UNIX™ REVIEW

THE PUBLICATION FOR THE UNIX™ COMMUNITY

May 1985 $3.95

DISTRIBUTED RESOURCE SHARING

MODULAR.
INTEGRATED.
NOW.

Handle™ presents a modular software series for office automation.

Available now for several UNIX™ and XENIX™ based systems, the **Handle** family of office automation products may be purchased as individual modules, in combinations, or as a fully integrated system. The **Handle** product series is a powerful set of software tools designed for today's multi-user office environment. **Handle** integrated software can easily share data between modules in the series as well as with outside applications and databases. **Handle's** open architecture is built on a database foundation and is available to outside developers.

Handle Office Automation Software Series product modules include:

- **Handle Writer/Spell™.** Word processing with automatic spelling correction and verification.
- **Handle Calc™.** Virtual spreadsheet with up to 32,000 rows and columns.
- **Handle Graphics™.** Advanced business graphics.
- **Handle List™.** List processing, management, and forms.

AT&T 3B2/300 & 3B5

CONVERGENT TECHNOLOGIES MINIFRAME

DURANGO POPPY II

To be announced?

HANDLE
TECHNOLOGIES, INC.

Circle No. 300 on Inquiry Card

850 NORTH LAKE BOULEVARD / PO BOX 1913 / TAHOE CITY, CALIFORNIA 95730 / 916-581-5227

TM— HANDLE, HANDLE WRITER/SPELL, HANDLE CALC, HANDLE GRAPH & HANDLE LIST ARE TRADEMARKS OF HANDLE TECHNOLOGIES, INC. UNIX IS A TRADEMARK OF AT&T BELL LABORATORIES. XENIX IS A TRADEMARK OF MICROSOFT CORPORATION. MINIFRAME IS A TRADEMARK OF CONVERGENT TECHNOLOGIES. POPPY IS A TRADEMARK OF DURANGO.

UNIX V.2, FAST COBOL & PC-UNIX-HOST NETWORKING

THREE COMPLEX NEEDS ONE SIMPLE SOLUTION

When we designed the UNITE product line, we did it with the systems integrator in mind. For example, we knew if we packed the power of a minicomputer into a chassis the size of an IBM PC, gave it PC to UNIX to MAINFRAME communication and the ability to run COBOL programs under UNIX super fast, we'd really have something.

Well, we did it. It's called the UNITE 8i.

We think you'll find that it's the optimal, multi-user, multi-functional business computer for systems integration on the market.

True, the UNITE 8i is small and powerful. But like all CYB computers, it's also versatile, reliable and priced just right.

PERFORMANCE

The UNITE 8i utilizes a 10MHz MC68010 virtual processor to access up to 12 megabytes of RAM, including up to 4 megabytes of no-wait state memory. A second MC68010 (12.5 MHz) is a high-performance VPM I/O accelerator, which greatly increases the speed of the computer by off-loading the majority of the I/O functions from the main CPU.

The UNITE 8i's third source of power comes from its full implementation of UNIX System V.2 with demand paging, virtual memory, Berkeley 4.2 networking and Ethernet support.

PC NETWORKING

If you've designed your vertical solution to run on a multi-user system in a business environment, you've probably been faced over and over again with an installed base of PCs that your customer wishes to retain.

UNITE solves your problem with the friendliest DOS to UNIX interface available today.

Our UNITE networking software allows PCs to fully access your vertical package and still run DOS application software. Plus, it provides PC to IBM mainframe and DEC VAX communications.

VERSATILITY

If there's one thing systems integrators need, it's versatility. The UNITE 8i is only one of several supermicros from CYB. Each one can be configured with a variety of memory, disk, tape, language (C, BASIC, COBOL, FORTRAN-77, Pascal, LISP) and communications options.

And if you have COBOL programs requiring fast execution we offer two alternatives. Run them under UNIX with PHILON's FAST/COBOL. Or choose the RM/COS operating system option. Either way, you can't lose.

RELIABILITY

CYB computers are rugged. We enclose only the highest quality materials and components in our sturdy, heavy-gauge aluminum cabinets.

After successful completion of rigorous pre-testing and burn-in, we back every system with a full warranty.

Please call or write for more information and ask about our *free* technical training for systems integrators.

Eight-user systems begin under $10,000*.

CYB SYSTEMS, INC
6448 HIGHWAY 290E
AUSTIN, TEXAS 78723
(512) 458-3224

*OEM Pricing

CYB...Single Source Solution for Systems Integration

UNITE is a trademark of CYB SYSTEMS, Inc. UNIX is a trademark of AT&T Bell Laboratories. Ethernet is a trademark of Xerox Corporation. DEC and VAX are trademarks of Digital Equipment Corporation. IBM is a registered trademark of International Business Machines Corporation. PHILON is a trademark of Philon, Inc. RM/COBOL is a trademark of Ryan McFarland Corporation.

Circle No. 299 on Inquiry Card

How we improved Structured Query Language.

Actually, we didn't change a thing.

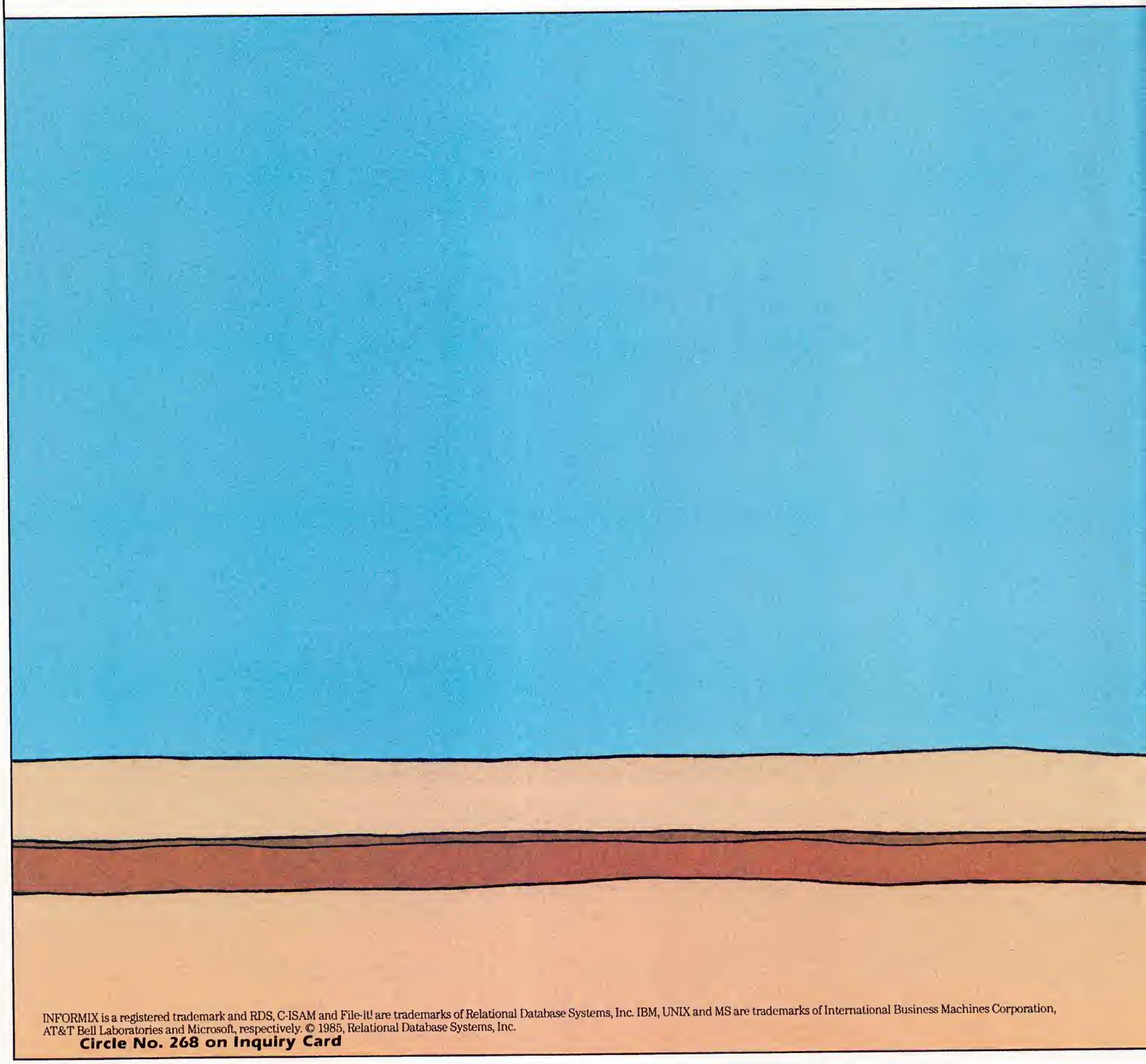
We just combined it with the best relational database management system.

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



Circle No. 268 on Inquiry Card

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.

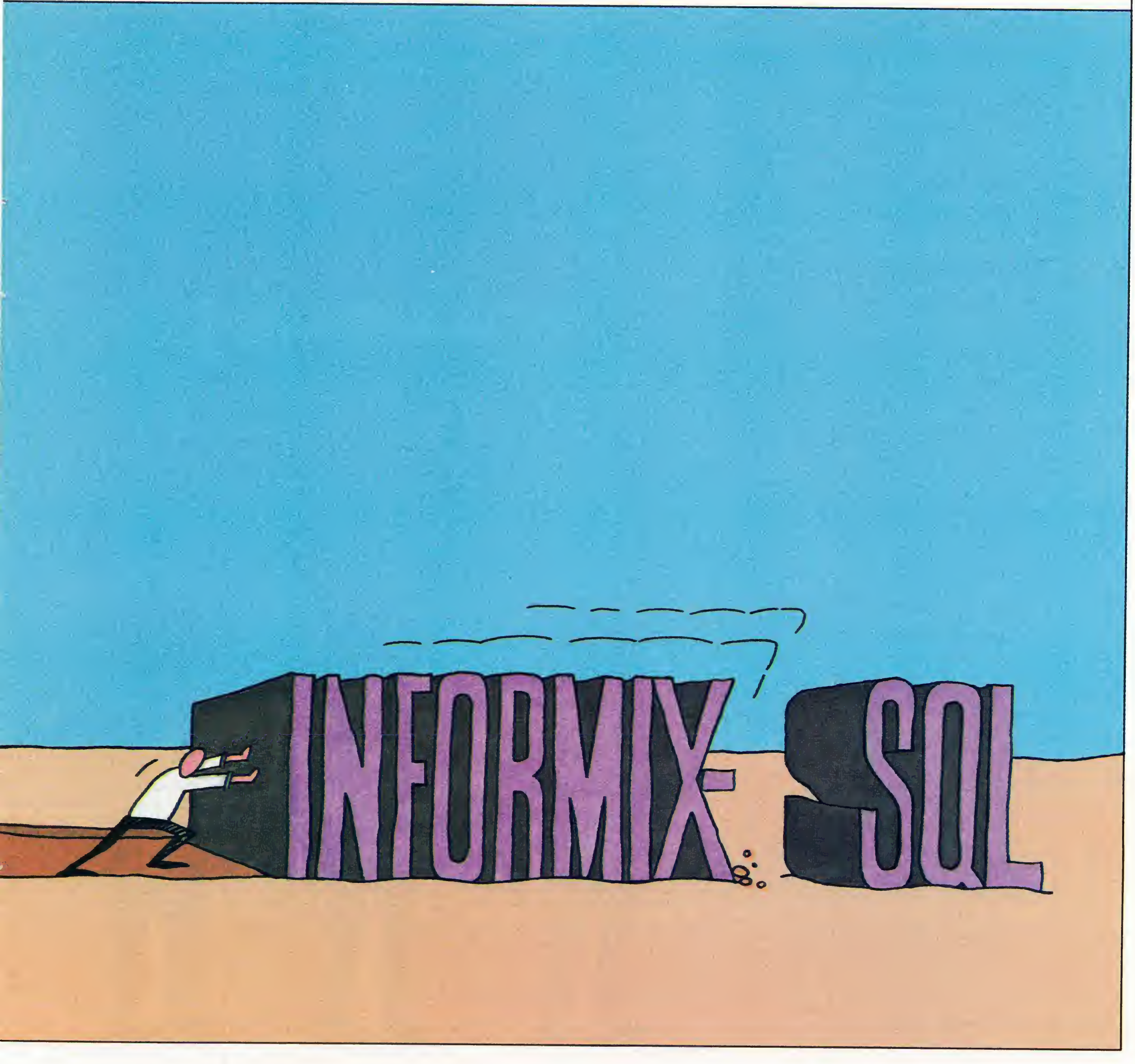
So call us for a demo, a manual and a copy of our Independent Software Vendor Catalog. Software vendors be sure to ask about our new "Hooks" software integration program. Our number: 415/424-1300.

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 5
May 1985

DEPARTMENTS:

Cover art by Stephen G. Luker

FEATURES:

DISTRIBUTED RESOURCE SHARING

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.

UNIX REVIEW is sent free of charge to qualified individuals in the U.S. The publisher reserves the right to determine qualification. For those not qualified, there is an annual (12 issues) subscription charge of $28 in the U.S., $35 in Canada and $48 in all other countries (surface mail). 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 EDITOR:
Roger Strukhoff
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 Technology 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, Bell Laboratories
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:
Colleen M. Y. Rodgers
Sales/Marketing Assistant:
Anmarie Achacoso

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

BPA membership applied for in March, 1985.

VIEWPOINT

The dawning of an era

"Distributed resource sharing" is one of those golden phrases that journalists and marketing people love dearly. Like others that roll off the tongue easily, it's not always clear what people mean when they use it. But don't write off "distributed resource sharing" as just another content-free cliche; it actually describes a significant new direction in computing.

Not every user believes that the promised environment looks quite like distributed processing, but precious few are willing to go back to the timesharing solutions of days gone by. The reduced need for sharing under distributed schemes is the clincher.

Sure, users still have to queue up to use printers, disk drives, and a whole parade of other peripherals, but everyone also can have his or her own CPU. This translates directly into greater autonomy and improved (or at least more regular) response times.

Of course, in a perfect world where the sun shines every day and taxes are collected only on alternate leap years, users wouldn't have to share anything. But, alas, financial and spatial constraints suggest that day will be some time in coming.

Distributed resource sharing itself took awhile before making a big splash. Certainly, the concept is not particularly new—Datapoint, for one, has been in the distributed processing game for almost a decade. But, new technologies and a general drop in component pricing over recent years have served to bring the day of "a CPU in every office" closer to fruition.

We're still in the midst of the movement, however. There's almost as many distribution methodologies as there are companies offering "solutions". The gamut ranges from networks of low-end PCs to systems of supermicro workstations.

This issue of UNIX REVIEW takes stock of where we're at and points to where we may be heading. Paul Leach, one of the designers of Apollo Computer's DOMAIN system, opens up with an article discussing schemes for improving individual and group productivity through distributed processing.

Subsequent articles by Steve Holmgren and regular columnist Bill Tuthill discuss the virtues of remote procedure calls. Holmgren, who is President of Communication Machinery Corp. of Santa Barbara, CA, also lays out possible future RPC directions.

The architectural issues behind distributed file systems are the concern of another piece jointly authored by Gary Sager and Bob Lyon, two of the key figures behind Sun Microsystem's Network File System.

Dave Buck wraps up with an article focusing on a crucial local resource often overlooked by people in the UNIX community—the mainframe database. Drawing on his experience as Chairman of D.L. Buck and Associates, a company specializing in UNIX drivers for systems that need to interface with IBM machines, Buck discusses the options for accessing large databases.

All in all, the issue should give you something to connect with the next time associates drop references to distributed processing in conversation. They might not know what they're talking about, but you will.

Mark Compton

IBM's® Unix™-based system of the future is the CIES system of today.

IBM knew that someday even small businesses would catch on to the fact that PCs just aren't enough. Too little power. Too little memory.

IBM's solution: a UNIX-based multi-user system. Someday.

It's the same solution CIE Systems came up with over two years ago. And it's a solution we've been perfecting ever since.

The new CIES 680/100 and CIES 680/200 are not in the future. Not something you need now, but can't get. They're here. Today.

They're here with multi-user expandability of from one to 40 users, doing different jobs simultaneously.

They're here with up to 2 Megabytes of memory.

They're here with up to 300-plus Megabytes of hard disk, along with a floppy disk drive and a streamer tape drive for backup.

They're here with all the power. All the memory. All the customer support. All the ready-to-run applications a business needs, including a complete general accounting program, word processing, electronic spreadsheet, even a complete medical practice program for physicians.

Not someday. Today. From CIE Systems, the ever-growing giant in supermicros backed by C. Itoh & Company Ltd., the fifth largest corporation in the world with over $60 billion in annual sales.

For more information on the system you need today, not someday, just call or write.

CIE Systems, Inc., 2515 McCabe Way, Irvine, CA 92713-6579. (714) 660-1800. Call toll free 1-800-854-5959. In California, phone 1-800-432-3687.

See us at Comdex
Booth # 1740 GWCC East Hall

™ CIES 680 is a Trademark of CIE Systems, Inc.
UNIX is a Trademark of Bell Laboratories
® IBM is a Registered Trademark of International Business Machines Corp.
© 1984 CIE Systems, Inc.

C.ITOH
Computers

CIES SYSTEMS SALES OFFICES

Southwest
2515 McCabe Way
Irvine, CA 92713
(714) 660-1800

Northeast
Executive Mews, M64,
1930 East Marlton Pike
Cherry Hill, NJ 08003
(609) 424-6925

North Central
1 Cross Roads of Commerce
Suite 305
Rolling Meadows, IL 60008
(312) 392-1331

South Central
17311 Dallas Parkway
Suite 230
Dallas, TX 75248
(214) 248-8355

Southeast
4501 Circle 75 Parkway
Suite 1190 A
Atlanta, GA 30339
(404) 953-1876

New England
400 Amherst St.
Space #41
Nashua, NH 03063
(603) 881-7031

Northwest
2700 Augustine Dr.
Suite 238
Santa Clara, CA 92504
(408) 748-0452

England
Beacon House
26/28 Worple Road
Wimbledon, SW19, 4EE
01-946-4960

Circle No. 267 on Inquiry Card

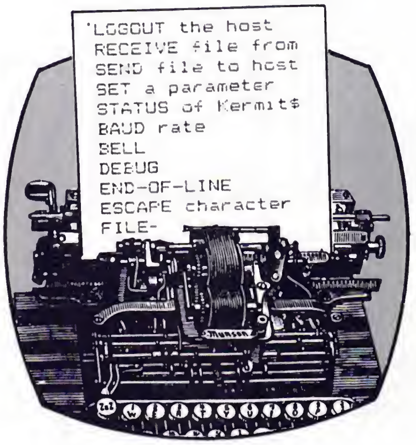
THE MONTHLY REPORT

AT&T pushes desktop UNIX

by Roger Strukhoff

Remember Lily Tomlin's classic character, Ernestine the telephone operator? Remember when she ran amok in the control room, knocking out service to Peoria while gloating, "We don't care ...we don't have to...we're the phone company."? Well, AT&T certainly doesn't have that luxury today. Since its 1984 divestiture, the company has been as subject to the cruelties of the marketplace as any other company. But a certain arrogance apparently persists at Ma Bell, as was demonstrated by the company's March announcement of the UNIX PC Model 7300.

Jack Scanlon, AT&T vice president for computer and work station systems, touted Bell-developed UNIX and belittled the commercial prospects of MS-DOS technology. Reminding one of Jerry Ford's refusal to acknowledge that Poland had come under Soviet domination, Scanlon maintained that MS-DOS is not the answer to business applications and will eventually die. "It's obvious that UNIX provides the total office solution," he said. "Harnessing UNIX on a desk (will) usher in a new era for PCs." Scanlon went on to praise the multitasking capabilities of UNIX, saying a computer running it can "match the way people actually think, work and act at the office."



AT&T's efforts to "civilize" UNIX (using AT&T's term) resulted in the 7300's Apple Macintosh-like interface. A mouse, icons, and superior bitmapped graphics may in fact be key to the 7300 becoming a real thoroughbred for the business user, but the initial lack of applications software could keep this horse in the gate for awhile. AT&T announced a total of 28 programs in its start-up library. Of the 28, there are two word processors (Microsoft Word and the AT&T UNIX PC Word Processor), two spreadsheets (Multiplan and Supercomp 20), one database (dBase III), three graphics programs, and AT&T's electronic mail. Fully half of the programs are programming and development tools.

Another problematic area involves the decision to equip the standard 7300 with only a 10 MB hard disk. A 20 MB option is also offered, but even that is small for a UNIX box. A 40-MB disk made by an outfit called Bell Technologies (no relation to Ma Bell) offers a more reasonable alternative, but the disk adds more than $2500 to the cost of the machine.

The 7300 gained notoriety under the "Safari" code name while under design at Convergent Technologies of Santa Clara, CA. It runs unbundled UNIX System V, as does AT&T's 3B2 supermicro, and is fired by a Motorola 68010 microprocessor running at a very brisk 10 MHz. The base system, priced at $5590, includes half a megabyte of RAM and a 10 MB hard disk. A more reasonable system, offering a full megabyte of RAM and a 20 MB hard disk, costs $6590. Half-megabyte RAM expansion cards cost $1195. All systems have three expansion slots.

The 7300 will concurrently support three users. Operating system software comes in three flavors: the core system with a telephone manager, UUCP, word processing, and other basic functions; a development tools package, with **sort, merge,** and ISAM; and a utilities package, including a C language compiler, 68010 Assembler, and SCCS.

The new machine's 12-inch green CRT monitor swivels and tilts, and has bitmapped, 720 x 348-pixel resolution. The bitmap-

"Nothing works together," they cried out.

"It will now," said Alis.

With Alis,™ everything works together. Text, spreadsheets, drawings, business graphics and database information work together in a single, always editable document.

Alis combines the advantages of integrated PC applications with the information-sharing benefits of communications-based OA systems. It offers the most advanced total office solution ever.

Alis makes it easy for people to work together. It provides integrated electronic mail, calendar and meeting scheduling, and revolutionary Automatic Office Assistants™ to aid management information monitoring and decision making.

Written in "C," Alis is initially available on UNIX* to large OEMs. It's destined to bring sanity to the mad tea party world of office automation.

*UNIX is a trademark of Bell Laboratories. Applix, Alis, and Automatic Office Assistants are trademarks of Applix, Inc.

Alis™

The next-generation office software system from APPLiX

Finally, some answers in Wonderland.

APPLiX, INC., 112 TURNPIKE ROAD, WESTBORO, MASSACHUSETTS 01581 (617) 870-0300

Circle No. 298 on Inquiry Card

ping program, from Graphic Software Systems of Wilsonville, OR, illustrates AT&T's belief that business wants Macintosh-like machines that offer good graphics and an icon-driven interface.

Communications naturally play an important role in the AT&T UNIX micro. A built-in modem operates at 300 or 1200 bits per second. The telephone manager, part of the core UNIX software package, has features such as repeat dialing, directory dialing, last number redialing, and data call setup to remote computers. The 7300 is capable of simultaneous voice and data communications, with alternation allowed. A call management service records call length and lets the user take notes for later review.

AT&T also announced three products other than the 7300: the Starlan departmental local area network, an enhanced 6300, and a terminal for use with the 7300.

The terminal, dubbed the Personal Terminal (PT), costs $1795; it's $1895 if you'd also like a keyboard. Really. Paraphrasing an old line from Apple Computer Corp., Jack Scanlon termed the PT "the workstation for the rest of us." It has a unique, soft touch-screen user interface, with tactile response. There are no infrared light grids. Scanlon said the terminal is aimed at a "boss-secretary" environment, in which voice and verbal messaging capabilities are predominant. It runs on Starlan and in a 7300 multiuser environment.

Enhancements to the 6300, meanwhile, give AT&T a machine that's squarely in competition with the IBM PC/AT. Among the improvements are an expansion card providing voice/data and other communications features; the availability of Microsoft Corp.'s XENIX 3.0; the addition of an Intel 8087-2 match co-processor; a new mouse; and a 20 MB hard disk option that includes a half megabyte of RAM. An enhanced 6300 with an 8087-2 co-processor is priced in the $6000 range.

In keeping with some of the biggest and smallest names in the industry, AT&T used a pre-announcement to trumpet the arrival of Starlan. Some might be so unkind as to call this a "vaporware" tactic. Starlan won't be available until the fourth quarter of this year. It will support both UNIX and MS-DOS machines; AT&T is not completely writing off IBM just yet. Company strategy calls for pushing the enhanced 6300 as a server for MS-DOS machines, and either the 7300 or 3B2 supermicro as the UNIX server.

AT&T's efforts to civilize UNIX resulted in the 7300's Apple Macintosh-like interface.

The LAN runs on twisted-pair using CSMA/CD access, running at 1 megabit per second (mbps). As many as 10 nodes can be daisy-chained; the theoretical limit for connections is 1200. At its heart is an Intel 82588 controller chip. Connection costs are in the $700 to $800 per node range. This is in the general area, if not at the high end, of connection costs for Ethernets and for the IBM PC Network. AT&T worked with the IEEE in developing this local area network, and is proposing the Starlan specifications as an 802.3 standard for 1-megabit LANs.

AT&T voice and data products, including the four announced at the March conference, are distributed through three basic channels: direct sales, value added resellers (VARs), and computer specialty retailers. AT&T has a direct sales force of about 7000 people who sell from about 200 sales offices throughout the country. The company expects about half its sales to be generated by direct sales. The second channel, VARs, includes about 60 companies working on about 50 regional and specialized market segments. There are also about 1000 retail outlets. AT&T expects that number to grow to about 1600 by the end of 1985.

CORVUS WAXES AND WANES

Proving itself to be a multitasking company, Corvus Systems took over Onyx + IMI, Inc. within weeks of cutting back its own staff. The Corvus-Onyx + IMI merger combines the unique capabilities of these two San Jose-based companies; "it's a good business fit," claims Corvus chairman and CEO Michael D'Addio. Onyx was the first company to make a UNIX-based microcomputer, while Corvus is best known for its storage devices and Omninet twisted-pair wire local area network.

Holders of Onyx + IMI stock will receive shares of Corvus common stock at an exchange ratio of between 1.25 and 1.3 to 1. Corvus and Onyx each have about 10.4 million shares of common stock outstanding; the companies' stocks were within 10 percent of each other in worth on a recent business day, with total worth in the neighborhood of $60 million. Once the merger is final, Corvus' new board will comprise seven people; the four from Corvus' present board, and three of four from Onyx's board. One member currently sits on both. Onyx's former

president, Fred Bialek, left the company a few weeks before the merger announcement.

Onyx has been under financial pressure, apparently due to its inability to introduce new products. The company's stock price had declined close to 90 percent in recent years, before rising again after the merger was announced. Onyx's early successful product was the 8000, the first UNIX-based microcomputer when it was introduced about five years ago. But the company could not follow up. An MC68000-based UNIX machine, aimed at competing with the DECs of the world, was scrapped at the last minute because management felt it was aimed at the wrong market; another, lower-priced 68000-based system is pending.

> Onyx had been planning a major reorganization prior to its merger with Corvus.

Onyx had been planning a major reorganization prior to the merger with Corvus. Many employees were (and are) being transferred to Medford, OR, while others will move to the Corvus complex.

Corvus' decision to diversify was announced within weeks of a round of staff cutting. About 50 jobs were eliminated, or 12 percent of the company's total work force. The company said it was consolidating different job functions in all areas of the company in this cutback. ■

UNIX SYSTEM V

TRAINING

For 15 years, we've taught our own people to use the UNIX™ System. Now we can teach yours.

WHY AT&T FOR UNIX SYSTEM TRAINING?

AT&T offers the most current and comprehensive training on UNIX Systems.

AT&T provides the best learning environment; one terminal per student; evening access to facilities; and expert instructors.

AT&T has the breadth of courses your staff needs to unlock the full power of UNIX System V.

AT&T courses signal your commitment to improving productivity with high-quality training for your employees.

AT&T COURSES OFFER:

The same training and methods we use to teach the UNIX System to our own people.

Rigorous classes designed to teach specific skills for job-specific applications.

Five areas of instruction ranging from introductory to advanced levels for Managers/Supervisors, Users, Systems Administrators, Applications Developers, and Systems Programmers.

Frequent class offerings so you won't have to wait for the courses you want.

Conveniently located training centers in Princeton, NJ; Columbus, OH; Lisle, IL; and Sunnyvale, CA. Or we'll bring our courses to your company and hold the training at your convenience.

For more information, a catalogue, or to register for classes, call 1-800-221-1647, Ext. 87.

AT&T

©1984 AT&T Technologies, Inc.

HCR/PASCAL, WIRTH ITS WEIGHT IN C

PASCAL Originally designed by Niklaus Wirth, is now available for a wide range of UNIX™ processors. HCR/PASCAL conforms closely to industry standards, passes all conformance tests in the PASCAL Validation Suite. Supports multiple module programs, a dynamic string package, and direct random file access.

C is the standard language of UNIX, HCR/PASCAL is written in C and translates PASCAL into C producing efficient optimized code. This approach allows direct interaction with the UNIX environment and offers a high degree of portability.

