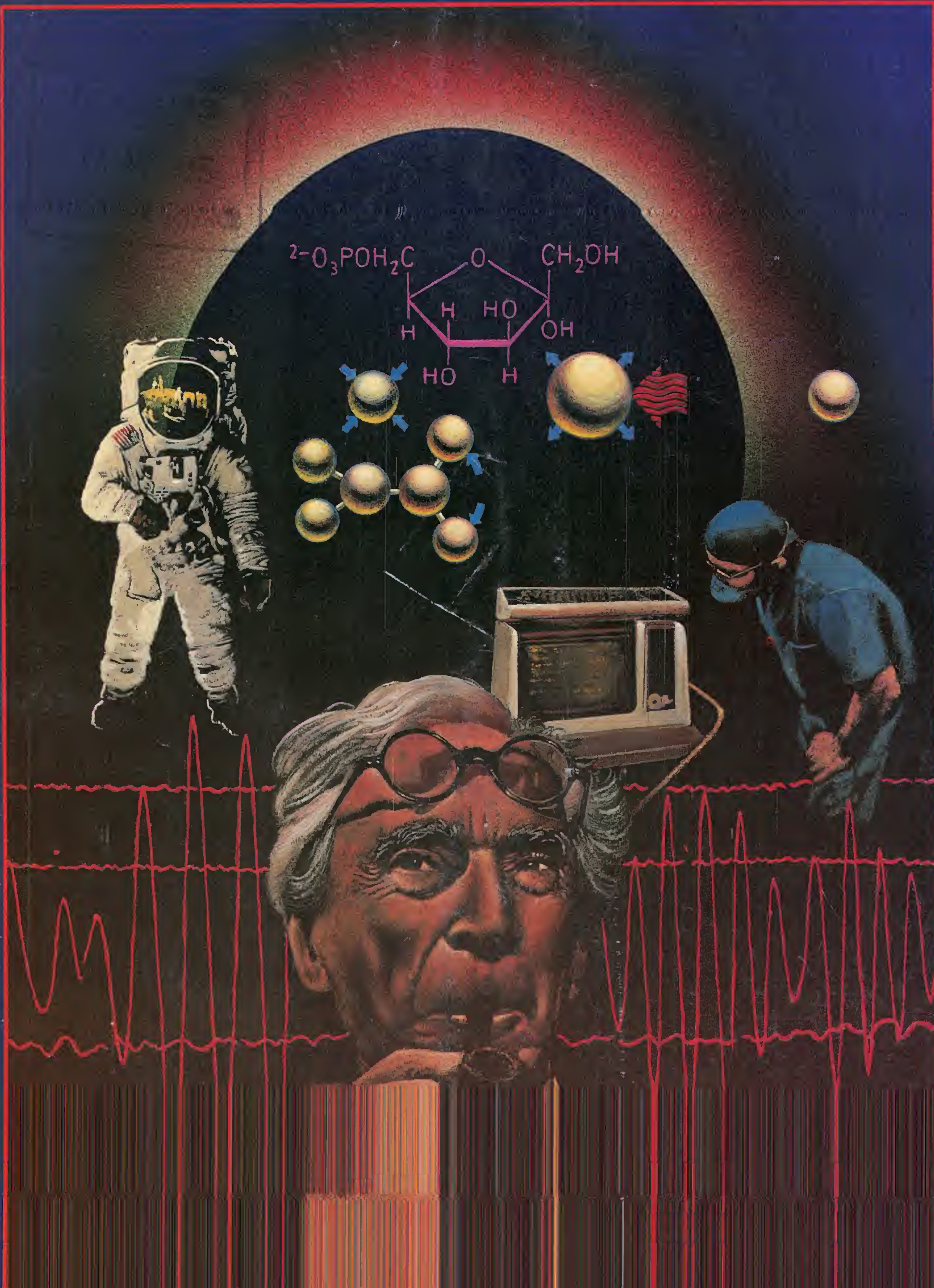
UNIX™ REVIEW

THE PUBLICATION FOR THE UNIX™ COMMUNITY

June 1985 $3.95



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

Circle No. 1 on Inquiry Card

*OEM Pricing

CYB...Single Source Solution for Systems Integration

How we improved Structured Query Language.

Actually, we didn't change a thing.

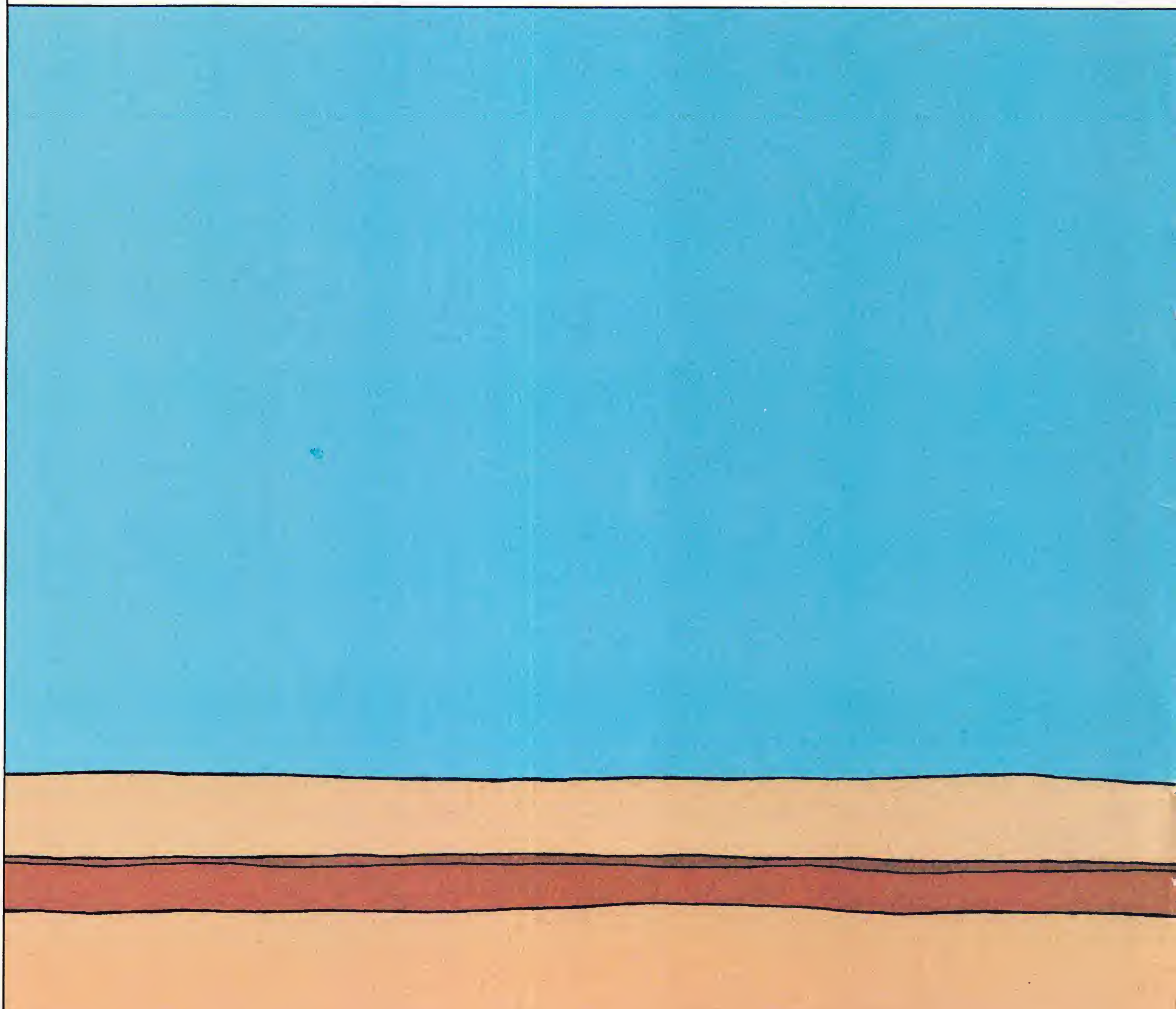
We just combined it with the best relational database management system.

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



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.

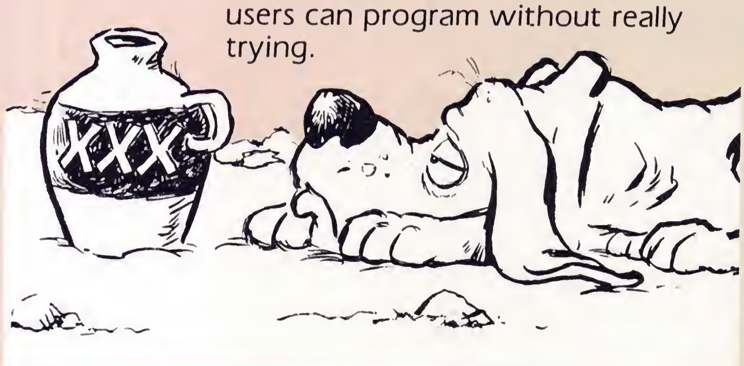
UNIX™ REVIEW
THE PUBLICATION FOR THE UNIX COMMUNITY

Volume 3,
Number 6
June 1985

DEPARTMENTS:

Cover art by Heda Majlessi

FEATURES:



UNIX APPLIED

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.

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.

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

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

BPA membership applied for in March, 1985.

VIEWPOINT

A small departure

A note attached to a recent subscription order asked, "Why do you guys take so much pleasure in beating up UNIX?" It took me by surprise.

I suppose I should have expected something of the sort since the magazine does have an independent voice (although one that I hardly think could be characterized as *hostile*). Being independent may not be what readers expect, given that trade journals typically exalt the professions they cover, but I would argue that it is responsible journalism.

Having adopted this self-righteous pose, let me now climb down from my pulpit and make a small confession. The issue you are now holding contains material that smacks of unabashed boosterism. Yes, it's true. As you make your way through the magazine, you'll read about the wonderful things that have been done with UNIX, the wonderful things that are yet to come, and the wonderful people that have made it all possible.

Why? Largely because I'm sick of hearing about what an applications wasteland UNIX is. It's just not true, folks—and it's time to put the misconception to rest. In addition to being a favored development environment, UNIX actually has been used—with great success—as an execution environment. Honest!

I'm afraid that even some systems and software developers, otherwise counted as UNIX aficionados, may be totally unaware of this side of the system. It's for these people that we dedicate this issue.

Leading the charge on popular misconceptions is Tom Athanasiou, a UNIX programmer whose survey of unusual applications includes some real eye-openers. I call them "eye-openers" not just because they demonstrate the power of UNIX, but because they demonstrate just how pervasive the system has become.

Marc Rochkind, the gentleman who brought you SCCS, offers an article that tells how one can whip together software from scratch, using the UNIX basket of tools. Some might call the approach "programming without programming". But, then again, they might not after reading Marc's piece.

Another article contributed by Peter Langston, late of Lucasfilm Ltd., tells of how UNIX has facilitated and influenced the development of computer games that stand at the vanguard. State-of-the-art in computer games? Glad you asked.

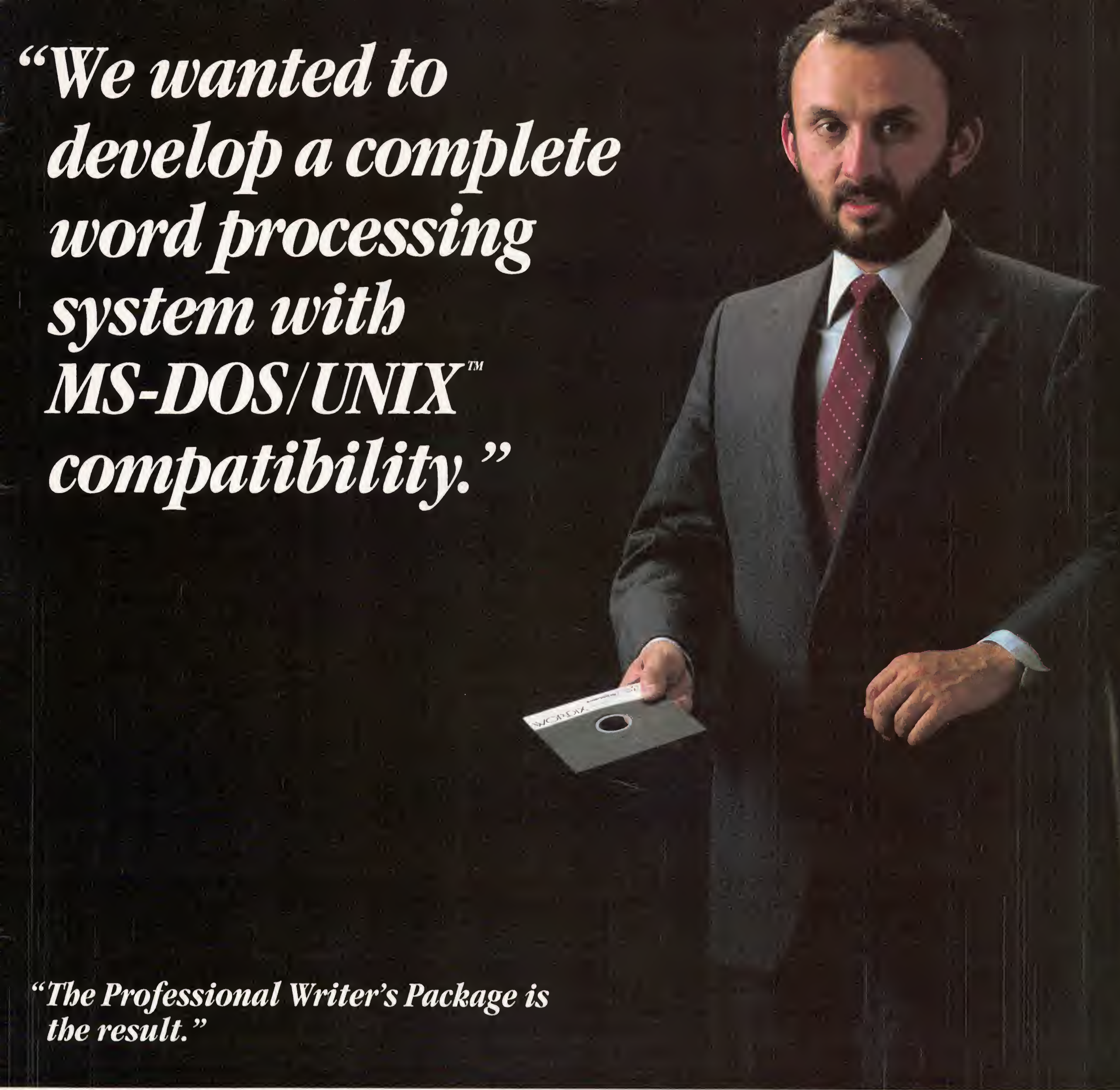
Finally, we get into some real, roll-up-your-sleeves UNIX applications with an article authored by Tony Cuilwik, head of AT&T's Information Systems Laboratories in Columbus, OH. The piece considers the variety of UNIX applications you set into motion every time you pick up a telephone. Did I hear "real-time" whispered out there? You betcha!

To conclude the issue, Ned Peirce interviews Berkley Tague, the man largely responsible for the spread of UNIX within the national telecommunications network. As founder of what was to become the UNIX Support Group, Berkley has closely tracked the evolution of UNIX applications within AT&T. He speaks about how history may well repeat itself outside the AT&T empire.

Before leaving off, acknowledgements for the contributions made by Ken Arnold, John Mashey, and Dennis Ritchie are also in order. Technical input like theirs makes the magazine possible.

But, enough of my sermonizing. It's time you partake of the these readings, taken from the true church. Let us all now sing the praises of UNIX.

Mark Compton



Iván Pérez-Méndez
President, Emerging Technology

"Drawing on years of word processing experience, we've designed a program with all of the features necessary for producing everything from simple letters and memos to complicated documents, like technical manuals and dramatic scripts.

"At the same time, we realized that you don't want to spend a lot of time learning a new program or thumbing through documentation. So the **Professional Writer's Package** comes with on-line help, on-line tutorials and a shell that takes you step by step through all phases of document preparation and file management.

"And the files created with the **Professional Writer's Package** are compatible with MS-DOS® files. Since the **Professional Writer's Package** has already been used successfully at thousands of installations, you get a proven product with proven capabilities.

"If you want to learn more about the **Professional Writer's Package**,™ call or write today."

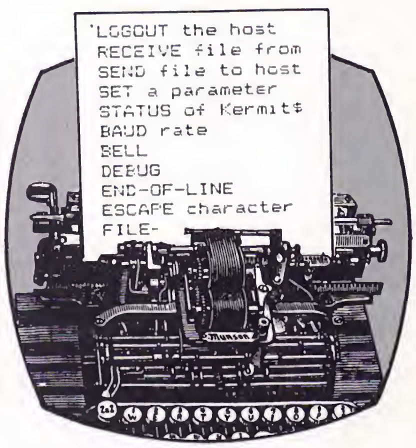
EMERGING TECHNOLOGY 4760 Walnut Street, Boulder, Colorado 80301 303/447-9495, 800/782-4896

Circle No. 72 on Inquiry Card

THE MONTHLY REPORT

Systems Expo loses heart in San Francisco

by Roger Strukhoff



Attendance at the UNIX Systems Expo/85-Spring show in San Francisco April 24 to 26 was, to put it charitably, modest. There were a number of important companies missing, most notably IBM, Hewlett-Packard, Charles River Data Systems, Gould, Intel, Masscomp, Perkin-Elmer, and Zilog. AT&T was there, but in a booth more appropriate for a modest magazine publisher than a corporate Leviathan. Turnstyle attendance totalled about 2000, according to Computer Faire, Inc., producer of the show.

Among the product announcements at the show: Interactive Development Environments (IDE) introduced its enhanced Rapid/Use prototyping and development system, its Troll/Use relational database management system (RDMS), and the Transition Diagram Editor; Unify Corp. expanded the range of systems that runs its RDMS; and Computer Consoles Inc. (CCI) announced a floating point/math accelerator for its Power 6 family.

The show's lethargy may affect Computer Faire's plans for a fall Systems Expo show, scheduled for October 3-5 in Boston. Talk on the floor indicated that Computer Faire might combine the Boston show with its 8th Northeast Computer Faire, scheduled for Boston on September 26-29. This approach may run up against its own problems, though, since the Northeast Computer Faire comes just one week after National Exhibition's UNIX EXPO in New York. Space at Boston's Bayside Exposition Center is also said to be in short supply—which is a comment both on the size of the auditorium and the popularity (and size) of the Northeast Computer Faire.

UNIX EXPO, meanwhile, is basking in the glow of AT&T's recent announcement that it will launch a series of UNIX tutorials at that show. This will be a first for AT&T.

"For 15 years we've taught our people to use the UNIX system," said AT&T spokesman Stephen Payne, noting that the company's planned public tutorials will add "a new group of end-users, programmers, and applications developers" to its list of students.

The tutorial program has 20 sessions and will include a series of computer laboratory courses that provide hands-on experience with UNIX hardware and software. The scheduled topics include: introductions to the UNIX system, shell programming, C language and system tools; sessions on **awk, make, SCCS, terminfo,** and **Lex** and **yacc;** as well as special topics such as text processing, data communications, database management, device drivers, internals, and UNIX applications for managers.

STRATUS USF TURNS HEADS

A true highlight of UNIX Systems Expo/Spring was the first public appearance of Stratus Computer's UNIX Systems Facility (USF) for its Continuous Processing family of fault tolerant computers. USF is integrated with the Stratus Virtual Operating System (VOS), which has been extended to support UNIX application programs. The VOS extensions are made at the kernel level to avoid degradation of performance between UNIX and native VOS operations. VOS facilities also available to USF users include demand-page virtual memory, indexed files, record locking, access-control lists, and transparent networking. Non-UNIX ap-

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

VERDIX ADA DEVELOPMENT SYSTEM

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

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

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

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

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

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

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

IT'S VALIDATED. USE PROVEN. AVAILABLE NOW!

VERDIX
Ada Development System

Verdix Corporation
Westgate Research Park
7655 Old Springhouse Road
McLean, VA 22102
(703) 448-1980

Circle No. 31 on Inquiry Card

plications can also run on a Stratus/USF system.

The USF shell is a ported copy of the UNIX standard shell with extensions providing access to the VOS commands. Applications can be ported from other UNIX-based systems "with minimal effort", according to Stratus. Software development can be done in COBOL, BASIC, PL/1, Fortran, Pascal, C, using **SCCS** and other UNIX tools. Communications options abound in USF, taking in Stratus SNA, 2780/3780, IBM HASP/RJE, 3270 terminal emulation, and X.25 and X.29 terminal support, as well as local area and wide area network schemes from Stratus.

VOS was developed by engineers who had worked on the Multics program sponsored by MIT, Bell Labs, and GE back in the 1960s. "The blending of VOS with USF permits users to avoid some of the frequently voiced drawbacks of UNIX . . . (namely, its) loose concept of record management (and its unsuitability) for real-time database updating," according to Stratus vice president of engineering Robert A. Freiburghouse. "The joining of VOS with USF gives users access to . . . a network-wide distributed file system."

/usr/group-IEEE EFFORT SEEKS COOPERATION

The first meeting of the recently-formed /usr/group Technical Committee will be held during this month's Usenix Conference in Portland, OR. The exact time and place has not yet been decided. Heinz Lycklama, the committee's chairman, can be contacted at 213/453-8649 for the latest details.

The Technical Committee actually represents a merger of the /usr/group Standards Committee and the IEEE P1003 Standards Committee. Its mission is to define an appropriate interface to the operating system for UNIX-based applications programs. Specifically, the "effort must focus on immediate technical details . . . to achieve a consensus towards a standard," according to Lycklama. "The Technical Committee will focus on those issues which would tend to stagnate a standards effort."

There are five general categories of concern: the networking interface, distributed file systems, graphics, bitmapped terminals, and internationalization.

Membership on the /usr/group Technical Committee is open to anyone with "a bonafide interest" in this effort, Lycklama said.

WE'RE SORRY, YOUR CALL CANNOT BE COMPLETED AS DIALED

In a Monthly Report item in April, we printed an access number for browsing through a listing of software items in the AT&T UNIX Toolchest. Unfortunately, we got the area code wrong, inadvertently directing some of our readers to a shop in Brooklyn rather than an electronic software store in New Jersey, much to the bemusement of the New York shopkeepers, who didn't know what to make of all those modem tones. To access the AT&T UNIX Toolchest, dial this number (honest): 201/522-6900.

Roger Strukhoff is the Associate Editor of UNIX REVIEW. ■

The First Name In Integrated Office Automation Software

XED®

- Mail System
- Telephone Directory
- Menu Processor
- Word Processor
- Forms Manager
- Spreadsheet

Certified and Deliverable Since 1981

XED® INTEGRATED OFFICE SOFTWARE

Box 3938 • Chatsworth, CA 91313 U.S.A. • (818) 884-2000
FAX (818) 884-3870 • TWX 910-494-1716 • Intl. Telex 292 662 XED UR

Circle No. 5 on Inquiry Card

THE HUMAN FACTOR

From now on, consider it . . .

by Richard Morin

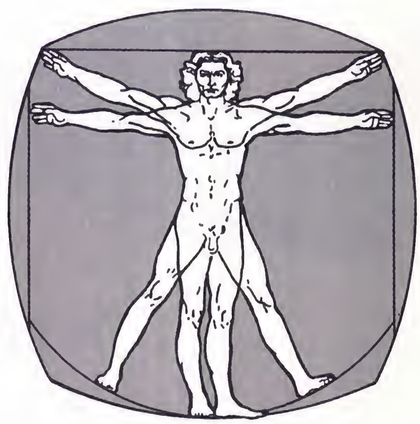
It is difficult, and even dangerous, for commercial institutions to relinquish control of standards. Minor changes in standards can have enormous financial effects. It is far safer, in the short run, for companies simply to set their own internal standards and live by them.

