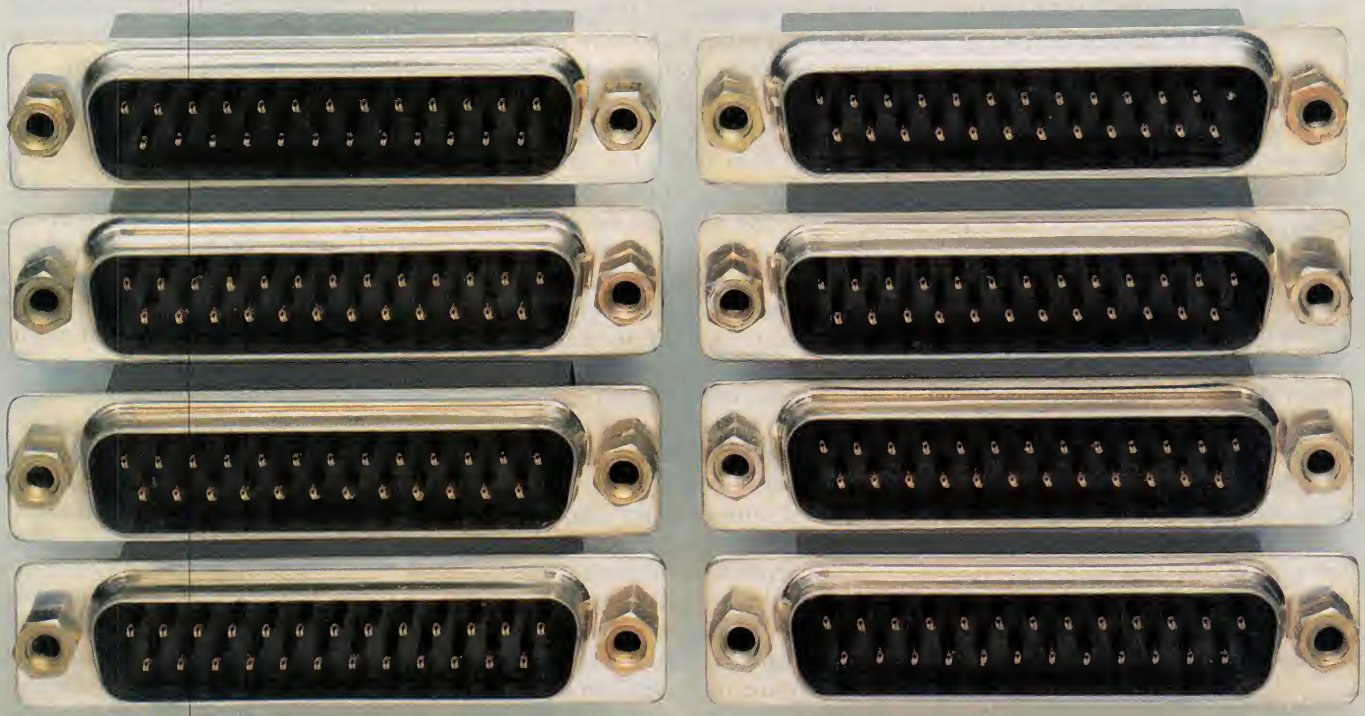
Pictures can be small, but maps are big. Of course, records can also be very large. What if the database is enormous? The census is an example of an enormous database. You probably have a billion bytes or more of data. I don't think anybody's very good at taking data from the census and simply putting it into a database system—not without writing a lot of special code first. What's more, there are only certain questions you can ask economically for a very large database. If I have a 10,000-byte database or whatever—you can look at every character and think hard. But if you have a giant database, that's just not possible. That's a problem with databases that exists no matter what system you implement your work on. As your view of the world changes—and as your customers' view of the world changes—your database will get weirder and weirder. One of the arguments in favor of doing things in a relational way is that you can do it more easily than you would be able to with a CODASYL model, one of those arrangements like a hierarchical database that's been wired into concrete.

There are also many philosophical questions—take the Bell Lab phone book database, for instance. Each employee has a unique payroll account number, so there is no way you can ask the database system reliably to see if somebody is listed in your database twice if the records have different keys, the system believes the people are different. That's because they *can't* be in the database twice.

One of the things that happens when people start talking about networks is that they have this idea that you want to be able to find things in the network. People tend to get carried away with the notion of using a database system as a name server that will allow them to look up nicknames and stuff. The fundamental problem is the difficulty in distinguishing among all the people named "David S. Johnson". You're probably going to have a lot of them. But you don't want to find all the people named Dave at Bell Labs during your search for just one of them.

REVIEW: *You would get mostly*

AST Eliminates XENIX's™ Biggest Limitation



AST's complete range of advanced technology enhancement solutions let you tap the full power and efficiency of XENIX.