UNIX is a powerful yet flexible operating system environment. HCR/PASCAL is available today on a diverse range of UNIX hardware: AT&T 3B™ series, the NCR Tower,™ DEC PDP-11/VAX,™ and others. HCR has a growing line of UNIX software including business applications. We back up all our software with full support. To find out how we can put HCR/PASCAL, C, and UNIX together for you, call or write:

Human
Computing
Resources
Corporation

HCR

10 St. Mary Street, Toronto, Ontario, Canada M4Y 1P9 (416) 922-1937

UNIX is a trademark of Bell Laboratories. PDP-11, VAX, and DEC are trademarks of Digital Equipment Corporation. AT&T 3B is a trademark of American Telegraph & Telephone. NCR Tower is a trademark of NCR Corporation.

Circle No. 297 on Inquiry Card

THE HUMAN FACTOR

Fast prototyping and UNIX

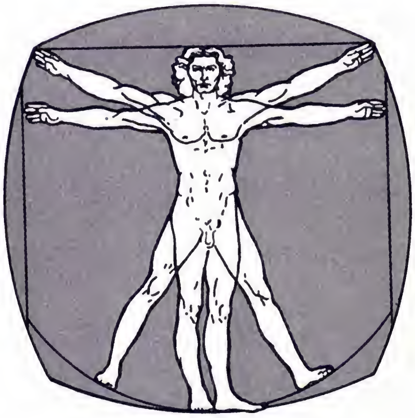
by Richard Morin

We have had a house cleaning in the computing industry. The structured methods advocates have been here, cleaning, polishing, and ordering. The smell of disinfectant is still overpowering. Swarms of **goto**s have been banished to a few dark corners. We have been reminded of the need for order, and have been taught the virtues of top-down design and stepwise refinement.

Even in maverick UNIXland, the effects are noticeable. The **lint** command has evolved into a fierce guardian of type checking. Even C shows signs of tampering. The **sccs** and **make** commands have emerged to help us manage our ungainly projects.

Still, a certain element of bottom-up programming has survived. It wears a suit and tie, to be sure, but it exists nonetheless. Cloaked in respectability, it now refers to itself as fast prototyping.

Fast prototypers make all the necessary bows to the structured establishment. They write modular code, using all manner of block structuring. They almost never use **goto**s.



Their chief resistance to the structured establishment is reflected in the way in which they approach projects: they skip the formal process of specification, with its structured walkthroughs, stepwise refinement, and so forth. Instead, they break off a piece of the problem and try to solve it. If that works, they try another piece. Pretty soon, the whole problem has been solved.

A certain degree of planning and analysis must be performed, to be sure. For one thing, the prototyper has to be careful to build up the solution in a modular fashion. For another, the order in which things are tried is critical.

It is very tempting to solve the easy parts of a problem first. Having all that code written is very comforting, but somewhat misleading. It does no good to solve the easy 90 percent of the problem if the remaining 10 percent is insolvable.

MANAGEMENT RESISTANCE

One might worry, with some justification, about management attitudes toward fast prototyping. We have spent the last decade teaching managers to expect structured methods. Now we have to convince them to permit this new approach.

Fortunately, several powerful arguments are available. First, prototyping allows one to find out very quickly whether a problem is, in fact, understood. Next, prototyping allows one to find out whether a problem is solvable. By attacking hard parts first, one finds land mines quickly. Reinforcements can then be brought in to save the day. Occasionally, a strategic retreat is advisable.

The analysis and design phases are critical. If either of them is going to fail, the manager needs to know about it quickly. Polishing, whether for appearance or efficiency, can wait.

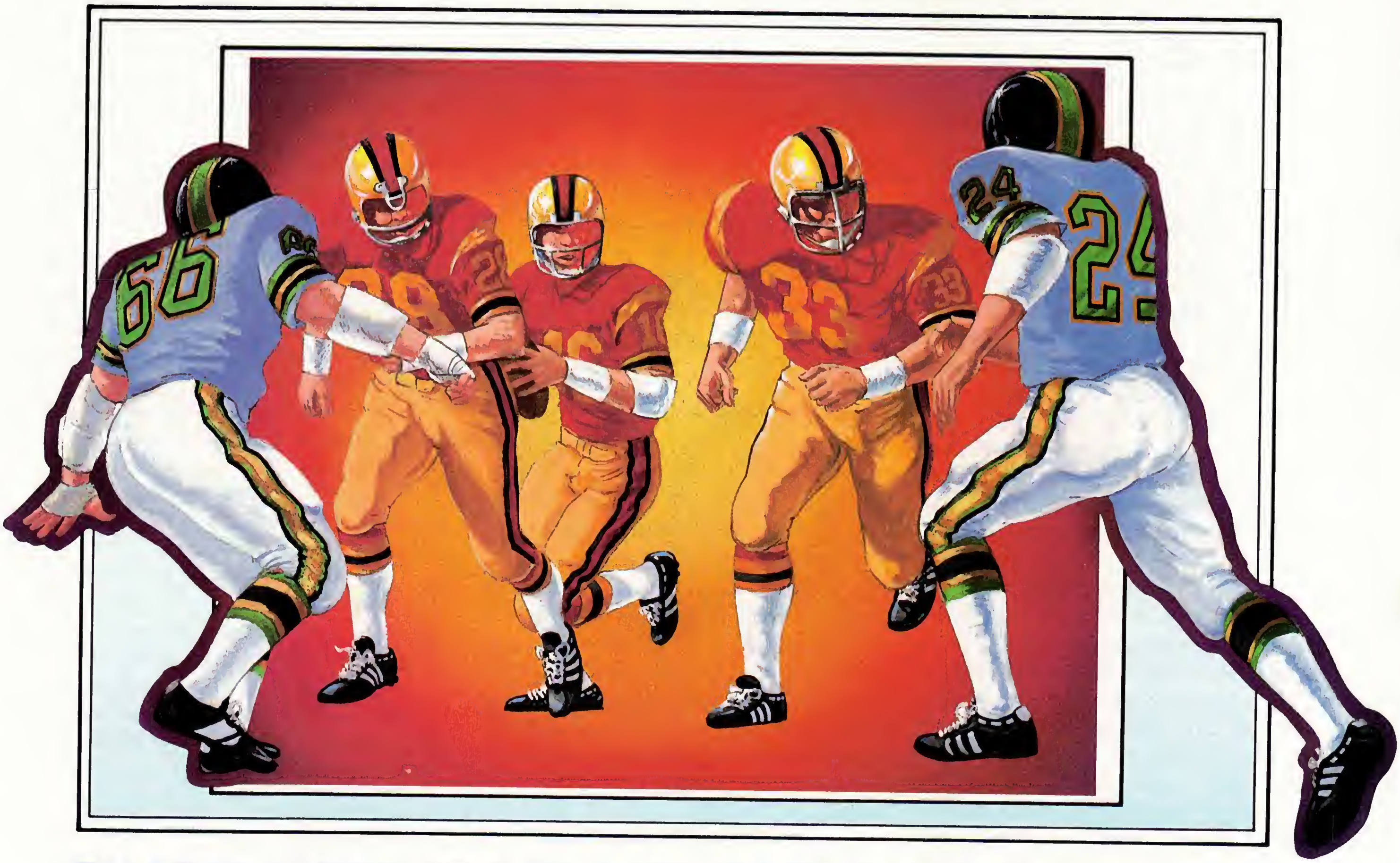
Finally, if a prototyping approach is forbidden, then the product will by definition become the first prototype—a curious phenomenon not unknown in software engineering.

LANGUAGE CHOICE

C, the lingua franca of the UNIX community, is not a particularly good fast prototyping language. For one thing, it has to be compiled and linked, which tends to discourage spontaneity. For another, it is rather picky about details such as variable initialization and so forth.

Use of C is often simply a neces-

ONLY PERFORMANCE COUNTS. ON THE FIELD AND IN BUSINESS.



THAT'S WHY IBC IS THE ONE COMPUTER YOU NEED WHEN YOU NEED MORE THAN ONE.

One thing counts on the field. Winning. Outperforming everyone in your class to become a champion. Execution. Maximum individual effort. Integration into a single team movement. Performance. Success. It's the same in business. Just win.

Your business needs a computer to help you team perform. One that lets more than one player get fast, efficient solutions to business problems. Solutions that one PC or even networked PCs can't give you. Either economically or functionally. Solutions that IBC multiuser computers will. And for less money.

Choose from four basic IBC multiuser computer families. They support UNIX and OASIS operating systems. Some support up to 32 users. Each is upgradable. They cost less than multiple PCs. About $1,500 less per user. Need more users? Add them for less than $600 with a dumb terminal. IBC systems have more memory: 256K to 8 Mbytes RAM, 12 Mbytes to 1.3 gigabytes hard disk storage. They share data. They share peripherals and software. They do more. They do it right. And they run faster. Benchmarks will prove it.

We have dealers who specialize in your business. Not department store retailers. Dealers who have the software you need. Specialists who've installed over 10,000 IBC multiuser business systems since 1979. Service? It's covered nationwide by IBC's authorized service centers.

Looking for your first computer system? Outgrown your present one? Are you a reseller addressing vertical markets? IBC has what you need. Call now for the specialist dealer nearest you. Call 1-800-4-MULTIS. Because IBC is the one computer you need when you need more than one. And you do.

IBC INTEGRATED BUSINESS COMPUTERS

"When Performance Is Everything"

21621 Nordhoff Street
Chatsworth, CA 91311
818/882-9007
Telex: 215349

Circle No. 277 on Inquiry Card

sary evil, however. Systems programming in UNIX is done almost exclusively in C. The occasional assembly language routine is not a very heartening exception.

Applications programmers are under no such constraints; they are free to use any language their system supports. The occasional system call will cause problems, of course. But Fortran and Pascal are used quite commonly for applications coding anyway.

The really radical approach, however, is to avoid compiled languages entirely. No compiling, no linking . . . just immediate execution. Talk about instant gratification!

Fast prototyping in the UNIX environment is thus a matter of writing interpreted scripts. These are then handled by **awk**, **sed**, a shell, or some other language processor. The scripts usually use a combination of languages and utilities.

Being interpreted, scripts are free to play fast and loose with programming techniques. They can rewrite their own subroutines, invent variables dynamically, and so forth. Using commands as operators, and files as data objects, they can be very powerful and very terse.

PROGRAMMER RESISTANCE

It is often hard for traditional programmers to accept this sort of coding. Pascal programmers worry about the lack of structured methods. C and assembly language hackers worry about efficiency.

Even questions of machismo enter the picture. "Real programmers code in compiled languages" and all that. In the seminars that I give on fast prototyping, I find myself explaining that I've *done* Fortran (about 25K lines), assembly language (about 15K lines), and C (about 5K lines). I then explain that I still use compiled languages. Some things require the expressive power and/or efficiency of a compiled language such as C. I even write an occasional assembly language routine.

C, the lingua franca of the UNIX community, is not a particularly good fast prototyping language.

Before I do, however, I try to use higher level tools. They allow me to increase my effectiveness, at the (possible) expense of my computer's efficiency. Often, the existing tools are as fast as the code I would have written myself—sometimes, they're actually faster.

This seems like a reasonable tradeoff. If more efficiency is required, a bit of profiling will show where it can be achieved. Recoding can thus be kept to a minimum.

A LOOK AT SHELLS

The UNIX notions of shells and shell programming are certainly not unique. Every operating system has some sort of command language. UNIX shells are better developed than most of these, however.

Command languages perform a number of tasks. They allow the invocation of user programs and system services, and specification of files and/or devices. Most support both batch and interactive use.

Most command languages also support some forms of control flow. In VMS, this is limited to IFs and **goto**s. In IBM JCL, even **goto**s are missing. UNIX shells have a broad range of control flow primitives.

UNIX shells also support modular programming. Shell scripts can invoke other scripts, or even invoke themselves. What's more, they can invoke scripts that they have written or modified.

By judicious use of quotes, a shell script can pass data into embedded **awk** and **sed** scripts. Other, more devious means allow retrieval of results.

The Bourne shell (**sh**) and the C shell (**csh**) are the current standards. The Bourne shell is better for most programming tasks, while the C shell is better for interactive use.

There are also some interesting alternatives on the horizon, including the experimental Functional Programming (**FP**) shell, and the practical Korn shell (**ksh**). In addition, there are a number of interpretive languages with shell-like capabilities.

FUNCTIONAL PROGRAMMING

The Functional Programming shell is described in a paper written by Manton Matthews and Yogeesh Kamath (Usenix Proceedings, Summer, 1984). The shell implements John Backus's Functional Programming language, using files as objects and programs as functions. The motivation of the authors is explained in their paper as:

> Our primary objectives in developing the FP-Shell were:
>
> [1] to provide a framework which is easy to understand and modify for the study of functional systems,
>
> [2] to extend the ability of the UNIX shells to combine programs by including other functional forms,
>
> [3] to investigate applications in which the extra

combining capability is utilized and gather usage statistics.

UNIX's use of pipes and filters is very close to the idea of composition of functions. The UNIX **sh** or **csh** command:

```
tbl paper | eqn | troff
```

would be rewritten in FP shell as:

```
vtroff o eqn o tbl: paper
```

An algebraic notation, by way of comparison, might render it as:

```
vtroff ( eqn ( tbl ( paper ) ) )
```

The interesting thing about the FP shell, however, is that a number of operations other than *composition* are supported. These include *construction*, *apply to all*, *conditional*, *insert*, *while*, *constant*, and *binary to unary*.

The FP shell thus allows commands that could not be written under, say, **sh**. The construction operator, for instance, allows the FP shell command:

```
C o [B2,B1] o A : input
```

This feeds the "input" into A, simultaneously piping the result through B1 and B2 before piping the result into C.

This kind of language design experimentation would be much more difficult in almost any other operating system. Thus, the flexibility of UNIX allows the UNIX community to be a sort of distributed computer science laboratory.

THE KORN SHELL

For an immediately practical alternative to the current shells, many people are looking at the Korn shell, **ksh**, described in D.G. Korn's paper "Introduction to KSH" (Usenix Proceedings, Summer, 1983). Korn shell advocates claim that it is better than **sh** for programming, better than **csh** for interactive use, and faster than both.

Some significant **ksh** programming features are aliases, arrays, built-in arithmetic, functions, enhanced I/O, variable attributes, and full **sh** compatibility.

In addition, a number of interactive features have been added. The Korn shell supports job control and command re-entry à la **csh**, and in-line editing surpassing it. A **ksh** user has the choice of using modified forms of either **vi** or **emacs** on the command line.

OTHER LANGUAGES

A number of interpretive languages can be used as UNIX shells. In point of fact, any interpretive language can be used as a shell. It is simply necessary that certain features, such as command invocation, I/O redirection, and so forth, be supported.

A later column will examine a number of these languages, comparing their features both as general programming languages and as shells.

Mail for Mr. Morin can be sent to the Canta Forda Computer Lab, P.O. 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. ■

Without FORTRIX™, moving up to 'C' can cost you a bundle!

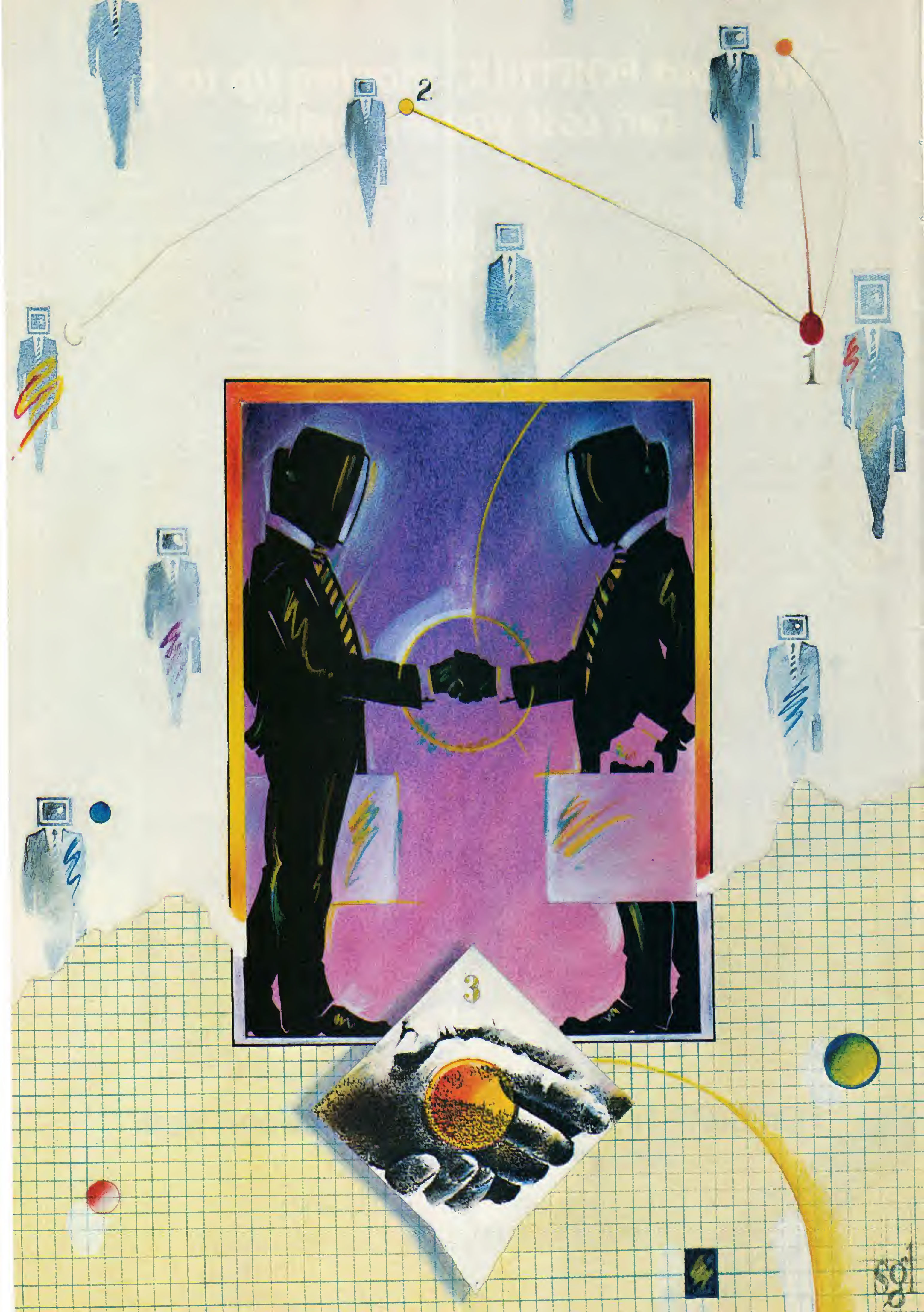
The bundle we're referring to consists of your existing FORTRAN programs and files. Costly items you'll have to discard when you move up to C, **unless** you save them with FORTRIX™!

Here at last is a program that automatically and rapidly converts FORTRAN code to C code, allowing you to salvage your FORTRAN material at approximately 600 lines per minute. This incredible speed allows a single programmer to convert, debug and put into operation a typical 50,000 line package in only one to two weeks. Plus, the resulting "C" program will run 15% to 30% faster than the original FORTRAN program, while occupying 35% less disk space! And the system even helps you learn coding in C language as you compare your own familiar FORTRAN programs with the corresponding C language programs generated by FORTRIX!™

There's a complete selection of FORTRIX™ versions to suit the full range of user requirements: Original FORTRIX™-C, which translates FORTRAN code to C code, allowing input data files to remain fully compatible with your new C program; FORTRIX™-C+, with the added ability to handle COMMON and EQUIVALENCE statements, character handling and direct I/O; FORTRIX™-C', the complete FORTRIX™-C+ package configured for non-UNIX* systems including VAX/VMS; and FORTRIX™-C/micro, standard FORTRIX configured for use on the IBM PC and compatibles.

FORTRIX™ has already been installed on 26 different brands of hardware, so whichever FORTRIX™ version meets your needs, you can be sure it will exceed your expectations in terms of speed and cost savings realized. Why not act now to save **your** bundle? Get full technical details, plus references from among over 100 satisfied licensees, from Jim Flynn at (212) 687-6255, Extension 44, or write to him at Rapitech Systems Inc., Dept. A2, 565 Fifth Avenue, New York, NY 10017.

FORTRIX™ Fortran-to-C Conversionware℠ from


2
1
3
sgl

THE WHYS AND WHEREFORES OF DISTRIBUTED RESOURCE SHARING

In some sense, all networks exist so that some resource might be shared. Even the earliest telecommunications networks, such as the Sabre airline reservation system that connected large numbers of geographically dispersed terminals to a central computing facility, were developed with the shared use of a computing resource in mind.

Since then, though, the term "network" has come to identify an interconnection of peer computers where the aggregate computational capability of the system is equal to the sum of the power of all of its parts. Among the resources of the individual machines that the network can draw upon are computing power, information, and peripheral devices. To achieve such a system, mere physical interconnection is not sufficient; the trick is to create a system that makes resources available in a convenient and efficient way to the entire network. The extent to which this is done depends not only on the skill of the system's implementers, but also on their objectives and the environment in which the system is expected to run.

We can perhaps see this best by examining two extremes in the resource sharing spectrum: the VAXcluster from Digital Equipment Corp. and the ARPAnet. The VAXcluster is a loosely coupled multiprocessor system consisting of several VAXen connected by a 70 mbps computer interconnect bus (the CI bus). Intended to cover only a small area, it is strictly a machine room network. The VAX cluster's software design creates the illusion of a single machine packing the combined power of the CPU, storage, and I/O capabilities of all of the cluster's machines.

The ARPAnet, on the other hand, is a nationwide network of totally autonomous, heterogeneous hosts interconnected by 56 kbps links. Despite the fact that the ma-

An overview of the burgeoning network concept

by Paul J. Leach

chines have several different architectures and are separately owned and administered, ARPAnet's network software is a success because it allows architectural chasms to be bridged in such a way that resources can actually be shared. Thus, ARPAnet users can exchange mail with other users, copy files from other hosts, and login to other systems on the network (the last two capabilities, though, require accounts on the other hosts).

ADVANTAGES OF DISTRIBUTED SYSTEMS

One would expect a distributed system for resource sharing to be more complex than a centralized system. The VAXcluster and ARPAnet networks are two examples that bear this out. One question this raises, of course, is: why go to all the extra trouble? Let's study the question briefly by looking at some of the advantages offered by a distributed system. Potential gains include incremental expandability, increased reliability, autonomy, and higher performance. (Since a system must be organized properly to actually achieve these advantages, careful scrutiny to determine the extent to which they have been attained may reveal strengths and weaknesses in a system's design.)

Incremental expansion refers to the system's ability to increase its capacity gracefully in small increments—which is to say, the granularity with which the network can be expanded. Also to be considered is the degradation in available computing power that additional loads bring.

At one end of the spectrum, the granule is the whole system. When the continual addition of users or applications overloads such a centralized system, the only options are to replace the system with an upgraded model (if one exists) or to try to split the users up into groups that need little or no communication with each other—which is to say, groups that can get along with isolated systems of their own. Even short of system overload, it should be noted that with systems of large granularity, each additional user represents a reduction in the average amount of computing power available to every other person in the user community.

At the other end of the spectrum, within, say, a network of personal workstations, the granule is the single user. When a new user is added to such a system, a workstation is added as well. The computing power available to each person thus remains constant. Somewhere between these extremes in granularity, most networks of small, timesharing computers might add a new machine for every five to 15 new users.

> All networks exist so that some resource might be shared.

Gains in the reliability of a distributed system can come in two forms: graceful behavior in the presence of failures, and high tolerance for faults. Because a system is a network of machines that can fail independently, the failure of a single node clearly deprives the system of that machine's resources, but the rest of the system should be able to continue operation. (This assumes, however, that the system is designed such that the resources that are lost are not critical to the rest of the network.)

Again, a look at examples at polar extremes should be illustrative. The two opposite ends of the failure behavior spectrum consists of a centralized system and a network of personal workstations. When the centralized system fails, all of its users are denied service whereas, when a personal workstation fails, only its owner is out of luck. (As a hedge against this, keeping a "hot standby" can be used as a fairly inexpensive way of getting back on the air quickly.) More advanced distributed systems allow important resources to be replicated—duplicate copies of files, multiple printers, and the like, so that if one instance happens to be down, another can be substituted.

The third potential advantage of distributed systems, autonomy, represents the right of components of a system to operate independently of the whole. Autonomy is one of the requirements for achieving good reliability: if machines can't operate independently enough, then one machine's failure will adversely affect many other users.

Independent operations are also valuable in helping to maximize individual and group productivity. In most organizations, each group has responsibilities to others, but the advantage of allowing each group to determine independently how best to meet those responsibilities has been borne out time and again. It's virtually axiomatic that the freedom to create innovative solutions leads to better performance. Centralized computer systems simply do not provide the best environment for such freedom. In such systems, any major new use affects all current users. Those users (or those who represent them) may thus need to be consulted or taken into consideration before work proceeds.

In a distributed system, it is possible to distribute the authority as well as the computing power necessary to manage this kind of issue. The users of each machine

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

Circle No. 293 on Inquiry Card

can decide how best to make their machine serve their purposes—usually without seriously impacting users on other machines. Within the limits of a personal workstation, of course, the user desiring a change need consult no one before making it.

The performance advantages mentioned above stem from exploiting the parallelism available in a network of many machines. The simplest technique to increase a network's performance is just to include more users—each running in parallel with the others. A more complicated method calls for splitting a single application up into many pieces, each of which can run on a separate machine. Given the current state of the art, there is no automatic way of accomplishing this task; the creator of an application must explicitly program it to be executable in parallel. Once the decomposition is accomplished, though, the system can help solve the problem of allocating machines to run the computation.

THE ADVENT OF PERSONAL WORKSTATIONS

Although we have concentrated thus far on the general topic of distributed resource sharing systems, there is one particular form of distributed systems that deserves special attention: the personal workstation network. They are the outgrowth of work done by researchers at Xerox PARC, MIT, and CMU who envisioned in the mid-to-late 1970s that computing would be done differently in the 1980s and 90s. True to the researchers' projections, more and more people have turned from timesharing machines to the high-speed local area networks that link workstations together.

From Xerox PARC have come the Alto and Dorado workstations, as well as the Ethernet network; from MIT have come the Lisp Machine and ChaosNet; and from CMU has come a proposal for the SPICE (Scientific Personal Integrated Computing Environment) project. It's interesting to note that the SPICE project proposal includes the telling phrase, "the era of time-sharing is ending." There are many outside of CMU who would agree.

A typical personal workstation, or node, in such a system has three major features: a processor with high computing power and a large virtual address space; a high resolution graphics display subsystem; and a high speed local area network connection.

> The only thing predictable about many timesharing systems is that they will slow to a crawl.

BOOSTING INDIVIDUAL PRODUCTIVITY

A fast CPU will provide predictable, fast response—even at 3 pm, when the only thing predictable about many timesharing systems is that they will slow to a crawl. When this response time is consistently less than one second, productivity can increase significantly. Much of the enthusiasm for workstations turns on this very point.

A graphics subsystem on the backplane with the CPU can provide the user with communications capabilities that are orders of magnitude better than those that a 9600 baud line can provide. A powerful CPU, a large address space, and robust graphics combine to enable the workstation to run large, sophisticated, highly interactive applications tailored to the task at hand. In addition, multiwindow user environments make it possible to execute several such tasks concurrently in a non-preemptive way. (See [3] for a survey of editing with and without windows. Section 1.8 describes integrated window environments.)

A good user interface toolkit, moreover, can help with the otherwise daunting task of creating applications that have high-quality graphic interfaces. Such applications can be quite demanding, using nearly all the workstation's resources; hence, a pagable operating system kernel is desirable, so that infrequently used OS functions can be paged out, freeing memory for use by the applications. In this fashion, the CPU, OS, and graphics hardware software combine to optimize the productivity of the individual.

RESOURCE SHARING FOR GROUP PRODUCTIVITY

The network is the pathway to optimal group productivity, since it is the means by which workstation users can cooperate effectively. When correctly implemented, networks provide the unified, comprehensive computing environment users need in order to share (and safeguard) information, programs, and peripherals with the same sort of ease, efficiency, and reliability generally characteristic of timesharing systems. (See [1,2,4] for examples of integrated network systems such as this.)

Convenient information sharing requires that the system support location-transparent file access protocols rather than ARPA-style file transfer protocols. This is because location-transparent file access allows users to get at

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. 292 on Inquiry Card

remote files as easily as they can access local files. Accessing a remote file under the ARPA-style file transfer protocol, by way of contrast, requires that the file first be copied to the local node, modified, and then replaced. Convenient sharing also requires a uniform namespace in which filenames must always refer to the same file, meaning that every file in the network must be unique. Without a uniform namespace, it would be cumbersome to exchange programs or shell scripts containing filenames because of the possible need to translate them before use.