Proprietary standards allow companies to define rules under which they will operate. The companies retain control of the rules, change them as needed, and remain safe from outside interference.

The rules can also be used to isolate customers from the temptations of other vendors. When an installation has a major investment in a proprietary product, it can be prohibitively expensive to abandon it. This allows the vendor to demand a premium price.

The cozy world of proprietary standards has its own problems, however. When a vendor locks in its own customers, it also necessarily locks out other *potential* customers. Dominant vendors may not care about this, but smaller vendors are keenly aware of it.

It is also expensive to maintain proprietary standards. Supporting a proprietary operating system, for instance, often entails providing lots of features that users have come to expect. Not every vendor can afford to do this.



Customers are also becoming wary of proprietary standards. A major conversion effort can be a chastening experience. It is also annoying, at least, to pay exorbitant amounts of money for proprietary products. Finally, customers often have needs that cannot be met by a single vendor, and proprietary systems work against multivendor solutions.

We are thus beginning to see a real movement toward cooperative industry standards. The UNIX industry is a major part of this movement, but it is not alone. Let's look at a sampling of the standards now emerging in the computer industry.

FIRST, UNIX

UNIX itself is a standard that defines a file system, a set of system calls, a user interface, and so forth. The standard is reasonably consistent across the industry. Porting of UNIX applications and personnel is already easy, and is getting easier. Some parts of UNIX are rapidly becoming more uniform. In time, a very good standard should exist.

AT&T's marketing arm would have us believe that System V is already the UNIX standard. In a sense, it is right. System V is gaining broad acceptance from the UNIX vendor community. Even the 4.xBSD vendors are being forced to support a degree of System V compatibility.

Behind the scenes, however, System V is still very much in flux. AT&T is adding assorted features in an effort to appease Berkeley UNIX aficionados. Other features are being imported from Dennis Ritchie and friends over at Bell Labs.

Nor is AT&T ignoring the rest of the industry. Indeed, the company is cooperating with /usr/group, IEEE, and others in defining system calls. In short, AT&T's System N (pick a number) has a good chance of being a well defined, well accepted industry standard.

It is gratifying, if somewhat surprising, to see this kind of effort from AT&T. The company has a substantial reputation for its Not Invented Here syndrome. While it has not yet entirely overcome the affliction, it is trying.

UNIX™ and UNIX Training

From the world's acknowledged UNIX experts:

UNITY under VAX/VMS

- UNIX environment for DEC VAX
- Correct UNIX functionality
- No VMS modifications
- Easy installation

HCR Human Computing Resources Corporation

From the leading video-based UNIX training company:

UNIX Fundamentals for Programmers

- Self-paced video instruction
- 15 module course
- Complete text/workbook
- Hands-on exercises

COMPUTER TECHNOLOGY GROUP

TWO FOR THE PRICE OF ONE!

Call HCR immediately—offer expires July 31, 1985.

Human Computing Resources Corporation

HCR

10 St. Mary St., Toronto, Ontario, Canada M4Y 1P9 Tel. (416) 922-1937 Telex 06-218072

UNITY is a trademark of Human Computing Resources Corp. VAX and VMS are trademarks of Digital Equipment Corporation. UNIX is a trademark of Bell Laboratories

Circle No. 32 on Inquiry Card

NEXT, NETWORKS

The computer industry has a number of networking standards of course, covering hardware configurations, protocols, data formats, and so forth. Unfortunately, it is still problematic to get computer systems from different vendors to communicate over a network.

The UNIX community has accepted the combination of Ethernet and TCP/IP as a *de facto* industry standard for local area networks. UNIX vendors race to see who can come up first on the trade show Ethernets. AT&T hasn't shown up on a trade show net yet, but it's working on it.

A range of network activities is supported by Berkeley's TCP/IP software. Users can copy files, log in remotely, and build applications that pipe data across the network. Other features, such as distributed file systems and remote procedure calls, can be layered on top of TCP/IP.

The TCP/IP-Ethernet combination is not expected to dominate the UNIX industry for long.

We thus have a working standard with at least reasonable characteristics. Networks can be built quickly and easily out of standard components. One might well assume that the industry would breathe a collective sigh of relief and leave things alone.

Surprisingly, this isn't happening. The TCP/IP-Ethernet combination is not expected to dominate the UNIX industry for long. Instead, the ISO's OSI model is almost guaranteed to be the next protocol standard—it already is an ANSI standard.

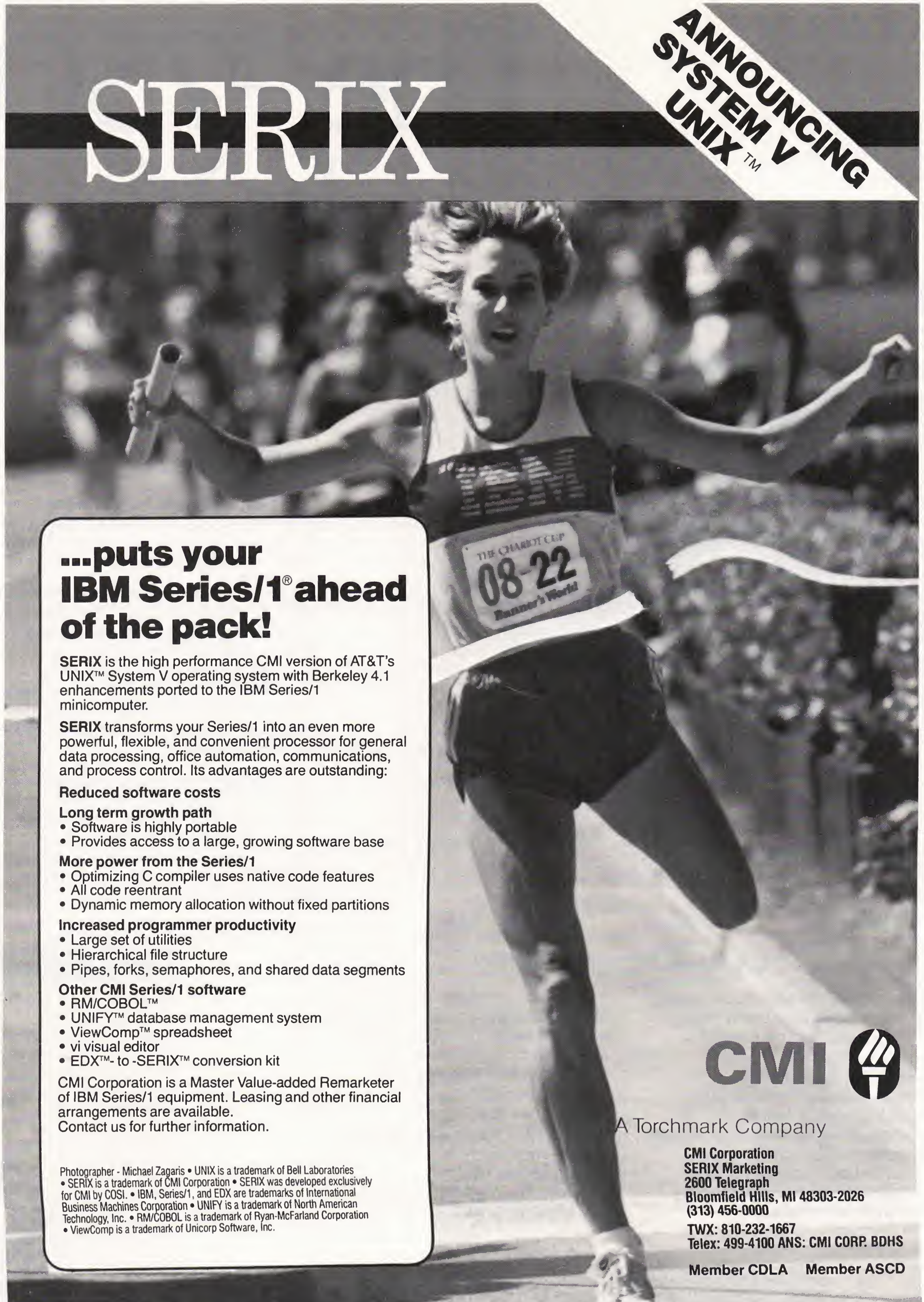
There are technical, political, and economic reasons for this. An informal comparison of TCP/IP and OSI may be instructive.

TCP/IP is not a new standard, and it is showing its age. Although it provides for most current needs, it has flaws that keep it from being a good longterm solution. OSI is a well-developed standard, it was recently designed, and it has room for growth and change. This will allow efficient implementations both now and in the future.

DARPA developed TCP/IP for its own needs, not for those of the diverse international computer industry. But OSI is an international standard, developed in accordance with the needs and desires of a variety of vendors, users, governments, and academic institutions.

These same participants now have a vested interest in seeing that OSI is widely supported and used. The vendors like it, the academicians like it, and some very influential customers (General Motors and Boeing, for example) are demanding it. With that kind of clout behind it, even Big Blue will have to support it.

Surprisingly, this leaves the UNIX community in pretty good shape. Gateways can be built between OSI and existing networks,



Circle No. 3 on Inquiry Card

The $750
In Your

AT&T 3B2™

One select investment can put you on top of UNIX™ applications development. Today. No more waiting for the future.

That one investment is UX-Basic™. It increases your productivity and gets your programming done on time, and on budget.

One complete, easy to learn, professional programming language will run all your applications without change, on micros to mainframes.

UX-Basic gives you all the power of UNIX to develop your business and scientific applications. You get productivity tools to save you time and money including a full featured line editor, automatic syntax checking, on-line help, meaningful error messages, and real-time debugging, for fast program development and modification.

UX Software, Inc. has prepared for the IBM PC/AT™ and AT&T 3B2™ and AT&T UNIX PC™, a limited 1,000 copy edition of the UX-Basic Development System, exclusively for software developers and UNIX users, and code named the 9000 series.

UX-Basic Features Profile

- automatic file and record locking
- abbreviated command input
- automatic backup copy
- BCD arithmetic
- C-ISAM™ bundled in
- common and local variables
- date and time functions
- device and terminal independent
- European format standards
- formatted business printing
- long variable and label names
- matrix operations
- modular structured code
- multi-line functions and subroutines
- multi-line IF/THEN statements
- multi-tasking and multi-user
- OASIS™ Basic compatibility
- recursive routines
- re-entrant code
- sequential, direct and indexed sequential files
- shell access
- software protection options
- sixteen dimension arrays

IBM PC/AT™

UNIX is a Trademark of AT&T Bell Laboratories
UX-Basic is a Trademark of UX Software, Inc.
C-ISAM is a Trademark of Relational Database Systems, Inc.
OASIS is a Trademark of Phase One Systems, Inc.
3B2 and UNIX PC are Trademarks of AT&T Information Systems
IBM PC/AT is a Trademark of International Business Machines Corporation

Investment Future

The full UX-Basic Development System includes:

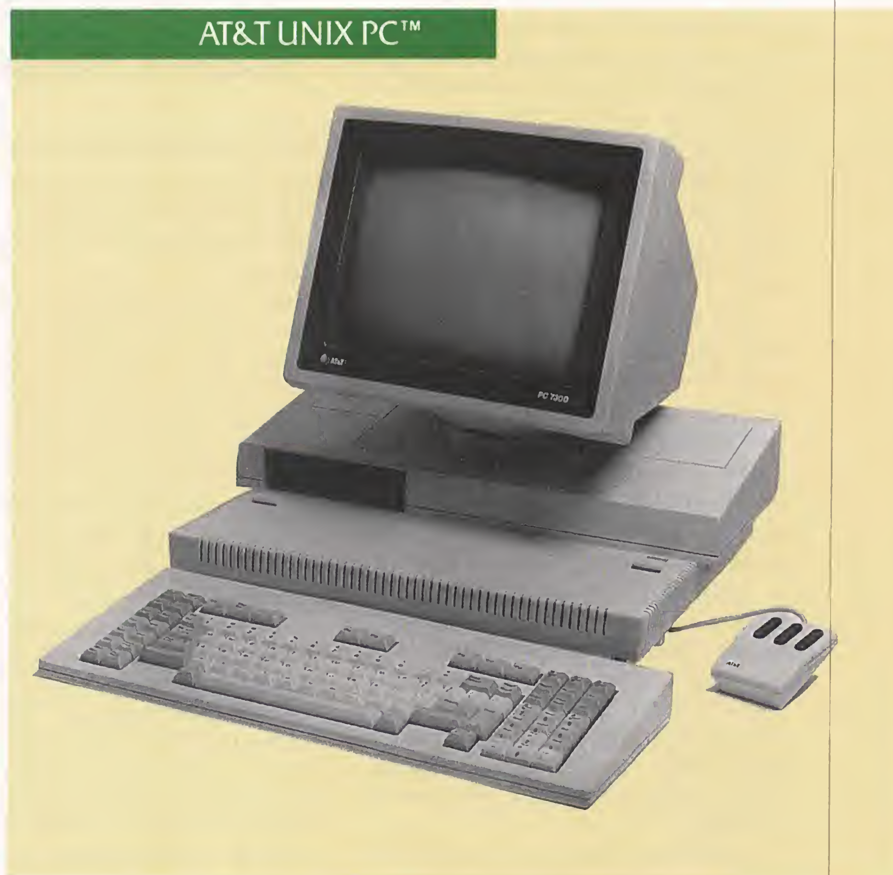
- UX-Basic: interpreter for creating source code.
- UX-Compile: the pseudo-code compiler.
- UX-Runtime: for executing compiled code.
- UX-Utilities: demonstration and maintenance programs.
- UX-Basic User Manual.

In addition to this, the 9000 series UX-Basic Development System edition provides extended service and support.

You will receive:

- A demonstration version of UX-Basic, including sample business programs, that you may duplicate and distribute.
- Direct product support from UX Software Inc., for 180 days.
- Free product updates for one full year.
- A subscription to /usr/uxsoft/notes, the insider's technical notebook for UX-Basic programming.

Order UX-Basic today. There is no more waiting for the future.



☑ I want to invest in my future.

Please rush me a 9000 series UX-Basic Development System, complete with sample business programs.

NAME ____________________

COMPANY ____________________

ADDRESS ____________________

____________________ TELEPHONE ____________________

☐ AT&T 3B2 ☐ AT&T UNIX PC ☐ IBM PC/AT

PAYMENT ☐ AMEX ☐ MC ☐ VISA ☐ Check

To arrange for immediate delivery, call now (416) 964-6909.

UX Software, Inc.
10 St. Mary Street, Toronto, Canada M4Y 1P9
Telex: 065-24099 UXSOFT TOR
(416) 964-6909

Circle No. 71 on Inquiry Card

and OSI support can be added to UNIX implementations. Finally, the existing level of networking expertise will not be lost, and should help in the transition.

Incidentally, Ethernet will not be the only winning hardware standard. It is not well suited to real-time applications because there is no guarantee that a given message will arrive in a finite amount of time. Token passing methods, as promulgated by IBM and others, may well take over there.

It is also not a cheap system, for wiring or electronics. An Ethernet system can cost several hundred dollars per node. Twisted-pair (telephone wire) systems, good to about a megabit per second, are much cheaper. The hardware used on the new AT&T 7300 is a good example.

Finally, Ethernet is not a high bandwidth net by today's standards. Installations needing higher speed communications will move to fiber optic, broadband, or other technologies. Still, Ethernet does work, and it fills a current need, so expect to see it around for a while.

If OSI succeeds in its design effort, however, this won't be a problem. Such hardware variations will be invisible to the software, and will simply function as compatible parts of a single system.

FINALLY, GRAPHICS

UNIX doesn't have much in the way of graphics standards. The **graph** and **plot** utilities aren't very impressive as graphics tools. They aren't even very good as filters, since they communicate in binary. System V adds some graphics support, but not much.

Many vendors of UNIX graphics workstations have proprietary graphics software. Others have decided to support the ACM Siggraph Core standard, to one degree or another. Fortunately, the international computer community is once again coming to the rescue.

> **UNIX doesn't have much in the way of graphics standards.**

On the graphics standards panel at the Dallas Uniforum show, the consensus ran overwhelmingly towards GKS. The panelists acknowledged that it was deficient in 3D support, but insisted that the problem was being addressed. Core was shunned because of its lack of raster primitives, as well as its imprecise definition.

As a recently developed international standard, GKS has many of the benefits ascribed above to OSI. It supports current hardware far better than does Core. It was developed by, and has the support of, a wide range of organizations. Expect to see it on almost all graphics workstations in the near future.

There is another graphics standard on the way—the PostScript language, developed by Adobe Systems (yet another PARC spinoff). PostScript is being marketed as an efficient and resolution-independent way of sending print images to raster printing devices. It is all of that, to be sure, but it is also far more.

PostScript describes a print image by specifying the algorithm needed to reproduce it. A processor embedded in the printer normally executes the algorithm, producing a bit image to be copied onto paper. A very small description can thus produce a highly complex image, reducing data transmission bandwidth.

A variety of software, workstation, laser printer, and typesetter vendors are supporting the language, with more on the way. The current list includes Apple, Mergenthaler Linotype, QMS, and Sun. With the low cost of microprocessors, PostScript could conceivably be installed on anything from a 150-dot-per-inch dot matrix printer on up. In any case, dumb devices can still rely on the main processor to generate bit images, albeit at some cost in data bandwidth.

PostScript is a complete programming language, augmented with a large number of resolution-independent graphics operators. It is stack-oriented, like Forth, and its syntax is rather similar.

A number of object-oriented and other features serve, however, to make the two languages rather different in practice. In addition, PostScript is not really intended to be used directly as a programming language. Instead, it provides an extraordinarily flexible and powerful intermediate language for describing graphic images.

PostScript descriptions, being plain ASCII text, can be generated, stored, retrieved, and edited easily and efficiently by graphics applications programs. It is this use, in the hands of imaginative software developers, that will show the true value of the language.

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. ■

PLANNED SYNERGY

We are absolutely dedicated to the synergy that results from our hardware running with our UNIX derived operating system. It makes our products both versatile and powerful.*

We are looking for UNIX professionals to become a part of our planned synergy. If you are a systems, technical support, or instruction professional, and take pride in meeting complex challenges, we invite you to join our TECHNICAL TEAM. We have opportunities in Dallas, Texas and in the San Francisco Peninsula area.*

For immediate consideration, please send your resume to: Professional Employment—UNIX, MOTOROLA Information Systems, P.O. Box 819091, Dallas, Texas 75381-9091 or call 214-888-2114.

Equal opportunity is our pledge and practice.

**UNIX is a trademark of Bell Laboratories.*

MOTOROLA

Four-Phase Systems

THE WIDE WORLD OF UNIX

A sampling of the system's varied applications

by Tom Athanasiou

The same characteristics that have made UNIX a *de facto* standard—its portability, flexibility, and ease of customization—have also made it an ideal host for a wide variety of applications. In many ways, UNIX has taken on a life of its own; as the longstanding software development environment of choice, it has catalyzed the production of an unusual variety of devices, networks, and configuration options. It has also inspired a number of applications that might best be described as being somewhat off the beaten path.

This is an account of some of those applications, none of which is quite in the mainstream of commercial or military life. Yet none has been chosen frivolously; some are scientific and show the virtues of easy customization and portability, while some depend simply on the ubiquity of UNIX. Taken together, they speak volumes for the simple but open-ended "toolkit" approach UNIX exemplifies.

UNIX AND NUCLEAR TREATY VERIFICATION

The use of computers for military purposes is familiar enough, but what is less known is that computers also have been used to help contain the nuclear arms spiral. Since nuclear treaties depend on the mutual ability of vying powers to verify compliance, the role that advanced electronics and computers play is critical.

In the US, compliance verification is the job of the Center for Seismic Studies, a direct contractor to DARPA (the Defense Advanced Research Projects Agency). When founded in May, 1982, the center was charged with the development of verification techniques for the proposed comprehensive test ban treaty between the US and USSR (in the event it was ever signed). According to Raleigh Romine, a self-described "systems hacker" at the center, the first step in the center's effort to modernize was to get off "a bunch of aging 360s and move over to a group of VAXen running UNIX.

"The idea of verification is that you want to be able to discriminate between an earthquake and a nuclear explosion, and our mission is to get finer discrimination," Romine explained. "Currently, the limit is 150 kilotons underground; if they go to the comprehensive test ban it's basically zip."

To accomplish its mission, the center has multiple real-time links to satellites throughout the world, satellites that relay signals from arrays of strategically located seismometers. "And we take tapes, UUCP links to Golden, CO, stuff on Telex, Telenet, whatever we can get," Romine said. The data aquisition is controlled by three DEC 11/44s, all running 2.9BSD.

An analysis of the data is performed by VAXen running "basically Berkeley 4.2". A variety of algorithms has been used to discriminate among various kinds of seismic waves. In fact, the development of better algorithms is one of the center's primary tasks. "The only thing that's really classified is just how fine we can discriminate," Romine said.

UNIX AT LUCASFILM

When Lucasfilm's Droidworks set out to design a computer-based film editing system, its objective was to produce a system that looked like traditional film editing equipment to the user. Film editing, in the words of one former Lucasfilm employee, "has its own gurus".

Editdroid is a videotape/video-disk film editing system that runs on 32-bit Sun boxes under 4.2BSD with a "slightly modified kernel". But users have no way of even knowing they're employing UNIX. The user interface is based on a Macintosh-like bitmap display (with icons, windows, and boxes—the whole business) as well as a trackball. Rob Lay, head of the Editdroid development team at Droidworks explained, "We've taken the film environment and modeled it in the bitmap display . . . The editors still have those little pieces of film, but now they're icons." Time is still marked in terms of "frames".

With the new film editing process, after the uncut films are written onto video disks, Editdroid is used to do an electronic


$^{2-}O_3POH_2C$
O
CH_2OH
H
H
HO
OH
HO
H
MAJLESSI

edit that yields a video tape and a cut list. After the videotape is approved, the original film comes out of the vault, and negative cutters use Editdroid's cut list, which holds to an industry standard cutters understand, to generate a production master.

Why did Lucasfilm go to all the trouble to produce an electronic editing system? "Because it's 10 to 15 times faster than traditional methods," Lay said. Why UNIX? "Because of the richness and efficiency of the programming environment," Lay responded. "We can do drivers for new devices very fast. We're able to turn around projects really quickly by using already existing packages." Why Berkeley UNIX? "Because three years ago when we started the project, it was the only reasonable system that had demand paging."

The first versions of Editdroid were released in January of this year. There are now three in commercial use in Los Angeles. One is being used to cut the television show "*Fame*".

UNIX AND HEART DISEASE

Few areas have been as heavily influenced by computers as medicine. While in many cases the benefits of this infiltration have been dubious, no one can question the significance of at least one new UNIX-based method for the cheap, early, and non-invasive detection of heart disease.

The still unnamed system, developed by the Berkeley team of cardiologist Aaron Dinaburg and statistician Barry Zuckerman, relies on the cheapness, power, and flexibility of UNIX. Built on a real-time kernel developed by ADAX (a small Berkeley R&D company) and run on a Plexus P-35, it works by first collecting five different kinds of information in real-time and then subjecting the information to statistical analysis. The data collection is entirely non-invasive: EKG and respiration pickups, heart sound recorders and pressure sensors are all attached to the patient's skin. Meanwhile, the ADAX real-time kernel reads data from the sensors into RAM.

The analysis is too complicated for real-time, so once 20 heartbeats worth of information is collected, the ADAX kernel yields to the Plexus machine's standard System V kernel. At this point, the machine is ready to perform what Zuckerman referred to as "a fairly hairy" analysis. "We collect 36 individual measurements per beat, and then, using standard methods, pick those providing the best separation between sick people and well people and reduce them down to eight," he explained. "Using these eight, we classify each heartbeat as either sick or well or in between. With one minute of sampling we get about 20 beats, and are down to about one percent chance of error in the diagnosis.