More Users With AST-FourPort/XN.™ Add up to eight extra XENIX-compatible ports—including standard cables and drivers—to IBM® PC-ATs in XENIX multi-user, multi-tasking operating environments.

This low-cost solution provides up to four individually addressable RS-232 serial ports on a single card. Install two AST adapter boards into your "console" PC for a total of eleven simultaneous users, including IBM's three.

Supplied XENIX Drivers. Installation is a snap too. Simply insert one or two single boards. And use our "driver installation script" to automate linking our XENIX drivers to your XENIX kernal.

More Memory and I/O With AST's Advantage!™ and RAMvantage!™ Here's two expansion cards that let you take full advantage of the PC-AT's high performance bus and extended memory capabilities. Use our RAMvantage! to add up to 3.0 Mb of parity checked memory. Or select Advantage! for 3.0 Mb of memory and several popular I/O functions.

Both take only a single slot, and offer AST's built-in Split Memory Addressing™ which rounds out base memory to its maximum 640 Kb and simultaneously extends memory at 1 Mb and above.

Higher Performance With AST's Colossus™ Disk Subsystem. Expand the AT's mass storage capacity with this 74 Mb Winchester disk drive and streaming tape cartridge backup. You get the highest performance available—30 millisecond average access time with direct disk to tape data transfer at 5 Mb per minute.

Whatever your needs, AST's commitment to PC-AT enhancement provides the total XENIX solution. For more information call our Customer Information Center (714) 863-1333.
AST Research, Inc., 2121 Alton Avenue, Irvine, California 92714. TWX: 753699 ASTR UR.

AST-FourPort/XN

- *Adds up to eight XENIX-compatible user ports*
- *Includes expansion cable to four standard DB25 connectors*
- *XENIX Drivers Supplied*
- *Data rate individually programmable to 9600 Baud*

RAMvantage!

- *Memory expansion—128 Kb to 3.0 Mb in a single slot*
- *User upgradeable with 64K or 256K memory chips*
- *Split Memory Addressing rounds out AT's system memory to 640K and continues expansion at 1 Mb*
- *Supports AT's full program processing speed*

Advantage!

- *All the memory expansion features of RAMvantage!*
- *Up to 2 serial ports*
- *Parallel printer port*
- *Optional game port*

Colossus

- *74 Mb disk capacity formatted, expandable to 370 Mb*
- *30 msec average disk access time*
- *60 Mb streaming tape cartridge backup*
- *Utilizes SCSI technology*
- *XENIX drivers available Fourth Quarter 1985*

AST-FourPort/XN, Advantage!, RAMvantage!, Split Memory Addressing and Colossus trademarks of AST Research, Inc. IBM registered trademark of International Business Machines Corp. XENIX trademark of Microsoft.

AST RESEARCH INC.

Circle No. 54 on Inquiry Card

useless information?

WEINBERGER: Well, I do not know if it's useless information, but it's pretty hard, given a list of full real names, to disambiguate people. Get out the phone book to convince yourself of that. Relational databases can be troublesome—even as ideal objects.

REVIEW: *What about the relational model attracts you?*

WEINBERGER: I find the relational model fairly simple to understand. The hierarchical model sounds like the UNIX system directory tree, but it's not; it's nothing at all like the UNIX directory. All the layers are of different types.

The point is to keep things just as simple as possible—perhaps even a little simpler so as to better restrain you from the tendency to make things complicated.

REVIEW: *The question of UNIX suitability for databases then comes down to whether it's unsuitable or not. Considering that it's being used for database solutions today, it's clearly not unsuitable. It's just that the database solutions aren't being built into UNIX.*

WEINBERGER: That's right.

REVIEW: *So you don't need to abandon UNIX whenever you face a problem that requires a database.*

WEINBERGER: Right. There's a graduate student at Princeton who is building a file system that you can think of as a database file system. When you read and write using it, locking is handled for you automatically. The system knows about transactions, and it has a UNIX way of dealing with these things. A lot of the conventional trappings of transaction processing simply aren't there. It's all handled for you whether you ask for it or not.

REVIEW: *How is he implementing it?*

WEINBERGER: It looks like a piece of a disk, but there's a hook built in that notices when reads and writes occur, and handles locking and logging automatically. He can tell what each process is doing since he's in the kernel. The scheme does not use /usr/group standard record locking, but the record locking works just the same. It's all done secretly for you, or to you, depending on how you've asked for it. The goal was not to implement a database system under UNIX. It was to get fairly realistic measurements comparing different forms of concurrency control and crash recovery.

The concurrency information is essentially the core of the research. There's a lot of talk about that. I think the consensus is that the most efficient means for doing concurrency control and crash recovery happens to be the way that big commercial systems use rather than all the other fancy techniques people have thought up. Not everybody agrees, of course.