Efficient sharing demands a network of approximately the same speed as a hard disk, as well as protocols efficient enough to effectively utilize such speed. (See [5] for a survey of local area networks.) Reliable sharing requires concurrency control to arbitrate simultaneous access to files. In this way, users can be assured of predictable results. (The DFS distributed file system for Xerox PARC [6] provides a good example of special facilities for reliable sharing.)

Although personal information is commonly stored in files, group information is often stored in databases. Thus, a good distributed DBMS is needed if networks are to be as conveniently usable as centralized systems. Like a good file system, a DBMS should provide location and concurrency transparency (although the concurrency control should probably allow a higher level of concurrency than would typically be expected of a file system.) In addition, a DBMS should offer replication and failure transparency. Replication transparency allows multiple copies of the database to be manipulated as if they were all combined into a single copy. Failure transparency means that operations on the database are reliable even if some or all of the nodes involved in the operation (or the network) fail. (Citation [7] discusses these concepts in more detail.)

Information is not the only resource in the network to be shared. Others include printers, magtape drives, and communications gateways (such as X.25). The standard means of making these available to all the nodes on the network is to construct a server, a process that runs on the node to which the resource is physically attached. This server can then service requests from other nodes to access the resource in question. Requests, which can come from any node in the network, are sent to the server via the network's interprocess communication (IPC) facility.

UNIX alone is not sufficient for the support of integrated networks of personal workstations.

Other resources that can be shared include idle workstations lying about in the network. If the good news is that workstations don't slow down at 3 pm, the bad news is that they don't speed up at 3 am, either—unless a server keeps track of workstations that are idle and allocates them to computations that can use the extra CPUs. This can be very useful in the instance of jobs specifically programmed to execute on multiple machines in parallel, or perhaps batch subsystems set to queue non-parallel jobs for later execution on idle machines. Note that this all raises a potential autonomy conflict: if the system becomes over-zealous in efforts to utilize spare capacity, it might try to re-allocate a node—shortly after its user turns away to answer a phone call. Under such circumstances, the predictability of response time would disappear.

Another aspect of group productivity to consider involves network administration. A network user registry enables users to identify themselves quickly to all machines in the network without having to go through the pain of maintaining a separate database of users on each machine. Network troubleshooting and maintenance tools are also necessary since the network is the most vital part of the overall system.

UNIVERSAL INTERCONNECTION

Just as personal workstations are more useful when connected with other personal workstations, they are even more valuable when integrated with the other types of computers. Not everyone needs the power of a personal workstation. Some can make do with a PC, while others need facilities personal workstations simply can't provide (like those afforded by a Cray 1, for example). What's more important is that all the various units are connected. A network's utility takes a quantum jump when everyone working on a problem is included. The level at which these users are interconnected is also important: the network must be more integrated than the ARPAnet if it is to be most effective. Thus, a network should interconnect all the PCs, workstations, and mainframes of the organization it serves, even if all the machines are running different system software.

The late 1980s will see networks with 10,000 nodes that incorporate all the computing resources of entire organizations.

What's more, as long haul communications speeds go up and costs drop, efforts to extend the intimacy of today's local network environment to geographically dispersed computing environments will surface.

SYSTEM DESIGN CENTER AND HERITAGE

Before running out to select a resource sharing network, consider the design center and heritage behind it—as well as the features within it. To make an analogy, almost any two makes of automobiles have the same features: four wheels, an engine, a steering wheel, brakes, and so forth. Nevertheless, the differences between a Mercedes and a Chevette are obvious to nearly everyone. The respective design centers behind the two cars explain the differences best: the Mercedes was designed to be a high-quality, powerful, luxury sports sedan, while the Chevette was intended for cheap transportation. Going one step further, it's important to note that a manufacturer that has only made luxury cars will face a steep learning curve if it wishes to become a successful manufacturer of low-cost cars (just as the converse is true).

The best way to illustrate this in the networking case is to look at some examples, taken from the point of view of a personal workstation network. A system designed originally to operate a machine room network intended to contain a maximum of a dozen or so machines might make use of algorithms that do not scale well to a network of hundreds of personal computers. Other examples can be found of single-process PC operating systems that have had considerable difficulty being stretched to support a multiprocess multiwindow user environment. A former timesharing system, with its emphasis on the efficient support of many users at dumb terminals, would similarly have a long way to go if it was to provide a highly interactive graphics user environment.

A litany of other examples could be trotted out. For all of that, though, a system's design center and heritage never present an insurmountable obstacle. All systems migrate over time, after all, but vestiges of them often linger. Thus, looking at a system's history can often tell you where its weak (and strong) points lie.

THE UNIX ROLE

Where does UNIX fit into all of this? In the 1980s, support for the UNIX program environment is essential for applications portability—just as Fortran was essential in the 1970s. The UNIX environment is much richer than that of Fortran, though, and it is thus capable of supporting far more complex applications without sacrificing portability. UNIX, therefore, is likely to replace Fortran as the standard vehicle for supporting applications if it can itself ever become truly standardized.

Despite these credentials, UNIX alone is not sufficient for the support of integrated networks of personal workstations. Its heritage does not include either graphics or networking; hence, even today, neither System V nor 4.2BSD support integrated graphics or windowed user environments. As a matter of fact, System V doesn't support net-

Continued on Page 90

DISTRIBUTED FILE SYSTEM STRATEGIES

The options and their implications

by G. R. Sager and R. B. Lyon

A major drawback of using more than one computer to work on a problem requiring the cooperation of two or more individuals is that few systems support transparent sharing across machine boundaries. Rather than applying the tools and procedures they find useful when operating on the same machine, cooperating users on different machines must often resort to extraordinary means to accomplish tasks. Furthermore, users typically cannot move easily from one system to another and still have access to their own environment and files, so the mere availability of accounts on all the machines working on the problem provides no solution.

One step that moves toward a solution is the extension of the concept of a file system onto a network to allow transparent shared read/write file access by machines on the net. In this article, we discuss some of the important issues in the design and implementation of such a file system. There is not enough space to consider all of the issues and alternatives, so our approach is to present several of the major issues, look at them from a UNIX point of view, and offer specific examples taken from Sun's Network File System (NFS).

As an aid to the presentation, let's first look at some terms and concepts that will be used: a *server* is a machine that provides file system resources to a network; a *client* is a machine that accesses file system resources over a network; a *user* is a person "logged in" at a client; an *application* is a program or set of programs that execute on a client (usually on behalf of a user); and a *workstation* is a client machine that typically supports one user at a time.

DEFINING THE INTERFACE

An obvious way to define the file system interface on the network is to extend the operating system semantics onto the network. The primary advantage of such a *distributed* approach is *transparency*: the entire seman-

Illustration by J. Kell Davies

tics of the file system are preserved intact. A stumbling block, though, is that most extant operating systems (including UNIX) were not designed to be distributed, giving rise to a number of problems we will discuss later.

The *services* approach offers an alternative: simple, well-defined services are provided to the net by server machines, and are accessed from client machines by users and applications. This approach emphasizes the interfaces presented to the net, rather than the implementations at the endpoints, resulting in an *open system* that can be used by a variety of operating systems and machines.

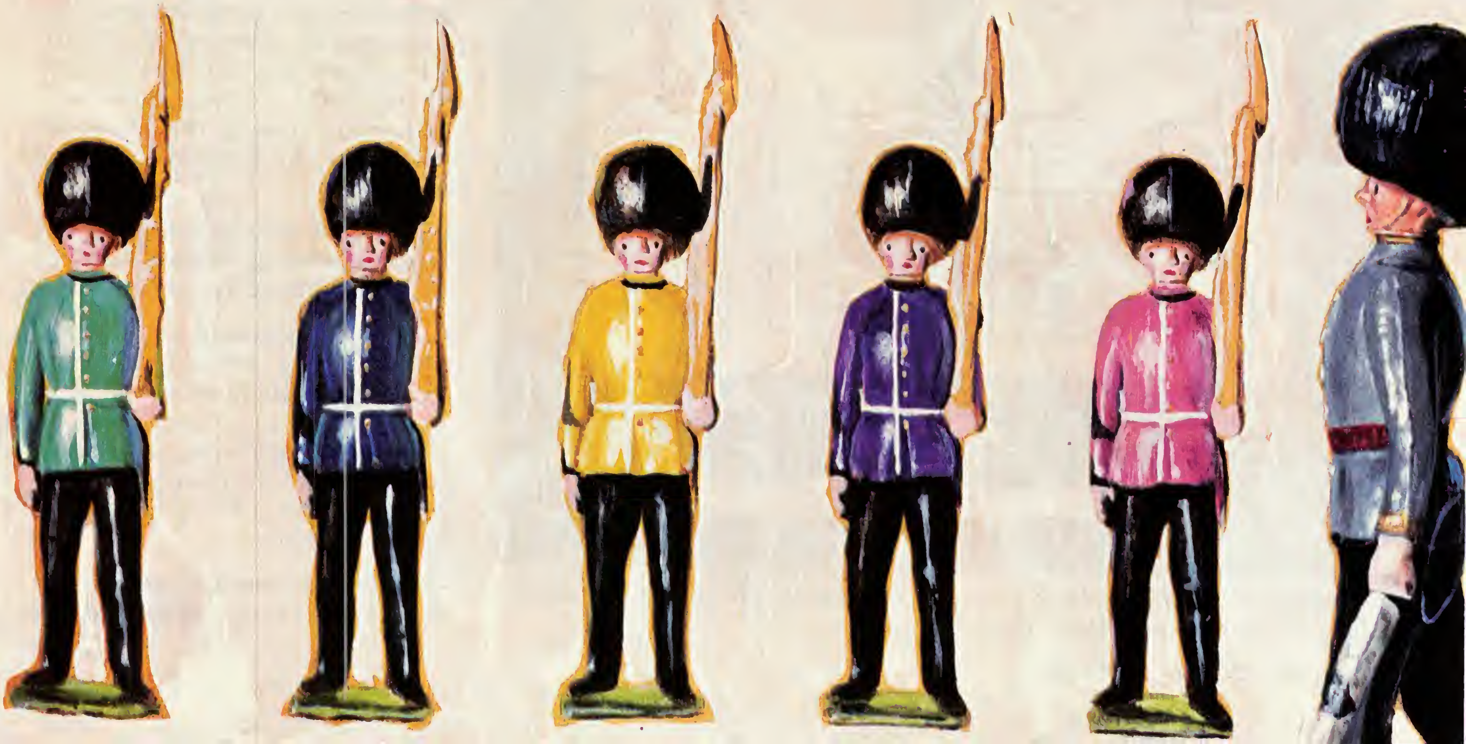
The NFS, for example, defines a generic file system protocol using the Remote Procedure Call (RPC) and the eXternal Data Representation (XDR) [1,5] package to present the protocol to the net in an operating system and machine-independent way. Thus, it is possible for a variety of operating systems or machines to share files across the net as clients or servers. Since one advantage of the services approach is its ability to connect unlike systems, it is important to keep the semantics of the file system service general. Simplicity makes it easier to implement client and server interfaces; low entry cost encourages future development of a rich variety of client and server systems.

> The primary advantage of a distributed approach is transparency: the entire semantics of the file system are preserved intact.

The disadvantage of the services approach is that transparency may be more difficult since users or applications may become aware that service from the net differs from similar service available locally. In the case of the NFS and a UNIX client, the client operating system provides the layer between the net and the user or application so that transparency is maintained.

ACCESSING THE INTERFACE

Introducing remote file system access into an extant operating system is made more difficult when the operating system defines and controls its own file system. UNIX presents a classic example of this problem. In Sun's UNIX implementation of the NFS, the problem is solved by cleanly separating file system operations from the semantics of their implementation (Figure 1). This clean separation is known as the *vnode* interface. Above the *vnode* interface, the operating system deals in *vnodes*, while below the interface, the file system may or may not implement *inodes*. From a *vnode*, the operating system uses the *vnode* interface to connect to a virtual file system (VFS). A VFS is in many ways analogous to a device driver: there is a well-defined interface to the rest of the operating system, and within that



interface, a great variety of devices can be supported.

A local VFS connects to file system data on a local device. The remote VFS defines and implements the NFS interface. The remote VFS uses the RPC mechanism to access the NFS server. RPC allows communication with remote services in a manner similar to procedure calling mechanisms available in many programming languages. NFS and RPC "high-level protocols" are described using the XDR package. XDR permits machine-independent representation and definition of high-level protocols on the network.

The arrows of Figure 1 show the flow of a request from a client (upper left) to local file systems (lower left) and to a file system on a server (lower right). In the case of access through a local VFS, requests are directed to file system data on devices connected to the client machine. In the case of access through a remote VFS, the request is passed through the RPC and XDR layers onto the net. On the server side, requests are passed through the RPC and XDR layers to an NFS server, which in turn uses the *vnode* interface to access a local VFS and service the request. This path is retraced to return results.

PERFORMANCE

A remote file system with poor performance will not gain wide use or acceptance, despite what other advantages it might offer. Therefore, performance must be considered from the beginning; some useful design guidelines are:

- move work from the server to the client whenever possible.
- allow client caching of data blocks, especially for read-ahead.
- allow short-term caching of file status information on the client.
- avoid excessive locking and synchronization requirements.
- minimize reliability and recovery overhead, especially for the servers.

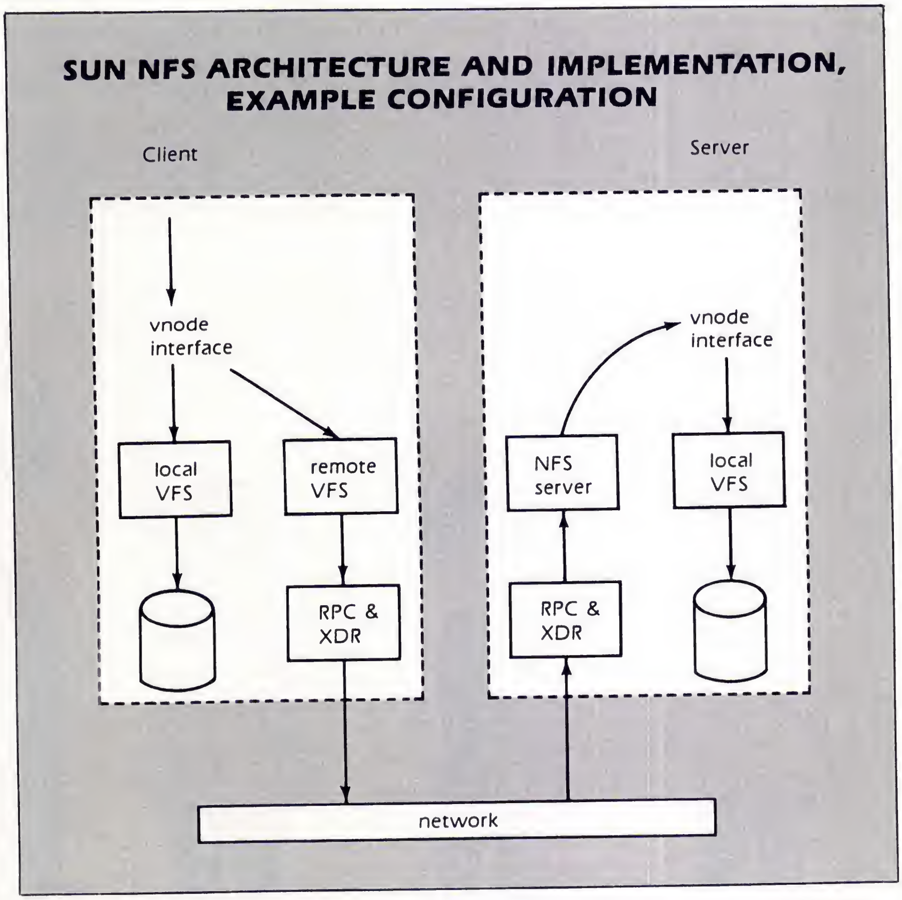


Figure 1 — *An illustration of one way to separate file system operations from the semantics of their implementation.*

FILENAME SYNTAX AND SEMANTICS

Remote filenames should look like names provided by the client operating system; thus, a UNIX client should use the UNIX pathname syntax and semantics, and access to remote files should come through a **mount**ed file system.

A less obvious design decision comes in the division of responsibility for pathname interpretation—that is, when a client encounters a **mount** point that goes onto the net, does it pass the remainder of the pathname to the server for interpretation or does it continue to interpret the pathname itself, element by element?

Important advantages are gained by having the client do pathname interpretation. They include:

- the server may be running a different operating system and use different syntax for pathnames.
- a client can **mount** a private directory on a remote directory. This has transparency implications, since applications requiring private spool areas would otherwise have to be rewritten to run on a "distributed" version of the operating system.
- a client can **mount** a remote file system in a remote directory. This

means that the client, rather than the server, takes responsibility for maintaining information related to the file tree it sees on the net.

- complex service arrangements involving more than one server and client are prevented, simplifying performance, recovery, authentication, and deadlock properties of the network.

INPUT/OUTPUT REQUESTS

There is a choice between three basic options that a file I/O request can exercise: it can ask for an entire file, blocks, or bytes.

Asking for an entire file means the file is transferred to the client when it is opened, and transferred back when it is closed. This implies that the client has enough local storage to hold the file, which can be a problem for a small diskless workstation that's dealing with large files.

Block-level requests mean that the block size is defined to be part of the interface presented to the network, and that blocks are transferred as a part of read and write operations. Most UNIX systems can be configured to use a new block size and some deal simultaneously with file systems using different block sizes. Other operating systems may have a preconceived notion of block size and find it difficult to deal with the size presented by the server.

The third choice, which the NFS uses, is to employ byte-level requests. This means that the server is willing to accept requests starting at any byte in a file and running to a length of anything between 1 and some maximum number of bytes (defined by each system). Byte-level naming presents the greatest flexibility to the client since it can simulate either of the previous methods. Of course, requests of certain sizes

server and the packet sizes on the network.

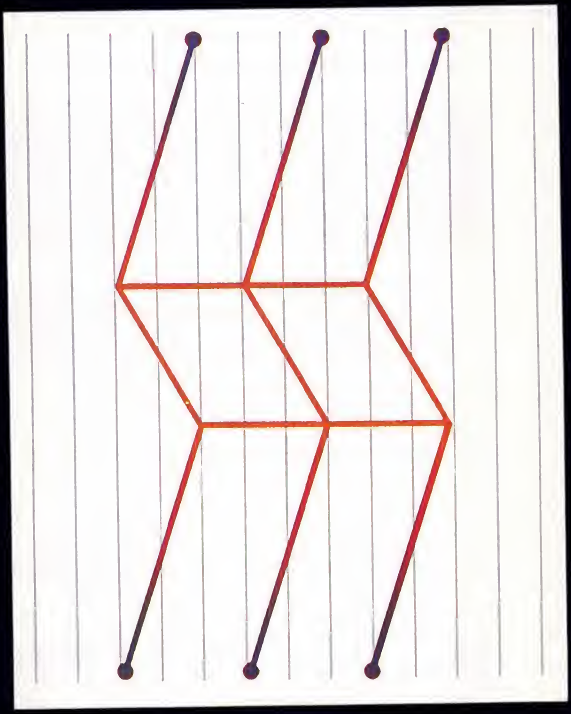
The choice of request size also has implications for synchronization. For example, byte-level locking is problematic if the interface requires file or block-level transfer. This is because the unit to be locked is smaller than the basic unit the interface deals with.

> **The services approach emphasizes the interfaces presented to the net, resulting in an open system.**

RELIABILITY AND RECOVERY

Most popular operating systems are written to run on a single computer, and they tend to assume that they know and control everything within their domain. As a result, use of the distributed approach to extend an operating system or any substantial part of its functionality onto a network requires a great deal of consistency and synchronization (*state*) among participating machines. With care in design and implementation, such a system can perform well in normal circumstances. However, it is unlikely that it can perform well in the face of failures. As the number of machines increases, it becomes more likely that at least one will experience difficulties. Furthermore, the effort required to maintain and recreate state across a failure and recovery of a client or server increases with the number of machines. Thus, in a large net, substantial amounts of computing resources will be spent in recovery activities.

With a distributed approach, there is an implicit trust between systems that state will be properly maintained. In the case of a workstation environment, this is not a safe assumption, as the machines



often are not under the control of a disciplined operations staff. Also, the interface is made more complex by the need to maintain and recreate state. Hence, the implementation of a client or server becomes more difficult. These two facts may ultimately eliminate PCs as clients.

The NFS approach defines the interface such that file servers are *stateless*. This means that servers do not remember from one transaction to the next anything about their clients, the transactions they completed, or the files they operated on. For example, there is no NFS *open* operation, as this would imply that servers remember what files are open. Of course, the UNIX client interface uses an *open* operation, but the information gained in the operation is remembered only by the client for use in later NFS operations.

The major advantage of a stateless server is the robustness it displays in the face of client, server, or network failures. Should a client fail, it is not necessary for a server (or human administrator) to take any action to continue normal operation. Should a server or the network fail, clients can retry NFS operations until the server or network is fixed. Once the server or network becomes operational, clients may resume operation as though no failure occurred.

A stateless interface should also support *idempotent operations*. This means that applying an operation more than one time has the same result as applying it only once. Thus, retrying "failed" operations is safe. For certain operations, such as *read* or *write*, idempotence is easy. Others, such as *remove* or *rename* are difficult, but engineering approximations can be obtained by having the server keep a cache of recent operations to recognize retries and by having the client ignore certain error returns on retried operations.

On the negative side, it is difficult or impossible to provide stateless versions of features like locking or special files. This complexity is required by very few applications, and it is often the case that complex features are "not quite right" for any given application. The NFS approach is to provide these types of services as companion "cafeteria-style" offerings so that only those customers who require them will pay the extra cost in performance and complexity. Furthermore, these services can be built using a "toolkit" philosophy so that customers can easily tailor features to fit their own requirements.

> With a distributed approach, there is an implicit trust between systems that state will be properly maintained.

ADMINISTRATION AND MAINTENANCE

If one takes the view that the distributed file system is being used to join extant systems, then it is natural to propose that each system provide a mapping from other systems to its own view. For example, the */etc/passwd* file on each system can be made to map a remote **uid** and **gid** to a local **uid** and **gid**.

This quickly becomes a nightmare to administer; when a new user is added to the collection of systems, an */etc/passwd* entry must be made on every system. And having different numeric **uid**'s and **gid**'s for the same user causes difficulties when that user moves from system to system.

The NFS approach is to "flatten" administrative data, so that the cost of obtaining sharing can be paid just once at the beginning. Since many multicomputer installations already attempt to maintain common administrative data for their systems, the effort required may be small.

In order to ease the task of administration, a companion service called the *Yellow Pages* (YP) is provided with the NFS. From the point of view of the servers and clients, the YP is a centralized read-only database. For a client, this means that an application's access to the data served by the YP is independent of the relative locations of the client and server.

The YP is a collection of cooperating server processes that use a simple discipline to distribute data among themselves. Thus, the servers share the load of providing access to data, and the failure of a server need not disable the network. The YP does not implement a true distributed database since, for every relation in the database, one YP server is designated to control the update of data for the entire collection of YP servers.

Thus, the administration of an entire network of servers and clients is done from a single point of contact. Should the control server fail, an alternate server can be designated as the control. The policy for distributing changes through the network yields a weak form of consistency: the databases across the network will be consistent after a "reasonable" time has elapsed. A system administrator can have changes distributed periodically according to a schedule, or can have them

Continued to Page 94


HELLO

THE UNTAPPED POTENTIAL OF REMOTE PROCEDURE CALLS

A Courier of good tidings

by Steven F. Holmgren

Though the use of remote procedure calls is not yet widespread, these mechanisms for the remote execution of software offer a rich form of system interaction. Those who have RPC capabilities can execute remote software on a subroutine-by-subroutine basis. This allows the user to obtain finite data or data descriptions by calling remote subroutines with a finite series of parameter data in a specified execution environment. Among the tasks accomplished by the calling procedure are the packaging of the parameter data, the establishment of a remote execution environment, the transmission of the packaged request, the receipt of the completed execution, the unpackaging of the results, and the delivery of those results to the initiator of the call.

Unfortunately, hype from network vendors about the generic capabilities of their products has tended to cloud common perceptions about remote procedure call mechanisms, and networking in general. Broadly speaking, networking today offers the ability to exchange text-based files and emulate remote terminals. From a user's viewpoint, this is hardly an easy environment to work within. It forces users to understand the topology of their local network; the location, format, and sizing of any interesting data within that network; and the command syntax and sequencing of whatever operating system might be necessary to manipulate the data. Moreover, the user of today's typical network product must have some form of access authorization to get at data in the first place. This may require users to have a number of different login identifiers, with passwords that could change monthly.

This sort of arrangement simply isn't the "tie all your machines together" panacea that many users might have thought they had purchased. The user is not usually chomping at the bit to jump into the middle of the file transfer process or to provide remote password information; rather, what the user wants is to acquire whatever applications, data, or software is necessary to get the job done.

The remote procedure call model offers vastly increased flexibility in processing system interaction. It has been long held as an ideal for communications developers to work toward. With an operational remote procedure call capability, users can actually select from a variety of remote machine services (such as **sort**) instead of chafing under the constrictions of the limited set of file-by-file or remote terminal interactions that are commonly found in "modern" networking products.

Subroutine call and return processing typically takes a few milli-

Illustration by Amy Lanset

seconds in a modern processor. In the remote procedure call environment, parameter encode/decode processing and transmission overhead can take as long as 500 milliseconds. This performance differential has historically made it impractical to use remote procedure calls for system interaction. The recent advent of 10 mbps Ethernet communications and high speed 32-bit microprocessors have created a set of processing economics that allow users to communicate data reliably between applications at over one million bits per second. This processing and transmission performance has led to a renewed interest in remote procedure call models of system interaction.

Because Xerox became one of the earliest progenitors of readily available high-speed, local communications when it first offered a 2.94 mbps version of the Ethernet, it is not surprising that some of the best known work in the area of remote procedure processing stems from the Xerox Network System (XNS) family of protocols.

The remainder of this article will develop a reference model for remote procedure call protocols, discuss the XNS Courier model, and finally suggest two evolutionary paths that the technology might follow.

REMOTE PROCEDURE CALL REFERENCE MODEL

The overhead required to process a remote procedure call is significant, and the performance of that overhead processing has a lot to do with the success or failure of a remote call implementation. A brief look at the component steps of a remote call should lay out the scope of the problem and serve as a framework for future discussions.

1) establish an execution environment.
2) request procedure and parameter encoding.
3) request transmission.
4) request procedure and parameter decoding.
5) execute task.
6) encode reply.
7) transmit reply.
8) decode reply.
9) tear down execution environment.

EXECUTION ENVIRONMENT

The establishment and subsequent destruction of an execution environment is required for the management of a reliable, consistent communications path between the initiating procedure call environment and its companion execution environment. At minimum, this requires the construction of a reliable communications circuit from the initiating network node to the executing network node. Data for this purpose includes: a network name, address, and route to set parameters for a network connection request. Once remote service availability has been established, authentication information is required (a username and password, for example). Finally, user profile information, such as pathname search rules or name alias information, needs to be exchanged if the execution environment is to be properly conditioned for the procedure calls themselves.

> **The remote procedure call model offers vastly increased flexibility in processing system interaction.**

Each implementation and some protocol specifications require a policy decision about when to establish and tear down an environment. The choices include: 1) a high overhead policy in setup and teardown after each request, 2) the establishment of a global remote environment whenever any execution is taking place, and 3) the destruction of environments on a least-recently-used time basis—in effect caching other environments under the assumption that they might be needed.

PARAMETER ENCODING AND DECODING

When communicating with a different breed of systems, it is important to express binary and character information in a consistent, standard fashion. Typically, this involves the creation of a data structure that describes the parameter type (*binary character*, *binary integer*, or *complex*, for example) followed by the actual parameter value. Complex information concerning the length of a parameter (such as the size of a character string or the number of array elements) is also encoded in a prepended field followed by the parameter itself (or, in some cases, by a series of parameters). Various forms of encoding and decoding software exist to transform arbitrary information into an encoded string or to take an encoded string and produce localized versions of the decoded parameter information.

In a technology where end-user performance is strongly associated with end-user acceptance, it is important to realize that the encoding and decoding process is extremely time consuming. Further, since the process is performed four times for each remote procedure call, it should be noted that its overhead is magnified and that particular attention needs to be paid to optimizing the software that implements it.

REQUEST AND REPLY TRANSMISSION

Transmission bandwidth is also critical to the overall performance of a remote procedure call system. Procedure call performance must fall within some reasonable time boundary or users will find other ways to accomplish their work. It is suggested that 10 mbps communication channels be the minimum performance communication path in any commonly used environment. Further, signaling bandwidth must be coupled with a high-speed network processing engine in order to ensure reliable data communications on a packet-by-packet basis between network nodes. As a rule of thumb, application level performance must approach a one million bit per second data rate in order for general response communications to take place.