"What is really interesting about this is that we can identify developing heart dysfunction before surgery becomes the only possible remedy. Lots of people have hypertension and not all of them get sick, but hypertension is known to be an extraordinary risk factor in the development of heart disease. With this method, we can identify people that show subtle evidence of developing heart problems—while it's still early enough for them to reform their lifestyles."

> The use of computers for military purposes is familiar enough, but what is less known is that computers also have been used to help contain the nuclear arms spiral.

One further note for UNIX fans: before UNIX, the data was collected on a PDP/11 and shipped to a multi-megabuck IBM mainframe, where it was computed. With the advent of UNIX supermicros, this sort of computation-intensive diagnosis has gone from being possible to actually being practical.

UNIX AND DRUG DESIGN

Many observers of new technologies believe that, ultimately, the biological revolution will surpass the computer revolution in significance. But, as the discipline of drug design intensifies, new biological technologies become dependent on computers.

Tom Ferrin is the facilities manager of the Computer Graphics Lab in the Department of Pharmaceutical Chemistry at the University of California in San Francisco. He's been using UNIX-based computer graphics systems for molecular modeling since 1976, but the recent emergence of new, extremely powerful graphics terminals has made powerful, new methods possible for the first time.

Ferrin's lab uses two of these new devices, one an Evans/Sutherland machine and the other a Silicon Graphics Iris. Both have very powerful hardware graphics devices, capable of interactively rotating, translating, scaling, and otherwise altering complex color graphic images. These devices are controlled by a version of 4.2BSD containing an altered scheduler capable of simple real-time operations. UNIX also mediates the human interface, keeping the terminals from overwhelming the user with information.

Graphics are extremely important in drug design, as they are in many scientific applications, because of their ability to present massive amounts of information in a manner easily comprehended by mere mortals. It is estimated that fully 60 percent of our brains are related, somehow, to vision. Needless to say, drugs, and the world of proteins within which they operate, are complex enough that anything that promises help in understanding them is very welcome indeed.

Because UNIX can be customized so easily, and because it has spawned the creation of such a large family of compatible devices, UCSF was able to put together a system—with a minimum of special programming—that's capable of successfully simulating real experiments and predicting their outcome. In this way, a graduate student at UCSF has been able to develop analogs of Thoroxin (Human Growth Hormone) far more specific in their efforts than the natural hormone itself. As a result, it may soon be possible to stimulate the lung growth of premature infants, thus speeding the development of normal breathing.

UNIX AT RAJNEESHPURAM

Rajneeshpuram, in central Oregon, is no marginal, sixties-style hippie commune. Now a fair-sized town, it has two hotels (one cheap and one expensive), a bookstore, a boutique, an electronics shop, a beauty salon, a school, multiple delicatessens and restaurants, a clinic, a disco, a newspaper, a Chamber of Commerce, public utilities, an extensive public transportation system, a local peace force, and a software shop—which, if nothing else, lends new meaning to the term "UNIX guru".

"Sarito" is a longtime UNIX consultant whose tutelage has made it possible for Rajneesh Computing to build a UNIX-based "currency card" system that serves the nearly 4000 local residents and 15,000 far-flung pilgrims who wish to substitute plastic for "dirty money". Customers who deposit their money at the Rajneesh Financial Currency Card Office (no interest paid: local deposit of inter-bank wires accepted) can simply present their cards for signature validation when making purchases in Rajneeshpuram and go happily on their way, presuming they have sufficient funds, of course. Participating merchants can produce invoices by way of a local terminal and printer.

> **Many observers of new technologies believe that, ultimately, the biological revolution will surpass the computer revolution in significance.**

"This is a reliable, multivendor transaction-processing system, made with off-the-shelf equipment," claimed Sarito. Since the Rajneeshpuram population varies so widely, the system has been designed for easy reconfiguration so as to use fewer terminals and fewer computers during non-peak times. It currently runs on six 68000-based machines, four Altos computers and two Plexus boxes. The Altos machines are connected by way of an 800 kilobit AltosNet; the Plexus machines talk by way of Ethernet boards. Everything has been kept simple, with no mag-strip readers in place as yet. In keeping with this, the Rajneeshpuram system uses only simple terminals that pass escape sequences to drive their associated printers.

Rajneesh Computing, which contracted with Rajneesh Financial Services Trust to build the system, is a typical diversified UNIX shop. It uses UNIX because of the system's rich programming environment, and because there are so many small, well-built UNIX machines available. The shop sells commercial programming to other businesses at Rajneeshpuram. Sarito's kernel expertise, meanwhile, is sold to a large company that wishes to remain unnamed; and a variety of custom applications—such as the currency card system—are available for purchase. As an aside, it may be interesting to note that the currency card system may someday be sold to other organizations with "a fairly strictly delimited environment."

One possibility is Club Med.

UNIX AND SOLAR PHYSICS

The solar physics community is a perfect microcosm of the geographically rambling community of computer users. Solar physics involves a lot of number crunching, and the numbers to be crunched originate in telescopes and other devices that must be as far as possible from city lights.

Phil Scherrer, a solar physicist at the Wilcox Observatory at the Stanford Center for Space Sci-

Continued to Page 92

HOMEBREWED

Illustration by Chris Peterson

UNIX APPLICATIONS

How to program without really trying

by Marc J. Rochkind

There are four ways to get an application for your UNIX system. One, you can buy an off-the-shelf package. Two, you can hire an outside programming shop to develop a custom application just for you. Three, you can program the application yourself, using a language such as C or BASIC. Or, four, you can develop the application yourself without programming.

This last alternative is seductive. It seems to offer the advantages of a customized application without the expense, development time, and longterm maintenance burden of programming. One increasingly popular approach to performing this miracle is to use an application generator, or fourth-generation language, that allows you to specify what your application is supposed to do, without saying how. There are several such products available for UNIX, and, from what I have heard, they seem to work well for the class of problems for which they were designed.

The subject of this article, however, isn't the use of an application generator product, but rather the use of the UNIX shell and text processing commands as an application generator. Can you develop your own applications—without programming—using just the UNIX commands that come with nearly every UNIX system? And if so, will the result be acceptably efficient, robust, and maintainable?

To shed light on this question, I'll present a simple office accounting application implemented entirely with the UNIX shell and a few common commands. This example will serve to expose many of the key advantages and disadvantages of this approach to application development. Next, I'll compare the non-programming approach to a more conventional approach using a programming language. I'll conclude with some observations about whether application development without programming really works, and make some recommendations.

A CHECKBOOK/EXPENSE ACCOUNTING SYSTEM

Let's say we want to develop a simple accounting system that can calculate our checkbook balance and total expenses for each of a number of expense accounts. There are actually two checkbook balances that we want: the current balance, reflecting all the checks that have been written; and the bank balance, reflecting only those checks that have cleared the bank. The bank balance is used to compare our accounts with those the bank maintains on a monthly statement.

To track expenses, each account has a three-digit number. For illustrative purposes, these six accounts will do:

```
110    income
310    rent
315    utilities
320    advertising
325    office supplies
330    travel
```

A typical small business, of course, would have several dozen such accounts.

Our check register can be kept on a UNIX system by simply entering a line of text into a file for each transaction (starting balance, check, or deposit). Any text editor can be used. The first part of the register for June, kept in a file named *acct*, is shown in Figure 1.

The first column has a "c" if the transaction cleared—that is, if it is present on the latest bank statement. The second column contains the date (not actually used in our examples). The third column reflects account numbers. In the fourth column, we've listed the amount, positive for checks and negative for deposits. The fifth column contains comments. The first line of the example doesn't represent an actual transaction, but simply provides the starting balance.

A few minutes with a calculator will show that the current balance is $2359.03 and the current bank balance is $4386.05 (because the last statement didn't include three checks).

To calculate the current balance with UNIX commands, we just need to find a way to add the fourth column of numbers. Calculating the bank balance is only slightly harder—we have to select those text lines with a "c" in column one, and add the fourth column of those entries. To get the total expenses for a specific account (310, say), we have to select the entries with that number in column three, and then total the fourth column. The basic operations we'll need then are *selection*, based on the data in a particular column, and *addition* of a column.

To start with, let's calculate the current balance. I was planning to use the **cut** command to select particular columns to process, but I discovered that although this command is present in UNIX System III and System V, it's missing from 4.2BSD. So we've learned our first lesson right away: you can't count on all UNIX commands being present on all systems.

Fortunately, I was able to use the **awk** command

c	-	-	-1456.78	
-	6-1	310	525	office rent
c	6-1	325	13.26	file folders
c	6-2	315	54.90	telephone
c	6-2	315	15.67	electric
-	6-2	325	3.79	paper clips
c	6-3	110	-3200	
-	6-4	330	1498.23	Chicago trip
c	6-4	310	124.50	furniture rent
c	6-5	325	12.40	pens & scratch pads
c	6-2	315	50	answering service

Figure 1 — *An example of input for a check register program.*

instead of **cut**. This command line isolates column four from the file *acct* and writes it to the file *col4*:

```
$ awk '{print $4}' <acct >col4
```

Here's what's in *col4*:

```
-1456.78
525
13.26
54.90
15.67
3.79
-3200
1498.23
124.50
12.40
50
```

Next, we have to add these numbers. The **dc** (*d*esk *c*alculator) command is ideal. The only problem is that we first need to place addition signs after each number (except for the first) and a **p** (*p*rint) command at the end. Then we can feed the result into **dc**, which is a reverse-Polish calculator. We'll use the **sed** command to edit in the addition signs as well as the **p**:

```
$ sed -e '2,$s/$/+/' -e '$s/$/p/' <col4
-1456.78
525+
13.26+
54.90+
15.67+
3.79+
-3200+
1498.23+
124.50+
12.40+
50+p
$
```

Unfortunately, we don't have enough space here to explain the **sed** command line in detail. However, if you don't understand it, you may already be asking yourself how you are supposed to develop applications without programming when you have to deal with commands like **sed**, which seem to be even more complicated than programming itself! It's a very good question, and we'll come back to it later.

Anyway, now that we have the input to **dc**, we can just pipe it in. But wait! Some of the numbers have a minus sign, whereas **dc** requires an underscore. So we use the **tr** command to translate minus signs to underscores:

```
$ sed -e '2,$s/$/+/' -e '$s/$/p/' <col4 | tr '-' '_'
_1456.78
525+
13.26+
54.90+
15.67+
3.79+
_3200+
1498.23+
124.50+
12.40+
50+p
$
```

We can combine **sed**, **tr**, and **dc** into a single pipeline that adds a column of numbers. We'll put this into a file named **add** and mark it executable (with **chmod**) so we can use it like a command:

```
sed -e '2,$s/$/+/' -e '$s/$/p/' | tr '-' '_' | dc
```

Now we can use **awk** to cut out column four and **add** to add it up. The command file *addcol4* will be useful:

```
awk '{print $4}' | add
```

And, finally, a command file named *currbal* that calculates the current checkbook balance should be added:

```
addcol4 <acct
```

Here's *currbal* in action:

```
$ currbal
-2359.03
$
```

To calculate the bank balance, we just have to select the entries with a "c" in column one and pipe the output into *addcol4*. We'll use the **grep** command to do the selection, as shown by the example in Figure 2. Here's the command file *bankbal* (compare it to *currbal* above):

```
grep '^c' <acct | addcol4
```

And here's the calculation of the bank balance:

```
$ grep '^c' <acct
c       -      -      -1456.78
c       6-1    325    13.26       file folders
c       6-2    315    54.90       telephone
c       6-2    315    15.67       electric
c       6-3    110    -3200
c       6-4    310    124.50      furniture rent
c       6-5    325    12.40       pens & scratch pads
c       6-2    315    50          answering service
$
```

Figure 2 — *The sort of output one might obtain with* ***grep***.

```
$ bankbal
-4386.05
$
```

There are several ways to total the expenses for a single account. We'll do it this way: first, we'll use **awk** to cut out columns three and four (the account number and amount). Second, we'll use **grep** to select the lines with the specified account number. Third, we'll use **awk** to cut out just the second column (the amount, which was the fourth column originally). Fourth, we'll use **add** to total the list of numbers.

Let's look at this in stages. Here's the first execution of **awk**:

```
$ awk '{print $3, $4}' <acct
- -1456.78
310 525
325 13.26
315 54.90
315 15.67
325 3.79
110 -3200
330 1498.23
310 124.50
325 12.40
315 50
$
```

Assuming we want account 315, we next use **grep** to select the corresponding lines:

```
$ awk '{print $3, $4}' <acct | grep '^315'
315 54.90
315 15.67
315 50
$
```

Then we use **awk** again to get just the amount column:

```
$ awk '{print $3, $4}' <acct | grep '^315' \
   | awk '{print $2}'
54.90
15.67
50
$
```

And finally, we use **add** to get the total:

```
$ awk '{print $3, $4}' <acct | grep '^315' \
   | awk '{print $2}' | add
120.57
$
```

We'll put this command line into a file named *expense* and use the argument reference **$1** instead of the constant 315:

```
awk '{print $3, $4}' <acct | grep "^$1" \
   | awk '{print $2}' | add
```

When we execute the command file *expense*, the argument we supply will replace **$1** in the argument to **grep**. Here's *expense* in action:

```
$ expense 315
120.57
$
```

To get totals for all five expense accounts, we can use the **for** statement that is available with the Bourne shell (a similar statement is available with the Berkeley shell). The command file named *expenses* looks like this:

```
for acctnum in 310 315 320 325 330
do
        echo 'Total for account' $acctnum
        expense $acctnum
done
```

And finally, the total we get for each account comes out like so:

```
$ expenses
Total for account 310
649.50
Total for account 315
120.57
Total for account 320
Total for account 325
29.45
Total for account 330
1498.23
$
```

CRITIQUE OF THE CHECKBOOK/EXPENSE ACCOUNTING SYSTEM

Several questions are burning: is our implementation of high or low quality, or something else altogether? Was the process that we used programming? Was it harder or easier than conventional program-

```
c       -       -       -1456.78
6-1     310     525             office rent
c       6-1     325     13.26        file folders
c       6-2     315     54.90        telephone
c       6-2     315     15.67        electric
-       6-2     325     3.79         paper clips
c       6-3     110     -3200
-       6-4     330     1498.23      Chicago trip
c       6-4     310     124.50       furniture rent
c       6-5     325     12.40        pens & scratch pads
c       6-2     315     50           answering service
```

Figure 3 — *A commonplace error in input.*

ming? And, the leading question: can you reasonably expect to develop your own UNIX applications by simply stringing command lines together, as we just did? We'll try to answer each of these questions in turn.

First, what about the quality of our implementation? Speed isn't much of a problem. True, the overhead is enormous. It took eight processes just to calculate the bank balance, and we made a separate pass through the database to total each account, rather than gathering up all the totals in a single pass. But there is little data in a simple accounting system like this one, and the command files are probably run only a few times per month. So speed really doesn't matter.

A more serious issue is robustness in the face of error. For example, suppose we made a mistake entering the second line in the *acct* file like the one shown in Figure 3. Try to guess how the *currbal* command reacted to this error. Would you believe:

```
$ currbal
empty stack
empty stack
stack empty
145 is unimplemented
stack empty
stack empty
- 01427.-0365
$
```

(Note the two messages "empty stack" and "stack empty"—along with the mysterious last line.)

Since data entry errors are quite common, we must say that this response to a simple input error is unacceptable. We would have to add additional commands to each of the command files to check for various errors—not necessarily all possible errors, but certainly the most common ones.

Another aspect of quality is maintainability. How easy would it be to modify the command files as our needs change, or as bugs are found? Actually, it's easier to understand the command files than it is to write them in the first place, although if the command files were much larger, this might not be true. Certainly, adding more account numbers or more data columns could be easily handled. It seems fair to say that command files don't seem inherently more or less maintainable than conventional programs. The length of the command file or program and the skill with which it was written are probably more important than the actual language.

Now for our second question: was this programming? To answer this, let's list some key elements of the programming process:

1) Specify what the program is to do (which is to say: what outputs are to be produced?).
2) Design the file structures and data entry transactions.

```
awk '
{
        if ($1 == "c")
                bank += $4;
        curr += $4;
        expense[$3] += $4;
}
END {
        print "Bank balance = " bank
        print "Current balance = " curr
        for (acctnum in expense)
                print "Total for account " acctnum " = " expense[acctnum]
}' <acct
```

Figure 4 — *A potential application of* ***awk.***

3) Code the program.
4) Test the program.
5) Document the program.
6) Release the program and provide ongoing support.

In our command file implementation of *currbal*, *bankbal*, and *expenses*, we performed all six steps! Furthermore, only step three was different from the way it would have been had we programmed in C or BASIC. I would say that what we have done was indeed programming, although our language was somewhat unusual.

The third question we want to answer is: is command file programming harder or easier than conventional programming? I don't believe this question can be answered in a general way—it depends too much on the skills and tastes of the individual doing the work. For example, if you know BASIC and don't know **sed**, then you might very well think that command file programming is much harder than conventional programming. If you know most of the UNIX commands and also know C, you might use either technique, depending on the application. If you've never programmed and don't care to learn, but you know the UNIX shell and some commands, you might feel much more comfortable with command files than with programs.

But wouldn't a conventional program for the checkbook/expense application be much longer and more complex than the short command files we showed earlier? My guess is that a C program would be. However, C is not the only programming language available under UNIX. An **awk** program that calculates both the balances and the totals for each account is shown in Figure 4. It's only a dozen lines long! Here's the output from a sample run:

```
Bank balance = -4386.05
Current balance = -2359.03
Total for account 330 = 1498.23
Total for account 315 = 120.57
Total for account 325 = 29.45
Total for account - = -1456.78
Total for account 110 = -3200
Total for account 310 = 649.5
```

So command files aren't necessarily simpler or shorter than conventional programs.

Finally, can you reasonably expect to develop your own applications without programming, using command files? My answer is no. The programming process with command files is essentially the same as it is with conventional languages. For some tasks, command files are straightforward, but they can be just as complex as any program. UNIX commands aren't easier to learn than conventional languages. (They probably are easier to learn than C, though—but then C is one of the hardest languages to learn.)

CONCLUSIONS

I wouldn't be justified in drawing conclusions from just the few examples shown in this paper, so I won't. I'll draw them instead from my 13 years of experience in programming UNIX, both with command files and with C.

First of all, both conventional programming languages and command files have their place. Command files are best when most of the processing is done by the commands themselves, rather than by shell statements or filters like **sed**. For example, we used **dc** only to add a column of numbers. To do that, we had to use **sed** to insert the addition operators and the final **p**, and then we had to use **tr** to change minus signs to underscores. Our application would have been much more straightforward if there were a standard UNIX command to add up a column of numbers. As a general rule, when a command file has complicated arguments like "2,$s/$/+/", you know you've gone too far.

Secondly, I find that when I'm trying to develop a command file, I have to first ask myself "Can this be done?" It's like a puzzle. Then, more often than not, even after working the whole thing out, I find I was wrong about some command—or, as in the case of the **cut** command, I find that a command I wanted to use is missing. I don't have this problem when I program in a conventional programming language. I ask myself not "Can this be done?", but rather, "What is the best way to do this?" There are sound, teachable methodologies that can be used with conventional languages. With command files, everything seems like a trick.

Thirdly, conventional languages seem to be fairly well defined. There is usually a standard textbook, if not an official standard. But what is the definition of the command file language? There isn't any. Its statements and operators are determined by whatever happens to be on your particular system. You need to be thoroughly familiar with your UNIX manual to use command files effectively. Furthermore, your command files are unlikely to be portable to other versions of UNIX.

Finally, when you program with command files, you must state *how* the processing is to be done, not just *what* processing is to be done. Command files are not application generators or fourth-generation languages. You have to be a programmer to use command files effectively. The UNIX shell and the rich collection of UNIX commands make up a programming language that's available to you. Only you, though, can decide if it's the *right* programming language for you.

Marc J. Rochkind worked at Bell Laboratories for 12 years. He was part of the team that designed the Programmers' Workbench version of UNIX, and is best known as the creator of the Source Code Control System (SCCS). Mr. Rochkind has just completed work on a new book, "Advanced UNIX Programming," which will be published by Prentice-Hall this summer. His company, Rochkind Software Corp., Boulder, CO, recently completed work on RIDE, a new C-like programming language for business applications. ■

COSI

T A N G O™

Use Tango to:

- Connect IBM and compatible PC's running DOS to UNIX systems.
- Offload processing to PC's.
- Control data and applications on remote PC's.
- Distribute processing between UNIX and PC's.

Buy Tango for:

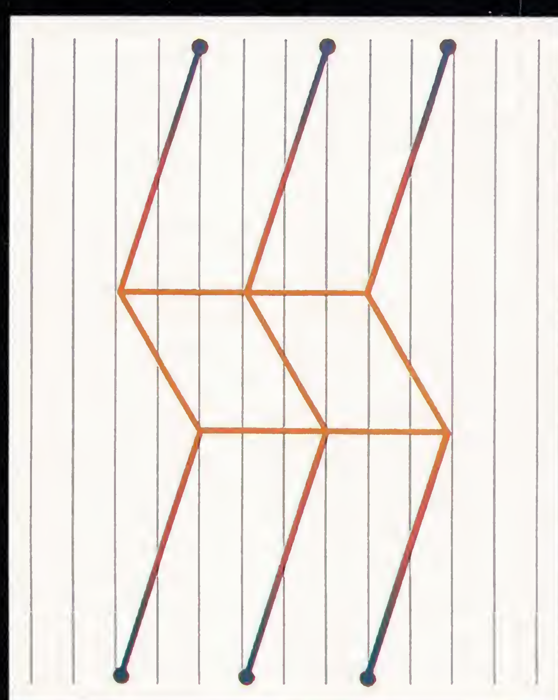
- Execution of DOS programs on the PC under UNIX control.
- Simple elegant file transfer under error correcting protocol.
- DEC, IBM, and Tektronix (graphics) terminal emulation.

Tango utilizes a standard RS-232 serial port on the PC and connects to the UNIX computer via a modem or direct connection.

COSI

313 N. First St.
Ann Arbor, Michigan
48103
(313) 665-8778
Telex: 466568

Tango is a trademark of COSI.
UNIX is a trademark of Bell Laboratories.