REVIEW: *Do you believe that there's only one solution?*

WEINBERGER: I don't think that there's only one solution, but I think at the moment there's a clear-cut first choice if you're trying to build a fairly high performance database system. You use locking instead of optimistic concurrency control, and you use logging instead of shadow paging or whatever the other alternatives are.

REVIEW: *Doesn't shadow paging buy you a bit in terms of recovery?*

WEINBERGER: Yes, shadow paging makes a lot of aspects of recovery easier. I had a package that did some kind of shadow paging in index trees, and we never had to roll back a transaction. It was wonderful. Until you committed new pages, they were invisible. The file got bigger, but you couldn't see the new pages logically. The problem is that you have to write whole pages to do that, and if your transactions are small, the log records will be small and won't take up whole pages. Then, if you've carefully got your file organized so that you get several disk blocks per revolution but your shadow pages end up getting placed someplace else, you're going to use the heads a lot. That makes the scheme seem noticeably less efficient. But it's not clear that's always important. Shadow pages really do offer a lot of conveniences.

But, for high performance it looks like locking and logging are going to win. Maybe it's not a consensus; it's probably controversial. But anyway, that's the way it looks to me.

REVIEW: *Are you still working on databases?*

WEINBERGER: No. I ran out of things I wanted to do. I'm still prepared to speculate about databases, though.

REVIEW: *Do you have any speculations in mind?*

WEINBERGER: You've heard a lot of them already, put forward as facts. We haven't really mentioned networking much, though. We've talked mostly about questions related to distributed databases. Of course, there's a lot of research being done on distributed database systems, but it's never been quite clear that the notion itself is really a good idea. Distributed programs are hard to understand. But you always have to face up to the question of networking when you talk about database systems. That really complicates things. ■

September 23-27 Computer Technology Group, Boston and Washington, DC: "Berkeley Fundamentals and **csh** Shell". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

September 23-27 Bunker Ramo Information Systems, Trumbull, CT: "Advanced C". Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

September 23-27 Information Technology Development Corporation, Cincinnati: "UNIX Systems Administration". Contact: ITDC, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.

September 25-27 Computer Technology Group, London: "Advanced C Programming Under UNIX". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

September 25-27 Interactive Systems Corp., Santa Monica, CA: "UNIX Architecture—A Conceptual Overview". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.

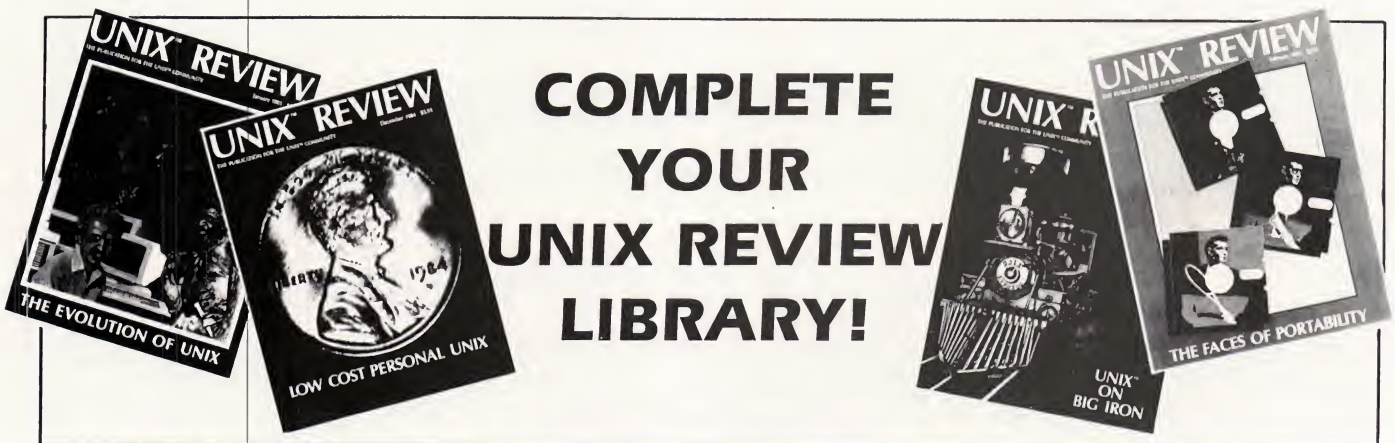
September 30-October 4 Computer Technology Group, London: "Berkeley Fundamentals and **csh** Shell". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

September 30-October 4 Bunker Ramo Information Systems, Trumbull, CT: "Intro to UNIX". Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

September 30-October 4 Interactive Systems Corp., Santa Monica, CA: "The C Programming Language". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.

September 30-October 11 Information Technology Development Corporation, Cincinnati: "C Programming Language". Contact: ITDC, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.

Please send announcements about training or events of interest to: UNIX Review Calendar, 500 Howard Street, San Francisco, CA 94105. Please include the sponsor, date and location of event, address of contact, and relevant background information.