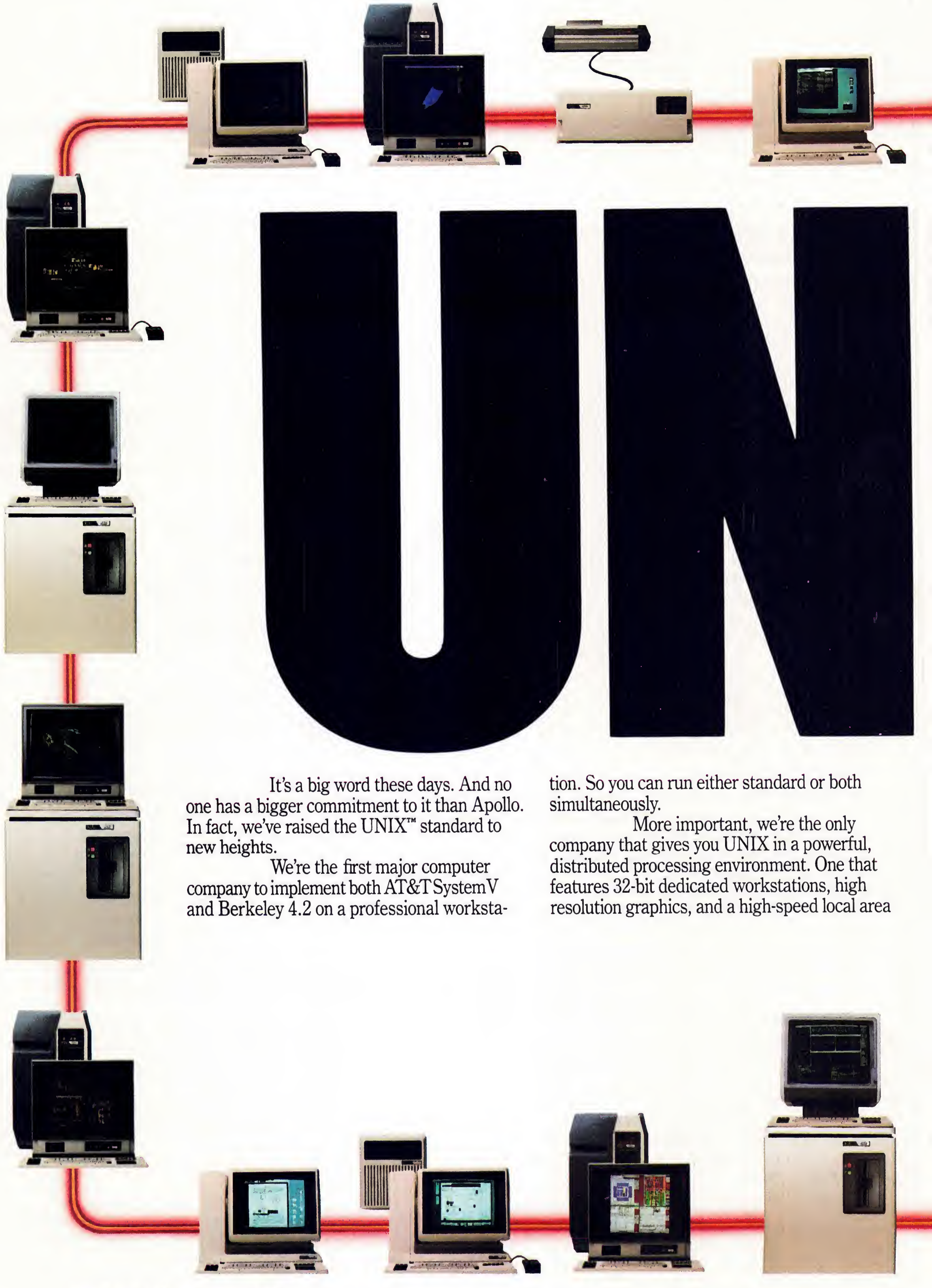
COURIER: THE REMOTE PROCEDURE CALL PROTOCOL

The Courier defines remote programs that are uniquely numbered. Each program contains a series of procedures that may be invoked as a service to network users. Within each remote program, remote procedures are uniquely numbered and the appropriate return and error values are defined.

Environment management is handled by the creation of a reliable data stream at the beginning of a "session". The data stream is kept open until the session ends. Reliable data stream communications are supported by the Xerox Sequenced Packet Protocol.

A formal grammar specifying Boolean, cardinal, integer, long integer, string, and untyped data types handles the parameter encoding and decoding. In general, Boolean data types are single-bit entities in a 16-bit field, cardinal types are 16 and 32-bit unsigned binary entities, integer types are 16 and 32-bit binary entities, and strings are delineated by a 16-bit field followed immediately by 8-bit characters. Given these basic data types, a number of "constructed" types can be defined to allow for arrays, enumerations, sequences, records, and choices. These synthetic types are combinations of the basic data types with a different organization. Using this scheme, all parameters, procedure identifications, and return values are virtualized. Even though attention has been paid to the need for minimizing the encoding and decoding process, there is still significant overhead in support of what Courier names the *object layer*.

In addition to the object layer, Courier defines a message stream layer to add context to the data. The message stream contains




IX

four data types: *call*, *return*, *reject* and *abort*. The *call* and *return* types are used as part of the normal processing of a procedure request and reply. The *reject* request indicates a system-level failure in the processing of a request, and *abort* is used to indicate programmatic level errors.

Operationally, Courier sets up a Sequenced Packet Connection, selects a unique number to represent the remote program to be called, and selects a unique number to represent the remote procedure to be called within the remote program. It then encodes the parameters and issues a *call* request. A *return* reply will then be received with any associated data.

The unique program number is assigned in much the same way that unique Ethernet addresses are assigned. The unique procedure number that is assigned within a remote program is a 16-bit value selected by the implementor. If there is one particularly common criticism of the Courier, it relates to this manual assignment of a unique number to all programs. The practice is still conceivably workable, but given that most of the software in the world has been generated during the last 30 years, one wonders how many "billions and billions" of remote programs will be sold and thus need to be numbered over the next 30 years.

Courier is fairly basic in that it uses a single procedure call-per-interaction and accomplishes remote processing by grouping remote procedure calls. Procedure selection and parameter typing are made as efficient as possible on a call-by-call basis so that any real improvements in the practical implementation of the protocol must come from superior communications line performance, encoding performance, or execution performance.

There are, however, global optimizations that could be made by extending the protocol to add global intelligence to each procedure request. In this way, the number of request and reply transactions can be reduced without dampening the level of remote processing.

PROCEDURE CONTROL PROTOCOL

One of the first ways in which the protocol can be extended is by adding remote evaluation of procedure return data. This should occur before the data is encoded for return, so that the initiator might have an opportunity to modify the parameters and reissue the call at the execution site. Constructs to do this could be patterned after modern program command languages such as the standard UNIX shell. Simple capabilities such as *while*, *do*, *if-else*, and the remaining set of industry standard program control modifiers would thus become as applicable to the control of remote processing as they are today to the semi-automatic handling of commands in a generalized timesharing environment.

DISTRIBUTED PROCESSING PROTOCOL

In the Courier protocol, remote procedures are selected from a fixed list of procedures. The list is organized manually by remote procedure numbers under each remote program. There are a number of ways to make this manual process more flexible.

National's Series 32000™: First In Design, First To Production, First To Market.

Now You Can Be The First To Build On It.

You're a software innovator with extensive UNIX™ experience. You want to work with emerging technologies. And the only opportunity you're interested in is the opportunity to make a difference.

You belong at National Semiconductor's advanced product development center in Portland, Oregon. We're developing a UNIX operating system performance-tuned to Series 32000-based board-level computers. Then we'll expand it to include graphics and network support within the AT&T System 5 framework.

It's one of software's most exciting challenges. To meet it, we need software development professionals who possess a BS/MS in CS or EE, or equivalent, plus at least three years of experience in one of these areas:

UNIX Internals

You will specify hardware design and implement UNIX on new hardware.

LAN Implementation

Experience designing and developing Ethernet systems, plus implementation knowledge of TCP/IP, ISO protocols important.

Graphics Kernel Development

You will define and implement hardware to support GKS or ANSI CORE graphics kernels and applications.

Utilities

You will port UNIX utilities to new hardware and software.

Diagnostic Programming

Work with design and test engineering to write user-friendly hardware diagnostic programs.

CAD Support

Put your Pascal programming experience to work producing programs to make CAD equipment easier to use.

Software Support

General OS experience plus knowledge of UNIX operating system needed to develop operating system, networking and graphics packages.

The Pleasures of a Start-Up Without the Pains of Growing Up.

What can we offer you? High energy. Top rewards. Unsurpassed technical resources. In fact, all you expect from an invigorating start-up environment. Plus the added benefit of solid support from National Semiconductor — a technological leader for nearly twenty years.

To apply, send your resume to NATIONAL SEMICONDUCTOR, Portland Development Center, Professional Employment, Dept. UR, 15201 S.W. Greenbrier Pkwy., Beaverton, OR 97006.

*UNIX™ is a trademark of Bell Laboratories.

Together, We'll Do Even More.

National Semiconductor

Circle No. 266 on Inquiry Card

The use of globally distinguished program and procedure names instead of numbers is one such way. One can also add naming domains, regional administrations, and sub-hierarchies to the naming process. For all this, though, the user is still essentially limited to selecting an existing remote procedure from a pre-defined list.

There is a larger optimization possibility that extends the list of pre-defined procedures to include the execution of whatever data is included as a parameter to the request itself. In effect, this sends a procedure to a remote system as part of a *call* request and lets the remote system support its execution with the appropriate parameters, data, and resources. Some form of programming language would need to be included in each remote procedure call request to make this approach work. The flexibility derived from such an organization and the possibility of simultaneously processing operations would substantially enhance the coupling of processing resources.

The range of possible applications involving a series of distributed machines and a distributed procedure protocol exhausts the imagination. For example, an application might allow a user to formulate a database request that searches through the database for selected data, isolates the selected data, and returns a reply when complete. A user could also send requests off to a series of machines, each processing in parallel. As the replies were returned, the user could then integrate the associated data into its final form. An alternative application might involve the transmission of a text editor to the data so that data modifications might be performed "on site".

One final application could include the use of a program capable of transmitting itself to a series of different sites for the purpose of gathering opinion data from users. While moving from one site to another, it could issue intermediate results and status reports about its progress. When such a program had polled all of its users, it could return to its initiating site to report final results.

CONCLUSION

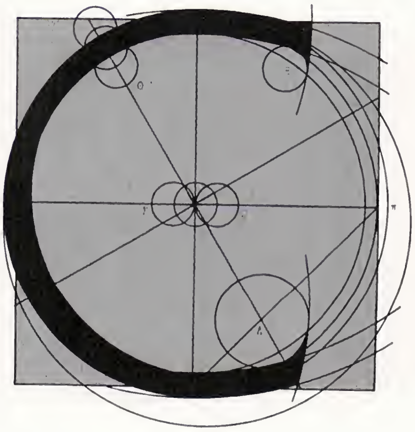
Today, resource sharing is most widely supported by text-based file exchange and remote terminal interaction. The remote procedure call concept offers a much

Continued to Page 92

C ADVISOR

Remote procedure calls

by Bill Tuthill



Last month's column was a discussion of how 4.2BSD sockets permit communication between processes on the same machine, and how they can link processes on different machines. As demonstrated by the examples, sockets are not easy to use, though they may be efficient to implement. The remote procedure call (RPC) mechanism is an alternate means of inter-process communication. Since RPC is easier to use than are sockets, and since it has been implemented with sockets, one could say that RPC is higher-level.

Because RPC is new to most system developers, it's hard to predict what kind of popularity it will achieve. An important early RPC implementation was Courier, used in Xerox Network Systems. This article discusses a recent implementation by Sun Microsystems, heavily influenced by Courier, but based on UNIX.

In the RPC model, programs call a procedure that sends data packets to a server, where a dispatch process services incoming requests. When a request is completed, the server process sends back a reply, and the procedure call returns to the client program. Figure 1 shows what sockets look like; Figure 2 shows what RPC looks like.

At the simplest level, programs can make RPC calls as if they were normal procedure calls on the local machine. In order for this to work, the system has to be furnished with library routines such as **rnusers()**, which returns the number of users on a given host. Figure 3 shows how to obtain the number of users on any machine connected to the network. If a system hasn't been furnished with the proper routines, things are a bit more difficult. First, it's necessary to set up a daemon on the server machine, and register the required routines. Remote procedures are registered with a 96-bit quantity that encodes program, version, and procedure number. This way, when it receives a remote procedure call, the daemon knows what routine it should call to service the request.

For example, suppose there is a remote database, structured like */etc/passwd*, containing information about all users on the network. Figure 4 shows how the daemon registers the service. The **registerrpc()** call establishes a correspondence between a given C routine and a particular RPC procedure number. In this example, we used the assigned program number PWPROG, version number PWVERS, and procedure number PWPROCNUM. These numbers are not cast in concrete, but server and client must agree on their values. The final two parameters are input and output routines for packaging network data going to the **netpw()** routine. The **svc—run()** call is a library routine that goes into a wait state in order to receive RPC requests coming over the network. It should never return, because this program runs as a daemon.

The **netpw()** routine is quite simple, calling the library routine **getpw()** to retrieve an entry from the password file, given a user ID. Because **xdr—string()** needs the address of a string pointer, rather than merely the address of the string's first element, we return the address of **pwline**, which points to the **buf** array.

After setting up a server machine, programs on the client machine can call the network service as demonstrated in Figure 5. The **callrpc()** routine takes eight parameters: the remote machine, the program number, the version number, the proce-

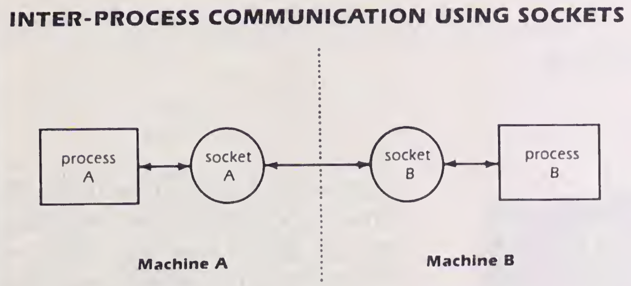


Figure 1 — *A diagram of a socket.*

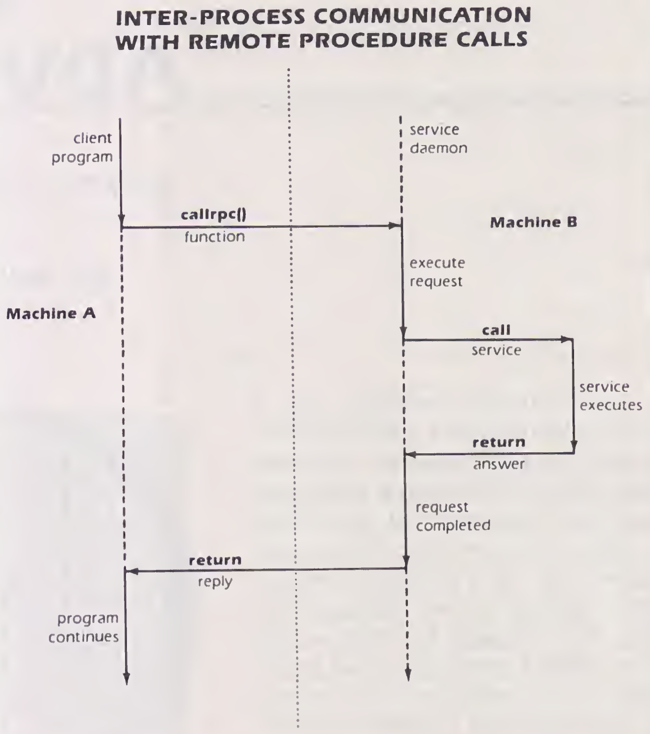


Figure 2 — *An RPC diagram.*

```
#include <stdio.h>

main(argc, argv)
int argc;
char *argv[];
{
        unsigned num;

        if (argc < 2) {
                fprintf(stderr, "usage: %s hostname\n", argv[0]);
                exit(1);
        }
        if ((num = rnusers(argv[1])) < 0) {
                fprintf(stderr, "error calling rnusers\n");
                exit(-1);
        }
        printf("%d user%s on %s\n", num, (num > 1) ? "s" : "", argv[1]);
        exit(0);
}
```

Figure 3 — *A program for obtaining the number of users on any machine connected into the network.*

```
#include <stdio.h>
#define PWPROG  0x20000000
#define PWVERS          1
#define PWPROCNUM       1

char **netpw();
int xdr_int(), xdr_string;

main()  /* register remote prodecure */
{
        registerrpc(PWPROG, PWVERS, PWPROCNUM, netpw, xdr_int, xdr_string);
        svc_run();              /* never returns */
        fprintf(stderr, "error: svc_run() returned!\n");
        exit(1);
}

char **
netpw(uid)/* given uid, return line from password file */
int *uid;
{
        static char buf[BUFSIZ], *pwline = buf;

        if (getpw(*uid, buf) != 0)
                strcpy(buf, "bad password entry");
        return(&pwline);
}
```

Figure 4 — *The way in which a daemon establishes a correspondence between a C routine and an RPC procedure number.*

```
#include <stdio.h>
#define PWPROG0x20000000
#define PWVERS 1
#define PWPROCNUM 1

int xdr_int(), xdr_string();

main(argc, argv)
int argc;
char *argv[];
{
        char *pwline;
        int uid;

        if (argc != 3) {
                fprintf(stderr, "usage: %s hostname uid\n", argv[0]);
                exit(1);
        }
        uid = atoi(argv[2]);
        if (callrpc(argv[1], PWPROG, PWVERS, PWPROCNUM,
            xdr_int, &uid, xdr_string, &pwline) != 0) {
                fprintf(stderr, "error calling callrpc\n");
                exit(2);
        }
        printf("%s\n", pwline);
        exit(0);
}
```

Figure 5 — *An example of how programs on a client machine can call for a network service after a server machine has been set up.*

dure number, the input argument type, the address of the input argument, the output argument type, and the address of the output argument. Input and output argument types are in fact routines that package network data.

One problem in network data transmission is that different machines on the network have different architectures. For example, even though the VAX and the M68010 have the same word size, the 68010 is forward byte-ordered, while the VAX is backward byte-ordered. Consider the **writer** program, shown in Figure 6, and the **reader** program, shown in Figure 7. Piping the output of **writer** to **reader** gives identical results on a 68010 or a VAX.

```
sun% writer | reader
0 1 2 3 4 5 6 7
sun%
---
vax% writer | reader
0 1 2 3 4 5 6 7
vax%
```

But if **writer** is run on a 68010, and **reader** is run from a remote shell on a VAX, the results are bogus:

```
sun% writer | rsh vax reader
0 16777216 33554432 50331648 67108864
  83886080 100663296 117440512
sun%
```

Identical results are obtained when executing **writer** on the VAX and **reader** from a remote shell on the Sun. These results occur because the byte ordering of long integers differs between the VAX and the Sun, even though word size is the same. Note that 16,777,216 is 2^{24}—when four bytes are reversed, the 1 winds up in the 24th bit.

It is evident that RPC won't work across different architectures unless data is encapsulated in a machine-independent way. The **xdr—int()** and **xdr—string()** routines in the **registerrpc()** and **callrpc()** examples above are part of Sun's external data representation (XDR) package, a vehicle for transmit-

```
#include <stdio.h>

main()                  /* writer.c */
{
        long i;

        for (i = 0; i < 8; i++) {
                if (fwrite((char *)&i, sizeof(i), 1, stdout) != 1) {
                        fprintf(stderr, "failed!\n");
                        exit(1);
                }
        }
}
```

Figure 6 — *The **writer** program.*

```
#include <stdio.h>

main()                  /* reader.c */
{
        long i, j;

        for (j = 0; j < 8; j++) {
                if (fread((char *)&i, sizeof(i), 1, stdin) != 1) {
                        fprintf(stderr, "failed!\n");
                        exit(1);
                }
                printf("%ld ", i);
        }
        printf("\n");
}
```

Figure 7 — *The **reader** program.*

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.

Choose the best. Choose MLINK.

CMI

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

Please send me the following version(s) of MLINK:

_____ Altos 486/586/986 Xenix ($250)
_____ AT&T 3B2 Unix System V ($500)
_____ DEC PDP-11 SCO Xenix ($500)
_____ IBM PC/XT/AT or compatible ($250)
_____ Send me more information

MLINK is available for many systems.
Please call for current version information.

Introductory offer - we pay shipping!
Outside North America add $15.00 (U.S.)

___ Check enclosed ___ MasterCard ___ Visa
___ COD (Registered checks or money orders only)

Card # ______________________ Exp. __________

Signature ______________________

Ship to:
Name ______________________

Address ______________________

City/State/Zip ______________________

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 is a trademark of Microsoft Corp. CP/M is a registered trademark of Digital Research.

Circle No. 263 on Inquiry Card

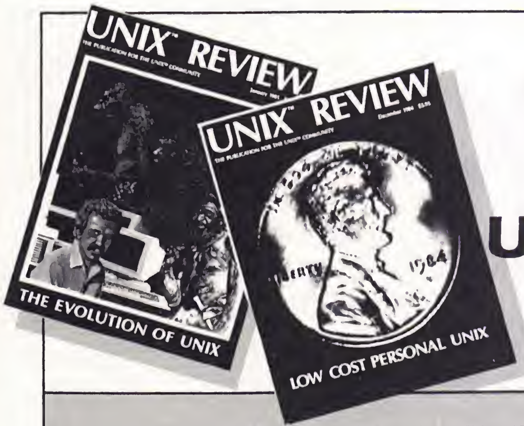
ting RPC data in a machine-independent way. Conversion from one machine representation to XDR format is called *serializing*, while the reverse process is called *deserializing*. XDR routines are bidirectional, in the sense that the same routine can serialize or deserialize data. Here is a list of some built-in routines:

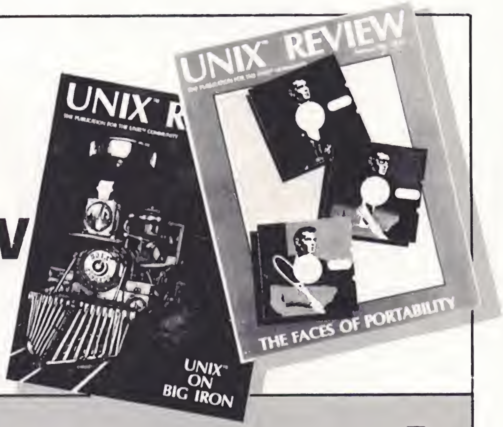
```
xdr_int()       xdr_float()
xdr_long()      xdr_double()
xdr_short()     xdr_enum()
xdr_u_int()     xdr_bool()
xdr_u_long()    xdr_string()
xdr_u_short()   xdr_bytes()
```

The routines in the left column are used for integers, long integers, and short integers (signed and unsigned). The routines in the right column are for floating point and double precision numbers, enumerated types, Booleans, null-terminated strings, and byte-counted strings. Structures and arrays can be serialized and deserialized by combining the primitives above.

The advantage of RPC is that it allows useful facilities, such as distributed databases and network file systems, to be implemented without undue difficulty. The disadvantage is that it adds a level of complexity to the system, and provides features that make sockets redundant. Without pre-packaged library routines, RPC is harder to use than are local procedure calls. Furthermore, administrative effort is required to ensure that remote procedure numbers do not conflict with each other.

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). ■





Environmentalists.

Whatever your environment.

Lachman Associates, Inc. provides programming services with expertise in your software environment. LAI consultants bring together several hundred man-years of experience to meet your software development needs.

Varied systems and proven experience.

We provide complete development services for hire. Our expertise spans all of the major microprocessor, minicomputer and System/370 operating systems. LAI has been instrumental in a number of UNIX ports, as well as major projects involving: ● high-reliability operating systems development ● a distributed transaction processing system ● development of a new UNIX-like operating system ● local area networks using Ethernet and TCP/IP ● inter-netting SNA and non-IBM hosts ● distributed file system research ● a terminal front-end processor ● device drivers ● relational database systems ● compilers ● advanced debuggers ● comprehensive support for UTS ● enhanced features for MVS and JES ● a multi-processor architecture evaluation ● system performance measurements ● development of a heterogenous networking system ● advanced graphics software. Our related services include market studies, product analysis, customized technical training, documentation and software research and development.

Your place or ours.

LAI depth and experience, gained from these varied projects, works for you in many ways. Our knowledgeable consultants work to supplement your staff under your management, with focused experience that delivers from day one. We can also provide complete project design and development, from initial analysis through coding to complete documentation, using our own technical supervision.

Put the LAI team's experience to work for you . . . whatever your environment.

MAKING THE IBM CONNECTION

Proposals for orchestrating UNIX networks and mainframe databases

by David L. Buck

The recent popularity of the UNIX operating system in the commercial marketplace is at least partially due to the ease of developing and porting applications to the computers that run it. That popularity may be lost when it comes to large companies or data systems, however, if users cannot find ways to integrate the use of their UNIX systems with the use of non-UNIX mainframes and their various computing environments. Fortunately, manufacturers of UNIX-based computers are gradually taking steps to make cross communications convenient.

Departments within large companies may find they can use UNIX computer systems to do their own work—with the blessing of their MIS groups since the load on corporate mainframes can thus be reduced. However, these departments will be limited in the scope of their work so long as they cannot share data with other systems. Users need access to mainframe data and computing resources, while the MIS department must ensure that access is controlled.

UNIX systems provide for the controlled sharing of data and computing resources with the UUCP suite of utilities. But UUCP has the disadvantage of not being compatible with most other non-UNIX computer systems, and of having only an asynchronous protocol.

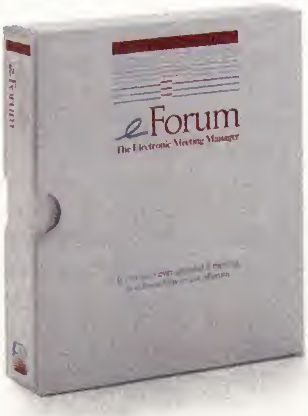
Most UNIX system suppliers have recently started to recognize the need for their systems to communicate with non-UNIX mainframes, and so have implemented the protocols that are most commonly used by those systems, including IBM's Binary Synchronous Communications protocol (Bisync) and Systems Network Architecture (SNA).

Interestingly enough, IBM is not included in the set of manufacturers bridging UNIX to IBM protocols. Though it recently announced the availability of UNIX System V on that old IBM workhorse, the Series/1 minicomputer, it did not announce the availability of any synchronous communications software for the UNIX port, specifying UUCP instead as the primary means of transferring data to and from UNIX as it runs under VM/370. Meanwhile, IBM *did* announce SNA networking extensions for the IBM PC that allow users of the small machine to get into SNA via a Series/1 computer running the standard IBM Series/1 operating system.

Even after Bisync and SNA are implemented on a UNIX machine, much remains to be done in achieving effective communications with non-UNIX mainframes. Most of the software utilizing these protocols on a UNIX system emulate terminals of one type or another, and so the ability to have a UNIX application interact with a database on a mainframe (host) system is limited by the capabilities of the emulated terminal. Broadly speaking, these terminals can be classified as either *interactive* or *batch*.

Illustration by Hyon Kim

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. 281 on Inquiry Card**

NETI

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)

INTERACTIVE COMMUNICATION FACILITIES

An interactive terminal is one that allows a user to work directly with host application programs. IBM's 3270 family of terminals is popular among IBM customers for interactive use since it offers synchronous communications and full buffering. Typical 3270 applications send a screen buffer to the terminal, which provides the user with protected fields (such as "Name" and "Address") and unprotected fields that the user may use to enter data. After updating the unprotected fields on the screen, the user may then press any of several keys in order to transmit only those fields that have been modified back to the host application. The amount of data actually sent is thus minimized and the communications line to the host is kept free, except when the user signals that the screen is ready to be transmitted.

One common means of connecting a UNIX system to an IBM system is by way of a 3270 terminal emulator running as a UNIX application (in much the same way that the **cu** utility allows use of a local terminal on a UNIX system to emulate an asynchronous terminal connected to another system). Such emulators use a Bisync or SNA driver on the UNIX system to connect to the IBM host. The 3270 terminal emulator allows users to switch from running UNIX applications to running host applications.

Data sharing in this mode is generally limited to what the user can do through the keyboard—although it can also sometimes capture the current screen display into a file, or capture data sent from the host application to an emulated printer attached to the 3270's emulated controller. The **cu** utility provides a means of data sharing that is similar in terms of its ability to capture received data in a file, and send data out as ASCII files.

More sophisticated users also demand that there be a programmatic interface to the terminal emulator so that an application program might be able to emulate a user typing at the emulated terminal. With this approach, an application that's assisted by the host application can query a database or do limited data entry or modification.

The programmatic interface scheme requires that the UNIX application know all of the user interactions required to log onto the host application as well as all those needed to request or modify the data records—a tall order that could require knowledge of the format of several screens of data. Moreover, minor changes in the host application will require corresponding modifications to the UNIX application. Typically, these changes can be either easily described to the user in a written newsletter or explained in a message displayed on the screen.

> Most UNIX system suppliers have recently started to recognize the need for their systems to communicate with non-UNIX mainframes.