The PC-to-UNIX™ Connection

THE WAY IT SHOULD BE DONE

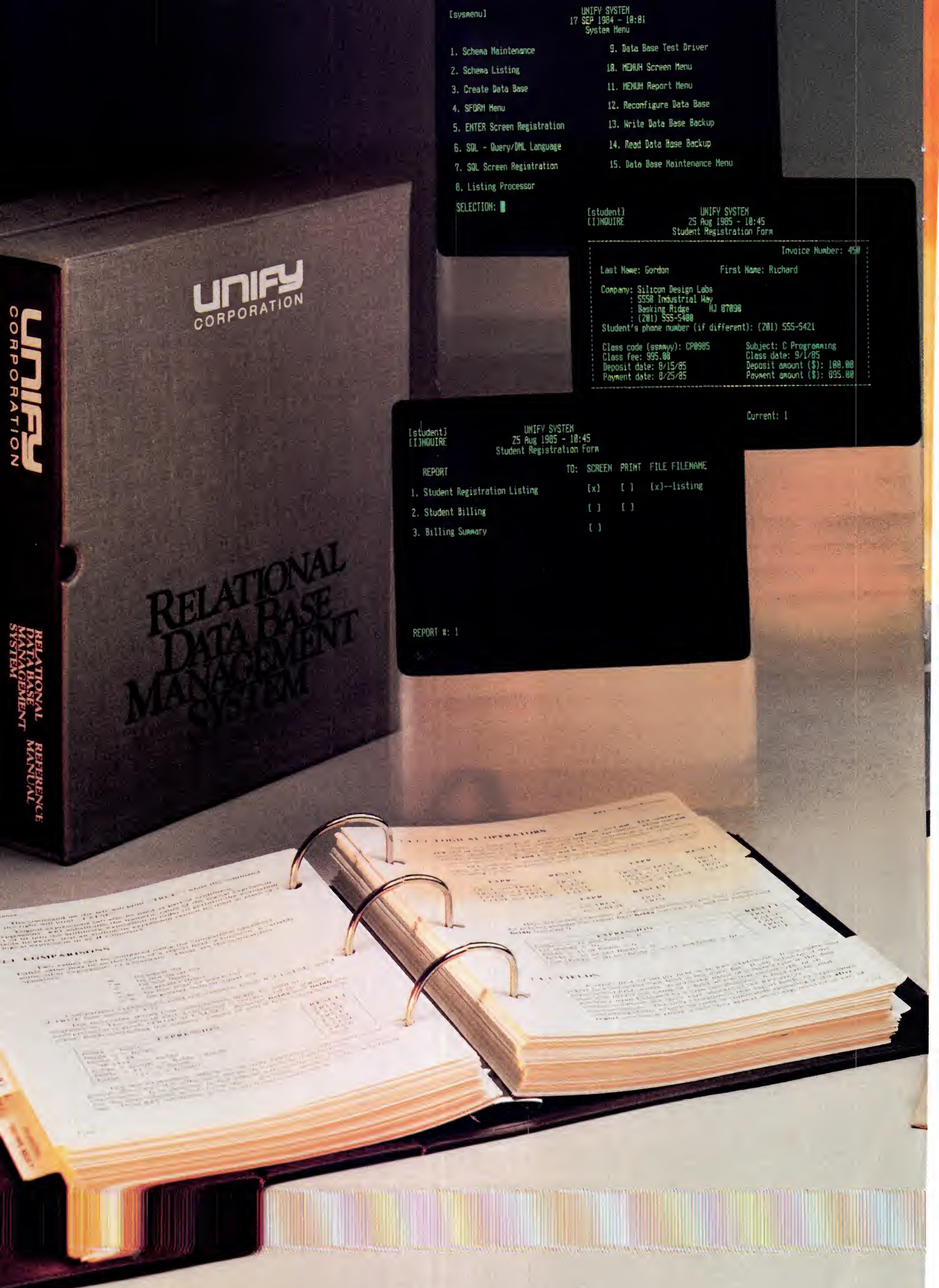
Serious UNIX influences on the development of games

by Peter S. Langston

From its inception, the challenge before the Lucasfilm Games Group has been to explore ways of applying the technological and methodological expertise developed at Lucasfilm (principally for film production) in a new entertainment medium—video games. A major tool on which that expertise has been based is the UNIX operating system. It is an interesting coincidence that this operating system's early development was heavily influenced by games as well as the interests of game designers.

In this article, I describe ways in which aspects of the UNIX system aided and influenced the development of two video games, "Ballblazer" and "Rescue on Fractalus!". Not only was the development of these games aided by the presence of UNIX software tools, but many useful ideas were suggested by examination of the tools' thoughtful design. The initial decision to use a UNIX system was based largely on my personal preference as leader (and, at the time, sole member) of the games group; as the project progressed, we discovered more and more reasons to believe it had been the right choice.

Illustration by Hyon Kim



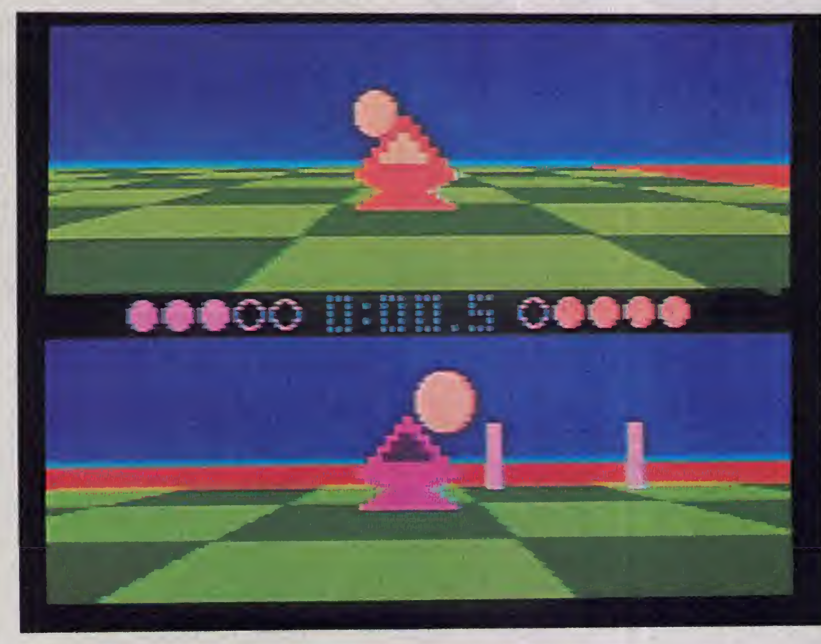
[sysmenu]
UNIFY SYSTEM
17 SEP 1984 - 10:01
System Menu
1. Schema Maintenance
2. Schema Listing
3. Create Data Base
4. SFORM Menu
5. ENTER Screen Registration
6. SQL - Query/DML Language
7. SQL Screen Registration
8. Listing Processor
9. Data Base Test Driver
10. MENUH Screen Menu
11. MENUH Report Menu
12. Reconfigure Data Base
13. Write Data Base Backup
14. Read Data Base Backup
15. Data Base Maintenance Menu
SELECTION:
[student]
[I]NQUIRE
UNIFY SYSTEM
25 Aug 1985 - 10:45
Student Registration Form
Invoice Number: 450
Last Name: Gordon
First Name: Richard
Company: Silicon Design Labs
: 5550 Industrial Way
: Basking Ridge NJ 07090
: (201) 555-5400
Student's phone number (if different): (201) 555-5421
Class code (ssmmyy): CP0985
Class fee: 995.00
Deposit date: 8/15/85
Payment date: 8/25/85
Subject: C Programming
Class date: 9/1/85
Deposit amount ($): 100.00
Payment amount ($): 895.00
Current: 1
[student]
[I]NQUIRE
UNIFY SYSTEM
25 Aug 1985 - 10:45
Student Registration Form
REPORT
TO: SCREEN PRINT FILE FILENAME
1. Student Registration Listing [x] [ ] [x]--listing
2. Student Billing [ ] [ ]
3. Billing Summary [ ]
REPORT #: 1
UNIFY CORPORATION
RELATIONAL DATA BASE MANAGEMENT SYSTEM
RELATIONAL DATA BASE MANAGEMENT SYSTEM REFERENCE MANUAL

HISTORY

"Months ago in a deserted warehouse in Marin County, George Lucas met with a lone programmer . . . " begins one of the many descriptions of the formation of the Lucasfilm Games Group. Needless to say, some journalistic license is evident in such accounts of the group's formation but when the purple prose is stripped away, what remains is the important part of the group's charter—to look at the booming industry of video games and see in what ways the "magic" of Lucasfilm could be brought to bear. That "magic" consists of two parts: an uncompromising attention to detail (leading to a sense of involvement and realism even in the most unrealistic of situations) and the use of high-tech tools to make possible the creation of formerly impossible sequences and images. One important tool supporting this endeavor was the UNIX operating system. When I was hired to form a games group at Lucasfilm, the benefits of using a UNIX system for large-scale graphics software development was well appreciated; what remained to be seen was whether those benefits would also apply to small-scale software development for "toy" machines.

The first project we undertook was a survey of the games industry to learn how games development was being carried out. We were amazed. Home video games were being produced in one of two ways. Often, a single programmer would spend a year or two of his spare time creating the game concept, graphics, sound, play mechanics, and assembler code (typically in a basement on a system with too little memory, too few floppy disks, no reasonable way to make a backup, and few if any debugging tools). Or a team consisting of a programmer and one or two helpers would be allotted two to three months to carry a game from concept to debugged implementation.

Programmers in either case often had no prior experience in computer programming and were plagued by flakey equipment and the lack of many aids that we considered basic: high-level languages, support tools, libraries of software, input/output redirection, hierarchical file systems, large storage devices, and an understanding of software design



A scene from "Ballblazer" (©1985 Lucasfilm Ltd.)

A scene from "Rescue on Fractulus!" (©1985 Lucasfilm Ltd.)

Photos courtesy of Lucasfilm Ltd.

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.

Circle No. 10 on Inquiry Card

methodologies.

When we looked at arcade ("coin-op") video games, the situation was different, but no better. The typical development cycle appeared to be:

1) come up with a game concept and get approval for it.
2) design the hardware.
3) build the hardware.
4) test the hardware.
5) write the software.
6) try the game out.
7) decide whether or not to continue.
8) debug and polish.

Although steps 3, 4, and 5 could be overlapped, it still took six to 18 months to get to step 6. At that point, it was too late to make major changes or to drop the project, which would entail a big loss. As a result, many uninteresting coin-op games were produced. We learned that the reason general purpose hardware was not used was that special purpose hardware is less expensive to manufacture than general purpose equipment—an important consideration when one wants to sell 10,000 to 100,000 of these games. Thus, a manufacturing savings of $1000 on each unit could mean a saving of between $10,000,000 and $100,000,000 overall. This combination of optimism and greed apparently blinded the industry to any thoughts of rational planning.

Media interest in Lucasfilm has always been high and so it was not long before we found ourselves being quoted in the press, pompously telling the games industry How It Should Be Done. In the midst of all this media attention, our group realized that since none of us had ever taken a video game through all the steps of production, we might in fact be setting ourselves up for some embarrassing surprises. "*Lucasfilm Computer Scientists Eat Words*" and "*Ivory Tower Talk is Cheap*" screamed the banner headlines of our nightmares.

To avoid such possibilities, we decided to design and implement one or two "throw-away" games as a combination rite-of-passage and reality check. These games would also serve as a way of identifying the areas that would profit most from the creation of some software tools. It's a credit both to the members of the Games Group and to the ideas we championed that the "throw-away" games have gotten such enthusiastic reviews, ranging from "George Lucas does it again!" to "setting a new standard in the industry". (OMNI magazine named one of the games in its "Top Ten Video Games of the Year", tied for sec-

Photo courtesy of Lucasfilm Ltd.

The boys after a day of zapping the Jaggis.

The Ones Who Showed The Way

The Place: Lucasfilm.

The Time: 1983 - 1984.

The Mission: To produce Games—not just any games, but Games that would show the games industry How It Should Be Done.

The Crew: Well, you can see for yourself. Pictured here are the guys fresh from a day's battle with the Jaggis ("Rescue on Fractalus!"). Included, from left to right, are Gary Winnick, Charlie Kellner, David Levine, Peter Langston (the guy holding the helmet), Noah Falstein, and Loren Carpenter. Among the others who made contributions were: Steve Arnold, Eric Benson, Steve Cantor, Ed Catmull, Terry Chostner, Mike Cross, Marty Cutler, Bob Doris, Gary Hare, Charlie Keagle, George Lucas, Lyle Mays, Pat Metheny, Lorne Peterson, Rob Poor, David Riordan, Richie Shulberg, and Steven Spielberg.

ond, even though the game had not yet been released.)

Of course, the games never did get "thrown-away" and in fact will appear in stores in the near future. As to whether they will show the world How It Should Be Done—we think so, but now you can decide for yourself . . .

THE HARDWARE

The Atari 800 computer contains a 6502 microprocessor chip, 32K to 64K of RAM, and several special purpose chips designed by Atari. The "ANTIC" chip handles DMA, some interrupts, vertical and horizontal scrolling, and the vertical line counter. The "GTIA" chip handles the graphics generation, graphics objects, collision detection, and the miscellaneous switches and triggers. The "POKEY" chip handles the keyboard, serial communications port, timers, hardware random number generator, potentiometers, and sound generators. The "PIA" chip handles the joystick controllers and interrupt lines.

An Atari 800 computer, complete with keyboard, power supply, and cabling that allows use of a conventional television set as the screen, costs about $100 in discount stores today. The 6502 microprocessor chip used in the Atari machine (also used in the Apple II and other home computers) costs about $1.50 (retail) in single unit quantities. All in all, the Atari 800 and the 5200 are *small* computers. Nevertheless, we found that writing good programs for a small computer often requires using tools that are very large. Such a discovery rules out using the Atari machines to run the tools. A much better arrangement is a hybrid that uses a large machine for editing and assembly, and a slave Atari machine for testing.

THE TOOLS

Many pieces of software were created by our group specifically for the game development projects. Typical of these were:

1) a cross assembler.
2) a library of macros for the cross assembler.
3) a pair of programs for downloading an Atari 800 home computer over a serial line from a DEC VAX 11/750.
4) a similar pair of programs for downloading an Atari 800 floppy disk over a serial line.
5) a simulation of flight dynamics and graphics on an Evans & Sutherland Picture System.
6) a program that turns an Atari 800 into a drum rhythm machine.
7) a music scoring system using **pic** and **troff**.
8) a storyboard editor that is written to run on a Sun Workstation.
9) a 6502 disassembler.

It was not surprising to us (although it was to many people in the industry) that the majority of our software ended up written in high-level languages. The cross assembler and its macro library were written in Lisp. The Atari end of the download programs and the drum machine program were written in our Lisp-like cross assembler. The rest of the tool software was written in C.

> Some aspects of the "state-of-the-art" were just too primitive for us to bear.

Lisp was chosen because it took only two weeks to implement a cross assembler that allowed both assembler macro definitions and arbitrary Lisp expressions to be included in the assembly task. It also meant we could extend or reconfigure the assembler as we discovered unforeseen needs and deficiencies.

The macro library had two important effects. First, it allowed us to write in a pseudo-structured assembler that included "*if-then-else*", "*for*", "*while*", and other familiar constructs. Second, it acted as a first step toward a subroutine library of commonly used routines. We found that many of the routines we included in the initial set never were used at all, while others that we added as we went along were used frequently.

The download programs made it easy to maintain the source on larger machines running UNIX. This allowed us to assemble the code and debug syntactic problems using many different tools before sending executables to the small Atari machines for debugging. Unfortunately, debugging on the Atari was not always convenient, so we designed a dual-ported memory card that would allow us to debug the running Atari program using software running on the larger machines. We discarded the possibility of simulating the Atari on the larger machines after discovering an incredible number of important undocumented eccentricities in the Atari hardware.

Simulations on the E & S Picture System allowed us to start fine-tuning flight dynamics and make some general decisions about ways to achieve the graphic results we wanted while the Atari graphics-rendering code was still being designed.

The drum machine program provided a workbench on which we could experiment with rhythmic patterns to be used in the games. We were able to include features that make commercial microprocessor-based rhythm units so popular and to add new features that suited our particular composition styles. Further, the program had the advantage of providing us with the exact sounds that we would be able to include in the games.

Although a music scoring system using **pic** is unavoidably unwieldy, it still allowed us to use familiar text editors, the Sun Workstation graphics screen, and the Imagen printer for manipulation and distribution of musical ideas and final compositions.

A computerized storyboard editor seemed like a particularly good idea because the pixel sizes and graphics modes of the Atari graphics system are quite eccentric. In fact, the Atari graphics are *so* eccentric that even the Sun Workstation's relatively high resolution screen was unable to capture all the "nuances". A further problem was the need to display the colors that appear on the Atari screen. Because of artifacting effects (the term "artifacting" refers to the interaction of colors in adjacent pixels on the screen), this would have been difficult to simulate even if we had had color graphics on the Sun Workstations. As a result, we found the existing, primitive graphics editors available for the Atari to be more useful than the more sophisticated but inappropriate editor that we had written for the Sun Workstation.

In our survey of game software, we found one or two programs that had unusually sophisticated graphics and arranged to speak with their authors. We were disappointed to discover that few game programmers are willing to discuss the technical details of their work or share insights they may have gained. (Not wishing to follow this clandestine approach, we went so far as to write an article called "Ten Tips from the Programming Pros", for the Spring 1984 issue of Atari Connection Magazine in which we described many of the techniques we used to write our games.) With a little work, we were able to write a disassembler that made feasible the difficult task of deciphering a 6502 executable.

Our intention had been to learn the steps necessary to produce a video game. As such, we did not want to try any radically new approaches; instead, we wanted to develop a list of ways to improve on the state-of-the-art. On the other hand, some aspects of the "state-of-the-art" were just too primitive for us to bear. As a result, we compromised by making only those improvements that were quick to implement, and required minimal long-range planning. The need for an improved macro assembler was obvious and it was relatively quick to implement, whereas the dual-ported memory cartridge turned out to be a larger project than we had expected, and its construction was therefore postponed.

UNIX AIDS

The ways in which the UNIX system aided the games design process can be grouped into two broad categories: software available on UNIX systems, and capabilities of the UNIX operating system itself. Available software included program languages, editors, communications programs, and other miscellaneous tools. The capabilities of the system included a hierarchical file system that made file sharing convenient, input/output redirection, and a general "permissiveness" that kept the system from getting in our way when we needed to do something strange.

Languages. Having languages like C, **awk**, the shell, and Portable Standard Lisp (PSL) readily available allowed us to choose a language to fit the requirements of any particular task. Some programs had to execute a specific, well-defined task as quickly as possible and were written in C (the download programs, for example). Other programs were more experimental and needed flexibility in the way they manipulated symbolic entities; these were written in Lisp. Still other programs had to be written quickly and would only be used once or twice; these were written as shell command files or as **awk** scripts. Had we been using typical games industry systems, we would have written all these programs in assembler (or perhaps Fortran if we were lucky), and we would still be debugging some of them today.

Editors. Although every class of Computer Science graduates seems to produce another editor-to-end-all-editors, none of these editors seems to have made it into common use in the video games industry. We, on the other hand, made much use of **emacs**, **vi**, and **ed**, enjoying the luxury of fitting the choice of editor to the requirements of the task at hand with few, if any, religious debates over the virtues of modelessness or the sins of meta-cokebottle-shift.

COMMUNICATIONS

Our group put in many 20-hour days, sometimes spending 16 hours at work and then going home to say "remember me?" to the family before logging in for another few hours of work. After a few months of this, it became hard to predict when any particular person would be in the office or even be awake. The UNIX mail system allowed us to keep in touch with each other and share ideas and status reports even when schedules didn't overlap. The ability to access files and programs through dial-in lines allowed occasional family visits and the speedy incorporation of middle-of-the-night inspirations.

Miscellaneous Tools. We used everything from **adb** to **yacc** at one time or another in this project. We got recalcitrant programs to work with **adb**, kept track of important deadlines using **calendar**, found the changes we made by falling asleep *on* our keyboards with **diff**, kept track of immense numbers of interdependent modules with **make**, printed cartridge dumps for the copyright office with **od**, scored music with **pic**, made global name changes with **sed**, prepared press releases using **spell**, **tbl**, and **troff**, borrowed tools from other sites with **uucp**, disassembled other designer's programs with **yacc**, and so forth. Because we had access to these tools, we were able to participate in parts of the production that normally are left either to specialists who have no other connection with the project or are left undone altogether. As a result, the games display an unusual coherence, with a common thread of design style surfacing throughout.

We also found that other parts of the Lucasfilm Computer Division had created tools useful to us. The Pixar (graphics processor) project had written a program to load PROMs that we found invaluable. The Digital Audio project had written a program (LUDS, pronounced "Lewds") to aid in circuit design and the creation of schematics, wire lists, and so forth for its design of the ASP digital sound processor. We used LUDS for our dual-ported memory design.

Finally, we used programs written for, or resulting from, other game projects. These ranged from programs that helped maintain large numbers of related source files to programs written to manipulate the masses of data produced by games like "Empire".

Although none of these programs was written with our specific needs in mind, each, like much of the software associated with UNIX systems, was written to make as few limiting assumptions as possible, and proved to be perfectly adequate for our needs.

File Sharing. Crucial to a team approach to software development is the ability to break programs up into separate modules and share access to them among the entire team. The UNIX operating system made this easy, although a little more concurrency control in the various editors would have avoided one or two minidisasters. Often, three people would be working on related parts of a game, passing files back and forth for advice or criticism while a fourth was compiling and testing the results of the others' changes.

I/O Redirection. Many of the tools that we used could not have existed on a system where the author of the software had to know what kind of device the data was coming from and to what kind of device it would go. Because we had to know less about the intended uses of a tool at the time we wrote it, we were often able to use an existing tool to solve a problem instead of having to create a new one. I/O redirection is one of the features of the UNIX operating system that often leads programmers to write more useful programs than they had intended.

UNIX INFLUENCES

The ways in which the UNIX system influenced our games development are subtle, as influences often are. While some of the members of the Games Group had not been exposed to the UNIX operating system before joining, the majority of the people in the group were longtime UNIX aficionados whose enthusiasm for the system was adopted by all.

Many of the influences we felt were philosophical. Although no one of them is exclusively associated with UNIX software, the group, taken as a whole, implies UNIX influence strongly. It would be impossible to consider the influence of the UNIX system without including the influence of people like Brian Kernighan and John Mashey, who have been eloquent spokesmen for many of the ideas and philosophies that permeate the system.

The concept often called "fail fast" seemed particularly relevant to games development. Much of the decision-making in game design comes down to making a yes-or-no decision about a potential approach. Such choices vary from "is this idea interesting enough?" to "can it be done at all?" We found it immensely effective to try solving the hardest parts first in deciding any of these questions, allowing us to discard doomed approaches quickly and devote our energies to potentially successful ones.

We always tried to make individual *tools*, each of which performed a single, general task well. We could then use a combination of such tools to perform a complex task rather than having to write a new program for each specific task. As a result, we had to write relatively few tools and ended up using each in many contexts—often in concert with tools written by others.

The programs we wrote generated little or no gratuitous output. Thus, when someone wanted to make a particular program a part of some larger construct, there was little or no effort spent trying to get rid of useless verbosity. Further, when diagnostics were needed, they really stood out, rather than being lost in a sequence of "program starting...", "Pass 1 completed", "Pass 2 loading...", and so forth.

4.2BSD

FROM NOW ON, CONSIDER IT SUPPORTED.

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

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

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

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

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

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

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

"We know UNIX™ Backwards and Forwards"

MT XINU™

UNIX™ SUPPORT FROM BERKELEY

Circle No. 9 on Inquiry Card

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

THIS AD IS FULL OF LIES.

Until today there *was* no integrated, multi-user productivity system that could be *shipped immediately*. As a result, a number of previously accepted statements about the multi-user software market are simply not true. For instance:

LIE 1.

There are no readily available multi-user integrated productivity solutions.

NO!

Prevail is a UNIX-based, multi-user business productivity system which is being shipped and used today. Prevail offers business users standard office automation products framed in a straightforward and easy-to-use window environment.

Prevail components include:

- ☐ Word Processing
- ☐ Spreadsheet
- ☐ Database Management System
- ☐ Report Writer
- ☐ Graphics
- ☐ Window and System Manager
- ☐ Communications
- ☐ Applications Development Language

LIE 2.

Nobody offers software that enables end users to tailor systems for their day-to-day operations.

WRONG!

Prevail combines applications development and business productivity tools and provides them in such depth and breadth that people with differing levels of needs and skills will tailor the system to the degree they wish.

LIE 3.

There is no system on the market which makes it easy for software developers and end-users to use the same environment.

NONSENSE!

Prevail removes the barriers that have been making departmental productivity so difficult to achieve. Because everyone has different needs and skills, Prevail offers people-oriented tools. Tools with power and flexibility.

LIE 4.

Software suppliers don't offer adequate post-sale support.

TOTALLY INACCURATE!

Inspiration Systems backs Prevail with outstanding customer service, training, and documentation. We'll be there to help you get started and to ensure that you make the most of Prevail. Day after day, week after week, we'll be here to answer your questions.

The truth of the matter is that Prevail is the only integrated, multi-user productivity solution that can be shipped to you today.

DISCOVER THE WHOLE TRUTH.

Discover **PREVAIL**

Inspiration Systems, Inc. · Production Plaza · Sewickley, PA 15143 · 412/771-4000

1

I'd like to discover the whole truth about PREVAIL.