COMPLETE YOUR UNIX REVIEW LIBRARY!

June/July 1983—UNIX on the IBM/PC .... ☐
August/September 1983—Sritek and Venix . ☐
October/November 1983—UNIX Typesetting ☐
December/January 1984—Vi and Emacs ... ☐
February/March 1984—UNIX Databases ... ☐
April/May 1984—Menu-based User Interfaces ☐
June 1984—Big Blue UNIX ..... ☐
July 1984—The AT&T Family ..... ☐
August 1984—Documentation ..... ☐
September 1984—System Administration ... ☐
October 1984—UNIX on Big Iron ..... ☐
November 1984—User Friendly UNIX ..... ☐
December 1984—Low Cost UNIX ..... ☐
January 1985—Evolution of UNIX ..... ☐
February 1985—UNIX Portability ..... ☐
March 1985—Performance ..... ☐
April 1985—UNIX Networking ..... ☐
May 1985—Distributed Resource Sharing ... ☐
June 1985—UNIX Applications ..... ☐
July 1985—Office Automation ..... ☐

Back issues are $4.95 each including postage. Payment in advance is required. Send this order form with check (US funds payable at US bank only) or credit card information to: REVIEW Publications, 901 S. 3rd St., Renton, WA 98055.
Additional $1.00/issue for foreign mail.

Name _____

Company _____
Address _____
City _____ State ___ Zip _____
M/C or VISA # _____
Exp. Date _____

ADVERTISER'S INDEX

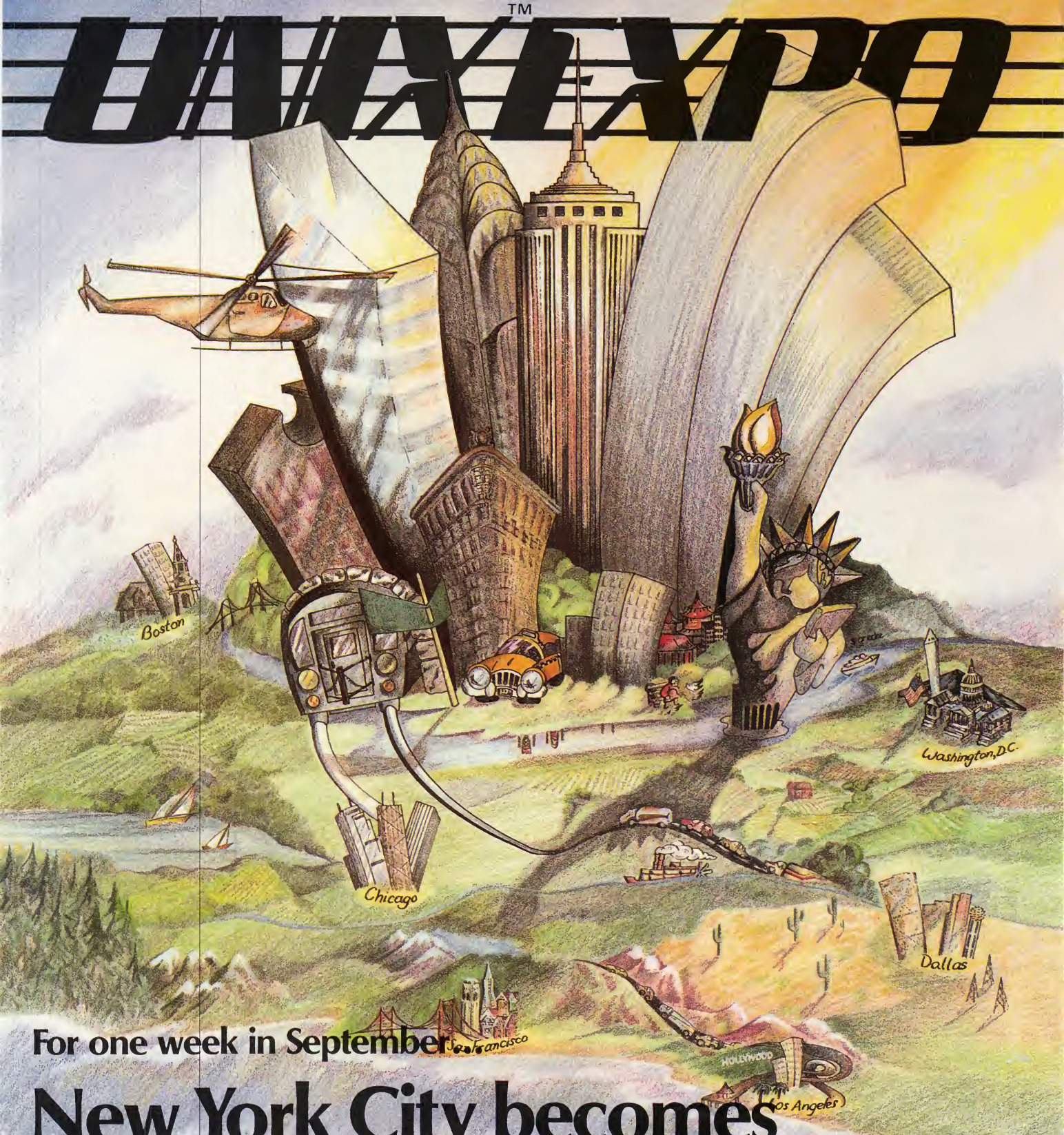
Adax Inc. 64
Artisan 91
AST Research, Inc. 101
AT&T Information Systems 47,73
B.A.S.I.S. 69
bbj Computer Services 75
Bell Technologies 86
Ceegen Corp. 20
Century Software 68
Charles River Data Systems 71
Cleo Software 99
CMI Corp. 67
Cogitate 98
Computer Cognition 93
Computer Methods 7
Concentric Associates 38
Corporate Microsystems, Inc. 21
COSI 83
DCC Data Service 98
Digital Equipment Corp. 10,11
DSD Corp. 39
Dynacomp 74
Franz, Inc. 90
Gould 85
Handle Technologies . . . . . Cover II
Image Network 89
Infosystems Technology, Inc. 97
Integrated Solutions 28,29
Mark Williams Company 9
Megadata Corp. 65
Microware Systems . . . . . 105,Cover IV
MIPS Software 93
Mt. XINU 22
NETI . . . . . Centerspread
Quality Software Products 89
R Systems, Inc. 31
Relational Database Systems 1,2,3
Santa Cruz Operation 62
Scientific Placement, Inc. 90
SHA Computers 86
Sperry Corp. 95
Unify Corp. 27
Unipress Software 13,15,17
Unitech Software, Inc. 18
United Airlines 57
University of Toronto 78
Uniworks 77,79,81
UNIX Expo 103
Verdix 23
Webco Industries, Inc. 106
Zanthe Information Systems . . . . . Cover III