The PC marketplace has popularized a variation on this theme: specialized software on the host and the PC are often used to perform data transfer and translation. For applications such as transferring small spreadsheet data files from one system to another, some PC products have a built-in interface to a 3270 terminal emulator that attaches to a corresponding spreadsheet data editor, file transfer program, or database query application running on the host. Since the same supplier provides both applications, there is no requirement for the user to modify the local application when the host application changes.

The overhead involved in receiving screen buffers and responding to them as a user makes for a slow interface to the host database—one that shouldn't be used to query or update large volumes of data, and thus little more than a kludgey way to connect application to application.

BATCH COMMUNICATION FACILITIES

A *batch* terminal typically consists of a card reader, a printer, and sometimes a card punch. IBM 2780 and 3780 batch terminals have been the industry standard since the late 1960s to early 1970s, and so have been emulated by such major equipment manufacturers as Honeywell, Data General, DEC, and Control Data Corp. Since most mainframes support this older Binary Synchronous batch terminal protocol, minicomputers and microcomputers alike have rushed to add its remote job entry (RJE) capability. The protocol used by 2780 and 3780 terminals is unique in the IBM communications realm in that it supports peer-to-peer communication; this allows one to build programs easily to transmit any file from one computer to another using a common synchronous protocol. IBM's more recently developed Systems Network Architecture batch terminals do not have this peer-to-peer capability, however.

The batch terminal implements a simple form of file trans-

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.

OEM terms available. Mastercard and Visa accepted.

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.

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. Japanese Distributor: Softec, Telephone: 0480 (85) 6565. Swiss Distributor: Modulator SA, Telephone: (031) 59 22 22.

UniPress Software

Your Leading Source for UNIX™ Software.

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

Circle No. 289 on Inquiry Card

fer in that it regards a collection of card images in a reader as a single file, and then sends them as a single stream of data, broken into blocks of convenient size. Printer output, as well, consists of a collection of print line images similarly segmented. Since the batch terminal typically can only deal with card images and print line images that have a well-understood and simple format, it is necessary first to reformat a data file from fixed or variable-length records into fixed-length records of 80 characters, and then switch it back again to its original form.

This relatively trivial task, though, provides a simple general-purpose file transfer mechanism for connecting machines of differing architectures and software. Since the Binary Synchronous protocol is synchronous and employs block error checking, it provides users with the sort of speed, convenience, and reliability that can't be found in the asynchronous environment.

This is because a bisynchronous transmission consists of eight data bits per byte, multiple bytes per block (on the order of 512 characters per block), with two bytes of synchronization overhead at the beginning of a block. Asynchronous transmissions resynchronize on each byte transmitted, with an overhead of two bits per byte. Under identical circumstances and using a 2400-bit-per-second modem, a bisynchronous protocol can transmit approximately 300 characters per second, while an asynchronous protocol tops out at 240 characters per second.

Interestingly enough, IBM is not included in the set of manufacturers bridging UNIX to IBM protocols.

Although bisync protocols and devices are far from the latest in IBM technology, the bisynchronous batch terminal emulation remains the most widely employed file transfer method between mainframes, minicomputers, and many microcomputers because of its ability to handle peer-to-peer transmissions. IBM's System Network Architecture (SNA), on the other hand, is a centralized network that does not allow direct peer-to-peer transfers without approval from its centralized control node. However, for file transfers between a mini or microcomputer and an SNA node, quite a few features employed by SNA and its batch terminals make it a more attractive alternative than binary synchronous communications.

For one, the 3780 bisync batch terminal reduces transmission times by employing the compression and expansion of sequences of blanks within a record. The 3770 SNA batch terminal, the one most closely related to the older 3780, can employ compression and expansion of sequences of any character. Additionally, the 3770 can further reduce transmission times when it's given a compaction table and compacted data.

"Compaction" is defined here as the substitution of one character for two adjacent "master" characters. Up to 16 master characters may be defined in a compaction table, with an inverse relationship existing between the number of master and non-master characters in the data stream. For example, if a file consists primarily of numeric data and punctuation characters, they could be defined as the master characters, resulting in a two-to-one compaction for most of the file.

The data link protocol employed by the 3770 (and most other SNA devices not collocated with the host computing facility) is IBM's Synchronous Data Link Control (SDLC). SDLC allows up to seven blocks to be transmitted with one acknowledgement covering one or more of the data blocks; what's more, other data can also be "piggy-backed" with the acknowledgement. This feature allows greater data throughput without the disadvantage of large block sizes. Thus, if a transmission error is detected, instead of having to retransmit an entire large block of data, only the small block with the error and those blocks that follow must be sent again.

The Bisync protocol allows transfer in only one direction at a time, meaning that a 2780 or 3780 terminal can either send a file from its card reader, or receive a file at its printer or card punch—but it can't do both simultaneously. The 3776 terminal, by way of comparison, can have up to six separate conversations going on a single communications link, with some conversations consisting of transmissions to the host, others involving the receipt of messages from the host. For transaction processing that

TEXT EDITING

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

NEW RELEASE

UNIPRESS EMACS™

VERSION 2

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

Circle No. 222 on Inquiry Card

UniPress Software
Your Leading Source for UNIX Software.

requires some response time from the host, the 3776 terminal has the advantage of being able to continue sending transactions even while the host is processing or responding to earlier transactions.

Data directed to the 2780 or 3780 terminal is destined to end up either at a card punch or printer, depending on the device selection code specified at the beginning of the file. The 3770 terminal family also offers the option of configuring diskettes and magnetic tapes, so these devices may be selected as well. Unlike the fixed width of a card image, or the limited maximum width (132) of a 2780/3780 print image, the 3770 printer width can be defined within the data. The 3770 tape and diskette devices, meanwhile, can have fixed record lengths of between one and 128 characters, as specified when the device is selected. The ability to select many different types of devices, and to specify certain ones for specific tasks simplifies applications that use a programmatic interface to a 3770 terminal emulator; data intended for a specific application, for example, could be directed to a particular medium and device address, allowing for easy co-existence of multiple applications.

APPLICATION-TO-APPLICATION INTERFACES

Via Interactive Facilities. The only reasonable way to achieve an interface between UNIX applications and IBM applications via interactive facilities is to have a programmatic interface to a terminal emulator on the UNIX side that is matched either by standard, static software on the IBM host, or user-developed software with a simple, general-purpose screen layout. The important consideration is that the overall interface remain constant.

The advantage of this approach is the access it provides to a large number of IBM applications. This owes to the overall popularity of the 3270 terminal, and the fact that either a Bisync or SNA interface may be used with most of those applications. Its disadvantage lies in the clumsiness of its interface for an applications program. This is due to the fact that it was designed for humans. Likewise, the interactive facilities approach lacks speed since the interface is screen-oriented rather than record-oriented.

Via Batch Facilities. Using batch file transfer capabilities, a trivial interface can be established between an IBM system and a UNIX application, without either of the applications needing to know a great deal about the presentation of data (as is the case under the interactive approach). The intention of the batch interface is to facilitate the submission of "jobs" to an IBM host, and to facilitate the receipt of the output from those jobs, so it is ideal as a UNIX application/IBM bridge. A UNIX application needs only prepare a "job", usually by sandwiching data between two fixed sets of "job control language" (JCL), before handing the results off to a batch terminal emulator. Its only remaining task then is to pick up the results at the batch terminal emulator when the output is returned.

For other—perhaps shorter—transactions, a transaction re-

quest can be made in one or more records, which can be sent as card images. One set of transaction requests quickly implemented at my company allows one UNIX system to request a wide range of tasks on a remote UNIX system. We can, for instance, have the remote system send a particular file back, receive a file, and give it a certain name, or execute a given UNIX command using standard input supplied by the transaction and then send back the standard output and standard error data in a response file.

Via IBM's SNA LU6.2. IBM has also given thought to providing an application-to-application interface between different nodes within an SNA network. The answer it's come up with is affectionately known as *LU 6.2*. Each addressable component of an SNA network capable of communications, called a *Logical Unit* (LU), has a "session type", or higher-level protocol, that it finds agreeable. A session type establishes the basic subset of SNA commands used for communications with other logical units; for instance, *LU type 2* is used to signify 3270 interactive communications, while *LU type 1* stands for 3770 batch communications. *LU type 6.2* is a recent addition to the list of session types that are intended for application-to-application communications. But although often mentioned, it is still not widely implemented.

Unlike the more traditional design-your-own approaches to interfacing applications, the *LU 6.2* session type has the ability to define the beginning and end of a transaction. It has also defined error recovery so that if an unrecoverable error happens mid-transaction, the steps taken from the point that's marked as the beginning of the transaction to the point where the failure occurred can be backed out. There are also a number of defined commands that allow users to create or destroy files; add, change, query, or remove records; and execute commands remotely.

This then seems to be the ideal

Continued to Page 95

RULES OF THE GAME

Your day in court

by Glenn Groenewold

When Susie Grep learns that her highly successful software, GrepStar, is being brazenly appropriated and marketed by Mastodon as a part of its Sabretooth II system, she is more annoyed than alarmed. As Susie sees it, a call to her lawyer should be all that's needed to put a stop to the infringement. After all, hadn't she been careful to comply with each requirement of the copyright law when distributing her program, and hadn't she rigorously guarded the confidentiality of the source code as her trade secret?

Understandably, Susie is upset when she later learns from her attorney that the problem promises to be a great deal more than an annoyance. Mastodon is no inadvertent infringer. On the contrary, it is taking the position that Susie's program could not legally be copyrighted and therefore is susceptible to use and sale by other companies. "You're going to have to go to court to stop them," her lawyer tells her.

At this point, assuming she is serious about stopping the ripoff, Susie is about to embark on what is likely to be a long and expensive education in just what it means to be a party to a business lawsuit. Right off, her lawyer will probably need a great deal of information in a hurry to prepare the documents initiating the suit properly. Having assembled this data, Susie might expect the next step to be a court hearing to consider the merits of her claim that Mastodon is making illegal use of her intellectual creation. Unfortunately, it's not usually that simple.

FIRST, LADIES AND GENTLEMEN . . .

Way back when, the trial of a lawsuit bore a certain resemblance to the Shootout at the OK Corral. Neither party had any way of finding out what evidence the other side might be able to produce at trial, nor could the parties determine whether their opposition was sitting on evidence it considered damaging to its case. Not surprisingly, in those days trials often were pretty suspenseful.

Changes have occurred gradually over the years—partly because of feelings that this style of conducting lawsuits wasn't exactly fair, and partly because dockets have become sufficiently crowded that courts no longer have time to allow litigants to conduct fishing expeditions for evidence at trial. So today it is possible to find out just about everything the other side plans to present in the way of evidence, and also to uncover anything the opposition might have in its possession that might be useful to prove *your* case. This is accomplished through a process called *pre-trial discovery*.

Since the kinds of discovery that are allowed vary according to what court system is involved—be it federal or one of the 50 states—the rules applying to a specific case may be different from those discussed here. But regardless of what court hears Susie's suit, she is apt to learn that discovery proceedings can be a pain.

Early on, she's likely to be confronted with page after page of questions from Mastodon's attorneys. Often, many of these queries will call for business details and other pieces of information that seem to have little or nothing to do with the subject of the lawsuit—and that Susie regards as nobody else's business in any event. She may also find it burdensome or even impossible to come up with many of the answers demanded.

Susie is facing a set of *interrogatories*. Though it may be hard for her to believe, this procedure was originally intended to make it easier for contesting parties to exchange facts with each other. But lawyers have found it necessary to make interrogatories more and more complex, both to guard against the possibility of evasive answers and to reduce the chance that something might be overlooked. There's also a suspicion that more than a few attorneys use lengthy interrogatories as a device to harass the other side.

> Though it may be difficult to believe, interrogatories were originally intended to make it easier for contesting parties to exchange facts with each other.

Ironically, given our (and Susie's) standpoint, it's the computer that has made it possible for lawyers to turn out many pages of such questions with little expenditure of effort. Once the computer has the names it needs to plug into the blanks, it can do the rest. These "boilerplate" interrogatories are deplored by most judges, but it's difficult to do much about them. A party to a lawsuit is entitled to learn more than merely the things that clearly can be used at the trial.

Information sought need only be "reasonably calculated to lead to the discovery of admissible evidence." And who can say whether a particular scrap of information might not lead to something that will be valid courtroom evidence?

Another discovery device is the request for *production of documents*. These demands are some-

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

SPREADSHEET

Q-CALC

Subject: Extraordinarily powerful spreadsheet with extensive math and logic facilities.

Powerful spreadsheet specifically designed to take advantage of the UNIX operating system. Q-Calc uses termcap to support any terminal. Interactive prompts and help text make it very easy to use.

Features:

- Extensive math and logic facilities.
- Large model size.
- Allows sorting and searching.
- Interfaces with the UNIX environment and user programs via pipes, filters and subprocesses. Spreadsheet data can be processed interactively by UNIX programs, with output placed into the spreadsheet.
- Q-Calc command scripts supported.
- Uses termcap.
- Optional graphics for bar and pie charts. Several device drivers are included to support graphics terminals.
- Available for the VAX™, Sun™, Masscomp™, AT&T 3B Series™, Cyb™, Apple Lisa™, Perkin Elmer™, Plexus™, Gould™, Cadmus™, Integrated Solutions™, Pyramid™, Silicon Graphics™, Callan™, and many more.

Price:

	Binary
VAX, Perkin Elmer, Pyramid, AT&T 3B/20	$2,500
(with graphics)	3,500
MC68000™	750
(with graphics)	995

Source Code available.

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: UNIX AT&T Bell Laboratories VAX Digital Equipment Corp Sun Sun Microsystems Masscomp Masscomp Cyb Cyb Systems Lisa Apple Plexus Plexus Computer Gould Gould Pyramid Pyramid Integrated Solutions Integrated Solutions Silicon Graphics Silicon Graphics, Cadmus, Cadmus Computer Perkin Elmer Perkin Elmer Callan Callan Data Systems AT&T 3B Series AT&T MC68000 Motorola

Circle No. 270 on Inquiry Card

UniPress Software
Your Leading Source for UNIX Software.

times incorporated in interrogatories, but they can be made at many stages during legal proceedings. Often the items sought are regarded by their owner as confidential proprietary information that can't be disclosed without irreparable harm.

If both sides are adamant, the judge must decide whether to order production of documents or information. A *protective order* can be requested as a condition of turning over items to the other party. If granted, this order requires that the party obtaining information treat it as confidential, under penalty of contempt of court. These orders are frequently used when it's contended that the information disclosed constitutes a trade secret.

However, the handling of confidential data turned over to lawyers in the course of litigation often results in problems. For example, in a recent highly-publicized suit, IBM contended that its opponent's attorneys harmed it by improperly disclosing such information.

STAR CHAMBERS

Lest you think that this exhausts the list of Susie's possible woes, the games have only just begun. At some point, Mastodon's lawyers are almost certain to want to take a *deposition* from Susie, and probably from her key employees as well. These are procedures by which the lawyers ask questions of potential witnesses under oath, though outside of a courtroom setting. While depositions can take place just about anywhere else, often a good deal of gamesmanship is involved in selecting their location. Very Important Persons frequently insist on having their depositions taken in their own offices. This is supposed to give them a psychological advantage. Most lawyers, having sat through hundreds of depositions, really don't care where they're held so long as the chairs are comfortable. The offices of one of the participating attorneys often are used, although sometimes a "neutral" site, such as a court reporter's office, is selected.

Depositions can be dreary. There's no judge present to rule on objections or hurry the proceedings along, so hours can be consumed as lawyers wrangle over the propriety of questions. Depositions may drag on for days, or in complicated cases—which claims of software infringement certainly are—for weeks. Though Susie has a proprietary stake in the outcome of these maneuvers, her employees do not. They can hardly be blamed if they weary of these tiresome games, and decide to accept offers of employment elsewhere.

Who can say whether a particular scrap of information might not lead to something which will be valid courtroom evidence?

It shouldn't be thought that these shenanigans are all one-sided. Susie's lawyer has probably been firing off interrogatories and demanding the production of documents on *her* behalf. It's likely, too, that her attorney has taken a whack at Mastodon's officers and employees by requiring *their* depositions.

But all this can become terribly expensive. Computer-assisted or not, Susie's lawyer will be devoting time to the preparation of interrogatories to Mastodon. Additional time will be spent assisting her in framing her replies to theirs. Moreover, Susie wouldn't want her deposition or the deposition of one of her employees to take place without her lawyer being present. Naturally, the attorney will expect to be paid for all this expenditure of time. Susie shouldn't be surprised if she finds she's incurred thousands of dollars in legal expenses without getting anywhere near a courtroom.

SNEAK PREVIEWS

Actually, Susie may not have all that long to wait before she has a *big* day in court. That's because the harm done to her will be irreparable if Mastodon is allowed to continue marketing its identical program during the period required for Susie's lawsuit to come to trial. To prevent this, her lawyer will most likely attempt to obtain a temporary injunction halting Mastodon's practices. This is an important stage of the legal proceeding, since its outcome often anticipates the ultimate determination of the case.

Preliminary proceedings such as this may not be as lengthy as the actual trial, but they obviously require careful preparation on the part of the attorneys. Time may be short, and there will be much that needs to be done. Lawyers expect to be well-paid for such heroics.

ON THE MERITS

When the great day of the actual trial arrives—certain to be several months and possibly some years after Susie's lawsuit was first filed—it may seem almost anticlimactic. But no matter how many preliminary skirmishes are won or lost, it's the war itself that counts. The trial, therefore, is no time to let up.

Trials involving such matters

as copyright or trade secret infringement are notoriously long and costly. Since neither juries nor judges can be expected to have any knowledge of what goes on inside a computer, it's necessary that all this be patiently explained by expert witnesses. In effect, the litigants will have to foot the bill for presenting a crash course in computer science to people who really may not care a fig about such things.

And, of course, Susie and her long-suffering but faithful employees will have to take the witness stand and go through the same testimony they have previously given in depositions. Human memory being what it is, chances are great that there'll be some sort of inconsistency in the testimony given on these two occasions—don't forget, a couple of years may have elapsed. No matter how inconsequential the discrepancy, the lawyer for Mastodon will be sure to pounce on it as evidence beyond any question that Susie or her employee is an utterly untrustworthy witness. Court rules prohibit violent retaliation for this sort of character assassination. Susie and her crew will have to bear up under it as best they can.

When the trial has finally ended, the decision will be issued. Let's be optimistic and assume that Susie is triumphant, and that, however unlikely, Mastodon chooses not to appeal. Otherwise, Susie will have lost more than simply the control of her software. Whether or not she has to pay damages to Mastodon, generally she will have to pay its court costs. These usually don't include attorneys' fees, but they do customarily encompass such items as the expense of depositions and the expense of obtaining the attendance of witnesses at the trial.

Need we add, Susie's lawyer will still expect to be paid.

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. ■

INDUSTRY INSIDER

Fresh concepts in output control

by Mark G. Sobell

As the number of computers and users has multiplied, so has the importance of the standards that tie them together. Until recently, though, there was no standard way to describe a typeset or laser-printed page.

From its inception, the UNIX **troff** formatter has produced only output specifically designed to drive a CAT phototypesetter. By various twists and turns, several other output devices have also been supported over time. But this still did not provide a way to move easily from one device to another.

Device Independent **troff** was thus designed to use intermediate files and a post processor to generate different types of output files capable of driving a variety of phototypesetters. This is fine in theory, but once you create an output file with one of these programs, you cannot change your mind about which device to print it out on. Nor can you be confident that your existing program will be able to drive any new output device you later purchase. That's because once a decision is made to use a particular output device, Device Independent **troff** is independent in name only.

To address this long-standing problem, Adobe Systems has designed PostScript, a page description language that you or an application program can use to write a program describing a page of output. When you send this program to a device equipped with a PostScript interpreter, you get output that matches your design. The technique is quite straightforward, but the ramifications are far-reaching.

To date, the PostScript interpreter has been brought up on the Apple LaserWriter, the QMS Lasergrafix, and on several of the Allied Linotype (formerly Mergenthaler) typesetters. With such a combination of output devices, you can use a Macintosh to create images that you can first proof on an office laser printer and then send off to a typesetting house for final production. Adobe promises that it will add new output devices to its list in the near future.

Adobe had to address several important issues to make its proposed standard functional. A single PostScript program must be capable of driving devices of various resolutions—anything from a 300 dot-per-inch (or as they say now, *spots*-per-inch) laser printer to a 1000+ dot-per-inch phototypesetter. And, because you may want to use the laser printer as a proofing device for the phototypesetter, line breaks and page breaks on the two must correspond. To maintain this correspondence, the Adobe software has been developed in such a way as to ensure that each character has exactly the same width, each line has the same number of characters, and each page has the same number of words—regardless of the device on which they're printed.

Another important issue is that of integration of text and figures—either line art or half-tone representations of photographs or other continuous tone art. Adobe has addressed both the resolution and integration issues by creating what it terms "a complete graphic imaging solution." PostScript handles text as graphics, so the issue of integration has actually become moot—the whole *page* is a graphic image. Because

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 distribut 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. 295 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.

text is considered to be just another graphic, you can rotate type to any angle you desire (how about a banner running at 45 degrees across the page?), and you can slant and distort type in many different ways.

Carrying the graphics concept to its functional conclusion, Adobe has chosen to store its fonts as outlines. Because PostScript has the ability to fill defined areas, it can create any of the characters it has outlines for. Most important, PostScript can use these images to create an image that takes maximum advantage of the resolution of the output device (or, the *marking engine*, as the latest terminology has it). Thus, the same PostScript program can drive a laser printer, a phototypesetter, or both, without modification.

Consider the implications. You can generate one output file (a PostScript program) and print it locally on your laser printer. You can then send the same output file to a colleague who has a different brand of printer or even a printer with a different resolution—and the program will produce the same page. Because a PostScript program is composed entirely of printable ASCII characters, you can transmit it over any system you can use for transmitting text.

How is Adobe propagating its proposed standard? For $30, Adobe will sell you a manual that tells how to write a PostScript program. Although you may want to go through this process a few times to gain a better understanding of the program, it's likely that in the general case, you'd be better off using or creating an application that writes the program for you, based on some higher-level input.

An example of this type of application program is Microsoft's Word package, which can generate output in the form of a PostScript program. Word runs on both the IBM PC and the Macintosh and can use equivalent PostScript program files from either machine to drive an Apple or QMS laser printer, or a Linotype typesetter.

Postscript opens the way to setting up a publications system that is not tied into one manufacturer's line of hardware and software. As a standard, it also means you can upgrade any component (computer, software, or output device) without needing to tamper with the entire system. Finally, the new standard offers a transparent solution to the end user that allows *any* application designed to work with PostScript to drive *any* output device rigged with a PostScript interpreter.

THE UNIX CONNECTION

For UNIX users, Adobe provides a set of programs that works with both **troff** and **ditroff** (Device Independent **troff**) to generate PostScript files. The result is that you can attach a laser printer (or a phototypesetter for that matter) to your UNIX system and drive it with **troff** or **ditroff**. This set of translation programs is called TranScript and is currently available for 4.2BSD systems. Adobe expects to release a System V version in the third quarter.

Again, the portability of output files developed under PostScript is of key importance. Using TranScript, you can generate an out-

EMERALD ONE™

SOFTWARE WITH COURAGE, BRAINS AND HEART

WHAT IS EMERALD ONE?

The most complete integrated office system available today, EMERALD ONE combines the most essential office tasks through six fully compatible and seamless sets of tools. EMERALD ONE runs on a broad range of mainframe, mini, super-micro and personal computers which use the UNIX™ operating system–the emerging standard for the office.

THE TOOLS OF EMERALD ONE

EMERALD ONE integrates your office tasks through:

1. *COMMUNICATIONS,* including Telephone Messaging and Electronic Mail systems,
2. *INFORMATION HANDLING* with EMERALD ONE's powerful Relational Database system,
3. *DECISION SUPPORT* features such as Business Graphics and the Electronic Spreadsheet,
4. *DOCUMENT PREPARATION* with Word Processing and a Cabinet, Document and File Folder system,
5. *TIME MANAGEMENT* tools such as the Personal Diary system and Meeting Scheduler and
6. *SYSTEM ADMINISTRATION* functions that allow a non-technical user to customize EMERALD ONE for the individual, work group and organization with ease.

Circle No. 290 on Inquiry Card

SOFTWARE FOR THE WORK GROUP

EMERALD ONE goes far beyond stand-alone personal computer software by linking individuals and their work groups. With EMERALD ONE, users work as a communicating group, not as isolated individuals. Whether it be a document, spreadsheet or personal diary entry, everything created with EMERALD ONE can be exchanged easily between individuals, work groups and beyond.

EMERALD CITY, THE PEOPLE BEHIND EMERALD ONE

EMERALD ONE is the result of an intensive, multi-year research commitment by Emerald City and its sister company Trigon Systems Group, one of the most respected consulting companies in office integration.
Attractively priced for distribution by hardware manufacturers, system integrators and OEM's, EMERALD ONE is fully supported by an extensive marketing program designed to assist distributors in penetrating the integrated office market. Emerald City offers the reality of a complete business solution, not just technology.

Emerald City, the company with courage, brains and heart.

EMERALD
CITY

Emerald City Inc.
20 Richmond Street East, Suite 700
Toronto, Canada M5C 2R9
(416) 863-9923

UNIX is a trademark of Bell Labs
EMERALD ONE is a trademark of
Emerald City

put file on your UNIX machine, use a modem to ship it off to an Apple or PC, and drive a printer from that machine—whether it's located across the room or across the country.

CONCLUSION

As with any innovation, PostScript will not become a true standard until it is generally accepted and widely used. Adobe has made a good start, though, by lining up an impressive array of manufacturers committed to the use of Adobe products. Besides Apple, QMS, Allied Linotype, and Microsoft, Adobe officials indicate that several other major players will soon make announcements. The next six months should show whether the "standard" has been blessed.

Software and hardware manufacturers will significantly determine just how far the new technology goes. Now, despite much improvement, programs such as Word cannot support extensive typesetting. (Word does not even offer hyphenation, a key ingredient to typesetting.) In a hardware vein, Apple must wrestle with the fact that the same Macintosh screen that supports only 72 dots-per-inch can drive a typesetter with a resolution of over 1000 dots-per-inch.

Giving the user complete control over a typesetter is not, in itself, that difficult. Providing increased user control while also maintaining ease of use is a stickier matter. This, then, is the challenge to which manufacturers must turn their attention.

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 "A Practical Guide to the UNIX System" (Benjamin/Cummings, 1984) and "A Practical Guide to UNIX System V" (Benjamin/Cummings, available May, 1985). He has been working with UNIX for over five years and specializes in documentation consulting and ***troff*** *typesetting. Mr. Sobell also writes, lectures, and offers classes in Advanced Shell Programming and* ***troff*** *macro development.* ■

Software Engineers

TOMORROW'S TECHNOLOGY TODAY

"There is some very exciting work underway here."

Richard Kornblum
Director of Systems Development
The Foxboro Company

At Foxboro we're pushing the state-of-the-art in a number of areas, developing some of the most advanced software systems anywhere in the world. Our Systems Development group is breaking new ground developing highly sophisticated software for fourth generation, state-of-the-art, microprocessor-based networks and operating workstations for process management and control systems.

To get this project underway, we have assembled the brightest, most innovative team of software professionals in an area long noted for technical genius. But we need more people who want to get in at the beginning and carry this program the rest of the way. To attract the best, we're offering the best, from compensation to a total environment designed specifically with the engineer in mind.

When Foxboro makes a commitment, we succeed. Make your commitment to a more exciting job today and you can share in that success.

Opportunities exist for:

- **Engineers**
- **Senior Engineers**
- **Principal Engineers**
- **Consulting Engineers**

to work in the area of:

Fault Tolerant Software

- Developing micro-based system diagnostics from definition to design on highly reliable fault tolerant systems.
- Experience in: fault isolation; error recovery; diagnotics; diagnostics for processor, memory, communications interface, disc memory or process IO interfaces.
- High-level language experience (Pascal and "C") required.

For these positions, please respond to Greg Page (617) 549-4046.

Opportunities also exist for experienced (5-plus years) software engineers to advance the state-of-the-art in:

- **Distributed Multiprocessing**
- **Local Area Networks**
- **Interactive Graphics**
- **Data Base Software**

Specific experience may include: real-time operating systems, High-level languages ("C", Pascal), distributed data bases, 16/32 bit microprocessor machine architecture, IEEE 802 LANS, X.25 and UNIX^{tm}

UNIX^{tm} is a registered trademark of AT&T Bell Laboratories.

For these positions, please respond to Fred DelGizzo (617) 549-4106.