ISI Inspiration Systems, Inc.
Production Plaza
Sewickley, PA 15143
412/771-4000

NAME____________________ (PLEASE TYPE OR PRINT.) COMPANY ____________________

ADDRESS____

CITY ____________________ STATE__________ ZIP__________

TELEPHONE ____________________ HARDWARE TYPE ____________________

☐ END-USER ☐ VAR/DISTRIBUTOR ☐ OEM

☐ I'M IN A HURRY, PLEASE ASK A SALES REPRESENTATIVE TO CALL ME

Circle No. 70 on Inquiry Card

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

We found simplicity to be an even better policy than honesty. Every time we were tempted to elaborate on some program, the result was an unending job of fine-tuning the elaborations—a task probably made more difficult because the elaborations never really belonged in the first place. When we stayed simple, it was easy to see what needed to be done, and it was easy to tell when we were finished and could go on to the next task.

UNIX ENVIRONMENT

The UNIX environment was perfectly suited to our work. When we needed a program, it was either already there or was easy to write. Perhaps that's because the UNIX system was designed by (game) programmers to make the job of program development as easy as possible. In any case, it made *our* job easy. We didn't have to fight with programs that *almost* did what we needed but had made some limiting assumption. Nor did we have to trick the operating system into letting us do something it thought we should not do.

We were heavily influenced by the software, philosophies, and concepts associated with the UNIX operating system. It could be argued that the time was right for these developments and that the UNIX system was just the vehicle for ideas that would have appeared anyway; but it's clear that without these, our games projects would have ended very differently. We would have spent more time chasing down blind alleys and would have examined fewer potentially profitable approaches. We would not have been able to participate in as many areas of the production as we did, thereby leaving many crucial design decisions to the same people who crank out the standard, bland products that flood the market. Worst of all, we probably would have had to throw away the "throw-away" games not only because we would not have had time to complete them, but because they would not have been as interesting as they are.

Peter Langston has been playing with computers for 20 years, more than half of them as a UNIX devotee. He now plays at Bell Communications Research in Morristown, N.J. His UNIX amusements have led him to both the academic/research and the business/commercial communities. While in the business sphere, he became involved with the first commercial UNIX license. Mr. Langston is also an inveterate toy maker, whose toys have touched thousands of lives; the one that's probably best known is "Empire", a complex, multiplayer, real-time, global simulation game. He holds a degree in Chemistry, but insists he remembers nothing of the subject. He was one of the creators of the world's first (and perhaps only) timeshared analog computer. ■

UNIX is a trademark of AT&T Bell Laboratories; Apple II is a trademark of Apple Computer Inc.; Atari is probably a trademark of Atari Inc.; DEC and VAX are trademarks of Digital Equipment Corporation; Picture System I is a trademark of Evans & Sutherland. Sun Workstation is a trademark of Sun Microsystems Inc.; Rescue on Fractalus, Ballblazer, Pixar, LUDS, and ASP are trademarks of Lucasfilm Ltd.

This article appears courtesy of the European UNIX Users' Group. ■

Stick this in your RAM.

All this common desktop clutter hides neatly away in your UNIX system's host computer. R Desk makes office communications far more efficient AND makes every user better organized, more productive.

With a keystroke, you summon your electronic desk calendar, your appointment book and your personal "to do" list. You can even set up meetings with others, automatically.

Or call up your Rolodex™—with as many as 8,000 names and addresses per user. Find the data you need and return at once to your program, or print out mailing labels!

There's a calculator built in. Plus an automatic telephone dialer that works with your modem. Even an electronic message pad, so phone calls are noted on terminals—instead of little pink slips of paper.

User IDs keep personal files private, while company lists (vendors, customers, etc.) can be open for everyone.

This powerful organizer runs on Motorola 68000 or Intel 80286 based systems. It comes on a standard 5¼" diskette. It's fully menu driven—but we'll include a how-to manual for diehards who insist on documentation.

Guaranteed to make you more productive in 30 days, or return for full refund!

R Desk™

BY R SYSTEMS, INC.

Circle No. 30 on Inquiry Card

Only $295!

Call toll free, (800) 527-7610. In Texas, call collect (214) 343-9188. Or send this coupon with payment today to:

R SYSTEMS INC.
11450 Pagemill Road, Dallas, Texas 75243

NAME __________

COMPANY __________

ADDRESS __________

CITY __________

STATE __________ ZIP __________

TELEPHONE __________

Send check or money order for $295. (Texas residents add sales tax.) Or charge to

___ VISA or ___ MasterCard

CARD # __________

EXP. DATE __________

SIGNATURE __________

UR685

REACH OUT & TOUCH THE UNIX SYSTEM

The execution environment that answers when you call

by Tony Cuilwik

Every time you place a telephone call, a computer using the UNIX operating system either monitors, measures, helps route, or ensures the quality of the call. The order is a tall one, given the standards people have come to expect when they "reach out to touch someone", but the fact remains that the fundamental integrity of the national telecommunications network depends on more than 1000 real-time, minicomputer-based systems that are built on a version of the UNIX operating system.

Among the varied and wide-ranging functions these systems perform are network performance measurement, automated network testing, circuit order planning, circuit order recordkeeping, automated trouble detection, automated or directed trouble repair, service quality assurance, quality control, inventory control, customer recordkeeping, and customer billing—as well as any number of other operational and administrative functions.

The systems vary from single-machine, local-access configurations to multiprocessor, distributed systems affording wide-area access. One characteristic they all share is the ability to present data to users in real-time.

In most telecommunication applications, a system supporting operations collects data in intervals ranging from tens of seconds to tens of minutes. This data is then held in memory in the telecommunications equipment itself or in special purpose microprocessors that are connected to it. Spontaneous events occurring within the network or resulting from human interactions with the systems require "real-time" responses that can be measured in seconds or minutes. The object is to guarantee a minimal acceptable human response time. This challenge has been met by tuning the underlying UNIX system.

Historically, the need for "real-time" operations systems was recognized in 1969-70, and the development of such systems began in earnest in 1971. Early in this period, it was determined that an operating system and environment should be provided to system designers, who would then only need to develop application-specific software. By 1974, several sites had chosen the UNIX operating system as this development environment. A few, meanwhile, had also selected it as an execution environment and were busy designing enhancements and improvements for the system.

Soon afterward, it became apparent that a common operating

Illustration by Daniel Hanson


ABC
DANIEL
HANSON 85

environment between projects would facilitate work. So, by 1976, a common "real-time" UNIX was supported across some of these projects. Among these were environments in which typical timeshared UNIX response times (up to a full second for scheduling and as much as several seconds for swapping) were simply unacceptable. Other system deficiencies that had to be "designed around" were the lack of good interprocess communications and the lack of an effective means for measuring and controlling system performance.

Major additions necessary to move the timeshared UNIX system into real-time applications included interprocess communications (name pipes, messages, semaphores, and shared memory), file access (logical file system, record access system), error recovery, power fail/restart, and line and terminal disciplines. These additions were developed, integrated, or donated to the common good by people developing specific systems. By 1979, there was an enhanced real-time UNIX system that was centrally supported, offering a collection of tools and a number of human/machine interface designs to protect system users from direct contact with UNIX primitives.

Two techniques commonly employed by designers of these real-time systems were:

1) the design of highly efficient device drivers that throttle the data flow and minimize UNIX processing time.
2) the design of robust and specialized database management systems that work around weaknesses in the UNIX file system.

Another technique involved designing software that minimized the use of time-consuming processes—like context switching—or that carefully managed inefficient functions such as disk reads and writes. These techniques made for dramatic increases in real-time performance that contrasted sharply with the results obtained from software that was merely designed according to "good programming techniques", but not "tuned" to the use of UNIX features.

Every time you place a telephone call, a computer using the UNIX operating system either monitors, measures, helps route, or ensures the quality of the call.

AN APPLICATIONS SAMPLER

Representative examples of system design can serve to illustrate the size, functional capabilities, and role that the UNIX operating system currently plays in the day-to-day operation of our nationwide communications network—as well as to illustrate the type of real-time performance that has been achieved.

One such system interfaces with electronic switching systems within each region (or company) to collect telephone traffic data for short-term monitoring and control of the network. The system's objective is to allow people to respond in real-time to network problems in order to maximize the call completion rate.

Polls are made of interconnected switching offices every five minutes for traffic counts, and calls are made every 30 seconds to get readings on any alerting events. The total telecommunications traffic for one or more states is collected by each system, producing nearly 3 MB of data that must be stored, analyzed, presented on a wallboard display, and formatted into output reports at the close of each five-minute interval.

The UNIX operating system serves as a software base for this system's 12 major software subsystems, which are comprised of over 70 programs and 1100 C language modules. The real-time nature of the system was achieved through the careful use of UNIX services; the design of special device drivers for the interface hardware; the development of an application-efficient logical file system; and the use of an executive control program to initialize the system, control real-time application processes, and maintain the heartbeat of the system.

Some other operations systems designs have actually integrated machines that were dispersed across several states into cohesive, company-wide networks using point-to-point data lines. One such system administers and maintains the millions of customer records that must be kept for individual residence and business telephones. This system is deployed nationwide and is used to respond to customer trouble reports ranging from isolated equipment problems to the restoration of service to whole geographic areas during crises such as storms or floods.

The system architecture consists of multiple front-end machines connected through 50 Kilobaud lines to a mainframe host. The front-end machines use the UNIX operating system to handle approximately 175,000 transactions per day, with peak loads of 28,000 per hour. The software architecture in these machines was designed to provide a high performance transaction processing environment. In particular, its key software components include: a new file system that emphasizes system recovery from failures and the protection of database consistency, a database management system that is tailored to the application, and a number of transaction processing tools.

The philosophy driving the development of systems like these was that the UNIX system should be used for those functions that made sense, but that a superstructure should also be built around the system so as to achieve optimal performance.

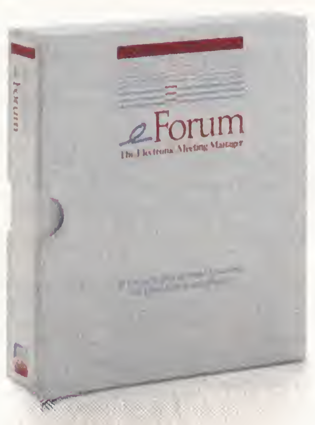
One of the first such systems to be deployed nationwide on AT&T computers provided a graphic report card on switching office performance and telephone service quality. This was used to pinpoint problems in the highly complex telecommunications network, including reports on problems that were statistically insignificant on a local basis but could be patterned on a global basis.

Today, this system uses an AT&T 3B20 minicomputer with 2 MB of main memory. Typically, some 2000 telephone offices are interconnected into the system through direct connections to major switching offices or through data links to other minicomputer support systems already at work collecting the needed data. These data sources are continuously polled for information, allowing the system to provide spontaneous reports to network analyzers as well as provide an on-demand analysis of the data.

The UNIX system running this application was tuned specifically for the task, and special drivers were written to collect the data. A human/machine interface was provided through clear, graphic presentation of data and the availability of multiple-level help functions able to assist analyzers at every step. An application-specific database management scheme was also provided to sort and analyze large volumes of data without placing heavy demands on the UNIX file system.

Up to 100,000 messages of 200 characters each are collected daily using this facility—with as

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. 6 on Inquiry Card**

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

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

many as 12,000 coming in during a busy hour. This data is patterned against a model of the network in order to determine correlations. More than 20 people analyzing the network data on an almost continuous basis depend on the system for the information they need to direct maintenance forces to problem areas.

CONCLUSION

Systems such as these—and many others as well—have been deployed throughout the telecommunications network and have become tightly integrated into its daily operation. They have typically achieved a 98 to 99.5 percent system availability ratio without the duplication of computing facilities. What's more, these systems have proven in all cases to be extremely cost effective in performing the tasks for which they were designed.

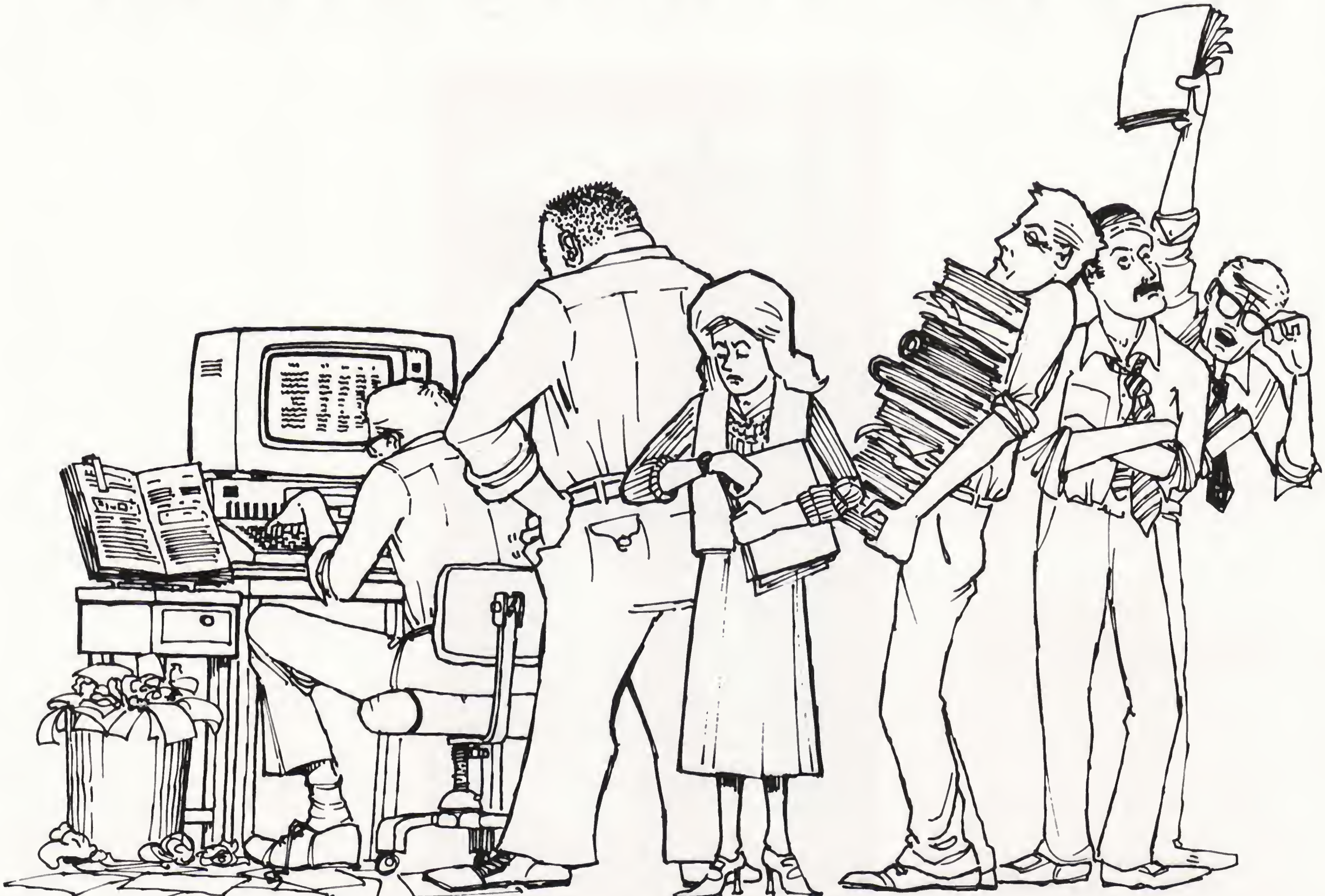
The fundamental integrity of the national telecommunications network depends on more than 1000 real-time, UNIX-based systems.

Over the past decade, many minicomputer systems have been designed and deployed using the UNIX operating system as a real-time applications processor. The UNIX system development environment has enabled many hundreds of system and software developers at AT&T Bell Laboratories to achieve high productivity in the design of these real-time systems. The wide range of hardware options, the ability to share a robust and far-reaching tool set, and the ease with which a system can be enhanced or adapted to changing user needs have contributed to this productivity.

For all that, the most important factor behind the ever-expanding use of the UNIX operating system as a development environment is probably the culture, or positive mindset, of the designers themselves. People building software systems in a UNIX system environment seem to consistently have an "I can do it" attitude. It's been this sort of confidence that has resulted in the development of the more than 1000 real-time systems currently in use in the telecommunications network. Each in its way has helped to move, measure, or improve the voice and data traffic we all generate each time we pick up a telephone.

Tony Cuilwik is the director of AT&T Information Systems Laboratories in Columbus, OH. Before coming to his current post, he served as the head of the Operations Systems Development Department at Bell Laboratories. In the course of his work with operations systems, Cuilwik has personally overseen the development of four UNIX-based support systems. ■

IBC IS THE ONE COMPUTER YOU NEED WHEN YOU NEED MORE THAN ONE.



If you want a computer to help you run your business, then you need a multiuser computer. Not a PC. It's that simple.

A PC is a personal computer, not a business computer. That's because PCs are single user computers with single user software. They're good for one person — but not good enough for a whole company. Even if the company is only two people.

Every computerized business has someone entering information someplace while someone else is looking up information somewhere else. That's two users. And every business has more than two users who need access to the computer. That's a multiuser computer environment.

The idea of multiuser computers is not new. They've been around longer than PCs. Companies like IBM, DEC, and HP have been selling multiuser business computers for years. So has IBC.

Recently the PC has gotten all the attention. It has a low price and can do a lot of new things for your business. But as soon as you buy the first one, you'll realize that you really need more. But don't buy more than one. Buy one multiuser business computer from IBC. It costs as little as a PC and can do more — a lot more.

If you want a good business computer, look at what IBM, DEC, HP or anyone else has to offer. But if you want that business computer at a great price, then look at what IBC has to offer. We've been building low cost multiuser business computers since 1979. And that's longer than anybody else.

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. 29 on Inquiry Card

THE UNIX APPLICATIONS STORY

An interview with Berkley Tague

UNIX critics commonly complain that the system suffers from a lack of applications software. These complaints sting because it's true that there are a number of markets where UNIX penetration is barely visible. But it should also be noted that there are a number of other markets that absolutely gush with UNIX bounty. The telecommunications field is a prime example.

As noted elsewhere in this issue of UNIX REVIEW, much of the nation's telephone network literally depends on the operating system. One of the key reasons this is true is because a man named Berkley Tague pushed for it.

During his 25 years with AT&T Laboratories, Tague has headed up departments responsible for computing services and applications development environments. In that role, he has consistently championed UNIX.

Today, he heads the Laboratories' Computing Technology Department in Whippany, NJ, where computing services that can be sold internally are developed. Among the development projects currently underway are ones related to CAD applications, floating point capabilities, and Fortran compilers.

UNIX REVIEW asked Ned Peirce, a systems analyst specializing in UNIX, to ask Tague for his views on the history and future of applications under UNIX. As Peirce discovered, clues about the future can often be found in the past.

REVIEW: *In your work with USG [the UNIX Support Group], you were involved with many of the earliest applications designed to run under UNIX. Based on that perspective, what would you say is the overall motivation today for developing UNIX applications?*

TAGUE: That's really pretty straightforward. The UNIX system is a very convenient environment for applications, and an easy environment for developing programs. It also promotes easy communication among people who are using the system to run applications—as well as among those who develop them.

Look at why we started on the UNIX system in the first place. It offered us an operating system environment that was independent of the hardware vendor—one in which we could move our applications from machine to machine with much less trouble. These software advantages came

Photos by George Peirce/Studio 84

in spades for hardware because big operations could suddenly be moved more quickly to newer machines, making it easier to keep up with the technology driving the industry.

REVIEW: *In your current post as head of the Whippany Computer Center, do you see yourself continuing to use a spectrum of machines?*

TAGUE: Absolutely.

REVIEW: *You don't see yourself migrating toward a shop full of large machines?*

TAGUE: No. Indeed, one of the services I offer is what we call an "allocated machine". There is something nice about giving customers their own computer systems that they don't have to share. You can say it is all yours, as distinguished from a mere piece of a larger machine.

REVIEW: *Your center doesn't have to manage the load?*

TAGUE: That is right. But we do have a "Fair Share Scheduler" that divides up machine resources so that we can allocate whatever fraction of a machine we wish. But, the most popular solution is still to get an appropriately sized mini and put your name on it.

REVIEW: *Do you also find yourself selling generic UNIX cycles?*

TAGUE: Yeah, we really do. We are coming out with more UNIX system applications of the PC variety. This will offer services for non-technical people, so there's a growing marketplace I am looking into. We are also getting more interested in UNIX-based office mechanization. I have direct responsibility for developing and propagating office mechanization tools in all the comp centers.

I also handle central support on graphics. Both are areas where we are looking to do new things. We are also in the process of moving the CAD/CAM folks over to UNIX systems.

REVIEW: *What do you see as being the most needed types of UNIX applications?*

TAGUE: I feel that the weakest area at the moment is applications software for non-technical users. By looking at PC catalogs, you can see many applications that have not been written for the UNIX environment yet. This software has got to get there.

REVIEW: *Before we get off too far into the future, could you retrace a little of the history behind applications under UNIX?*

> The UNIX system is a very convenient environment for applications and an easy environment for developing programs.

TAGUE: I think we're going to see the commercial marketplace replicate to some extent the experience we've had within Bell Laboratories. The UNIX system started in the Research community here, where it was seen as a nice environment to work in because it was designed primarily for sophisticated programmers doing program development.

Of course, the system wasn't exclusively for programmers, since there were many people who used it for numerical analysis and other applications not restricted to dyed-in-the-wool computniks. The fact that these people could use the system owed much to people like [Brian] Kernighan, who had a big interest in making things accessible to folks who were not gurus.

In any event, most of the system's users were involved in software development. There are people in Research who have been using it for years. These people continue to want it since it is very flexible and can be easily modified. Flexibility is far more important to these people than support.

Within the Labs, though, was a group with reasons for using the UNIX system that stem from somewhat different concerns. I'm referring to those involved in the development of applications for internal operations. We got started with the UNIX system in the early '70s, when other operating systems that were available for minicomputers weren't really much better than PC-DOS. Most offered little more than a file system and single-user capabilities. The human engineering, even if you were a guru, was pretty lousy. You even had to deal with things like numeric filenames in some of the systems.

The discovery that we had the need—or actually, the opportunity—in the early '70s to use these minis to support telephone company operations encouraged us to work with the UNIX system. We knew we could do a better job with maintenance, traffic control, repair, and accounting applications.

REVIEW: *Is that because the existing systems were being run in crude operating system environments?*

TAGUE: No. The existing systems were made up of people and paper. As you may recall, the phone business was in danger of being overwhelmed in the early '70s with the boom of the '60s. There was a big interest then in using computers to help manage that part of the business. We wanted to get rid of all of those Rolodex files and help those guys who had to pack instruments and parts back and forth just to keep things going.

REVIEW: *You mean you mechanized for the first time in the '70s?*

TAGUE: There had been mechanization of these processes to some extent for quite some time prior to that—but only in a local way. Take repair, for instance. A lot of it deals with keeping the connections straight between what we call the main distribution frames in the central office and the wires that tie residential telephones into the switch. Prior to the use of computers, "mechanization" consisted of somebody on a remote test bench using electrical meters and instruments to test lines. To get those connections made, an intercom was used to broadcast requests to a bunch of people standing around with alligator clips and soldering irons down in the wire center. The requests went something like, "Would you kindly connect jumper x to terminal y?" to get testing done.

The mini made an impact on this process in two different ways. First, we were able to get more instructions out to the people actually making the connections. And, at the other end, we were able to centralize information about entire systems and end-to-end circuits.

This meant that if I was responsible for keeping the Superbowl broadcast on the air between New Orleans and New York, I could—with a single console—view all the connections in that link and have access to all of the information automatically being collected about it. If something broke, I could immediately recognize that and orchestrate the process of getting it repaired. The repair itself would ultimately be left to a person working in much the same way as before.