COMING UP IN SEPTEMBER

Languages

- **The State of Compiler Technology**
- **Language Tools Under UNIX**
- **The Art of Source Code Maintenance**
- **C Standards Efforts**
- **The Lisp Connection**

UNIXEXPO™



For one week in September

New York City becomes the heart of the UNIX universe

Join the thousands of your colleagues who will seek answers to meet their business needs...and come away with a full understanding of UNIX solutions and applications.

Take advantage of the best UNIX has to offer:

- An exhibition featuring over 200 of the leading suppliers of UNIX based hardware, software and services.
- A tutorial program designed and developed by AT&T —the most respected source for the UNIX System.
- A conference examining the advantages of UNIX solutions in the business environment.

Plan now to attend and profit from THE PROVEN UNIX MARKETPLACE.

For all the details contact: UNIX EXPO 14 West 40th Street, New York, N.Y. 10018 Telephone: 212-391-9111. TELEX: 135401 DIMCOMM.

Circle No. 62 on Inquiry Card

UNIX™ is a registered trademark of Bell Labs UNIXEXPO is not affiliated with Bell Labs

CALENDAR

EVENTS

SEPTEMBER

September 18-20 National Expositions Inc., New York: "UNIX EXPO". Contact: Don Berey, 14 W. 40th St., New York, NY 10018. 212/391-9111.

September 26-28 8th Northeast Computer Faire, Boston. To be augmented with UNIX Systems Expo/85-Fall. Contact: Computer Faire, Inc., 181 Wells Ave., Newton, MA 02159. 617/965-8350.

TRAINING

AUGUST

August 5 LUCID, New York: "UNIX System Files". Contact: Alice Moss, 260 Fifth Ave., Suite 901, New York, NY 10001. 212/807-9444.

August 5-6 Computer Technology Group, San Francisco and Dallas: "Advanced C Programming Workshop". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

August 5-6 Interactive Systems Corp., Santa Monica, CA: "Advanced Commands for Programmers". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.

August 5-7 Computer Technology Group, Boston and Washington, DC: "UNIX Fundamentals for Programmers". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

August 5-9 Information Technology Development Corporation, Cincinnati: "UNIX for End Users". Contact: ITDC, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.

August 6 Computer Technology Group, London: "UNIX Overview". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

August 7-9 Interactive Systems Corp., Santa Monica, CA: "UNIX Architecture: A Conceptual Overview". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.

August 7-9 Computer Technology Group, Dallas and San Francisco: "Advanced C Programming Under UNIX". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

August 7-9 Computer Technology Group, London: "Unix Fundamentals for Non-Programmers". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

August 7-9 Digital Equipment Corp., Houston: "Comprehensive Overview of the UNIX Operating System". Contact: Digital Education Resources, 12 Crosby Drive, Bedford, MA 01730. 617/276-4949.