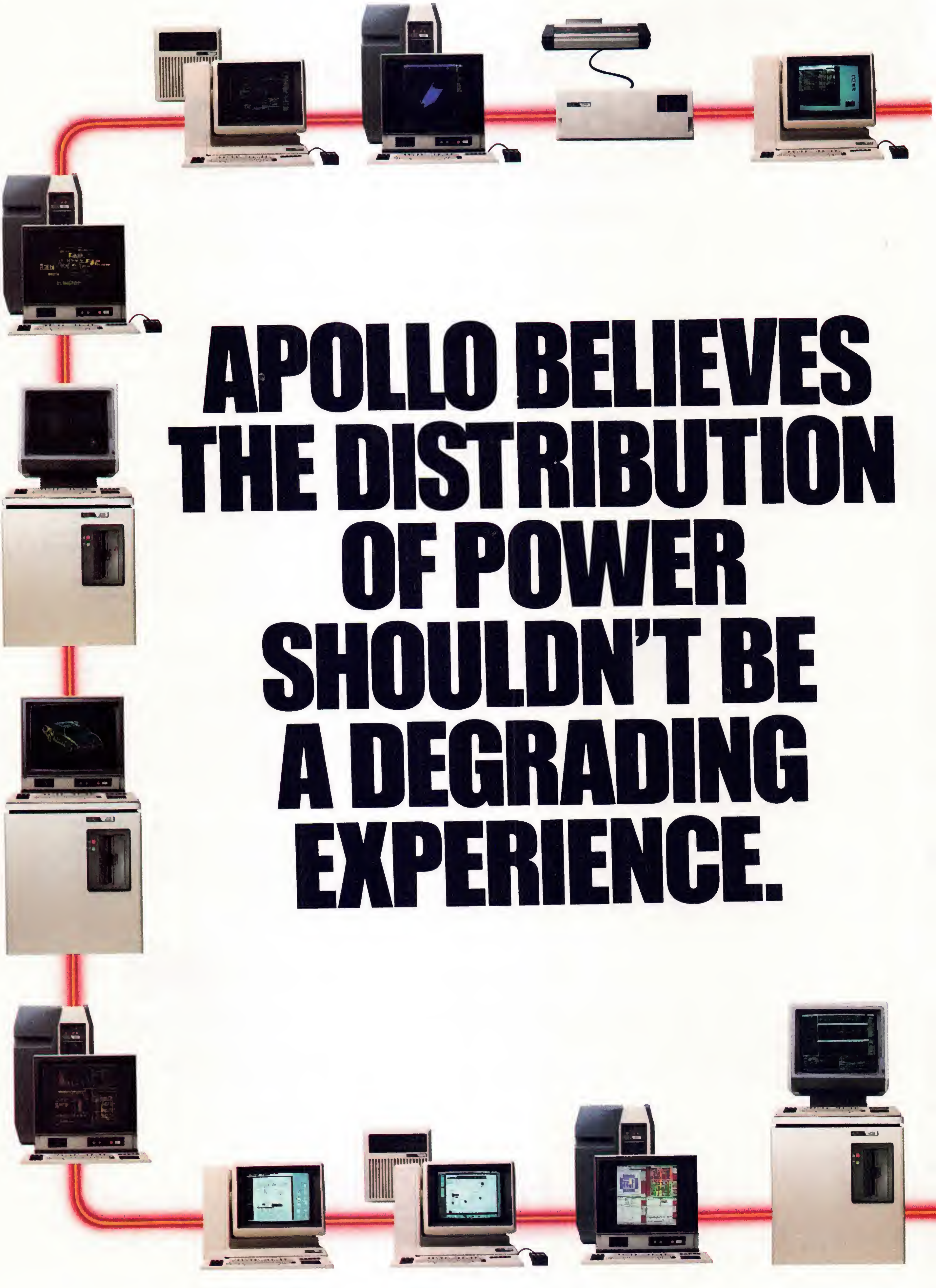
Send your resume today to the appropriate recruiter at Dept. UR5, Bldg. 6, 38 Neponset Avenue, Foxboro, MA 02035. Foxboro is an equal opportunity employer, M/F. We welcome resumes from minorities and protected classes.

If you have a PC or Terminal with 300 baud dial-up capabilities, then explore our opportunities in depth by calling our Career Data Base at (617) 549-4444 and enter carriage return. Our computers are manning the phones 24 hours a day, 7 days a week.

Our Career Data Base contains complete details on a wide variety of positions currently available at Foxboro.

FOXBORO

Circle No. 265 on Inquiry Card


APOLLO BELIEVES THE DISTRIBUTION OF POWER SHOULDN'T BE A DEGRADING EXPERIENCE.

That should begin to tell you what we think of timesharing on minicomputers and mainframes. An approach that has significant appeal when only a few people use a computer system. But a concept whose flaws become painfully apparent as soon as you try to distribute finite compute power to all the people who need it.

When you do, everyone has to sacrifice their share of that power. The result is system degradation. And the frustrations that come with slower processing, lack of adequate memory, and the additional time it takes to get work done.

There is, of course, an alternative. It's called DOMAIN® processing. And it's exactly what engineers, scientists, architects and other technical professionals have always clamored for. The computer industry's only known example of 32-bit dedicated workstations, each with the compute power of a mainframe, that let users transparently share both information and resources across a high-speed local area network.

Put another way, DOMAIN processing combines the best features of both timesharing and standalone computers, but none of their drawbacks. Sharing without sacrifice. Dedicated compute power without isolation from co-workers and their data.

With DOMAIN processing, every user gets his or her own computer system connected to a network. Each workstation delivers large machine performance and functionality, substantial data storage, and effortless interaction with the system. And when you add additional workstations, you actually increase everyone's power. Rather than diminish it.

The DOMAIN family of compatible workstations offers a wide range of products. From high performance color workstations for demanding applications such as solids modeling and finite element analysis to low-cost monochromatic workstations for software development and technical documentation.

Of course the true worth of any computer system is its applicability. A measure by which Apollo excels. Today more than 300 of the finest application packages available run under DOMAIN. State-of-the-art programs that cover electronic design, mechanical design and engineering, computer integrated manufacturing and much more.

So if you're running out of power to distribute to your product developers, we suggest you contact Apollo. You'll find the experience enlightening.

Call (617) 256-6600 x4497. Or write
Apollo, 330 Billerica Rd., Chelmsford,
MA 01824 MS32.

apollo

DOMAIN is a registered trademark of Apollo Computer Inc.

Circle No. 240 on Inquiry Card

DEVIL'S ADVOCATE

Stop the industry—I want to document!

by Stan Kelly-Bootle

Bertrand Russell was fond of paradoxes, especially his own. Who can forget the famous Russell paradox ("If A is the set of all sets which are not members of themselves, then 'A belongs to A' implies 'A does not belong to A'), which floored Frege and damn near killed Russell himself? It also ruined the dream of putting mathematics on a solid logical foundation.

Russell admitted to us later that the 20 years he and Whitehead spent trying in vain to resolve this paradox in *Principia Mathematica* left him scarred for life. Their theory of types sought to avoid the paradox by brushing it under the carpet. If certain sets of sets led to contradictions, then they had to be blackballed from the club.

" 'Fraid we blew it, Alf," Bertie announced suddenly. We were all sipping sherry, I vividly recall, in his Trinity College rooms. Whitey stormed out in a huff, leaving just Bertie, Witgie, C.P., Al (Turing), and myself. "There's a pretty paradox," quipped Al in his best mock Gilbert & Sullivan accent, putting a comforting arm around Bertie. Wittgie carried on poking at the empty fireplace as though nothing had happened. And yet we all sensed that Formal Systems Theory would never be the same again.

Poor Bertie did not live to see the UNIX resolution of the problem. Clearly, if one has directory files containing the names of files, they may or may not contain their own names. If they do, put them on Disk 1; otherwise on Disk 2. Then make a superdirectory file A, of all the directories on Disk 2—that is, for all those that do not contain themselves.

If we include A itself in this superdirectory, it would appear to belong on Disk 1 . . . and yet all members of A, including A itself, by definition, belong to Disk 2. But are we dumbstruck by this impasse? Do we rush off for 20 years to juggle with symbolic tautologies? No way in San Jose. This is hard-headed, feet-on-the-ground UNIX territory. File A is simply *swapped* to and from Disk 1 and Disk 2 as a background task—or perhaps it's just quietly erased when no one is looking.

Bertrand Russell was also fond of the Tristram Shandy Paradox. Tristram found that in trying to complete his autobiography, he was taking a year to cover each day's activity.

Given a tireless, immortal author armed with WordStar™ version Aleph 0, on a Turing Machine (no tm yet), and an endless supply of floppy tape, the question posed by philosophers and other layabouts is whether any part of Tristram's life would ever remain unrecorded. Or would the damned machine halt first? Sorry, I seem to be mixing my paradoxes.

What's undoubtedly true about the Tristram story is that, complete or not, the opus would be infinitely boring. The bulk of it would resemble the average modern novel, the plot of which revolves around a modern novelist trying to write a modern novel . . . this recursive narcissism pervades many fields of human endeavor.

When I was a fulltime folk singer touring the UK with other fulltime folk singers, we saw so little of the real world that we ended up writing and singing ballads about the tribulations of itinerant balladeers: "It's a Mighty Hard Road—50 Miles to the Next Gig", "My Agent Done Dropped Me", "Fretblood Blues", and other similar ditties.

The speculation on Tristram

AT&T just announced its UNIX™ PC. GSS software is on it. Just like on the IBM PC. The result is standard UNIX graphics, just like on PC-DOS.™

When AT&T and IBM needed graphics standards, they came to GSS. The emerging standards implemented by GSS for DOS and UNIX — the Virtual Device Interface (VDI) and Virtual Device Metafile (VDM),— form the foundation of the graphics strategy of IBM, AT&T, and others.

> ***"The GSS Virtual Device Interface endorsed by AT&T and IBM has established GSS as the primary source for graphics software tools in the small computer marketplace."***
>
> **Dataquest**

The first revolution in microcomputers was the development of standardized operating systems. Programs right off the shelf could run on different computers. But with only a few peripherals. And with limited graphics.

The second revolution is graphics and it belongs to GSS. Because we developed the graphics standards for AT&T and IBM, and because our GSS-DRIVERS,™ GSS-TOOLKIT,™ and GSS-SOLUTIONS™ provide you the edge you need to differentiate your products and stay competitive. And our products are available now, right off the shelf.

> ***"One of the most technically difficult areas to standardize is graphics. Having GSS support the VDI in both DOS & UNIX, will provide software developers greater program portability and shorter development cycles."***
>
> **Aaron Goldberg, IDC**

If you make software, computers, chips, or peripherals, you can also be a part of the standard.

The revolution continues.

25117 SOUTHWEST PARKWAY WILSONVILLE, OREGON 97070-9600
503/682-1606 TELEX 499-4839 GSS UI

™
GRAPHIC SOFTWARE SYSTEMS

IBM and PC-DOS are trademarks of International Business Machines, Inc. UNIX is a trademark of AT&T. GSS logo, GSS-DRIVERS, GSS-TOOLKIT, and GSS-SOLUTIONS are trademarks of Graphic Software Systems, Inc.

Circle No. 264 on Inquiry Card

Shandy arises from my recently completed Waite/Sams Primer on the Motorola M68000 family.

The emetic pace of the computer industry, especially in the last decade, presents a target not unlike Tristram's life, threatening to out-accelerate the traditional rate for producing timely and useful documentation. The M68000 chips themselves are Rocks Of All Ages, but some of their implementations and implementors fly forgotten as dreams. Before the ink is dry on the typical product document, hardware obsolescence (not excluding the demise of paper and ink), litigation, negative cash flow, rewrites, unwrites, mergers, sudden death, and the thousand natural glitches our trade is heir to, can intervene to vitiate the writer's efforts.

With this in mind, I sent out a carefully worded letter last September to all computer innovators (with an "Information Only" copy to IBM), urging a six-month moratorium so that my fellow authors and I could "catch up". The response was mixed, to say the least.

The BASIC standards committee, bless it, cabled immediately from a restaurant in Geneva, promising to remain "poised for action, awaiting your signal." AT&T wrote to say that six months was hardly sufficient ("Call that a moratorium?") because its planning charts allowed only a two-year granularity, but nevertheless agreed to rush out some "Freeze UNIX" bumper stickers. My only other success was Lotus Development Corp., which reluctantly offered to hold back Jazz for a while. The industry as a whole pressed on with its usual selfish, me-too, up-yours madness.

The only solution, it seems, is a break from traditional methods of information dissemination. In many user situations, online help files can speed up the instructional flow, but can they offer the convenience of books and manuals? Rapid browsability and informal, mid-job access are important but remarkably difficult to achieve. The helpee is not always sure where that tiny but cosmically vital piece of help is lurking. More often than not, a simple question invokes a daunting overdose of unhelpful diversions. The car owner wants to get from here to there without studying the molecular structure of hydrocarbons or the love life of Nicholas Carnot. All too often we are given the equivalent of "I wouldn't start from here if I were you."

Is there yet an electronic analog of flipping rapidly through a well-thumbed text so as to spot your previous annotations and yellow highlights?

On the broader educational front, many computer topics suddenly become fashionable. As the relative cost of computer components fluctuates, last week's hot debate on method A versus method B can quickly become as meaningless as a medieval angel-counting contest. Small wonder that ACM's SIG is planning a daily newsletter.

Computer Science in 1985 still reminds one of Ancient Egyptian Mensuration, coping quite well on an ad hoc, day-to-day, problem-solving basis, yet waiting for Geometry to arrive. We have our Thales' and Pytheas's *go leor*, even the odd Pythagoras (send a plain, stamped, addressed envelope for my listing), but where is Euclid or Euclidea hiding?

He or she is not required to *add* to the sprawling corpus of pre, post and praeter-structural gospels, but simply to step outside the cockpit and codify, codify, codify. We need to have machine-independent, language-independent, OS-independent, and logo-independent axioms. We need to have an acceptable deductive framework free from the influences of xenophobia, inventory levels, and sales commissions. The rewards will be great.

Remember that Euclid's *Elements* has held top spot in the all-time textbook bestsellers list for over 2000 years. Indeed, it is second only to the Bible in the combined lists.

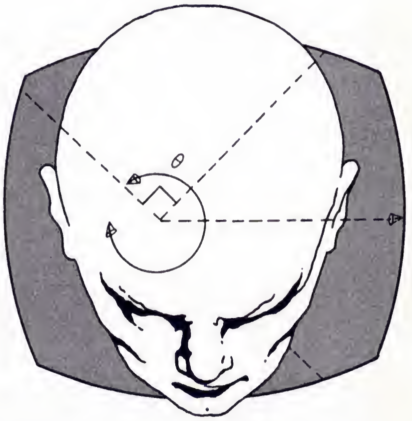
A *Principia Rationorum Digitorum* would bring similar stability to the computer trade. At least until a Lobachevski or Riemann pounces on the axioms.

Stan Kelly-Bootle has diluted his computer career by writing contemptuous folk songs for Judy Collins ("In My Life," Elektra K42009), The Dubliners and others. He is currently writing, with Bo Fowler, "The 68000 Primer" for the Waite Group, to be published by Howard W. Sams in the Spring of 1985. ■

PROBLEM SOLVER

Network servers under UUCP

by Bob Toxen



Because most network solutions on the market today provide essentially the same sort of service, it's sometimes difficult to appreciate just how many underlying communications mechanisms there are. Schemes that make use of directly-wired RS232 ports, public phone networks or Ethernet networks running XNS or IP/TCP are the most common. The organization that doesn't have several such options available on its various systems is rare. But none of these underlying mechanisms are suitable for all circumstances—making some higher-level facility necessary to tie them together.

This high-level facility should be able to operate over diverse hardware types, have a checksumming and retry capability for reliable transmission, a spooling mechanism to keep requests from getting lost if a system is temporarily down, and a way to acknowledge the completion of requests. At the very least, it must be able to transfer files and do remote program execution.

You may not realize it, but this high level facility is probably already at hand. UUCP meets all of the requirements quite well. Its chief cost comes in the few days of programmer time it takes to interface with XNS or IP/TCP in most implementations. This owes to the interfacing scheme of XNS and IP/TCP, which requires that access be made through custom library routines. For XNS, one can specify a device named *xns* in UUCP's *L-devices* and *L.sys* files for connection with systems communicating over XNS. In doing this, though, one should not place an exclusive use lock on the *xns* device, nor should one apply RS232-type **ioctls** (such as for setting the baud rate).

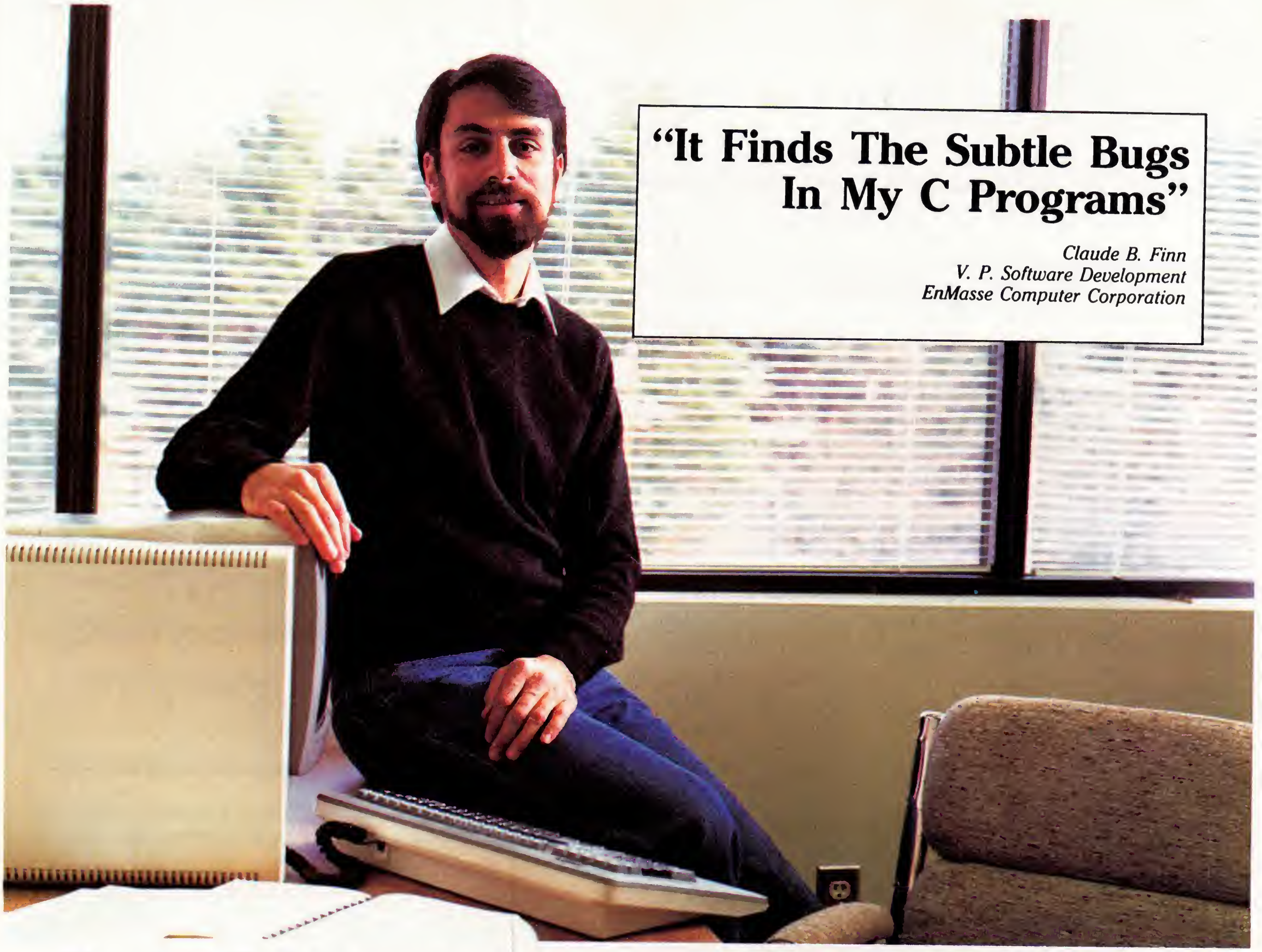
Once UUCP is configured, network servers can be developed. Since a single printer is generally shared among several systems, one often develops a printer server first. This and other servers can be often be simple shell scripts.

Assume the standard UNIX printer spooler, **lpr**, is used to actually drive the printer and that the system connected directly to the printer is called *bigiron*. On those systems not connected directly to the printer, one could remove the standard **lpr** program and replace it with a shell script to do remote spooling. The following script is typical:

```
cat $* | uux bigiron!lpr
```

If a list of files is supplied as an argument when the script is invoked, **$*** will expand to this list of files and they will be concatenated. If no arguments are supplied, **$*** will expand to a null string and standard input will be read. There are several features that could be added. One might be to tell **lpr** to print the name of the account invoking the script on the banner page (without such a script, the account name **uucp** would be printed on the banner page instead).

There are also other ways to make use of remote printers via UUCP. The **uux** program, for instance, is UUCP's own way of doing remote execution. Its first argument, a dash (-), tells **uux** to read its standard input (piped from **cat**) to EOF and then supply it as standard input to the remotely executed program. The second argument (*bigiron!lpr*) indicates which system to use for the remote execution and what program to execute. Under **uux** syntax, the system



"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. They may be purchased separately or as a group.

CCA Uniworks, Inc.
Productivity Tools for Programmers
20 William Street • Wellesley, MA 02181
Circle No. 279 on Inquiry Card

For more information or to place an order call our customer representatives at

800-222-0214

In MA (617) 235-2600
or mail this request form today.

Please send me information on:

- ☐ The Safe C Development Tool Family
- ☐ AI Development Tools
- ☐ CCA EMACS Unix/VMS Editor Environment
- ☐ Your complete line of state-of-the-art programming tools
- ☐ Please send license forms

Name ____________________
Title ____________________
Company ____________________
Address ____________________
City, State, Zip ____________________
Phone (   ) ____________________

CCA UNIWORKS, INC.
20 William Street • Wellesley, MA 02181

A Crowntek Company

Unix is a trademark of Bell Laboratories, VAX and VMS is a trademark of Digital Equipment Corporation, MS/PC-DOS is a trademark of Microsoft, Safe C is a trademark of Catalytix Corporation, CCA EMACS is a trademark of CCA Uniworks, Inc.

name (*bigiron*) should appear to the left of the exclamation mark (*!*) and the program to execute (**lpr**) should appear to the right.

To maintain security, the remote system will only allow certain commands to be invoked by **uux**. This list of legal commands appears in different places in different implementations. In most, the list is contained in a file in the */usr/lib/uucp* directory called *L.cmds*, *L-cmds*, or *uuxqtcmds*. In others, it is compiled into */usr/lib/uucp/uuxqt*, which is a program that a binary licensee must patch in order to change.

It should be noted that UUCP can communicate between computers from different manufacturers with different processors running different versions of UNIX. When a program is remotely executed (invoked) the binary that's used comes from the machine doing the actual work. This allows, say, a M68010 system to remotely execute a program on a VAX.

SERVERS FOR troff

With the current proliferation of laser printers, **troff** is becoming commonly available. A server under UUCP can be used to invoke **troff** on a system connected to a remote printer, even when the text to be **troff**ed resides locally. Thus, a **troff** server patterned after the **lpr** printer server is needed. It must have additional smarts built in, however, to deal with **troff** flags. This is handled by the script listed in Figure 1.

```
opts=""
system=bigiron
while [ "$1" != "" ]
do
        case "$1" in
          -*)
        opts="$opts $1"
        shift
        ;;
          *)
# troff a list of files
        cat $* | uux - $system!troff $opts
        exit 0
        ;;
        esac
done
# troff from standard input
uux - $system!troff $opts
exit 0
```

Figure 1 — *An example of a server that can be used to invoke **troff** on a remote system.*

MAIL SERVERS

A network mail server is already built into UUCP and **mail**. To send mail to someone, the account name and the name of the system it is on must be supplied. Thus, to send mail to an account called *jill* on the system called *onyx*, the following commands could be used. Under **csh**:

> None of the underlying communications mechanisms are suitable for all circumstances—making some higher-level facility necessary to tie them together.

```
mail onyx\\!jill
```

Under the Bourne shell:

```
mail onyx!jill
```

(See my column in the November, 1984, issue of UNIX REVIEW for a more detailed discussion of this facility.)

This sort of approach is fine for a set of five or 10 accounts where all the pathways are known and easy to remember. If one must communicate with 100 accounts distributed over a dozen workstations and a few VAXen, it's almost impossible to keep track of what system an account is on. (It is usually possible to track account names by using, say, a person's first name. If several people have the same first name, then the first letter of their last names can be part of their account names.)

What is needed is a mail server that knows where any given account is located. One could then send mail to *jill* and the server would know to send mail to *onyx!jill*. There is already the capability to do this on any UNIX implementation that has **Mail**, **sendmail**, or **delivermail**. With straight 4.1BSD, 2.9BSD, 4.2BSD, or a system with "Berkeley enhancements", one of these is probably available. To be sure, issue the command:

```
% ls -l /bin/Mail* /usr/*/Mail \
  /usr/lib/*sendmail /etc/delivermail*
```

With **sendmail** or **delivermail**, *aliases* can be added to the file */usr/lib/aliases*. These aliases are usable by everyone on the system. The syntax calls the name given to **Mail** and a colon (:), to be followed by a tab and the name to actually use. Thus, for our previous example, add the line:

```
jill:	onyx!jill
```

With this in place, mail can be sent to Jill by as simple a command as:

```
% mail jill
```

With **Mail**, aliases can be provided by similarly editing the file */usr/lib/Mail.rc* (or the file *.mailrc* in your home directory if you wish to limit the effects to your own account). For example, adding the line:

```
alias	jill	onyx!jill
```

will allow one to send mail to *jill* with the command:

```
Mail jill
```

Note the capital *M*. Even on some systems that do not have **Mail**, this last example will work with a program called **mail** rather than **Mail**. To see if a system's **mail** program has this capability, issue the command:

```
% mail -u oops
```

If the error message *"oops" is not a user of this system* returns, then the system has the ability to alias mail destinations.

SUMMARY

We've only scratched the surface. Many other UUCP servers can be developed as needed. One might create a "wall server", for instance, to invoke the **wall** program on a network of systems. This would allow the **wall** command to broadcast its standard input to all people logged into a network, meaning it could be used for making company-wide announcements. (Since **wall** is in the */etc* directory, either list it as */etc/wall* in UUCP's *L.cmds* file or link it to */bin/wall*.) For less urgent announcements, a server can be created for using the 4.2BSD **msgs** or System V **news** programs.

How many other uses can UUCP servers be put to? The possibilities are limited only by imagination.

Bob Toxen is a member of the technical staff at Silicon Graphics, Inc. who has gained a reputation as a leading expert on **UUCP** *communications, file system repair and UNIX utilities. He has also done ports of System III and System V to systems based on the Zilog 8000 and Motorola 68010 chips.* ■

THE UNIX GLOSSARY

Distributed processing and databases

by Steve Rosenthal

Note: only those aspects of these terms that pertain to distributed resource sharing are included in this listing.

access plan—the method by which a distributed system provides a local node or site with remote data. Data may be retrieved upon request, or requests may be buffered until a specified number accumulates or a particular stage of processing is reached.

atomic—an indivisible operation that must either be done to completion or aborted. Transaction updating is a typical example.

checkpoint—in a distributed network, refers to a point at which all outstanding transactions are recorded and new transaction logs are begun. If there are any subsequent system failures, recovery can be made by resubmitting transactions from that point.

commit—to accept a transaction into a database. In a distributed system, transactions are usually treated as atomic, being either completely accepted, or discarded. Once a transaction has been committed, its information becomes available to other stations in the system.

complete—said of a distributed database or a distributed set of programs, arranged so that all elements designed to be included in the global system are, in fact, incorporated into some local node on the system.

conceptual schema—the logical arrangement of items in a database or file. The physical record may be very different than the logical view.

concurrency control—in a distributed system, ensures that events that are input simultaneously do not interfere with each other or lead to the processing of incomplete records and files. The usual method is for the system to finish processing one transaction before starting another, or to use a system of locks or semaphores to restrict simultaneous use of the same information.

deadlock—in a multiuser or distributed system that uses a system of locks to reserve access to files, records, or system resources, describes a situation where two or more processes have seized the use of some key element, leaving no process ready to proceed.

disjoint—when speaking of a database or group of programs split up between nodes on a distributed system, refers to one that has no overlaps or duplications between nodes.

distributed processing—an arrangement of computers and related equipment that allows computation and decision-making to be done at more than one location. Most older UNIX systems connect terminals with no processing ability to a central computer, but increasingly the trend is to include processing ability in each location. So far, standard UNIX makes no provision for distributed processing, but a number of commercial and experimental systems have been built.

distribution transparency—the arrangement of a distributed system such that ordinary users are not aware of, nor need be concerned with, the physical frag-

UNIX™ & 'C' TRAINING

MANY UNIX-BASED SYSTEMS ONE UNIX TRAINING COMPANY

The Computer Technology Group provides the UNIX training solution. Training to fit the complexities of your UNIX-based system.

Three factors make the Computer Technology Group the experts in UNIX and 'C' language training:

- Experience, through training thousands of students worldwide in live seminars, with thousands more using our video training at their own locations.
- Extensive Curricula Supporting All UNIX Versions, creating a client base of manufacturers, software developers and end users.
- Quality of Instruction, with instructors and course developers who are experts in teaching UNIX and 'C', as well as in designing and implementing a variety of UNIX-based systems.