REVIEW: *You're talking about the person with a soldering iron?*

TAGUE: More likely someone plugging in an alternate module or pulling a manual switch and going to a backup system. Suddenly, their work became much faster because the information was all in one place—unlike earlier days when eight guys would have had to collect and sort out the trouble data in a series of phone calls before actually being able to get down to the business of working on solutions.

There were also other applications that started to surface in areas like cable and wiring layouts. The algorithms applying to these layouts were well known here at the Laboratories, but they were not the sort of thing you could usefully put into a manual. They were, however, easily put into computer programs. Optimum layouts could thus be generated using the computer to assess all the complicated engineering tradeoffs.

The value of computers was fairly evident and the economics of the minis were clearly there—people could actually afford them. But many people starting in with computers at that time were unsophisticated as far as software was concerned. I had a planning role and tried to get these people to understand that they were going to need a decent software environment to do their development jobs. What's more, if we faced the phone company with 18 different vendors and 19 different environments, neither the developers nor the phone companies were going to be able to maintain the thing once it got out in the field in large numbers. As a planner, I was trying to focus on a few vendors. At that time, it was primarily Hewlett-Packard and DEC, plus a few IBM systems.

The next question was: what are we going to do for an operating system? Vendor operating systems were available as a starting point; but a number of people had already started to build their own when they realized that what the vendors had was not adequate.

> I knew in my heart of hearts that once they got on UNIX, they wouldn't be able to do any better with the experience and the schedules they had.

REVIEW: *For what reasons?*

TAGUE: Well, for all sorts of reasons—ranging from inadequate file systems, to inadequate performance, to poor user interface. We sold those first application developers on UNIX simply by pointing out that the first job they were going to have to do was program development and that by using the UNIX operating system they could get that job done more easily. I did not argue with them about whether or not they should develop their own operating systems—knowing in my heart of hearts that once they got on UNIX, they wouldn't be able to do any better with the experience and the schedules they had. Indeed, that is what happened.

The other thing we pointed out to these people was that they could exploit UNIX portability, which first appeared around 1973.

REVIEW: *Was that when the portable C compiler was introduced?*

TAGUE: Yes, but not just the portable C compiler. The portable version of UNIX was also released at that time. If you did your development with UNIX on a prototype from vendor A, you could then go out for bids on 300 deployed systems since there was no need to stay with the same vendor. I don't think anyone ever took the opportunity to switch vendors, though most projects upgraded to newer vendor systems as they were offered.

REVIEW: *They continued using DEC?*

TAGUE: Yes. DEC quickly became the dominant vendor, but the portability still came into play because as DEC kept upgrading its systems, we were able to simply port the UNIX environment and the software we'd written for our applications. The people on one project had faced a loss of several months in their schedules when they had to make a transition from RSX-11B to RSX-11D. They switched to UNIX instead.

Portability is still a virtue in today's marketplace. An individual intent on setting up a computing operation now has a choice of vendors and the opportunity to benefit from a large number of value-added software vendors working to solve various problems. This is particularly important in the smaller machine market, where there isn't a single vendor that dominates in the way that IBM systems provide de facto standard for larger machines.

By getting UNIX out there as an open, accessible, licensed standard, AT&T has seen to it that

The weakest area at the moment is the applications software for non-technical users.

many vendors can play. This benefits the commercial user in the same way that we had originally intended our commitment to portability to benefit us.

Another important consideration at the time we adopted UNIX was support. I personally switched from a being a planner to being a supporter and developer of UNIX. I did this by creating the need and then volunteering to fill it. We put together the UNIX Support Group [USG] in September of 1973 and released the first C version of UNIX internally. [Generic I, II, and III were produced by these initial efforts.] In parallel with our efforts, the Programmer's Workbench gang under Rudd Canaday worked the same vein over in the BIS [Business Information Systems] area.

The Programmer's Workbench exploited and built on the classic use of UNIX for program development but in a broadened context. It made the necessary connections to feed IBM and UNIVAC over RJE links, for instance. But, by and large, it was focused on program development and manipulation of documents and data.

The systems developed for operations, however, were all characterized by two functions for which UNIX is popularly thought to be ill suited—processing transactions against databases and real-time capabilities for data collection. The interesting point that has often been missed outside of Bell Laboratories is that UNIX ain't all that bad in those areas.

REVIEW: *As a matter of fact, it is rather heavily used for those purposes.*

TAGUE: Indeed. A number of modifications were made to UNIX to allow for that, primarily by the developers at the Columbus Laboratories. As an example, some mods were made in the area of interprocess communication. These were required because the Columbus people were building ad hoc database managers for their systems. These proved to be fairly high performance database managers. They used UNIX and its file system to put a prototype together, but once they knew what it was supposed to do by dint of experience, they developed special purpose database managers. The typical manager was a pro-

cess that managed its own piece of the disk. This was done outside of the UNIX file system in order to pick up speed. The interprocess communication hooks were put in to permit communication between the fast database processes and other processes doing transactions. Named pipes and other stuff came out of that. While not all of the features developed at Columbus have been issued commercially, a subset of them have.

But an important part of the database business will require general-purpose database managers. Several are now offered in the UNIX environment.

I have a feeling that the ad hoc database manager still has a lot to recommend it, particularly among turnkey UNIX-based systems designed for those customer markets that need database management. If you really understand your problem and know what you want, such a system is hard to beat for performance.

REVIEW: *Do you think it will remain that way?*

TAGUE: Yes. It is not that difficult to build things under UNIX. We have plenty of experience in doing that and have found it to be a very effective way to proceed.

REVIEW: *As the hardware gets faster and faster does "real-time" start to occur in systems that were not designed to be real-time systems?*

TAGUE: Not really. We handled real-time in two ways. Most of the Columbus folks isolated the real-time part of the software in a special driver. UNIX is built to take on new drivers without a lot of trouble. A typical example from that era relates to the fact that our switching machines had been built to use a teletype as their output device. There were some lines coming out of our ESS machines that used teletype protocols, but these were really pretty hairy—they did all sorts of funny things that made sense on a model 35 tty but not when we ran them at 100 Kbaud speeds. You'd end up with a big burst of data that you'd lose if you couldn't catch it right off. Your job was to write a driver that

Portability is still a virtue in today's marketplace.

could catch that burst and then throw it up on the disk somewhere in a hurry so you could come back and look at it later at your leisure.

Basically, the gain comes in writing a driver that cuts the real-time part of the problem down to the point where the real-time performance required after that burst of processing is roughly equal to what you would get out of UNIX's timesharing response. In most cases, that solves the problem very nicely.

The other real-time issue in UNIX came up in quite a different area—the laboratory environment. There was a large group of users in basic research that needed help. People were using a large UNIX system and sharing it among several experiments. The trick here was to use a small PDP machine or some other microprocessor as a front-end. You could put the millisecond-level, real-time part of the operation on the microprocessor and then collect the data in the host UNIX system after buffering on the smaller machine. There was a version of UNIX invented at that time that would run on such a local micro. It contained some software supported C programs as though you had the whole UNIX environment available. In essence, the system would allow you to do things very fast locally until you asked for something like a file. It would then reach back to its "mother" UNIX machine and do the actual file access there. The file would then be moved down. People could write applications in C and put them down in this little box and do millisecond speed control. As long as they were careful about which commands they used and stuck to the local ones when speed was an issue, everything was fine. With care, you could control your experiment and collect your data from a single, normal UNIX console located next to a small, fast engine capable of controlling the experiment through very high-speed tentacles.

REVIEW: *That sounds like good experience for front-ending some of the larger machines.*

TAGUE: Yes. Very much so. What's more, if you think about the PC 7300 in this context, you can see that it represents an interesting opportunity.

REVIEW: *What other trends do you see?*

TAGUE: When I look at today's market, it seems that all the pieces are already there. I think that the external market for UNIX is going to repeat our internal history. People building turnkey systems in many parts of the world have already latched onto ad hoc solutions corresponding to those we used when mechanizing our office systems. The thing I think is still missing is the kind of software you find on the PC—the legion of small applications packages that range from general ledger programs, to spreadsheets, to user programs—or whatever. These must be programmed under UNIX with the same sort of non-technical interface that people have grown used to on the PC. There is no reason why this can't be done and, indeed, we've got a fair amount of experience internally doing this already. We made UNIX available some time ago for the non-technical parts of our internal market.

REVIEW: *You may have made a good first step toward taking that expertise to the outside market with the introduction of the 7300.*

TAGUE: Indeed, the 7300 with its menu shell is a good example of what we need. In mechanizing any of these processes for the non-technical marketplace, there are two pieces to the problem. One is that you need to include a veneer that speaks to the non-technical person, whether it be menus, or touch screens, or whatever else it is people are comfortable with. The other thing is to get the software to talk in a vocabulary that is reasonable for these people. I think if you begin to do that, you will find that the potential UNIX has in that market will actually begin to be realized. If you look around the world, you can see some such systems coming along already.

REVIEW: *Without using actual numbers, how would you characterize the overall maturity of UNIX in the application arena?*

TAGUE: The reliability and availability of UNIX was very good right from the start. Today, in my center, I expect to get a "99 percent plus" availability during prime shift. The 3Bs, in particular, do better than that. The thing that I think has changed is that the stack of work to be done—the modification requests, if you will—has been much reduced.

Among those requests are some saying, "I'd like an enhancement," and others saying, "Something works wrong and it ought to be fixed." The numbers of items in these piles have been in the thousands. There are probably between 5000 and 10,000 UNIX maintenance requests that have been handled over the years. There are still many that are open, but they are not visible to me anymore.

REVIEW: *So your overall impression is that things are good?*

TAGUE: Absolutely. It is an improvement over what we had in '73 by a long shot, and it is a definite improvement over what we had during my last tour in the comp centers at the end of the '60s as far as what a timesharing system will do and with what reliability. I don't think we have significant problems here. With a big piece of software, I don't expect it to have zero defects in my lifetime. But it certainly can have zero defects in a practical sense.

By getting UNIX out there as an open, accessible, licensed standard, AT&T has seen to it that many vendors can play.

REVIEW: *Do you find yourself relieved that there is now a UNIX support organization that is apart from the one you are involved in?*

TAGUE: Now that we have announced the thing commercially, it is absolutely essential that we have a field organization capable of supporting the expanding marketplace. In terms of what we do here, we use AT&T's services as backup for our own local support. The fact is, aside from a patch of trouble we ran into last September, we haven't had much need for those services. We do most things for ourselves here. Most of what we do is concerned with installing new systems and new versions of the system, as well as installing new applications and whatnot. I don't have anyone in my department who I would consider a real dyed-in-the-wool UNIX systems guru.

REVIEW: *Really?*

TAGUE: I have to be careful not to insult my people, but the point is that there isn't much need for a guru any longer. Our systems administrators today are MAGs [Members of Administrative Group], not MTS [Members of Technical Staff]. The focus is on the use of the system, not the system itself.

REVIEW: *Are you saying that, even in a center this size, you don't need a classic UNIX kernel hacker anymore?*

TAGUE: That's right—at least not the sort who *really* understands the kernel. We get our systems from the Chicago Indian Hill Comp Center, which adds in a bunch of stuff to produce suitable Comp Center editions of System V. They do very little to the kernel. They put a few hooks for accounting and the "Fair Share Scheduler" into the kernel, and even those might be gone by now. They put in a lot of extra commands, of course. We add in some local commands as well.

System reliability, in terms of fundamental reliability, is not an issue any longer. That is not true of all the applications under the system, however. There are application subsystems that still have some problems.

REVIEW: *What about security—particularly here at Whippany, where you have a lot of government work going on?*

TAGUE: When you start talking government security, you find very rapidly that the government has really only two modes of operating. One is where they run an unsecure system like everybody else and the other is where they have a classified requirement, in which case you essentially lock your system up in a safe. The terminals connected to such a secured system also have to be part of the safe, as do all the offices.

The security issue under UNIX

It is not that difficult to build things under UNIX.

has the following properties: first of all, security is an unsolved problem in search of an evolving solution. One way or another, it is something that has to be constantly watched, monitored, and improved as you go along. Certainly, one must consider the potential threat and the motivation to get at machines on a case-by-case basis. No one that I know of is doing international money management on UNIX yet. But people are doing it on systems that I suspect are no more secure, and perhaps less so, than UNIX. In any case, I haven't yet found anything in UNIX that makes it any more or less secure than any other operating system in principle.

In practice, UNIX offers some good news and some bad news. The good news is that because it is a fairly simple operating system written in C—which is a reasonably understandable language—one at least has a prayer when going over code in search of security holes. One also has a good chance of fixing them. Any attempt to do that with a traditional, large-machine operating system would prove to be a very large task. Most people working on secure operating systems have either started with UNIX or something of that ilk. They don't start with a major vendor's system for good reason. The other good news is that because of the way UNIX is built and the way its system administration capabilities are structured, it makes it easy to monitor activity.

The bad news is that the hacker can make use of the same toolkit. UNIX systems have traditionally been open, and are still often administrated in that way. Good system administration can provide users with as secure an environment as can be obtained on other systems.

REVIEW: *But for all these strengths, you indicated in earlier* UNIX REVIEW *interview that you once recommended against releasing UNIX to the outside world.*

TAGUE: The point I was trying to make in that interview was that, as a developer looking in on Bell Laboratories' Research and telling them about their business, I was wrong. I was overridden when they said, "We can ship this out to universities and it will work just fine." My concern was that after putting it out, some guy out at the Ohio National Bank who also happened to sit on our board would end up putting his payroll

Continued to Page 94

RULES OF THE GAME

The specter of antitrust

by Glenn Groenewold

GarageCo, Jack Megabyte and Helga Termcap's innovative software company, and Laidback, an enterprise begun by a collective of MIT graduate students, have independently developed programs capable of scanning a 1,000,000-word manuscript and correcting it to conform to the standards laid down by Wilson Follett in *Modern American Usage*. Although deplored by the Association of Magazine Editors on the basis that the livelihoods of its members are threatened, the competing software packages have proven to be tremendously popular with writers. Demand, in fact, has been so great that Behemoth, Inc., has concluded that it must include one of the programs in the built-in software for its Sycophant II home computer, due to hit the market in just eight weeks.

But Behemoth's intentions are revealed to Jack Megabyte by a disgruntled employee of the corporate giant, who tells him that the company plans to play GarageCo and Laidback off against each other. Behemoth expects to use its economic muscle to extract a license for one of the programs on terms that will enable it to market a combined hardware/software package at a far lower cost than the users would have to

pay to acquire the components separately.

Alarmed, Jack flies to Boston to meet with the graduate students. He convinces them that should Behemoth succeed in obtaining the rights to use one of the programs on the terms it seeks, both GarageCo and Laidback would be unable to license their software directly to users at profitable prices. Following this meeting, GarageCo and Laidback each refuse to discuss the licensing of their products with Behemoth's representatives.

But Behemoth does not take this rejection lightly. Knowing that it would take months for in-house development and testing of equivalent software for its Sycophant II, the company takes the offensive by filing an antitrust suit, claiming that GarageCo and Laidback, in refusing to deal with it, have conspired to restrain trade.

Though this illustration is not modeled on any actual case I know of, it's not entirely far-fetched. Our antitrust laws *do* provide a basis for bringing a lawsuit in situations similar to that just set forth. This will come as a surprise to a number of people who are under the erroneous impression that antitrust laws have as their main purpose the protection of small companies from big ones. It Ain't Necessarily So.

The courts have made it quite clear that these peculiarly American enactments are intended to preserve *competition*, not competitors. If the maximization of competition requires a ruling that has fatal consequences for one or more of the competing enterprises, well, that's unfortunate.

FROM THE EYE OF THE BIRD

There are several federal statutes to keep in mind when studying the antitrust area, which is considered to be a field of law unto itself. These enactments, and the ways they are applied, are incredibly complex, so it's not possible to do more here than touch on some of the high points. (And there are, remember, *state* antitrust laws, as well, which serve to complicate matters even further.)

From our standpoint, the most important federal statute is the Sherman Antitrust Act. This law has two parts, which are in some ways distinct and in some ways overlapping.

Sherman Section 1 forbids contracts, combinations, or conspiracies in restraint of trade. These all are things that necessarily involve two or more parties, so you can't violate Section 1 by yourself.

Section 2 of the Sherman Act deals with attempts to monopolize trade or commerce. This law does not make *monopolies* illegal as such, but instead prohibits activities that are purposefully designed to *create* or *preserve* a monopoly. It's not necessary that there be two to tango in order to have a Section 2 violation; you can do it all by yourself. It's quite possible, however, when two people are involved, for the same activity to constitute a violation of *both* sections of the Sherman Act.

Existence of a conspiracy under Sherman Section 1 can in fact be proven by circumstantial evidence. For example, though parallel conduct on the part of two competitors is not, in itself, enough to establish that they have conspired, evidence of meetings, trade association activity, or exchange of price information can complete the necessary proof.

Some practices are defined by Section 1 to be illegal *per se*. These include fixing prices, dividing customers and markets, forcing distributors or customers to take an unwanted product in order to receive a desirable one ("tying"), and refusing in concerted fashion to deal. But this doesn't exhaust the possibilities of illegal activity. *Any* action that *unreasonably* restrains trade can be grounds for a determination that Section 1 has been violated.

Section 2, the anti-monopolization portion of the Sherman Act, also can be more ferocious than it might seem at first glance. Possession of *monopoly power* has been defined by the courts as the holding of sufficient power to

There are a number of people engaged in modest enterprises who would be shocked to learn that the proscriptions of the antitrust laws might have anything to do with them.

control process or exclude competition in a *relevant market*. The joker is that the "relevant market" can be minuscule, so that even a small business enterprise can find itself in violation of Section 2 without great effort. Exactly what constitutes the "relevant market" for a particular product often can't be known until a court makes this determination in the course of an antitrust action.

The other federal antitrust law of primary importance for the computer industry is the Clayton Act. This regulates dealer arrangements primarily, and in some ways overlaps with Sherman Section 1. However, the Clayton Act is solely concerned with the sale or lease of commodities—not with services. Any *exclusive* franchising or leasing arrangement should always be looked at from the standpoint of Clayton Act considerations, as should any agreement that "ties" together two products for sale or lease.

Bear in mind that this is only a quick view of the top of the federal antitrust iceberg. There are other federal statutes in this field as well. Moreover, as mentioned, there is also the plethora of state laws that can't be covered here. Some of these other enactments lap over into the area of unfair business practices, and so it's difficult to segregate the antitrust aspects of business activity from the other legal ramifications this activity might suggest. (It's been said that the law is a "seamless web"; some who are not so charitable might choose to describe it as an "endless morass".)

WHAT IT MEANS

Having now gone over the ground that the lawyers for GarageCo and Laidback can be expected to cover in initial discussions with their bemused clients, we're in a position to understand the theory behind Behemoth's antitrust suit. Basically, the company has claimed that by refusing to discuss the licensing of their software, the two Lilliputian enterprises are engaged in a conspiracy that violates section 1 of the Sherman Act. Secondarily, Behemoth might also assert that GarageCo and Laidback are attempting to monopolize the market for software that conforms manuscripts to Follett, in violation of section 2 of the act.

How is this lawsuit likely to turn out? That's impossible to say, since the resolution of so many of the legal issues involved will depend on the court's determinations of fact. For instance, what is the "relevant market" for GarageCo's and Laidback's software? If it's determined to be the entire market for all software of all types, the two small companies will be home free so far as Section 2 is concerned, since clearly they control only a tiny portion of the market. But if the relevant market is defined as comprising only those persons interested in software that conforms manuscripts to the standards of Wilson Follett, they could be in trouble.

IMPLICATIONS OF ANTITRUST

Our antitrust laws represent one facet of the American dualism regarding business. We hold highly the idea of free competition for ideas and products, yet we also believe in enforcing property rights. These concepts inevitably collide on a number of fronts.

Copyrights, for example, represent our concern for the protection of a particular kind of property. Yet the aggressive utilization of the rights granted by copyright laws to the creators of intellectual property can place the copyright holder in antitrust jeopardy. In fact, in lawsuits for copyright infringement, it has become a standard tactic for the defendant to countersue by claiming antitrust violation.

Some commentators have criticized the antitrust laws, arguing that though they were intended to preserve competition, their actual effect has been to penalize those who compete effectively. Certainly there are a number of people engaged in modest enterprises who would be shocked to learn that the proscriptions of the antitrust laws might have anything to do with *them*.

But there's little likelihood that antitrust laws will disappear. To the contrary, regulation of business activity continues to increase, particularly at the state level. Anyone engaged in any business endeavor, on whatever scale, would be well advised to become acquainted with the requirements of any applicable laws of this type. It may very well be too late to pay attention once the court summons arrives.

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

On the fast track

by Mark G. Sobell

ELXSI has announced that it will be delivering a $2 million, 10-CPU system to Sandia National Labs in Albuquerque. CPU-intensive tests that ELXSI ran on the system reportedly produced *better* than a 10-fold increase in processing power over a single-CPU ELXSI system.

"This (testing) represents a breakthrough in multiprocessor technology," claimed ELXSI Vice President of Engineering, Dr. B. Kumar. "The limiting factor for multiprocessors has traditionally been the degradation in processing gain that results with each additional CPU. The System 6400 benchmarks show that we can implement multiprocessing on a very large scale and measure processing gains that even exceed the hypothetically optimal levels."

How can 10 computers hooked together perform better than 10 times as well as a single computer? "The performance gain is a result of a 'synergy' that allows each CPU to process more user work as CPUs are added," Dr. Kumar explained. "Load leveling and distribution of the message-based operating system over CPUs, which reduces system overhead to the CPUs as they are added, play a part in the results. While the applications used can also impact test results and this could reflect a best-case laboratory operation, it proves that ELXSI

linear performance is not just a theory."

According to Dr. Leonard Shar, vice president of R&D at ELXSI, each processing unit of the 6400 contributes a full measure of computing power to the overall performance. Shar attributes the performance of the multiple processor 6400 to the high-speed bus that ELXSI uses to connect processors with central shared memory and to the large amount of physical memory the machine enjoys. "The system that is being delivered (to Sandia) has been tested with 100 million bytes of central memory and can easily handle twice that amount, more than even the largest mainframe," Shar reported.

Though the machine that's going to Sandia is based on the proprietary EMBOS operating system, the 6400 can also run UNIX in both its single and multiple-CPU configurations. As a matter of fact, the 6400 can run both operating systems at the same time.

Does this all sound too good to be true? You can get started with a single processor ELXSI system for just $369,000.

Dropping down (just a little) from the stratosphere, Pyramid has announced an agreement to deliver several hundred machines to VMark Computer. The deal is reportedly worth about $10 million and will span three years. VMark will package its new, proprietary DBMS, uniVerse, on both the Pyramid single-processor 90x and dual-processor 90Mx. The system is designed to execute Prime INFORMATION and PICK-like applications in the UNIX environment.

Ed Dolinar, President of Pyramid Technology said, "Pyramid has resolved the compatibility issue between UNIX System V and 4.2BSD by offering both versions concurrently under OSx, Pyramid's proprietary UNIX operating system. VMark is now resolving the PICK vs. UNIX issue by offering both environments concurrently under uniVerse."

Why did VMark choose Pyramid? Roger Parsons, President of VMark, said, "We performed extensive multiuser benchmarks on hardware from a number of vendors, including AT&T, DEC, NCR, Convergent Technology, Gould, and Pyramid. We found that Pyramid offered the most solid, reliable UNIX implementation, and that it fully incorporated the features of both the AT&T System V and Berkeley 4.2 versions of UNIX."

Single-processor Pyramid systems start at $120,000.