August 8-9 Computer Technology Group, Boston and Washington, DC: "Shell as a Command Language". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

August 12-14 Computer Technology Group, London: "UNIX Fundamentals for Non-Programmers". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

August 12-16 Computer Technology Group, San Francisco: "Berkeley Fundamentals and **csh** Shell". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

August 12-16 Interactive Systems Corp., Santa Monica, CA: "The C Programming Language". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.

August 12-16 Computer Technology Group, Boston and Washington, DC: "C Language Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

August 12-16 Computer Technology Group, Dallas: "Berkeley Fundamentals and **csh** Shell". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

August 12-16 Bunker Ramo Information Systems, Trumbull, CT: "Advanced C". Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

August 12-16 Information Technology Development Corporation, Cincinnati: "INFORMIX Relational Data Base". Contact: ITDC, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.

August 15-16 Computer Technology Group, London: "Shell as a Command Language". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

August 19-20 Productivity Products International, Aspen, CO: "The Concepts of Object-Oriented Programming". Contact: Barbara Dunn, Productivity Products Int'l, 27 Glen Rd., Sandy Hook, CT 06482. 203/426-1875.

August 19-20 Computer Technology Group, Boston and Washington, DC: "Shell Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

August 19-20 Interactive Systems Corp., Santa Monica, CA: "Advanced Topics for C Programmers". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.

August 19-23 Bunker Ramo Information Systems, Trumbull, CT: "C Programming". Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

August 19-23 Computer Technology Group, London: "C Language Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

August 19-23 Information Technology Development Corporation, Cincinnati: "Bourne Shell Programming". Contact: ITDC,

9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.
August 20 Computer Technology Group, Chicago and Los Angeles: "UNIX Overview". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
August 20 Silicon Valley Net, Palo Alto, CA: "NFS: Network File System". Contact: Grant E. Rostig, PO Box 700251, San Jose, CA 95170-0251, 415/593-9445.
August 21-23 Computer Technology Group, Chicago and Los Angeles: "UNIX Fundamentals for Non-Programmers". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
August 21-23 Computer Technology Group, Boston and Washington, DC: "Using Advanced UNIX Commands". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
August 21-23 Interactive Systems Corp., Santa Monica, CA: "Advanced C Programming Under UNIX". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.
August 26-28 Computer Technology Group, Chicago and Los Angeles: "UNIX Fundamentals for Programmers". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
August 26-30 Interactive Systems Corp., Santa Monica, CA: "Ten/Plus Helper Writer Workshop". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.
August 26-30 Computer Technology Group, Boston and Washington, DC: "UNIX Internals". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
August 26-30 Information Technology Development Corporation, Cincinnati: "C Shell Programming". Contact: ITDC, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.
August 28-30 Digital Equipment Corp., Chicago: "Comprehensive Overview of the UNIX Operating System". Contact: Digital Education Resources, 12 Crosby Drive, Bedford, MA 01730. 617/276-4949.
August 29-30 Computer Technology Group, Chicago and Los Angeles: "Shell as a Command Language". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

SEPTEMBER

September 2-3 Computer Technology Group, London: "Shell Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
September 3-5 LUCID, New York, NY: "UNIX Shell Programming". Contact: Alice Moss, 260 Fifth Ave., Suite 901, New York, NY 10001. 212/807-9444.
September 4-6 Interactive Systems Corp., Santa Monica, CA: "INword Word Processing Workshop". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.
September 4-6 Computer Technology Group, London: "Using Advanced UNIX Commands". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
September 4-6 Computer Technology Group, London: "UNIX Internals". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
September 9-11 Interactive Systems Corp., Santa Monica, CA:

"UNIX Fundamentals". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.

September 9-12 LUCID, New York, NY: "UNIX System Administration". Contact: Alice Moss, 260 Fifth Ave., Suite 901, New York, NY 10001. 212/807-9444.

September 9-13 Bunker Ramo Information Systems, Trumbull, CT: "Advanced UNIX". Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

September 9-13 Computer Technology Group, Chicago and Los Angeles: "C Language Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

September 9-20 Information Technology Development Corporation, Cincinnati: "UNIX for Application Developers". Contact: ITDC, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.

September 10-12 Bunker Ramo Information Systems, Trumbull, CT: "Diagnostic UNIX". Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

September 10-12 Computer Technology Group, Boston and Washington, DC: "UNIX Administration". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

September 10-13 Integrated Computer Systems, Los Angeles and Washington, DC: "UNIX: A Comprehensive Introduction". Contact: ICS, 45405, Los Angeles, CA 90045. 213/417-8888.