ONE UNIX TRAINING COMPANY MULTIPLE DELIVERY SYSTEMS

Whether you're training two, 200, 2000 . . . you can select the most efficient and economical training solution for your unique environment:

- Public Seminars offered in major cities throughout the world.
- On-Site Seminars for training customzied to your system and to specific groups within your organization.
- Video-Based Training for consistent training that is always available at your location.
- Interactive Videodisc Training, which dynamically tailors courses to the individual—from novice to expert programmer.

ASK FOR OUR 48-PAGE COURSE CATALOG, WHICH PROVIDES:

- Comprehensive course outlines
- Course prerequisites
- Curriculum recommendation for multiple audiences
- Guidelines for cost-effective training media selection
- Current seminar schedule

CALL (800) 323-UNIX or
(312) 987-4082 in Illinois

™ UNIX is a trademark of Bell Laboratories.

COMPUTER TECHNOLOGY GROUP

Telemedia, Inc.

310 S. Michigan Ave., Chicago, IL 60604

Circle No. 291 on Inquiry Card

mentation of the system.

durable—said of processing arrangements that are designed to ensure that data will be reflected in the system files once it's been accepted by the system. This allows remote systems or locations in a distributed network to erase their copies of transactions once the transmitted data has been accepted by the location responsible for the record.

exclusive—a one-user-at-a-time limit on the right to access a shared file in a network or distributed system. Unlike private files, which can only be accessed by a single user, exclusive files can be used by others—at different times.

fragment—to divide a database into parts, each of which can be stored on a different device or system. Vertical fragmentation divides each record (or each row in a table-oriented database), while horizontal fragmentation divides the database such that one group of complete records (rows in a table-oriented database) remains together while another group is broken off into a separate piece.

fragmentation schema—the way databases or programs are broken up in a distributed system.

global optimization—in a distributed system, refers to improvements in the way data or programs are shared among various nodes or sites. The most frequent tactic is to cut the number of remote accesses, either by increasing locality or by assembling remote requests into blocks.

graceful degradation—a failure mode for a distributed system that allows users to access some local files and processing power—even when the coordination between sites breaks down. In this way, users avoid losing pending transactions.

granularity—the size of the data regions reserved by the software mechanisms preventing simultaneous writes, updates, or deletions by two or more users in a distributed system. Larger granularity is often easier to implement, but it tends to slow access to data in busy systems.

heterogenous—said of a distributed system in which not all nodes or sites are the same. Most commonly, the network will combine a central mini or mainframe with individual workstations.

homogenous—said of distributed systems in which each node or site is identical. A homogenous system often is comprised of identical workstations with no central server or main computer.

horizontal fragmentation—the division of a database or any other collection of data such that groups of complete records are stored at various locations.

isolation—the extent to which operations in process at one location leave all other locations unaffected.

locality—the degree to which the data needed in a distributed application can be found at the site that started the task (the site of origin). Complete locality means that all the necessary data can be retrieved locally.

local optimization—improving the way a node or site accesses or processes data in a distributed system. If the system is partitioned clearly, local optimization can occur even without any changes in global operation.

local reference—a request for data or programs located at the user's node or site in a distributed processing system.

location transparency—the arrangement of a distributed system such that ordinary users are not aware of, nor need be concerned with, where particular elements are located in their system.

log—a set of transaction records entered since the last checkpoint. One good practice is to log each transaction before making an update or performing an operation. Then, if a system failure occurs, the state can be reconstructed by resubmitting the log entries, using the checkpoint files as a starting point.

map—to implement the conceptual schema for storing data in a physical database. Although the physical database should act logically like the conceptual schema, it need not necessarily resemble it in form, order, or size.

redundancy—as applied to distributed systems, the extent to which data (or programs) are duplicated across the system rather than being partitioned among the component nodes.

remote execution—the execution of a program or process other than the one where the user is located.

remote reference—a request for data or programs located at some other node or site than the one where the user is working.

replication transparency—the arrangement of a distributed system such that users do not have to be concerned about whether the system keeps a single copy of each file or duplicates it at one or more nodes.

serialize—to process events that arrive simultaneously as if they had arrived one after the other. This is one of the principal methods of concurrency control.

shared—said of files and records that can be read by more than one process but are locked against writing or deleting. This mode is not yet a part of standard UNIX System V, but it is available on several other UNIX implementations.

site—a processing and user-interface node in a distributed processing system. The site of origin is where processes start, but processes can go on to invoke others at remote sites in the system.

site autonomy—the degree to which each location in a distributed processing system can work according to its own rules, schedules, and procedures. Greater site autonomy pleases users, but it carries the danger of database corruption or incompatibility.

transaction—an update, addition or deletion of a record in a distributed system. To prevent errors, some method of concurrency control must be used to coordinate processing of transactions so that records are not used while they are in the process of being updated.

vertical fragmentation—the division of databases or other collections of information such that some fields for the same records (or "attributes", if the database happens to be table-oriented) are stored in various locations.

Comments, questions, corrections? Please send them to Rosenthal's UNIX Glossary, Box 9291, Berkeley, CA 94709.

Steve Rosenthal is a lexicographer and writer living in Berkeley. His columns regularly appear in six microcomputer magazines. ■

RECENT RELEASES

HAPPINESS THROUGH CROSS TOOLS

Unidot has developed a family of cross development software tools for UNIX systems, including cross compilers and macro assemblers. Targeted microprocessors include chips from Intel (including 808X and 80X86), Zilog (Z80 and Z8000), and National Semiconductor (32000). The company says it can supply source code in most cases.

To facilitate development of complex software in assembly code, the Unidot assemblers support a flexible macro facility and conditional assembly facility, as well as up to 255 separately relocatable object sections, a full assembly listing, symbolic debugging, and a symbol cross reference listing.

The Unidot tools simplify programming microprocessors by using mnemonics compatible with those of the op-code and by using assembly language syntax. The pseudo operation syntax is common to all the assemblers.

Pricing varies with the host computer. A site license for multiple CPUs at the same location is 2.5 times the binary license fee. A source license, including a site license, is three times the binary license.

Unidot, Inc., 602 Park Point Dr., Golden, CO 80401, 303/526-9263.

Circle No. 246 on Inquiry Card

Qume QVT-101 terminal

SMART TERMINAL WITH SMART PRICE

The QVT-101 terminal from Qume Corp. is described by its manufacturer as a "smart, full-function editing terminal"; its price is $395. The terminal provides all the features of Qume's QVT-102, and emulates the Hazeltine 1500, Lear Siegler ADM 3A/5, and TeleVideo 910.

The QVT-101 features block mode data transmission, allowing entire screen loads to be dumped at one time. Multiple interface options allow it to operate remotely from a microcomputer, mini, or mainframe. This terminal offers 16 host or user-programmable functions.

A 14-inch non-glare screen, bidirectional printer port, RS232 interface, foreign character sets, and detached keyboard are all standard. Options include an amber screen, and current loop or RS422 interfaces.

Qume Corp., 2350 Qume Dr., San Jose, CA 95131, 408/942-4000.

Circle No. 247 on Inquiry Card

CAE FOR LATEST VAX SYTSEMS

CAE Systems has developed CAE 2000, a design automation tool with versions for the DEC VAXstation I and VAX 8600. The program can operate in standalone mode or as a node in CAE's WorkSystem network.

CAE 2000 uses DEC's latest VMS Release 4 and DECshell. Its features include schematic capture, logic simulation, circuit simulation, timing verification, and interfaces to other design tools.

The VAXstation I brings the power of a VAX to the desktop workstation, and supports Ethernet local area networks. CAE 2000 for the VAXstation costs between $22,000 and $50,000 depending on the application.

The 8600 delivers 4.2 times the power of an 11/780. CAE for it starts at $124,000.

CAE plans on July shipment for the programs. Similar implementations for the 11/750 and 11/780 are available now.

CAE Systems, Inc., 1333 Bordeaux Dr., Sunnyvale, CA 94089, 408/745-1440.

Circle No. 248 on Inquiry Card

Software development isn't a mountainous task once you eliminate the high C errors.

When you can find and fix bugs at the earliest possible moment, creating software stops being such an uphill grind.

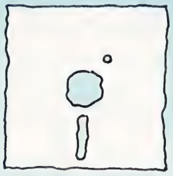
And the Smart/C Environment makes it possible. It's a complete, fully-integrated development environment for C that saves you from the creativity-inhibiting cycle of edit, compile, re-edit, re-compile, link, load, test, re-edit, re-compile, etc., ad infinitum. Smart/C puts the fun back in programming, because you spend your time creating... not waiting.

Here's why. Syntax errors are eliminated automatically as code is entered. Smart/C's highly integrated editor and interpreter allow you to interpret your program at any time in the creation process, so logic errors can be ferreted out as soon as the algorithm exists—long before any compile, link, or load.

The complete integration of the editor and interpreter means you can stop anywhere in the interpret cycle, edit, and then go right back into the interpreter exactly where you left off. Not only that, the screen-oriented user interface lets you see all operations, even interpretation, right on the listing of the code.

And to make maintenance programming easier, Smart/C's Migrator allows existing C code produced with any editor to be modified and run within the Smart/C Environment.

All of which makes Smart/C an excellent tool. It's flexible, non-restrictive, and lets you create elegant, readable, error-free programs that you can watch run with a great feeling of satisfaction.



Smart/C™

Free Demo Disk!

To fully appreciate Smart/C, you have to see it in action. For your free IBM PC MS-DOS demo disk, call us. Or write us on your company letterhead.
AGS Computers, Inc., Advanced Products Division,
1139 Spruce Drive, Mountainside, NJ 07092.
800-AGS-1313. In NJ, 201-654-4321.

Smart/C Features

The Smart/C Environment
- ☐ Fully integrated editor and interpreter
- ☐ Only one load brings them both in
- ☐ One command set
- ☐ Move between one another at will

Syntax Directed Editor
- ☐ vi-like command set
- ☐ Automatically provides formats for blocks, *for*, *case* and *if* statements

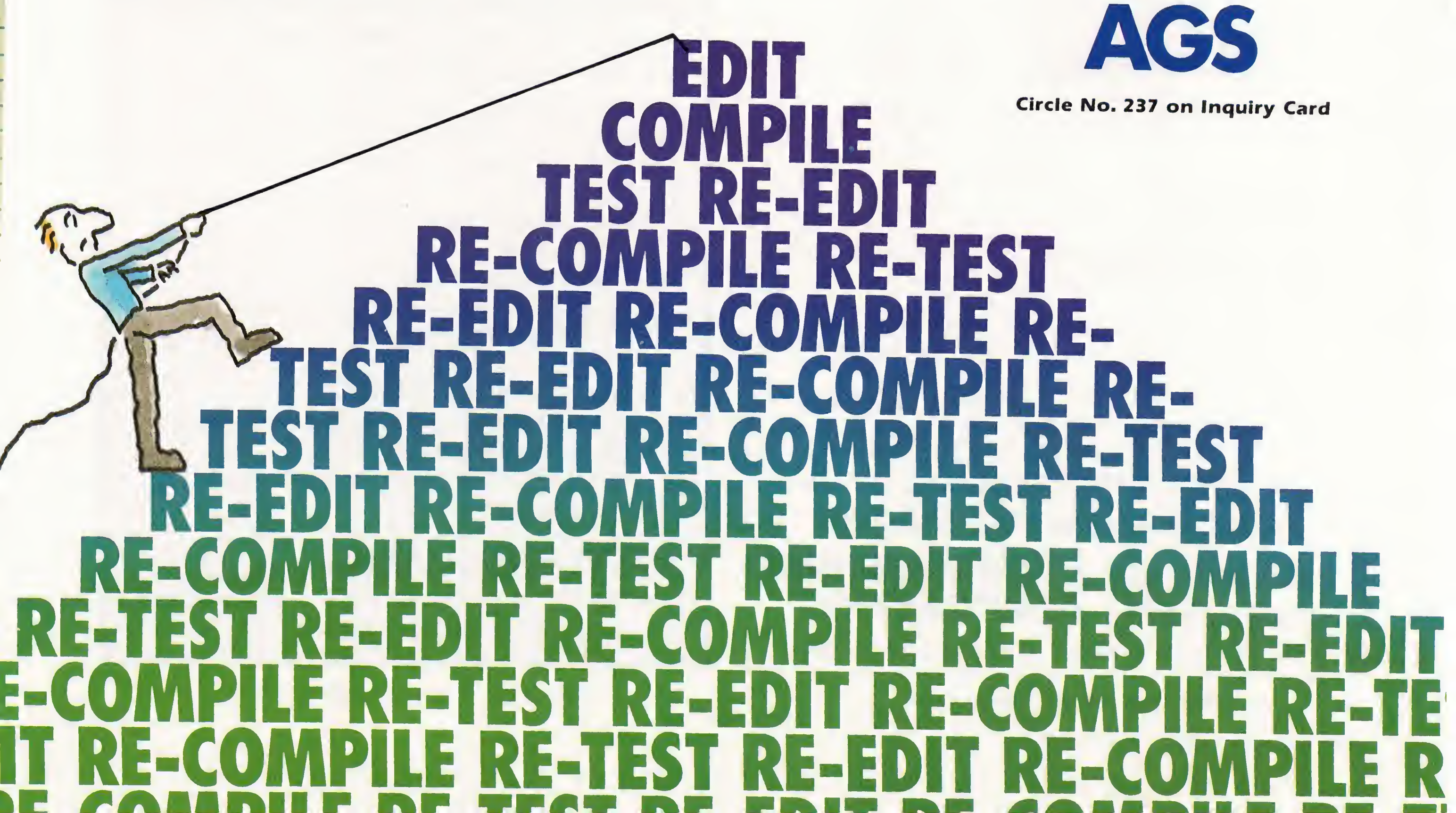
Interpreter
- ☐ Current module can call external modules during interpretation
- ☐ Has Include capability
- ☐ Totally precompilation—no incremental compile
- ☐ Can interpret partially defined files allowing for rapid prototyping
- ☐ Variable speed of interpretation
- ☐ Multiple windows with user-defined sizes

The Smart/C Migrator
- ☐ Allows C code produced with any editor to be interpreted by Smart/C
- ☐ Reformats for readability

Smart/C has been ported to UNIX™ System V Release 2, Berkeley 4.2, Xenix,™ and MS-DOS. Versions run on 8086- and 68000-based machines, as well as proprietary architectures. Smart/C runs on PCs, micros, supermicros, minis, and even mainframes.

AGS

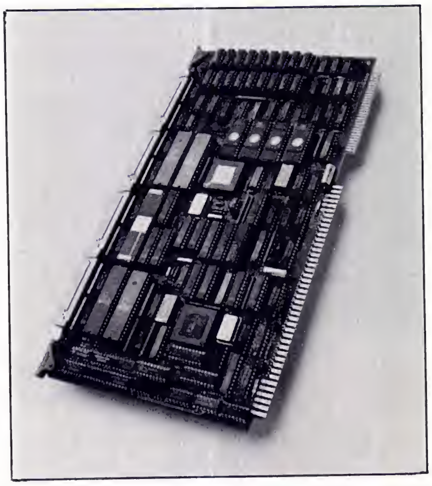
Circle No. 237 on Inquiry Card



OCTAL SERIAL BOARD SUITED TO 68000 UNIX SYSTEMS

SBE, Inc.'s M68COM octal serial board is an addition to the company's ModulasTen line of Multibus-compatible products. The board provides eight independent full-duplex channels, and is configured with either a 16/32-bit MC68000 or 68010 microprocessor. It runs at 10 MHz with no wait states for memory reads or writes.

The board is especially suited for 68000 UNIX-based systems, says SBE. It can be ordered with the UNIX-like Regulus operating system, and comes with the Probug debugger. Each channel on the board has three interrupt sources, as well as a "mailbox interrupt" that allows another Multibus master to generate a local interrupt.



SBE M68COM octal serial board

Four multiprotocol serial communications controllers on the board allow each of the eight channels to be programmed independently to run in synchronous, byte synchronous, and bit synchronous modes. The board supports X.25, SDLC, HDLC, and Bisync protocols. Each of the eight ports can be configured independently as RS232, RS422, RS423, fiber optic or direct-connect modem channels. On-board RAM is either 128K or 512K. Four other sockets accommodate as much as 256K of EPROM.

A 68230 parallel interface/timer (PI/T) chip provides an onboard timer, generating periodic system interrupts. The board supports programmable data transfer rates between 5 and 38.4K bits per second (bps), and external rates between 0 and 880K bps.

The board, configured with a 10-MHz 68000, 128K RAM, Probug, and four-channel DMA controller is $1,595 for a quantity of 100.

SBE, Inc., 2400 Bisso Lane, Concord, CA 94520, 415/680-7722.

Circle No. 230 on Inquiry Card

FSP MEANS FULL SCREEN PROCESSOR

FSP is the name of a new full screen processor for UNIX and XENIX-based systems from Information Concepts. It allows programmers to parse and display a Panel using a series of subroutines that can be called from any language. A Panel is developed by creating the screen image under any system editor, according to Information Concepts. Through the subroutines, functions such as Panel parsing, screen text display, screen field display, field validation, and post input response can be performed.

By calling FSPINP, an input processing routine, programmers can turn control over to the FSP screen manager to control the full screen data entry and editing process.

FSP also provides an Entry feature, permitting end-users to display an FSP panel and edit a standard operating system file.

A single license copy for FSP costs $950, or $240 annually on a lease.

Information Concepts, Ninth Floor, 1331 H Street, NW, Washington, DC 20005, 202/628-4400.

Circle No. 233 on Inquiry Card

REVISED C COMPILER FOR 8086 FAMILY

Whitesmiths, Ltd. has released an enhanced version of its C compiler for all systems with Intel 8086-family processors, including the 80186, 80286, and 80287 math co-processor.

Known as Version 3.0, the compiler reintroduces enhanced Pascal as an option. All compilers now also include systems interface library source code as a standard feature.

Other enhancements include: compiler and assembler source listings (including the ability to see high-level source code, assembler source code and generated code in one listing); source-level interactive debugging with breakpointing and variable display; closer adherence to an emerging ANSI C standard (including struct assignment enumerations, long identifiers, and the ability to declare arguments to a function for coercion and type checking); full ISO/OSI Level 1 implementation of Pascal including conformant array parameters; Pascal extensions such as string data type, the ability to open files by name, and an "otherwise" clause in case statements.

The design of Version 3.0 is based on architecture common to the company's entire compiler family, for the DEC PDP-11 and VAX, Intel 8080, Motorola 68000 and IBM 370 systems.

Whitesmiths, Ltd., 97 Lowell Rd., Concord, MA 01742, 617/369-8499.

Circle No. 234 on Inquiry Card

XENIX WINDOWS

A windowing interface called Viewnix from Five Paces Software allows IBM PC/XTs and ATs running XENIX to configure up to 10 windows. The windows can be expanded, contracted or moved as needed. Cut-and-paste capability is provided. The package's programming interface lets applications take control of windows in which they're placed.

Viewnix retails for $249.

Five Paces Software, Inc., 9635 Wendell Rd., Dallas, TX 75243, 214/340-4933.

Circle No. 229 on Inquiry Card

Circle No. 231 on Inquiry Card

Circle No. 232 on Inquiry Card


TCP/IP
for UNIX*
SYSTEM V
HAS ARRIVED
NEVER BEFORE AVAILABLE!

Creators of COMPUTER LANGUAGE Sponsor the C Expert Forum

C Seminar/ Workshop
September 16-18
Cambridge, Mass

Never before have so many leaders in the C programming field gathered for one event. The C Seminar/ Workshop will be an exciting forum on the latest technical innovations and C language developments. Best of all, you'll experience a practical, hands-on approach in small workshop sessions. All this in the beautiful autumn foliage of New England, just four blocks from Harvard Yard. The C Seminar/Workshop is brought to you by the publishers of COMPUTER LANGUAGE.

The cost for this comprehensive 2½ day event is only $695. Sign up by June 30th and receive a $100 early bird discount.

CURRICULUM

Speakers

Jim Brodie, *ANSI C committee chairman:* Overview of the ANSI Standardization Effort
P.J. Plauger, *author, ANSI C committee secretary:* Programming Style and C
Larry Rosler, *ANSI C language chairman:* Language Standardization Issues
Tom Plum, *author:* Efficiency of C Programs
Heinz Lycklama, */usr/group UNIX chairman:* UNIX Perspective on C
Leor Zolman, *compiler writer:* Porting C Programs between Operating Systems
Robert Ward, *C User's Group coordinator:* Structured Methods of Debugging C

Workshops

(Subject to change based on availability)

Seminar participants will be able to choose four from this list:

Debugging Techniques
Interpreters in a Development Environment
Programming for Portability
Efficient Code Generation
Cross Compilers
Network Data Base Theory and C
Object-File Formats for UNIX Systems
Philosophy and Methodology of Benchmarks
ANSI Standards: Questions & Answers
Code Readability and Organization
Asynchronous Communications
Writing Extensions to C
C / UNIX System Subroutine Interfaces
Porting C between CP/M, MS-DOS, and UNIX

C Seminar/Workshop Registration Form

Please enroll me in the C Seminar:

☐ Early Bird $595 (pay by 6/30/85)
☐ Single $695
☐ Multiple
(3 or more enrollments get $100 discount)
☐ I do not wish to enroll at this time but please send me more information.

Method of Payment:

☐ Check Enclosed
☐ Bill My Company

Make check payable to:
C.L. Publications Inc.

Name & title ____________________
Name & title ____________________
Name & title ____________________
Company ____________________
Address ____________________
City, State, Zip ____________________
Phone ____________________

UNIX

COMPUTER LANGUAGE Seminar
131 Townsend St.
San Francisco, Calif. 94107
(415) 957-9353

Circle No. 278 on Inquiry Card

WHYS

Continued from Page 27

working at all, and 4.2BSD networking is based primarily on the ARPAnet model.

This has two consequences. First, even the best applications that are totally portable across all Unices fail to make good use of the facilities of a personal workstation. Second, there is no universally accepted UNIX standard for graphics, windows, and networks. Hence, an application that wants to remain portable, even in the UNIX realm alone, must first isolate the system dependencies in libraries, much as Fortran programs that needed non-standard file system facilities used to isolate those dependencies.

Distributed resource sharing is becoming more popular—largely because, in conjunction with personal workstations, it is a powerful way of doing computing. In its current state, the field can be confusing to sort out since the requirements for resource sharing systems are not yet exactly clear. So new is the area that commonly agreed-upon buzzwords and lists of "knock-off" features have yet yet to be developed. The fact that this magazine issue is devoted to the topic may signify that a new phase is upon us.

REFERENCES

[1] E. Lazowska, H. Levy, G. Almes, M. Fischer, R. Fowler, and S. Vestal; "The Architecture of the Eden System"; *Proceedings of the Eighth Symposium on Operating Systems Principles*; December, 1981; pp. 148-159.

[2] P. Leach, P. Levine, B. Douros, J. Hamilton, D. Nelson, and B. Stumpf; "The Architecture of an Integrated Local Network"; *IEEE Journal on Selected Areas in Communication*; Vol. SAC-1 No. 5; November, 1983; pp. 842-857.

[3] N. Meyrowitz and A. van Dam; "Interactive Editing Systems, part II"; *ACM Computing Surveys*; Vol. 14 No. 3; September, 1982; pp. 353-416.

[4] G. Popek, B. Walker, J. Chow, D. Edwards, C. Kline, G. Rudisin, and G. Thiel; "LOCUS: A Network Transparent, High Reliability Distributed System"; *Proceedings of the Eighth Symposium on Operating Systems Principles*; December, 1981; pp. 169-177.

[5] W. Stallings; "Local Networks"; *ACM Computing Surveys*; Vol. 16 No. 1; March, 1984; pp. 3-42.

[6] H. Sturgis, J. Mitchell, and J. Israel; "Issues in the Design and Use of a Distributed File System"; *Operating Systems Review*; Vol. 14, No. 3; July, 1980.

[7] I. Traiger, J. Gray, C. Galtier, and B. Lindsay; "Transactions and Consistency in Distributed Database Systems"; *ACM Transactions on Database Systems*; Vol. 7 No. 3; September, 1982; pp. 323-342.

Paul J. Leach is a Senior Consulting Engineer at Apollo Computer, Inc., where he has helped design the DOMAIN system. His undergraduate work was done at MIT. Prior to becoming one of the founding engineers at Apollo, Mr. Leach worked in the research department at Prime Computer on computer architecture and operating systems. He has also served as an adjunct faculty member at Boston University. ■

A NEW DIMENSION IN DATABASE MANAGEMENT

MISTRESS is the fully relational database management system (RDBMS) for UNIX.* It features the Structured Query Language (SQL*) for the end user as well as standard programming interfaces to the C language for the DP professional. Advanced concepts include variable-length character fields, dynamic storage allocation, and B+ Tree indexing. MISTRESS has been designed exclusively for the UNIX environment and is totally written in C.

MISTRESS/32 is the advanced relational database management system for extended addressing UNIX products. MISTRESS/32 features enhanced capabilities for security, recovery and data integrity, as well as a fully integrated report writer and screen interface. MISTRESS/32 is the recommended system for more demanding applications.

*UNIX is a trademark of Bell Labs. IBM and SQL are trademarks of International Business Machines.

RHODNIUS Incorporated

10 St. Mary Street, Toronto, Ontario, Canada M4Y 1P9 (416) 922-1743 Telex: 06-986766 TOR.

Circle No. 288 on Inquiry Card

PROCEDURE CALLS

Continued from Page 42

richer form of system interaction by allowing users to group interactions in terms of more finely grained procedure calls. The advent of readily available, high-speed local communications and high performance microprocessors has sparked a renewed interest in the practical implementation of remote procedure calls.

The Xerox Courier remote procedure call protocol is representative of technologies available in the late 1970s. Courier takes a transaction-based approach in which each transaction corresponds to a remote call. The approach is rigid in the areas of execution environment management, procedure call identification and parameter encoding. While this rigidity makes the Courier simple to implement, the user needs to either increase the speed of transaction exchange or improve parameter encoding to limit system-level overhead in processing each transaction.

If remote procedure call protocols are to be more widely used in the 1980s, end-user performance will need to be improved. Alternatives for improving performance include the decoupling of transactions and remote executions by adding intelligence to each transaction. At minimum, this approach should allow users to add a processing control capability to each transaction by combining multiple, conditionally executed calls into each transaction.

This alternative can be expanded further to include executable software in each transaction. By programmatically examining the results of each procedure call at the execution site, these alternatives offer the possibility of making multiple calls without the need for intervening transactions. Increased performance can thereby result from a decreased number of transactions.

Remote procedure call technology is still in its infancy. A number of alternative improvements will be investigated and discarded before the concept attains widespread use. Along the way it is important that the concept be marketed on the basis of sound, demonstrated capabilities. The remote procedure call approach holds great promise, but to sell someone potential rather than actual capability is a serious mistake with ramifications for the entire industry.

Steve Holmgren is President of Communication Machinery Corporation of Santa Barbara, CA, a producer of high performance communication software and hardware. Prior to coming to CMC, Mr. Holmgren served as President of QMI, where he developed a single-chip TCP implementation. He holds a Bachelor's degree in Mathematics and Computer Science from the University of Illinois in Champaign-Urbana, where he went on to interface UNIX to the ARPAnet at the Center for Advanced Computation. He has also done advanced technical assessments and prototyping for military procurements through the Mitre Corporation. ■

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 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-based machines**

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.

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

Circle No. 286 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

FILE SYSTEMS

Continued from Page 33

propagate immediately.

CONCLUSION

We have covered only some of the major issues that must be addressed in the design and implementation of a distributed file system. The presentation is by no means complete; interested readers may want to obtain more information about the NFS [2], or read references on WFS [3] and DFS [4] to learn about successful distributed file system implementations. There are many similarities in the design approaches taken by WFS, DFS, and the NFS, so there is a large body of experience to demonstrate the viability of the approach.

REFERENCES

1) RPC and XDR are discussed elsewhere in this issue.

2) D. Walsh, B. Lyon, G. Sager, et al., "Overview of the Sun Network File System", *Proceedings of the Usenix Association Winter Conference*, January, 1985.

3) D. Swinehart, G. McDaniel, and D. Boggs, "WFS: A Simple Shared File System for a Distributed Environment", *Proceedings of the Seventh Symposium on Operating Systems Principles*, ACM order number 534790, December, 1979.

4) H. Sturgis, J. Mitchell and J. Israel, "Issues in the Design and Use of a Distributed File System", *Operating Systems Review*, Vol 14, No 3, July, 1980.

5) Documentation and user-level source for RPC and XDR are being posted to *net.sources*.

Gary Sager is Manager of the Systems Group at Sun Microsystems, Inc. Sager has worked for Bell Laboratories and AT&T Information Systems as principal architect of the Onyx/Pecos operating system; for LASL as a contributor to the DEMOS operating system; and for the Unversity of Waterloo as a co-developer of the Thoth operating system.