SPEED READING

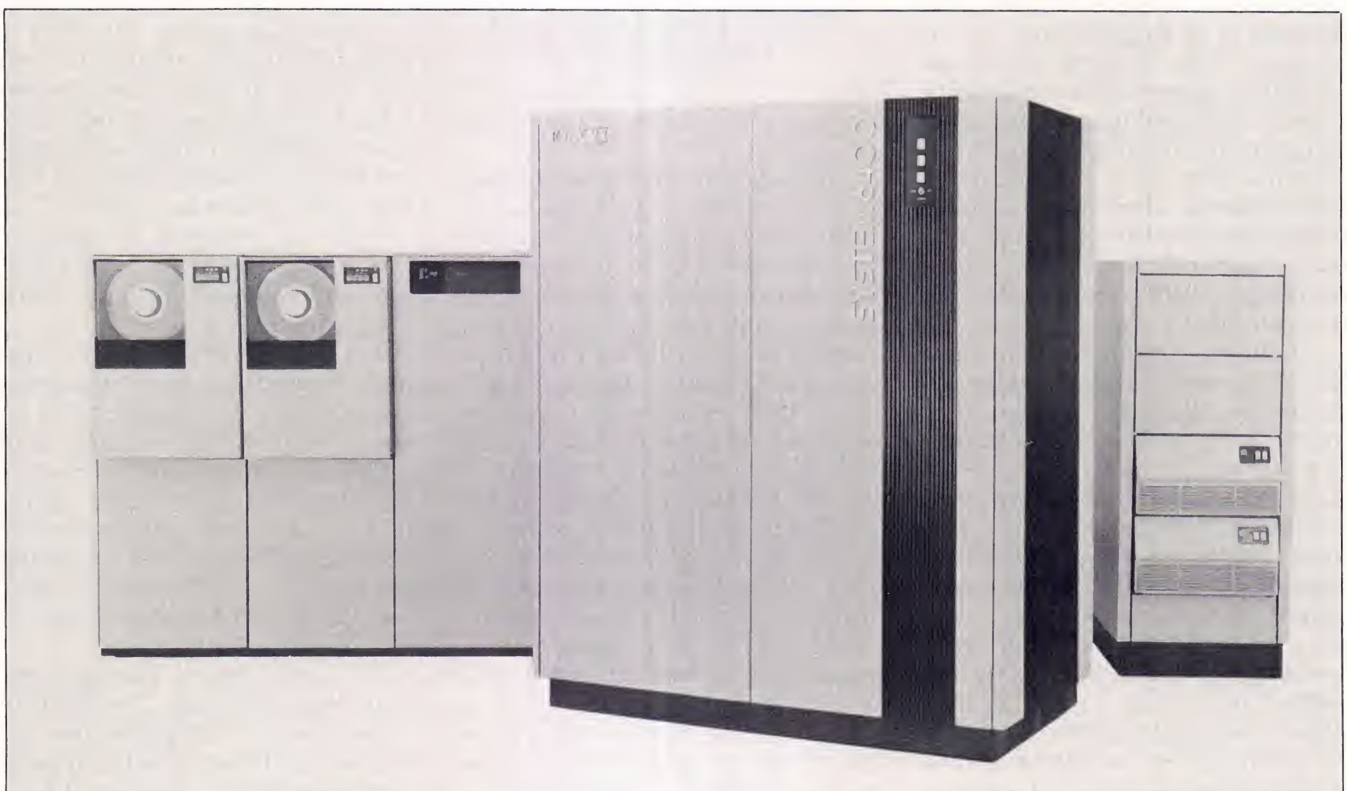
How long would it take you to find all the occurrences of *Klingon* in every *Star Trek* script ever written? From the sound of it, Gould must have had Scotty souping up the Warp Drive on its PowerNode computers since the company now claims you can find *Klingon*, or anything else, at otherworldly speeds. Gould calls its new capability *HyperSearch* and claims it can read a "properly encoded" database at speeds between 150,000 and 300,000 *English words per second*.

What exactly is *HyperSearch*? It consists of a single board and associated software for managing the text search function. How does HyperSearch work? Before you can perform your first search, HyperSearch encodes the text in a database—without the use of indices—so that when you perform a search, Hypersearch can speed through the data. Among other steps, the encoding process employs a data compression algorithm that reduces the amount of data kept online to between ⅓ and ½ its original size. In addition to taking up less disk space, the encoded data provides an extra margin of security because, Gould claims, it is impossible to decode compressed information without the vocabulary key that is generated when you encode the data.

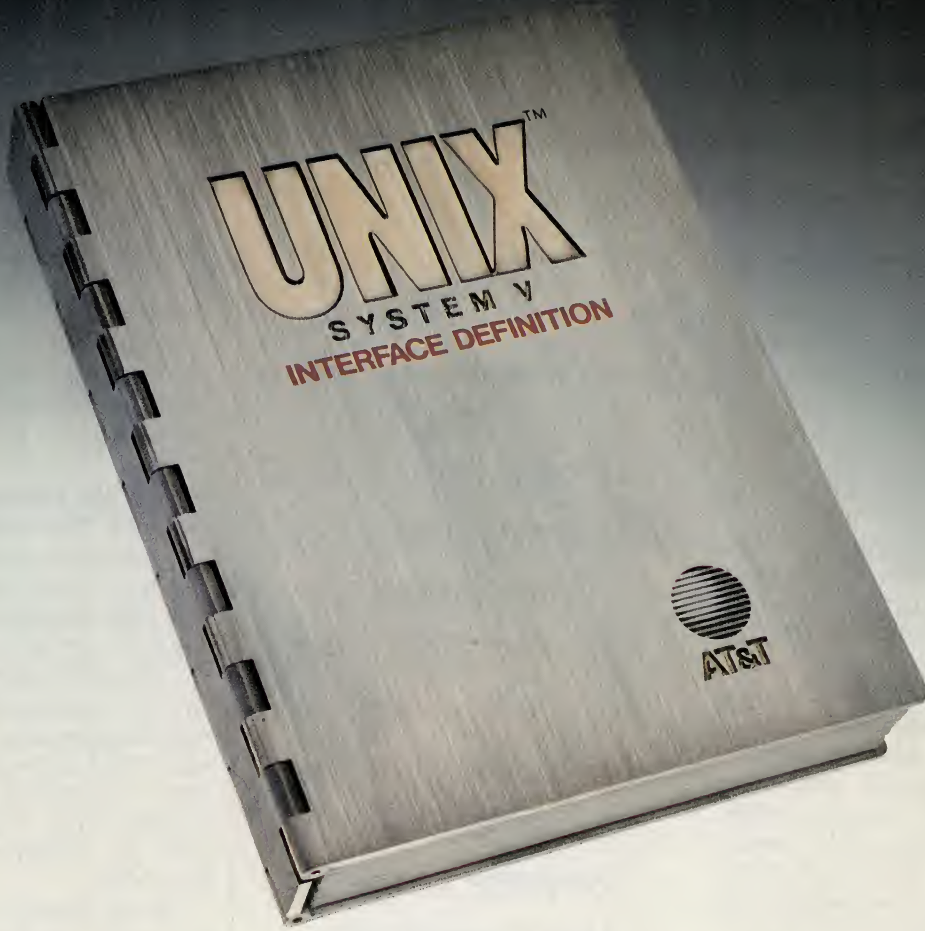
According to Bob Bergman, Gould's Director of Advanced Planning, "The fact that HyperSearch acts as a co-processor by handling the search process itself means that the host machine is free to run other UNIX applications concurrently." In truth, once a search is started, the CPU is totally out of the picture. The HyperSearch hardware uses DMA techniques to pull the data off the disk and then analyze it without the aid of the CPU.

Not only will HyperSearch let you look for words, it will also perform searches for two words occurring within a specified number of words of each other, or for two words that appear in the same sentence, on the same page, in the same chapter, or in whatever space you might name. What's more, you can perform as many as *eight* searches simultaneously (in parallel).

Even closer to Klingon Headquarters: Gould is known for the



The 6400, ELXSI's 10-CPU system that's said to offer otherworldly performance.



UNIX™ SYSTEM V. BEST-SELLING SOFTWARE IS WRITTEN BY THE BOOK.

How can you write software that's a clear best seller? By writing applications for the growing number of products based on UNIX System V. And by making sure your software conforms to the System V Interface Definition.

Your software written under the Interface Definition can run on current and future releases of AT&T's UNIX System V, as well as standard System V derivatives offered by AT&T licensees. With such a large potential market, it pays to go by the book.

Guidelines you can profit by

The Interface Definition precisely identifies features and functions that comprise standard System V. You'll have the information needed to write applications that are portable across current and future System V environments—without modification. Applications that are virtually hardware independent, from personal computers to mainframes.

Software developers will be able to sell more software to more customers for a broader range of hardware products. VARs will be able to offer more customers a wider choice of System V-based equipment—with a ready supply of software to run on it. And that means greater profitability.

A standard you can bank on

AT&T is committed to UNIX System V as a universal standard. It's gaining wide commercial acceptance.

We plan to preserve the integrity of UNIX System V and promote the compatibility between System V and its derivatives. Conforming to the System V Interface Definition ensures your applications will meet these high standards. And satisfy end-user needs.

To profit from the growing demand for software written by the book, order the System V Interface Definition—at $37.00 plus tax. Call 1-800-432-6600 and ask for Operator 383. You can charge your order to any major credit card.

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

AT&T
The right choice.

real-time operating system it uses to drive its computers as flight simulators. Mind you, these are not the simulators you run on your PC, but rather the *real* simulators that train pilots to fly 747s and fighters. Flight simulators are known as the ultimate test for a real-time system and UNIX is known for its lack of real-time capability.

Gould just rewrote its flight simulation software so that it runs under its UNIX operating system. It then tested a *real-time* flight simulator driven by UNIX. To make this a legitimate test, Gould chose to simulate a plane landing on an aircraft carrier. Think of the many variables: waves, runway motion, wind, plane speed—the whole unsettling experience. Now think of the number of dials and gauges the pilot must use (and that the computer must control). The upshot of all this is that the test was a success. So the next time someone tells you UNIX is not a real-time operating system . . .

DATA GENERAL JOINS THE FRAY

Data General has announced the Desktop Generation Model 45, which incorporates both a 68000 and a microECLIPSE microprocessor. The 68000 runs DESKTOP/UX (the UniSoft port of UNIX System V)—the microECLIPSE takes care of the I/O. The basic configuration of the Model 45 costs almost $12,000 (plus $1800 for a UNIX license) and has one floppy, a half megabyte of RAM, and a 15 MB hard disk (a little small for an eight-user UNIX system, no?). The amount of RAM and hard disk storage can be increased, and a cartridge tape is available.

This offering rounds out DG's UNIX line, which ranges from Venix on its portable Data General/One to MV/UX (hosted) and DG/UX (native) on its 32-bit workstations and superminis.

It's interesting that DG and Sperry, both long-time computer manufacturers with large followings, are now presenting their customers with a complete range of UNIX machines.

It's interesting that DG and Sperry, both long-time computer manufacturers with large followings, are now presenting their customers with a complete range of UNIX machines.

ON THE DOWNSIDE

Sydis, one of the premier phone-and-computer packagers, laid off 51 of its 128 employees in late April. Not too long ago, some of the founders of Sydis were rumored to be skipping down the halls singing, "We're in the money!" after IBM had acquired Rolm. The reason? Many of the founders came over from Rolm and had good reason (read "stock shares") to cheer for their alma mater. With AT&T and IBM in the phone-and-computer business, I wonder if there will be room for anyone else.

Black bunting was also in evidence around Fortune Systems because the troubled manufacturer has once again reorganized, this time laying off about 100 of its employees. To quote the president, Jim Campbell, "The goal of a profitable 1985 requires strong downsizing early in the year to lower the breakeven point . . ." Sigh.

SCHIZOID COMPUTERS OF THE FUTURE

A rumor perpetrated by International Resource Development has IBM developing the "lightening fast" PC 3MP for introduction in 1987. The *3MP* stands for three multiple personalities. The first operating system will be a proprietary IBM OS implemented in silicon. You will be able to plug in another chip if you want UNIX and yet another will give you MS-DOS.

IRD forecasts a similar future for AT&T, saying, "We may assume that AT&T will have rapidly realized its strategic mistake in not giving the 7300 UNIX PC an MS-DOS personality; probably this will be rectified within a year, by the addition of an optional add-on board for the 7300." IRD expects all future AT&T computers to be equipped with an MS-DOS personality "until MS-DOS fades away in the 1988 timeframe."

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

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.

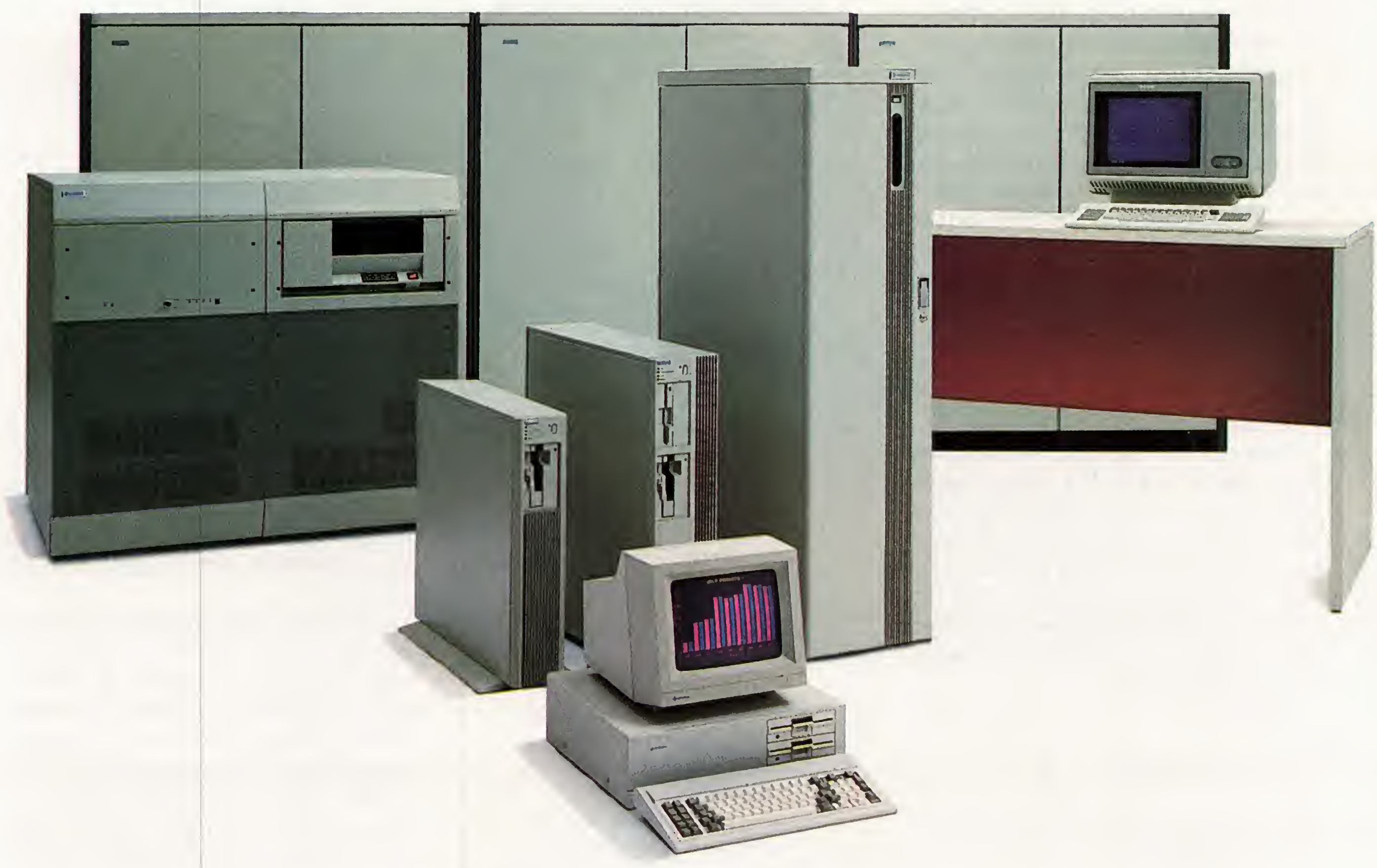
Of course, this linking of all your computer systems can add measurably to your productivity.

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

Introducing an idea that makes obsolescence obsolete.



The UNIX* operating system from PC to mainframe.

Circle No. 4 on Inquiry Card

DEVIL'S ADVOCATE

Style . . . either you've got it or you don't

by Stan Kelly-Bootle

"It was a tempestuous, nocturnal environment . . ." Sorry, I seem to have left my WWB (Writer's Workbench) in Novella Mode.

You may not yet be aware that as part of the drive to make UNIX System V a universally acceptable standard, AT&T has established a set of modes for the **style** command. In addition to the usual checks on grammar and work frequency, the enhanced WWB+ makes helpful suggestions from its built-in thesaurus, or rather, thesauri, since each mode has a specialized database of synonyms and paradigms.

Instead of simply prompting the author with the occasional *mot juste*, or warning that the word "environment" has already been used four times in the current sentence, the new system can offer global rewrites. They are as easy as, say, CTRL-B reformats are in WordStar.

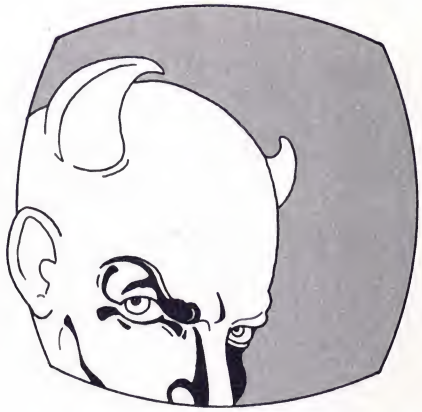
Using:

```
$ rewrite/c pome
```

for example, where /c is the Chaucerian modifier:

```
$ cat pome
```

gave me:



Prologue to the
Programmer's Tale

A Youthe ther was fulle berded
like a Parde,
Experiaunced in Wares bothe
Softe and Harde;
Fluant was he in Languages
obscourre;
Lange programmes hadde he
scribed, no-one more,
In Algolle, APL and Graspe
And divourse dialects of
UniHaspe.
Sonne's days he scorned to
laboure alle the night
Promoting Kernelles
left and right;
Of every Shelle fulle welle
he knew the Range
In deede, eek monthe his Jobbe
woulde change!
And everye time thatte he
moved Companye
His Income grew with
certaintye,
For oft he quoteth 'Those who
write Compilers
Are rarer thanne
Foure-Minute Milers!'

There are obvious dangers, of course. Expect a spate of systems manuals produced in */p* (for Proustian) mode. If your next Overdue Invoice starts:

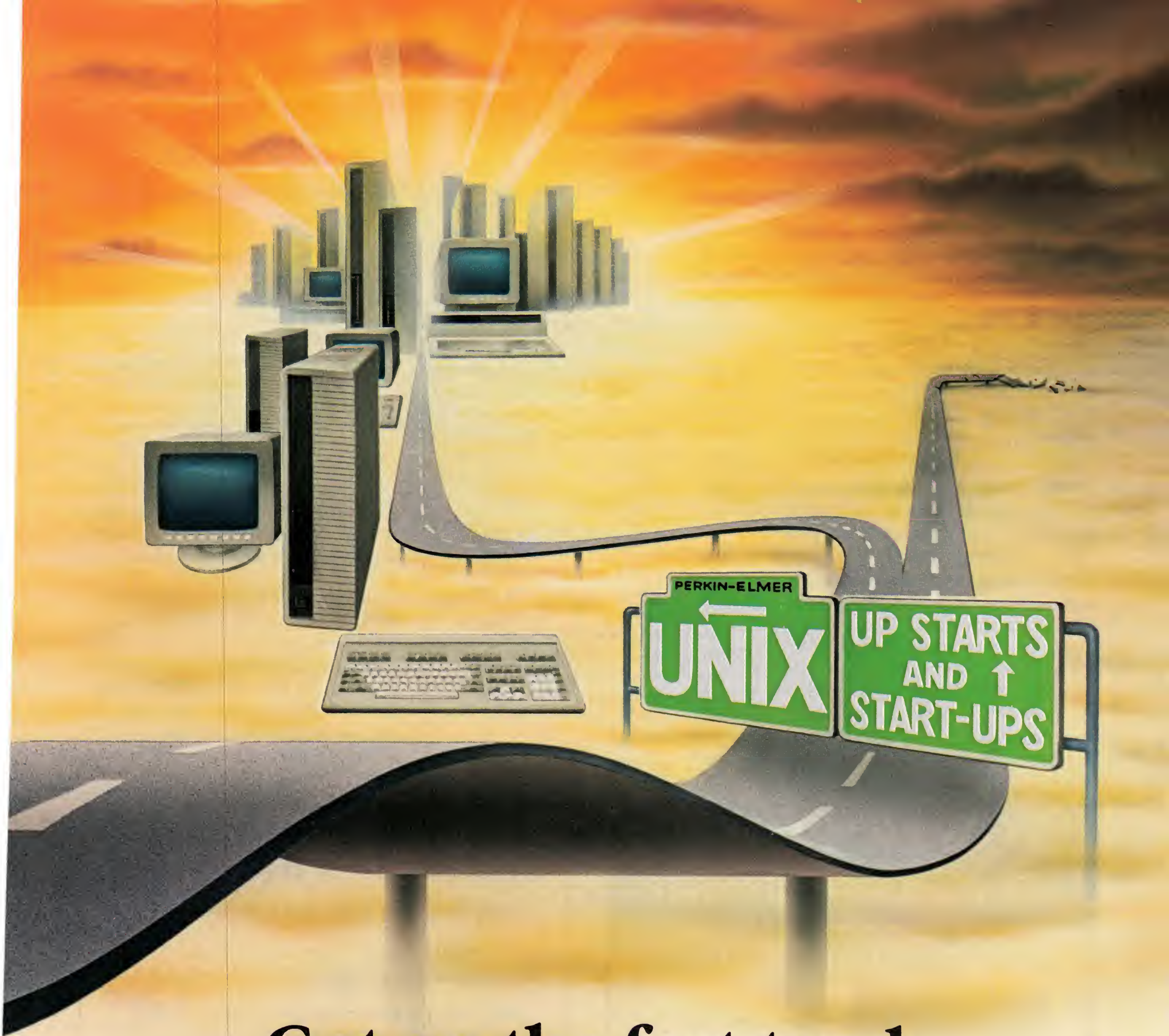
And thus we Bill like Bill yet Bill
no more,
For patience and thine credit
weareth thin.
As did the unfed Dido pining
for her love.

you can bet that some smart clerk has used the popular Shakespearean */s* switch.

Some options are clearly irresistible; witness the glut of spreadsheets set in Old Gothic Macfont.

When does an option become a maligned feature? Kernighan and Pike complain: "Creeping featurism encrusts commands with options that obscure the original intention of the programs." (*The UNIX Programming Environment*, Prentice-Hall, 1984, page 316). Whether you agree or not, you must admire the Churchillian splendor of the prose, produced without the aid of WWB+.

To my mind, the word "featurism" smacks of the subjective polemic, reminiscent of Lenin's use



Get on the fast track.

For low cost UNIX* applications turn to Perkin-Elmer's 7350A supermicro.

Perkin-Elmer gives you a head start in the growing world of UNIX system computing with the fastest supermicro in its class.** And we can keep you ahead because we're a Fortune 300 company with nearly a decade of experience with the UNIX operating system.

Move forward confidently with a fully supported, cost-effective solution; up to five users can share a single system. For the long haul, go the distance with the broadest range of compatible superminicomputers available today.

Why risk your future with start-ups or upstarts? Let Perkin-Elmer help put you on the path to success.

Choose the fast track: call 1-800-631-2154.

Or write to the Perkin-Elmer Corporation, Two Crescent Place, Oceanport, N.J. 07757. In N.J. call (201) 870-4712.

*UniPlus, UNIX, RM-COBOL and SIBOL are trademarks of UniSoft, AT&T Bell Laboratories, Ryan-McFarland and Software Ireland, respectively.