UNIX*-C COURSES

THE UNIX SYSTEM FOR END USERS
Sept 9-13 • Oct 7-11 • Oct 28-Nov 1

THE UNIX SYSTEM FOR THE DP PROFESSIONAL
Sept 16-20 • Sept 30-Oct 4
Nov 4-8 • Dec 9-13

C LANGUAGE PROGRAMMING
Sept 23-27 • Nov 18-22 • Dec 2-6

HANDS-ON SESSIONS IN ALL COURSES
COURSE FEE AS LOW AS $855
DISCOUNTS FOR EARLY REGISTRATION

COURSE LOCATION
WASHINGTON, D.C.
METROPOLITAN AREA

On-Site and customized courses also available. Write or call for course descriptions and registration information.

(301) 498-0722
WEBCO INDUSTRIES, INC.
14918 LAUREL OAKS LANE
LAUREL, MARYLAND 20707

*UNIX is a trademark of AT&T Bell Laboratories

CLIP AND SAVE

Circle No. 44 on Inquiry Card

September 12-13 Interactive Systems Corp., Santa Monica, CA: "Using the Shell". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.

September 16-17 Interactive Systems Corp., Santa Monica, CA: "System Administrator's Overview". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.

September 16-17 Computer Technology Group, Chicago and Los Angeles: "Shell Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

September 16-17 Computer Technology Group, Boston and Washington, DC: "Advanced C Programming Workshop". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

September 16-18 CL Publications, Cambridge, MA: "C Technical Seminar". Contact: Carl Landau, CL Publications, 131 Townsend St., San Francisco, CA 94107. 415/957-9353.

September 16-19 AT&T Information Systems, Callaway Gardens, GA: "UNIX OS: The First Step". Contact: AT&T Information Systems' Institute for Communications and Information Management, PO Box 8, Pine Mountain, GA 31822-0008. 800/247-1212.

September 17-18 Bunker Ramo Information Systems, Trumbull, CT: "UNIX/C Applications". Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

September 17-20 LUCID, New York, NY: "Advanced C Programming". Contact: Alice Moss, 260 Fifth Ave., Suite 901, New York, NY 10001. 212/807-9444.

September 18-20 Computer Technology Group, Chicago and Los Angeles: "Using Advanced UNIX Commands". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

September 18-20 Computer Technology Group, London: "UNIX Administration". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

September 18-20 Computer Technology Group, Boston and Washington, DC: "Advanced C Programming Under UNIX". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

September 18-20 Digital Equipment Corp., New York: "Comprehensive Overview of the UNIX Operating System". Contact: Digital Education Resources, 12 Crosby Drive, Bedford, MA 01730. 617/276-4949.

September 18-20 Interactive Systems Corp., Santa Monica, CA: "Interactive Networking Tools". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.

September 23-24 Computer Technology Group, London: "Advanced C Programming Workshop". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

September 23-24 Productivity Products International, Raleigh, NC: "The Concepts of Object-Oriented Programming". Contact: Barbara Dunn, Productivity Products Int'l, 27 Glen Rd., Sandy Hook, CT 06482. 203/426-1875.

September 23-24 Interactive Systems Corp., Santa Monica, CA: "Advanced Commands for Programmers". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.

September 23-27 Computer Technology Group, Chicago and Los Angeles: "UNIX Internals". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

"Nature Imposes Few Restrictions on Those Daring Enough to Lead"

ZIM is a fully integrated fourth generation application development system designed for leading system integrators and corporate and independent applications developers.

*COMPLETE DEVELOPMENT ENVIRONMENT
- -Report Writer
- -Forms Painter and Manager
- -Data Dictionary
- -Application Generator
- -Non-procedural Programming Language
- -Compiler
- -C Language Interface
- -Runtime System

*POST RELATIONAL
- -Entity Relationship Model
- -Powerful extension of Relational Model

*MAINFRAME POWER, FUNCTIONALITY AND PERFORMANCE
*APPLICATIONS PORTABILITY
- -MS-DOS, UNIX, XENIX, and QNX

*MULTI-USER
- -Full transaction processing control

*NETWORKING
*APPLICATIONS LIMITED ONLY BY HARDWARE
*BUILT-IN STRATEGY OPTIMIZER
*ENGLISH-LIKE LANGUAGE
*QUALITY PRODUCT SUPPORT

ZIM is a mainframe system that runs on micro-computers and on super micro-computers. If you want mainframe power, speed, flexibility and freedom from arbitrary limitations all at a micro price, talk to us about an evaluation system.
Dealer inquiries are welcome.