Bob Lyon is Project Leader for the Network File System at Sun Microsystems, Inc. Previous to Sun, Lyon worked at Xerox Corp. on the XNS transport protocols and the Clearinghouse network service. He also has worked at Bell Laboratories as a UNIX system administrator. ■

IBM LINKS

Continued from Page 59

interface to IBM hosts for application-to-application communications in the SNA environment, which is the norm today for IBM hosts since IBM no longer places emphasis on Bisync architectures. However, since it will be late 1985 or early 1986 before such interfaces are generally available for UNIX systems and applications, this should be considered more for planning than for immediate implementation.

SUMMARY

All three of the interfaces to IBM hosts mentioned here have their place in different environments. The interactive facility will be used by those who want to use their UNIX systems and local applications most of the time, but also need to query databases on IBM hosts from time to time. The batch facility allows a UNIX system to offload the host as a development facility. One possible scenario might find development and basic testing of COBOL applications occurring on a UNIX system, while full-scale testing with larger databases is left to the host. As previously mentioned, the ease of using this facility makes a batch application interface to the host a quick-and-dirty but fully functional job.

Finally, the application-to-application interface potential of *LU type 6.2* promises to provide a mature, professional facility for data and general resource sharing in heterogeneous network environments containing UNIX and IBM processors.

David L. Buck is Chairman of D. L. Buck and Associates, Inc., of San Jose, CA, a company that supplies hardware manufacturers with UNIX applications and drivers to interface their systems with IBM systems. Mr. Buck has been quite active in the /usr/group Standards Committee and the recently-formed IEEE P1003 Committee on standardizing the interface to UNIX and UNIX-compatible operating systems. As part of his work with the IEEE committee, Mr. Buck chairs the subcommittee on Systems Interface. He also offers seminars and classes on communications and UNIX. ■

CALENDAR

EVENTS

MAY

May 1 Yates Ventures, San Jose, CA: "AT&T, IBM, and UNIX: Desktop Directions". Contact: Glenn Chase, 3350 W. Bayshore Rd., Suite 201, Palo Alto, CA 94303. 415/424-8844.

JUNE

June 11-14 Usenix Association, Portland: "Usenix Conference and Vendor Exhibition". Contact: Usenix Conference Office, P.O. Box 385, Sunset Beach, CA 90742. 213/592-3243.

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. Contact: Computer Faire, Inc., 181 Wells Ave., Newton, MA 02159. 617/965-8350.

OCTOBER

October 2-5 UNIX Systems EXPO, Boston: "EXPO/85". Contact: Computer Faire Inc., 181 Wells Ave., Newton, MA 02159. 617/965-8350.

TRAINING

MAY

May 1-3 Interactive Systems Corp., Santa Monica, CA: "Customizing Ten-Plus". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404-9989. (213) 453-8649.

May 1-3 Center for Advanced Professional Education, Rochester, NY: "UNIX: A User-Oriented Evaluation Seminar". Contact: Center for Advanced Professional Education, 1820 E. Garry St., Suite 110, Santa Ana, CA 92705. 714/261-0240.

May 1-3 Computer Technology Group, London: "Using Advanced UNIX Commands". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 3 Specialized Systems Consultants, Bellevue, WA: "UNIX for Managers". Contact: SSC, P.O. Box 7, Northgate Station, Seattle, WA 98125. 206/367-8649.

May 6 NCR Corp., New York: "UNIX Operating System". Contact: NCR Corp., CASE-Special Orders, 101 W. Schantz Ave., Dayton, OH 45479. 800/845-2273 or 800/841-2273.

May 6-7 Computer Technology Group, Chicago: "Shell Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 6-7 Computer Technology Group, Los Angeles: "Shell Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 6-10 Interactive Systems Corp., Santa Monica, CA: "Ten-Plus Helper Writer Workshop". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404-9989. (213) 453-8649.

May 6-10 Bunker Ramo Information Systems, Trumbull, CT: "Advanced C". Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

May 7-9 Computer Technology Group, Boston: "UNIX Administration". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 7-9 Computer Technology Group, Washington, D.C.: "UNIX Administration". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 7-10 Integrated Computer Systems, Washington, D.C.: "UNIX: A Hands-on Introduction". Contact: Integrated Computer Systems, 6305 Arizona Pl., P.O Box 45405, Los Angeles, CA 90045. 213/417-8888.

May 7-10 Computer Technology Group, London: "UNIX Internals". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 8-10 Center for Advanced Professional Education, Denver, CO: "UNIX: A User-Oriented Evaluation Seminar". Contact: Center for Advanced Professional Education, 1820 E. Garry St., Suite 110, Santa Ana, CA 92705. 714/261-0240.

May 8-10 Computer Technology Group, Los Angeles: "Using Advanced UNIX Commands". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 8-10 Computer Technology Group, Chicago: "Using Advanced UNIX Commands". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 13 NCR Corp., Los Angeles: "UNIX Operating System". Contact: NCR Corp., CASE-Special Orders, 101 W. Schantz Ave., Dayton, OH 45479. 800/845-2273 or 800/841-2273.

May 13-14 Computer Technology Group, Boston: "Advanced C Programming Workshop". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 13-14 Computer Technology Group, Washington, D.C.: "Advanced C Programming Workshop". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 13-15 Interactive Systems Corp., Santa Monica, CA: "UNIX Fundamentals". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404-9989. (213) 453-8649.

May 13-17 Computer Technology Group, Chicago: "UNIX Internals". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 13-17 Computer Technology Group, Los Angeles: "UNIX Internals". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 13-17 Bunker Ramo Information Systems, Trumbull, CT: "Intro to UNIX". Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

IN A PLACE WHERE YOU CAN STILL FIND QUIET . . .

A WORKPLACE SHAPED BY LISTENING.

Most thinking people, whether Unix system professionals or supporting staff members, would prefer to work with a management team that's genuinely interested in their ideas.

Managers who'll be sure you get information about decisions that affect you, and then listen to how you feel about them.

Simple concepts . . . but finding a company that puts them into practice isn't easy. It's as tough as finding a quiet place to think.

You can in Utah.

At Sperry's Unix-based Micro Products Division, we have opportunities available in: UNIX Software programming, Systems design engineering, systems programming, SNA, DCA, OSI communications architectures, LAN technology communication, software engineering, and many others.

If you are interested, please send your resume to:
SPERRY
Employment Dept.
Attn: Dept. 1J
322 N. Sperry Way
Salt Lake City, Utah 84116

We are an equal opportunity employer m/f/v/h.

SPERRY

Circle No. 274 on Inquiry Card

May 13-24 Information Technology Development Corp., Cincinatti: "The UNIX System" and "The C Programming Language". Contact: ITD, 9952 Pebbleknoll Dr., Cincinatti, OH 45247. 513/741-8968.

May 15-17 Center for Advanced Professional Education, Silver Spring, MD: "UNIX: A User-Oriented Evaluation Seminar". Contact: Center for Advanced Professional Education, 1820 E. Garry St., Suite 110, Santa Ana, CA 92705. 714/261-0240.

May 15-17 Computer Technology Group, Boston: "Advanced C Programming Under UNIX". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 15-17 Computer Technology Group, Washington, D.C.: "Advanced C Programming Under UNIX". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 15-17 Computer Technology Group, London: "UNIX Administration". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 15-17 Digital Equipment Corp., Denver: "Comprehensive Overview of the UNIX Operating System". Contact: Digital Education Resources, 12 Crosby Drive, Bedford, MA 01730. 617/276-4949.

May 16-17 Interactive Systems Corp., Santa Monica, CA: "Using the Shell". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404-9989. (213) 453-8649.

May 20 NCR Corp., Los Angeles: "UNIX System Administration". Contact: NCR Corp., CASE-Special Orders, 101 W. Schantz Ave., Dayton, OH 45479. 800/845-2273 or 800/841-2273.

May 20-21 Interactive Systems Corp., Santa Monica, CA: "Advanced UNIX Commands for Programmers". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404-9989. (213) 453-8649.

May 20-21 Computer Technology Group, London: "Advanced C Programming Workshop". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 20-22 Center for Advanced Professional Education, San Francisco: "UNIX: A User-Oriented Evaluation Seminar". Contact: Center for Advanced Professional Education, 1820 E. Garry St., Suite 110, Santa Ana, CA 92705. 714/261-0240.

May 20-24 Computer Technology Group, Boston: "Berkeley Fundamentals and 'csh' Shell". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 20-24 Computer Technology Group, Washington, D.C.: "Berkeley Fundamentals and 'csh' Shell". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 20-24 Bunker Ramo Information Systems, Trumbull, CT: "Advanced UNIX". Contact: Bunker Ramo, Trumbull Industrial

OUR 5620 TERMINAL WILL CHANGE THE WAY YOU VIEW YOUR UNIX* SYSTEM.

If you want to get the most out of your UNIX System, get the single terminal that performs like six—the 5620 Dot-Mapped Display from Teletype Corporation. This terminal is an exceptional value because it offers graphics and capabilities you'd expect to find only in higher-priced workstations.

With its unique capability to divide the display into six windows, the 5620 makes excellent use of UNIX System V resources to greatly improve productivity. You create and control the size and position of each window simply, using the electronic "Mouse." Unmodified host programs then view the windows as separate terminals, making it possible for you to work on several things at once.

Just think how much easier that'd make it for you to prepare documents and graphics, do computer aided engineering or write and debug programs. Imagine, for example, while a program is compiling in one window, you edit the source code in a second window, check output in a third window, and send and receive mail concurrently in a fourth window.

The 15-inch diagonal display boasts 100 dots per inch, which gives you high resolution graphics and font capabilities. Complete with a full 32-bit processor and up to one megabyte of memory, you can also offload the host by running programs in the 5620.

As good as the 5620 makes Teletype Corporation look, it can make you look even better. To find out how, write Teletype Corporation, 5555 Touhy Ave., Dept. 3223-OO, Skokie, IL 60077. Or call **1 800 323-1229, ext. 615.**

TELETYPE: VALUE SETS US APART.

"Teletype" is a registered trademark and service mark of Teletype Corporation.
*UNIX is a trademark of Bell Telephone Laboratories, Inc.

AT&T
Teletype Corporation

Circle No. 273 on Inquiry Card

Park, Trumbull, CT 06611. 203/386-2223.

May 20-24 Specialized Systems Consultants, Bellevue, WA: "C Programming Workshop". Contact: SSC, P.O. Box 7, Northgate Station, Seattle, WA 98125. 206/367-8649.

May 21-23 Computer Technology Group, Chicago: "UNIX Administration". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 21-23 Computer Technology Group, Los Angeles: "UNIX Administration". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 22-24 Interactive Systems Corp., Santa Monica, CA: "UNIX Architecture: The Conceptual Overview". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404-9989. (213) 453-8649.

May 22-24 Computer Technology Group, London: "Advanced C Programming Under UNIX". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 28 Computer Technology Group, New York: "UNIX Overview". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 28 Computer Technology Group, Washington, D.C.: "UNIX Overview". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 29-31 Center for Advanced Professional Education, Costa Mesa, CA: "UNIX: A User-Oriented Evaluation Seminar". Contact: Center for Advanced Professional Education, 1820 E. Garry St., Suite 110, Santa Ana, CA 92705. 714/261-0240.

May 29-31 Computer Technology Group, Washington, D.C.: "UNIX Fundamentals for Non-Programmers". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

May 29-31 Computer Technology Group, New York: "UNIX Fundamentals for Non-Programmers". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

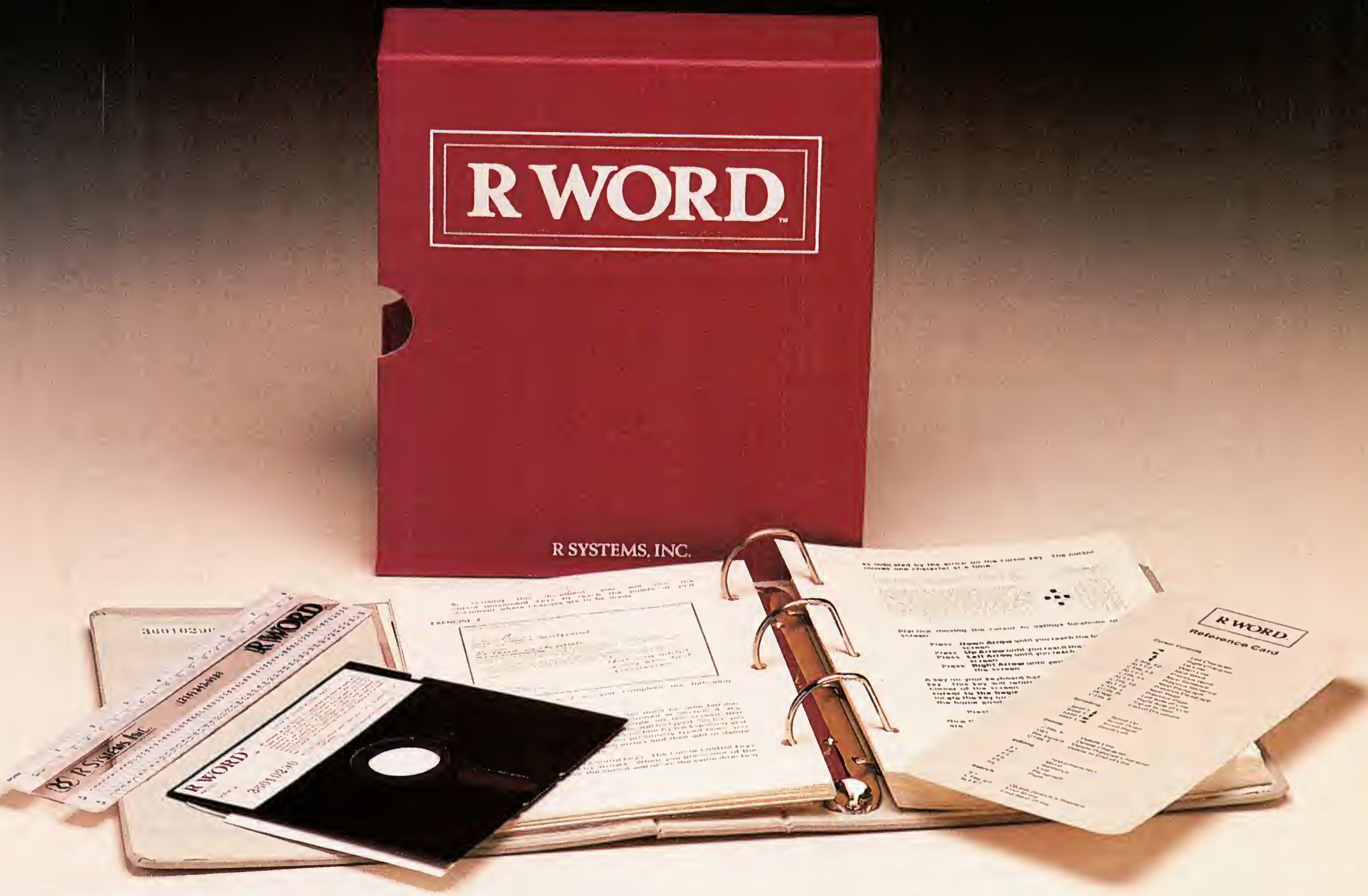
JUNE

June 3 NCR Corp., Chicago: "C Programming". Contact: NCR Corp., CASE-Special Orders, 101 W. Schantz Ave., Dayton, OH 45479. 800/845-2273 or 800/841-2273.

June 3-4 Specialized System Consultants, Seattle: "Hands-on UNIX for Non-Technical People". Contact: SSC, P.O. Box 7, Northgate Station, Seattle, WA 98125-0007. 206/367-UNIX.

June 3-4 Interactive Systems Corp., Santa Monica, CA: "System Administrator's Overview". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404-9989. (213) 453-8649.

June 3-4 Computer Technology Group, Los Angeles: "Advanced C Programming Workshop". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.



"R WORD is an excellent program. It is one of the easiest, yet full-featured, word processors I have ever used."

Bob Woodard of the University of Iowa continues in his letter to us about R WORD, "It beats the heck out of the word processing package available with our machine. R WORD is definitely the favorite of the people who have tried it."

We've received many letters like Bob Woodard's, praising the R Family of Office Automation Software. Now, we'd like **you** to meet R Family — three programs which are powerful, easy-to-use, well-documented, cost-effective, and supportable. And they're designed for single-users, multi-users, and multi-computer users.

R Office Manager

Suggested List $295

R Office Manager is the most powerful organizational and managerial package available. It consolidates and streamlines the basic office functions of today's complex corporate environment. R Office Manager coordinates phone messages, calendars, calculations, things-to-do lists, messages, names and addresses, and more.

R WORD II

Suggested List $895

R WORD II is designed for a production word processing environment which requires full-featured word processing, yet R WORD II offers amazing simplicity of use. Features include Help Menus, Cursor Controls, Editing Functions, Formatting, Three-Level File Structure, Printing Features, Mail Merge, Default Set Up, and Utility Functions.

The program also offers Global Search and Replace, Directories, Spelling Checker, Table Math, Headers and Footers, Automatic Table of Contents, Automatic Paragraph Numbering, Document Assembly, and much more.

R WORD III

Suggested List $1295

R WORD III offers all the features of R WORD II and R Office Manager, plus complete data base, file management, and report generation features which allow you to integrate any information found in other programs on your computer.

1-(800) 527-7610

Call us toll-free for more information about R Family. We are currently ported to and shipping R Family programs on the Tandy Model 16, NCR Tower + XP, Apple Lisa, Pixel, Plexus, and Altos 68000. Additional UNIX and XENIX ports are occurring weekly. We are also shipping for all MS-DOS and PC-DOS operating systems. Manufacture, distributor, dealer, national account, and porting inquiries are invited.

When you select word processing, office management, and office automation software, consider the benefits of R Family. This one set of programs can satisfy single-user, multi-user, and multi-computer requirements. R Family offers you consistency, versatility, and supportability, as well as the benefits of transportability and connectivity.

The R Family - single and multi-user office automation software:
DOS • UNIX • XENIX
COS • DX10 • DNOS

Circle No. 284 on Inquiry Card

R SYSTEMS, INC. **11450 Pagemill Road, Dallas, TX 75243 (214) 343-9188 (800)527-7610**

CALENDAR

June 3-4 Computer Technology Group, Chicago: "Advanced C Programming Workshop". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

June 3-5 Computer Technology Group, Washington, D.C.: "UNIX Fundamentals for Programmers". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

June 3-5 Computer Technology Group, New York: "UNIX Fundamentals for Programmers". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

June 3-7 Computer Technology Group, London: "Berkeley Fundamentals and 'csh' Shell". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

June 3-14 Information Technology Development Corp., Cincinatti: "The UNIX System" and "The C Programming Language". Contact: ITD, 9952 Pebbleknoll Dr., Cincinatti, OH 45247. 513/741-8968.

June 4-6 Bunker Ramo Information Systems, Trumbull, CT: "Diagnostic UNIX". Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

June 4-7 Integrated Computer Systems, Philadelphia: "Programming in C: A Hands-on Workshop". Contact: Integrated Computer Systems, 6305 Arizona Pl., P.O Box 45405, Los Angeles, CA 90045. 213/417-8888.

June 4-7 Integrated Computer Systems, San Diego: "UNIX: A Hands-on Introduction". Contact: Integrated Computer Systems, 6305 Arizona Pl., P.O Box 45405, Los Angeles, CA 90045. 213/417-8888.

June 5-7 Interactive Systems Corp., Santa Monica, CA: "Interactive Networking Tools". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404-9989. (213) 453-8649.

June 5-7 Computer Technology Group, Los Angeles: "Advanced C Programming Under UNIX". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

June 5-7 Computer Technology Group, Chicago: "Advanced C Programming Under UNIX". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

June 6-7 Computer Technology Group, Washington, D.C.: "Shell as a Command Language". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

June 6-7 Computer Technology Group, New York: "Shell as a Command Language". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

June 10 NCR Corp., Vandalia-Dayton: "UNIX Operating System". Contact: NCR Corp., CASE-Special Orders, 101 W. Schantz Ave., Dayton, OH 45479. 800/845-2273 or 800/841-2273.

June 10 NCR Corp., Minneapolis: "UNIX Operating System". Contact: NCR Corp., CASE-Special Orders, 101 W. Schantz Ave., Dayton, OH 45479. 800/845-2273 or 800/841-2273.

June 10-11 Bunker Ramo Information Systems, Trumbull, CT: "UNIX/C Applications". Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

June 10-14 Interactive Systems Corp., Santa Monica, CA: "The C Programming Language". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404-9989. (213) 453-8649.

June 10-14 Computer Technology Group, Los Angeles: "Berkeley Fundamentals and 'csh' Shell". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

June 10-14 Computer Technology Group, Chicago: "Berkeley Fundamentals and 'csh' Shell". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

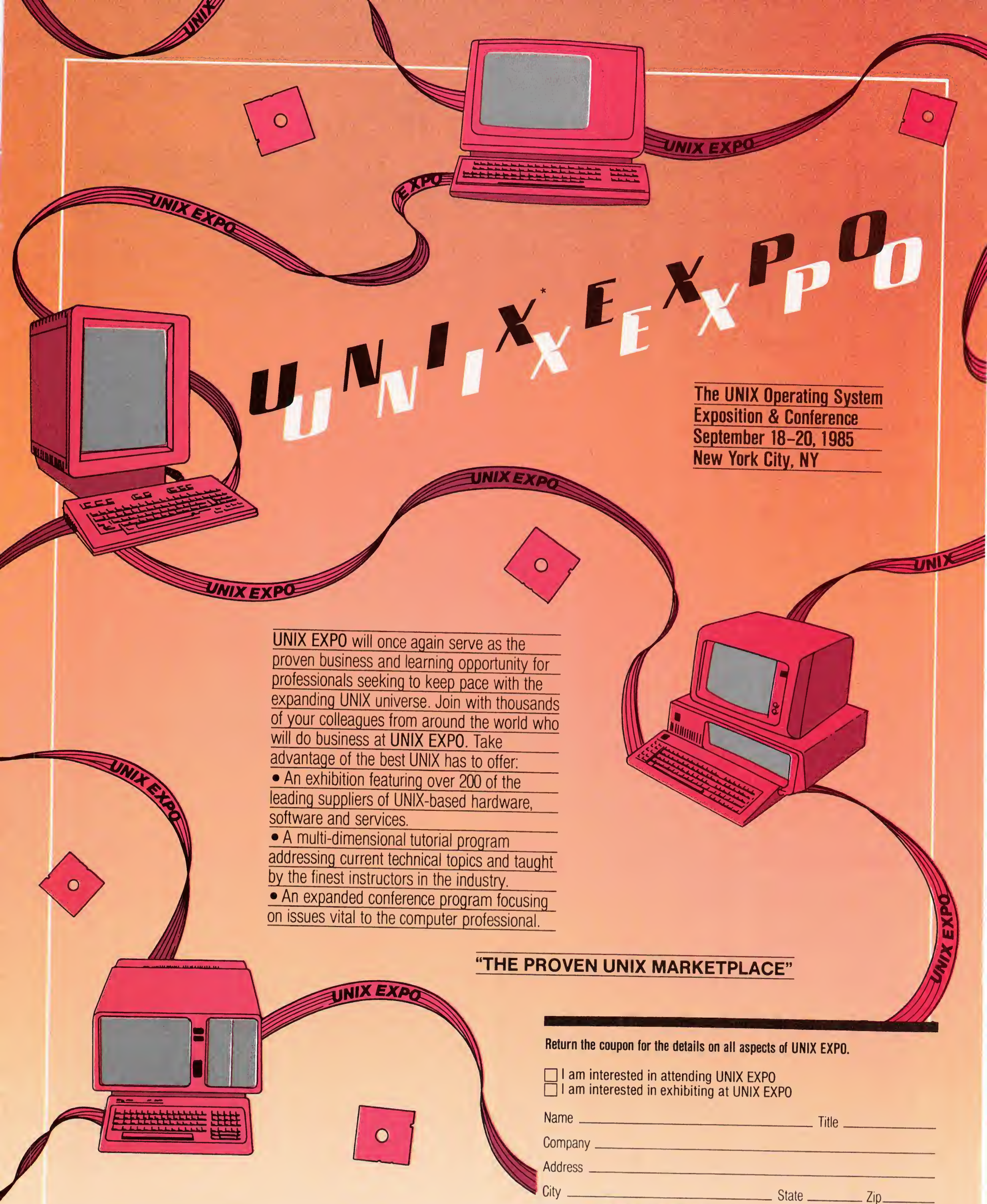
June 10-14 Computer Technology Group, Washington, D.C.: "C Language Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

June 10-14 Computer Technology Group, New York: "C Language Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

June 11 Computer Technology Group, London: "UNIX Overview". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, Ill. 60604. 800/323-UNIX.

June 11-14 Integrated Computer Systems, San Diego: "Programming in C: A Hands-on Workshop". Contact: Integrated Computer Systems, 6305 Arizona Pl., P.O Box 45405, Los Angeles, CA 90045. 213/417-8888.

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.



UNIX* EXPO

The UNIX Operating System
Exposition & Conference
September 18–20, 1985
New York City, NY

UNIX EXPO will once again serve as the proven business and learning opportunity for professionals seeking to keep pace with the expanding UNIX universe. Join with thousands of your colleagues from around the world who will do business at UNIX EXPO. 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 multi-dimensional tutorial program addressing current technical topics and taught by the finest instructors in the industry.
- An expanded conference program focusing on issues vital to the computer professional.

"THE PROVEN UNIX MARKETPLACE"

Return the coupon for the details on all aspects of UNIX EXPO.

☐ I am interested in attending UNIX EXPO
☐ I am interested in exhibiting at UNIX EXPO

Name ______________________ Title ____________
Company ______________________
Address ______________________
City ______________________ State ________ Zip ________

Return to: **National Expositions Co., Inc.**
14 West 40th Street, New York, N.Y. 10018
(212) 391-9111 • Telex 135401 DIMCOMM

*UNIX™ IS A REGISTERED TRADEMARK OF BELL LABS UNIX EXPO IS NOT AFFILIATED WITH BELL LABS

Circle No. 285 on Inquiry Card

ADVERTISER'S INDEX

COMING UP IN JUNE

UNIX Applications

- **Transparent Telecommunicatons Applications**
- **Boundary Conditions**
- **Historical Considerations**
- **Pervasive UNIX Uses**
- **Homebrewed Applications**

MOVING?

SUBSCRIBER SERVICE

Don't miss an issue of UNIX Review. Please enclose your current address label, or copy all information from the label and enclose it with your new address.

Write in new address below:

Name ____________________
Company ____________________
Address ____________________
City/State/Zip ____________________

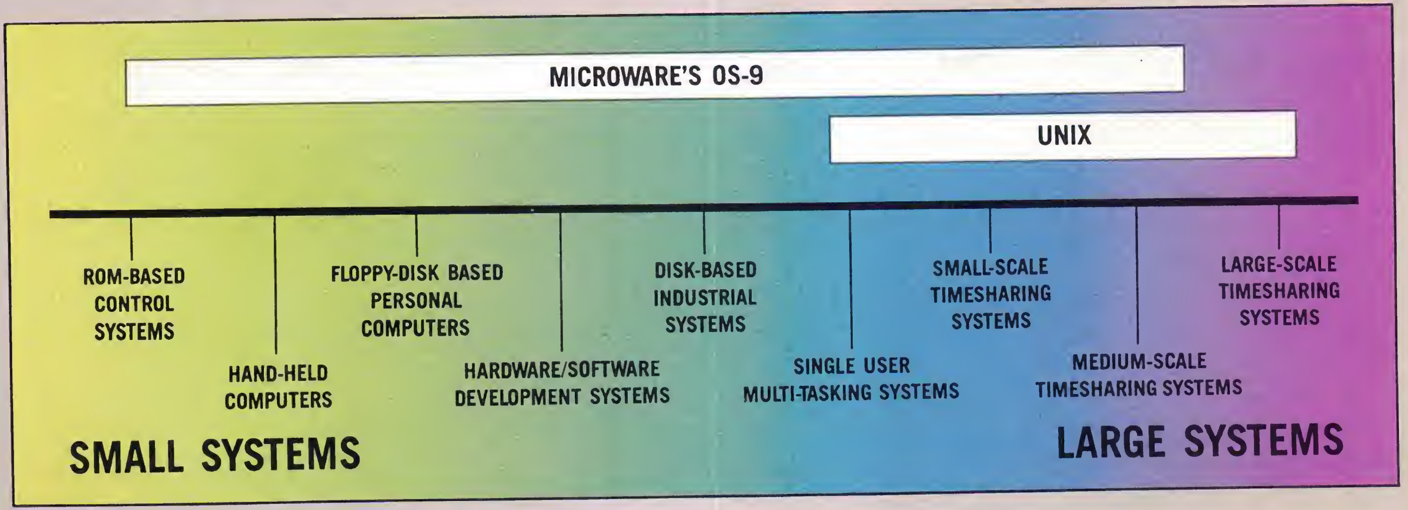
Send all subscription correspondence to:
Review Publications, P.O. Box 7439
San Francisco, CA 94120-9864

Circle No. 223 on Inquiry Card ➤

gic

A Complete UNIX™ Service

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.

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

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.

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.