ZIM
The Information Interface

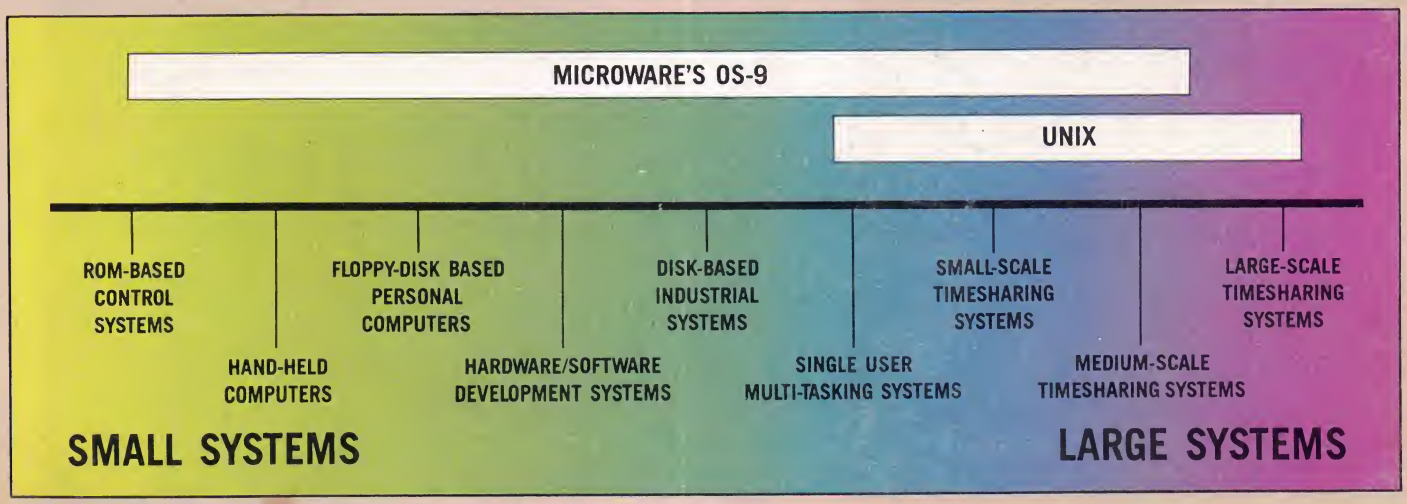
ZANTHE

1785 Woodward Dr., Ottawa, Ontario
K2C 0R1 (613)727-1397

MS-DOS and XENIX are Microsoft Corp. trademarks. UNIX is an AT&T trademark. QNX is a Quantum Software Systems trademark.

Circle No. 56 on Inquiry Card

Only Microware's OS-9 Operating System Covers the Entire 68000 Spectrum



Is complicated software and expensive hardware keeping you back from Unix? Look into OS-9, the operating system from Microware that gives 68000 systems a Unix-style environment with much less overhead and complexity.

OS-9 is versatile, inexpensive, and delivers outstanding performance on any size system. The OS-9 executive is much smaller and far more efficient than Unix because it's written in fast, compact assembly language, making it ideal for critical real-time applications. OS-9 can run on a broad range of 8 to 32 bit systems based on the 68000 or 6809 family MPUs from ROM-based industrial controllers up to large multiuser systems.

OS-9'S OUTSTANDING C COMPILER IS YOUR BRIDGE TO UNIX

Microware's C compiler technology is another OS-9 advantage. The compiler produces extremely fast, compact, and ROMable code. You can easily develop and port system or application software back and forth to standard Unix systems. Cross-compiler versions for VAX and PDP-11 make coordinated Unix/OS-9 software development a pleasure.

SUPPORT FOR MODULAR SOFTWARE — AN OS-9 EXCLUSIVE

Comprehensive support for modular software puts OS-9 a generation ahead of other operating systems. It multiplies programmer productivity and memory efficiency. Application software can be built from individually testable software modules including standard "library" modules. The modular structure lets you customize and reconfigure OS-9 for specific hardware easily and quickly.

A SYSTEM WITH A PROVEN TRACK RECORD

Once an underground classic, OS-9 is now a solid hit. Since 1980 OS-9 has been ported to over a hundred 6809 and 68000 systems under license to some of the biggest names in the business. OS-9 has been imbedded in numerous consumer, industrial, and OEM products, and is supported by many independent software suppliers.

Key OS-9 Features At A Glance

- Compact (16K) ROMable executive written in assembly language
- User "shell" and complete utility set written in C
- C-source code level compatibility with Unix
- Full Multitasking/multiuser capabilities
- Modular design - extremely easy to adapt, modify, or expand
- Unix-type tree structured file system
- Rugged "crash-proof" file structure with record locking
- Works well with floppy disk or ROM-based systems
- Uses hardware or software memory management
- High performance C, Pascal, Basic and Cobol compilers

microware® OS-9™

MICROWARE SYSTEMS CORPORATION
1866 NW 114th Street
Des Moines, Iowa 50322
Phone 515-224-1929
Telex 910-520-2535

Microware Japan, Ltd
3-8-9 Baraki, Ichikawa City
Chiba 272-01, Japan
Phone 0473(28)4493
Telex 299-3122

OS-9 is a trademark of Microware and Motorola. Unix is a trademark of Bell Labs.