**In nine UNIX benchmark tests, the 7350A outperformed the published results of similar supermicrocomputers in its price range, while comparing favorably with many other larger and more costly systems. Prices and specifications subject to change without notice.

SPECIFICATIONS & OPTIONS

- 8MHz MC68000
- 512KB to 3MB RAM
- Floppy Disk
- 15MB or 40MB Winchester
- 12" or 15" Monochrome; 13" Color Graphics Monitors
- 4 x RS-232C ports (Ethernet optional)
- UniPlus* port of UNIX System III
- MenuMaker/C Shell/AT&T Shell
- Word Processing, Spreadsheet, Relational DBMS, Graphics Libraries
- C, FORTRAN, BASIC-PLUS, RM-COBOL,* SIBOL*
- System prices start at $5,115 (quantity 100, U.S. only)

Circle No. 35 on Inquiry Card

PERKIN-ELMER

of "Kautskyism" or "revisionism". My features are nice to have, but it's those nasty *featurists* who add options I personally do not want. Yet the whole point of options is that they *are* optional. Standards for UNIX (or anything else) should never inhibit enhancements of the superset variety.

The need for a UNIX standard is indisputable. I want command X and option Y to work consistently wherever I go, but I don't mind if X' also equals X, nor if a new option Z appears. We just do not know yet what OS features might be deemed desirable by tomorrow's users on tomorrow's software. What is certain is that as memory prices decline and speeds increase, the old preoccupations with program size and efficiency will become less relevant. A $5 encrustment is no encrustment at all, and no barnacle neither, me brave hearties.

Reading about the new integrated voice/data UNIX systems prompts me to speculate about whether the new AT&T standards will include sections on the preferred formats for spoken messages. The Bell tradition suggests that we might soon be hearing:

> **"The disk block number you have reached, 770123, has been changed. The new number is 654310. Please make a note of the new number for all future references. The disk block number you have reached, 770123, has been changed. The new number is 654310. Please make a note of the new number for all future references . . ."**

or:

> **"The filename you have requested, FOOBAR, is unobtainable at this time. Please hang up and try later . . ."**

or even:

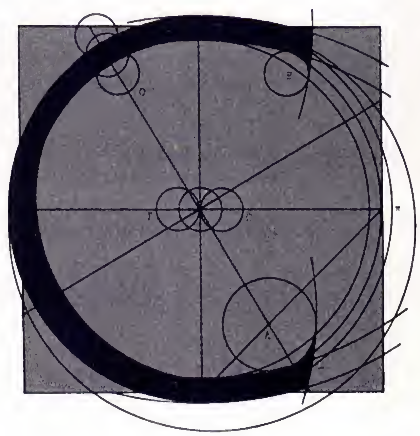
> **"The OS you are invoking, UNIX V, has been changed. The new OS is S1. Please make a note of the new name for future reference. The OS you are invoking . . ."**

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

C ADVISOR

Setting the user ID

by Bill Tuthill



The only patented UNIX feature is the **setuid** bit, which permits programs to change user ID, and thus adjust to whatever permissions they might encounter. This is a feature unique to UNIX that actually works as an *attribute* of a file. Although it is not possible to patent an algorithm, it is possible to patent a process. Because of this, Dennis Ritchie was able to patent the **setuid** bit since it is an algorithm *embedded* in a process. Figure 1 shows how the **setuid** bit fits into a short word that controls access permissions in an **inode**.

The UNIX kernel keeps track of two user IDs: the *real* user ID, which identifies the person who is logged in; and the *effective* user ID, which determines permissions at the moment. At login time, the real and effective user IDs are the same. The **setuid( )** system call changes both, but the **setuid** bit changes only the effective user ID. You can always determine someone's real user ID number by searching through the */etc/passwd* file for a login name, and then looking at the number listed after the second colon. This number remains constant no matter what modifications the effective user ID number might undergo.

C programs can set the user ID in one of two ways. First, file permissions can be changed with **chmod** so that the system sets the effective user ID upon program execution (this involves the **setuid** bit). An example of this follows:

```
% chmod 4751 program
```

As a result of entering a "4" in the thousands place, the **setuid** bit is turned on. By entering only a three-digit value, one leaves the bit off.

The real and effective user IDs can also be set by invoking the **setuid( )** system call from within a program. The first case is more common than the second. Generally, the **setuid( )** system call is used to set the user ID back to what it was originally.

In the first case, the effective user ID of the invoked process gets set to the ID of the owner of the executable file. In the second case, the real and effective user IDs may only be set to those belonging to the user who invoked the program. There are two exceptions to this, however. First, if the **setuid** bit is on, the real and effective user IDs may also be set to those of the executable file's owner. Second, if the effective user ID of a process is root (as in a **setuid(ROOT)** program), the real and effective user IDs can be set to anything.

On Version 7 and 4.2BSD, once the effective user ID is changed, the old one is forgotten and its permissions are lost. On System V, though, it is possible to switch user IDs an unlimited number of times. This makes the **setuid** mechanism much more useful because programs can alternate as needed between two simultaneous sets of permissions.

As a simple example, Figure 2 shows a program named *setuid.c* that prints the real and effective user IDs before and after calling **setuid( )**. Compile this program, and then change its mode to 4755. Next, log in as another user to see how it works. Figure 3 demonstrates how this interaction might proceed. When invoked by another user, the program sets the effective user ID to the identity of the program's owner, and then sets both the real and effective user IDs back to what they were originally.

				setuid bit	setgid bit	sticky bit	read owner	write owner	execute owner	read group	write group	execute group	read other	write other	execute other

Figure 1 — *How the* ***setuid*** *bit fits into the 16 permission bits in a short word that controls access permission in an inode. (The high four bits are not used.)*

The **setuid( )** of **getuid( )** doesn't need to be checked, since it can never fail.

With system calls on Version 7 and System V, the real and effective user IDs must be set together. On 4.2BSD, the real and effective user IDs may be set separately, using the **setruid( )** and **seteuid( )** system calls, respectively.

Any program that sets the effective user ID to root is a potential security hole. If a system programmer writes a **mail** program with a shell escape and makes it 4751 mode, for example, the subshell will have root permission—unless the program explicitly sets the user ID back to what it was originally. Programs that set the effective user ID to root (by being in the 4000 mode and being owned by root) should avoid shell escapes if possible.

Figure 4 contains a list of programs that are commonly **setuid** on most UNIX systems. Every now and then—particularly on systems with hostile users—you should run:

```
# find / -perm -4000 -print
```

in order to find all the **setuid(ROOT)** programs on your system. Listed in Figure 4 are some commands owned by root that are potential security holes; system administrators should monitor them closely. Programs that create, move, or remove directories, such as **mkdir**, **mv**, or **rmdir**, have to run as root in order to modify the directory node. Commands that change user ID, such as **login** and **su**, have to run as root in order to reset the real and effective user IDs. The **lpr** and **mail** commands need root access to protected spool directories, and the **passwd** command modifies the system password file, which had best be well protected. Programs such as **ps** and **df** that examine memory or disk drives should be **setuid(BIN)** or **setuid(CHECK)** so that memory and disk can be made readable only by **bin** or **check**, thus restricting them from the average user. If memory and disk were readable, users could read sensitive data simply by examining these devices. On some systems, **lpr** also runs as **bin** or **check**.

Programmers should avoid **setuid(ROOT)** programs if possible. In most cases, it's best to set the user ID to something else. The **safedit** program shown in Figure 5, for example, sets the user ID to **dbm** in order to create a secure system for data editing. In order to use it to its fullest advantage, you

```
main()          /* change this program to 4755 mode */
{
        printf("uid=%d euid=%d\n", getuid(), geteuid());
        setuid(getuid());
        printf("uid=%d euid=%d\n", getuid(), geteuid());
}
```

Figure 2 — *A sample program that sets and prints the real and effective user ID with a single call to* ***setuid( )****.*

```
% cc /tmp/setuid.c -o /tmp/setuid
% chmod 4755 /tmp/setuid
% who am i
review   tty01   Apr 29 22:14
% grep review /etc/passwd
review:gv9S14xHRLMro:292:10:Unix Review:/usr/review:/bin/csh
% login tut
Password:
% who am i
tut      tty01   Apr 29 22:14
% grep tut /etc/passwd
tut:W7Kh17fKdbbEY:508:10:Bill Tuthill:/usr/tut:/bin/csh
% /tmp/setuid
uid=508 euid=292
uid=508 euid=508
```

Figure 3 — *A sample session making use of the* ***setuid( )*** *call.*

```
38 -rwsr-x--x 2 root    system   19030 Aug 28  1984 login
23 -rws--x--x 1 root    system   11484 Oct 19 14:26 lpr
38 -rwsr-x--x 2 root    system   18984 Aug 28  1984 mail
10 -rwsr-x--x 1 root    system    4766 Aug 28  1984 mkdir
19 -rwsr-x--x 1 root    system    9378 Aug 28  1984 mv
31 -rwsr-x--x 1 root    system   15738 Aug 28  1984 passwd
11 -rwsr-x--x 1 root    system    5374 Aug 28  1984 rmdir
29 -rwsr-x--x 1 root    system   14492 Aug 28  1984 su
46 -rwsr-x--x 1 bin     system   23386 Aug 28  1984 ps
15 -rwsr-x--x 1 bin     system    7380 Aug 28  1984 df
```

Figure 4 — *Some programs that* ***setuid*** *to root. These are security holes that bear watching.*

```
#include <stdio.h>
#include <pwd.h>
/*
 * This program should be setuid(dbm) - owned by dbm, mode 4751.
 * The directory ~dbm/db and the file ~dbm/passwd are required.
 */

main(argc, argv)        /* safedit: check password for data editing */
int argc;
char **argv;
{
        char *p, *passwd, *getpass(), *crypt();
        struct passwd *pw, *getpwnam(), *getpwuid();
        FILE *fp, *fopen();
        char pwfil[BUFSIZ], dbdir[BUFSIZ], dbpasswd[BUFSIZ];
        char *editor, *getenv();

        if (argc == 1) {
                fprintf(stderr, "Usage: %s [-] filename\n", argv[0]);
                fprintf(stderr, " - prints encrypted password\n");
                exit(1);
        }
        if ((pw = getpwnam("dbm")) == NULL || geteuid() != pw->pw_uid) {
        fprintf(stderr, "Program should be setuid dbm\n");
        exit(2);
        }
        p = getpass("password:");
        passwd = crypt(p, "db");        /* encrypt password for validation */
        if (*argv[1] == '-') {
                printf("%s\n", passwd);
                exit(0);
        }
        sprintf(pwfil, "%s/passwd", pw->pw_dir);
        if ((fp = fopen(pwfil, "r")) == NULL) {
                perror(pwfil);
                exit(3);
        }
        while (fscanf(fp, "%s", dbpasswd) != EOF)       /* get last line */
                ;
        if (strcmp(passwd, dbpasswd) != 0) {
                fprintf(stderr, "Sorry, wrong password\n");
                exit(4);
        }
        sprintf(dbdir, "%s/db", pw->pw_dir);    /* change to db directory */
        if (chdir(dbdir) == -1) {
                perror(dbdir);
                exit(5);
        }
        umask(077);                             /* readable only by owner */
        if (editor = getenv("EDITOR"))
                execlp(editor, editor, argv[1], (char *)NULL);
        else
                execlp("vi", "vi", argv[1], (char *)NULL);
        exit(0);
}
```

Figure 5 — *An example of how to secure a directory of sensitive files.*

need to set up a **dbm** account on your system. After doing so, log in to this account, compile the program, and change its mode, as follows:

```
% cc safedit.c -o safedit
% chmod 4751 safedit
```

Now you can invoke the program from any other account on your system. The data files you create, in the directory ~*dbm/db* will not be accessible except through the *safedit* program. If other users know the password to the database directory, they could read and modify files you, or anybody else, create. A crypted version of the **dbm** password resides in ~*dbm/passwd* whenever required.

Let's look more closely at the program. With no arguments, **safedit** prints out a synopsis of its usage. Note that a minus sign as the first argument will produce an encrypted password, which the database administrator can put in ~*dbm/passwd*. The program then calls the library routine **getpwnam( )** to get a password for the *dbm* account, and checks to make sure the program is **setuid(DBM)**.

Next, the program reads a password typed by the user, using the **getpass( )** library routine, and crypts the password using the **crypt( )** library routine. The second argument to **crypt( )** specifies a "salt", which perturbs the DES encryption algorithm so as to make the password harder to decode. Then the program opens and scans the ~*dbm/passwd* file, then compares the encrypted password there with the one the user just typed in. If the passwords don't match, the program exits immediately. But if the passwords match, the program changes to the database directory, sets the permission mask to 077 (owner-exclusive access) with the **umask( )** system call, and then calls **vi** (or the editor specified in the **EDITOR** environment variable) to edit the database file.

A real database system would not use a generic editor such as **vi** to edit material in the database. But in other respects this program, despite its brevity, takes the right approach to data protection. First of all, it isn't **setuid(ROOT)**, so, apart from the database system itself, it is unlikely to become a security hole. Second, by being **setuid(DBM)**, it restricts data access to those who know the password. A more elaborate system of password checking, using initials and matching passwords for each user, could easily be implemented. Accounting files are good candidates for similar **setuid** treatment.

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). ■

THE UNIX GLOSSARY

UNIX applications

by Steve Rosenthal

Note: Only those aspects of the terms concerning applications are discussed in this listing. Many terms have additional meanings.

accounting package—a set of programs and procedures for entering financial information into a computer system, drawing the appropriate totals, printing reports, and often issuing statements and checks. Accounting packages are often sold as a series of modules, each of which may be priced as a separate add-on program.

ad hoc—said of queries to a database management system (DBMS) that are made directly by users rather than as part of a previously written program. Most of the more sophisticated DBMS packages for UNIX support an ad hoc query facility, frequently using a format resembling IBM's SQL (Structured Query Language). This feature allows users who are not programmers to get quick answers from the database, at least whenever they ask simple questions.

analytic graphics—when speaking of charts and graphs, "analytic graphics" refers to those designed primarily to help users make sense of data (in contrast with "presentation graphics", which are intended to help others see the user's view).

application—in the narrowest sense, "application" refers to a

program or set of programs intended to perform some useful work over and above the support and maintenance of the computer system itself. Examples would be accounting packages, spreadsheets, word processors, databases, and so on. In a looser sense, an "application" can be any program other than the operating system itself. In this context, compilers, debuggers, and so on would also be included. Compared to other operating systems, UNIX is particularly rich in the second class of applications. It is becoming richer in the first class, but this has not historically been the strength of the system.

applications development system—a software package intended to help users turn out applications programs, most often by using a description of needed inputs, outputs, and their relations rather than by specifying incremental steps—as is necessary with procedural languages. Some applications development systems for UNIX function as complete user shells, providing a user interface that substitutes for the normal UNIX environment. Compared to program generators, applications development systems are generally more sophisticated, but they usually also produce code that requires a more extensive runtime support package.

binary license—the common UNIX term for a license that does not include program source code, but does cover the machine-specific code actually read by the computer. In less commercial days, UNIX source code was often included with a program, and it's still often available at an extra charge. However, as UNIX becomes more of a mass-market item, more vendors are likely to adopt the position that source code will not be sold.

business graphics—a general term for charts, graphs, and other pictorial representations that are often used in business presentations. Because UNIX has no standard graphics interface, graphics programs tend to be hardware specific.

calendaring program—allows a user at one workstation in a network to schedule a meeting for several users, having the program automatically check each affected individual's calendar for conflicts.

command driven—said of software in which the user types in a series of commands in answer to a prompt instead of selecting options from a list on the screen. UNIX itself is command driven, as are many applications aimed at programmers. Most new applications aimed at more casual users employ a menu-driven (selection) format.

computer conferencing—using computers and data communications networks to connect users wishing to conduct a dialogue by way of messages that can be either stored or read in real-time. "Computer conferencing" is often known by its fancier name, "teleconferencing". Like UNIX, the strongest user base for teleconferencing exists in the research and academic fields (as well as in related industries). Moreover, UNIX and teleconferencing software both fit most comfortably on minis and supermicrocomputers. There are commercial UNIX-based teleconferencing systems available today.

database—often used as the short form of "database management system" (DBMS), a program that manages structured files of information. Strictly speaking, a "database" is actually the set of files managed by a DBMS.

database management system—a program or group of programs that provides access to information stored in an integrated group of data files. These systems are most often referred to by the initials "DBMS". To purists, a true DBMS stores data in a form independent of the specific programs that read or write the information. This allows different programs to access the same database. UNIX supports a range of DBMS packages, plus a number of simple file managers and list managers.

desktop software—refers to programs that substitute for common office desktop paper recordkeeping. Among the most common are software packages corresponding to calendars, memo pads, phone books, and alarm clocks. Desktop software has been more of a hit in the single-user personal computer market, but it is beginning to appear for single and multiuser forms of UNIX.

development language—a high-level language that is tailored especially for creating applications programs. Most sophisticated database management systems (DBMS) provide a development language. Some people also consider the set of UNIX shell commands a development language, with shell scripts regarded as the resulting applications.

file manager—a program that accepts, manipulates, and reports data according to a standard format. In ways, it is a simplified version of a database management system (DBMS) that does not provide for an independent data structure meant to be accessed by multiple programs.

fourth-generation language—a common term for non-procedural languages in which the user specifies inputs, outputs, and relations instead of the steps to be taken in manipulating the data. Database applications development and query languages are often billed as "fourth-generation languages".

full-text—said of information-retrieval systems that store and retrieve the entire contents of text documents, often by way of an index that lets the user find any occurrence of specified words or word groups. Indexing is usually done in batch mode, sometimes as a background process, but more often in UNIX systems as a process invoked through **cron** during the night hours.

hardware independence—for UNIX programs, this term refers to the ability to run a program on any hardware supporting the correct version of UNIX. Hardware independence is a goal that is met by degrees rather than as an absolute.

horizontal—as applied to software or hardware markets, this term pertains to functions that are generally part of a wide range of business or leisure activities. Products that apply to writing, database management, general information storage and retrieval, and arithmetic calculations are good examples. Software or complete systems designed for horizontal use must be more general, and may not solve any particular problem as efficiently as a more targeted solution would. But due to their bigger potential market, their cost may be low and the same effort expended in learning a single package may be applied to several different applications.

information retrieval—a general term for the recovery of any information from computer storage, and—more particularly—for the discovery and display of a document or section of text taken from a database of text.

integrated program, integrated software—a program or series of programs that accomplishes several standard functions and shares data and general command structures among the various modes. The most popular "integrated programs" are the ones that combine spreadsheets, word processing, databases, and graphing in some fashion.

menu-driven—refers to software that depends heavily on the presentation of lists of choices for users to pick from. Inexperienced users generally like this type of software, as it makes clear what the choices are at each step. Experienced users often prefer skipping the menus to make direct use of the commands themselves.

module—in reference to applications software, "module" refers to a package that performs a particular section of a task and is sold or documented separately from the software that performs the remainder of the task. For example, a complete accounting package may include modules for general ledger, accounts receivable, accounts payable, and so on.

project manager—a type of software that aids in the planning and scheduling of large, complex jobs. These packages accept descriptions of the resources and steps needed to prepare for various tasks, and then produce charts, diagrams, and schedules.

relational—as applied to databases in the strictest terms, "relational" refers to those that are conceptually organized in row-and-column format, with the data defined as the relation of one part of a record (a row) to a category or field (a column). In recent use, however, relational has come to refer to databases that can join or display two or more files based on a shared field or category.

runtime package—refers to the additional routines that are needed to execute a program, but which are not explicitly described in the program's source code because they are automatically provided during compilation or linking. Most programs written in a higher-level language, including most C programs under UNIX, assume a runtime package, which is why even a one-line program can be many kilobytes long.

screen editor—generally used to refer to an editing program that lets you move around in a textfile by location on the screen display, rather than in units of lines or units within lines. The UNIX **vi** editor is a screen editor, as are all recent commercial editors. Screen editors are also called "full-screen editors".

spreadsheet—a program that allows manipulation of numbers within a chart construct; each box in the chart is set either by direct input or as a calculated result of other entries. Because the complete chart can quickly and easily be recalculated after any value in it is changed, spreadsheets are widely used for budgeting and forecasting.

system software—in the narrowest sense, "system software" refers to software that is part of the operating system itself. In a wider sense, it means software that focuses on supporting the operation of the system's hardware and other software rather than on getting work done for applications that have an effect outside the computer system. Much of the system work in UNIX is done by a large set of utility programs, which—in the looser sense of the term—is a vital part of the "system software".

transaction processing—applications centered on small but frequent updates to large databases, such as those used in banking, billing, and reservations. Transaction processing is often done on distributed (multiuser) systems, and thus requires some means for keeping two or more users from updating the same record simultaneously. UNIX originally had no such locking mechanism, but most commercial implementations have added or are adding this feature.

VAR—short for value-added reseller, a person or firm that buys equipment, adds services or companion products (such as custom software or a number of integrated software products) and then resells the package. VARs have been a major factor in the commercial sales of mid-range UNIX systems.

vertical market—a market defined by industry or occupation rather than by the type of task a system might be required to handle. For example, kennel operators make up a vertical market that might be serviced by vertical software, while editing and word processing systems are solutions that are offered to horizontal markets. Vertical software theoretically offers complete solutions to narrow classes of people, typically by tailoring and integrating a wide variety of horizontal services. Many UNIX system houses specialize in these narrow vertical markets and customized programs.

word processing—the use of a computer to edit, store, and print text. Complete word processing programs generally combine the text editing functions performed by editors such as **vi** with the output formatting capabilities of programs such as **nroff**.

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

Steve Rosenthal is a lexicographer and writer whose work appears regularly in six personal computer magazines. ■

RECENT RELEASES

MICRO MAY BE NO SMALL FACTOR

Victory Computer Systems' MicroFactor II is a 68000-based UNIX System V supermicro. It has 2 MB of RAM, 190 MB of disk storage, supports eight users and Ethernet with TCP/IP. It comes with either a Z80 CP/M or Intel 8088 MS-DOS co-processor, and also features a floating point processor. The system is based on 16/32-bit VME bus technology, and has a six-slot card cage. Its base price is $7500.

Victory Computer Systems, 1610A Berryessa Rd., San Jose, CA 95133, 408/259-7370.

Circle No. 47 on Inquiry Card

The Victory MicroFactor II is a 68000-based System V supermicro.

SUN TAKES APPLE LASERWRITER TO CAD/CAM

An OEM agreement between Apple Computer and Sun Microsystems will bring the Apple LaserWriter printer into new areas. The LaserWriter, which uses Adobe System's PostScript software, will be especially aimed by Sun at publishing concerns desiring in-house preview of typeset documents, CAE/CAD/CAM applications, as well as architectural and construction applications.

The LaserWriter comes from Apple with 13 typefaces: Sun is licensing font libraries that contain hundreds more. The printer produces copy at a maximum speed of eight pages per minute with a resolution of 300 dots per inch. PostScript allows text and graphics to be scaled to any size and rotated. It operates through a serial port connection, and will be available to Sun customers this month, at a cost of $7750.

Sun Microsystems, Inc., 2550 Garcia Ave., Mountain View, CA 94043, 415/960-1300.

Circle No. 48 on Inquiry Card

OREGON COMPILES VAX/ULTRIX

Pascal-2 from Oregon Software has been issued in a new version, designed for DEC's Ultrix-32, running on VAX and MicroVAX systems. Pascal-2 development tools include a high-level interactive debugger, an execution profiler, and coding utilities. The program conforms to the international Pascal standard (ISO 7185), and supports all features of standard Pascal, including packed data structures and set types of as many as 256 elements. It allows calls to separately compiled routines written in Pascal, C, Fortran, or Macro, thus providing developers access to existing software libraries. Compile-time error checking ensures data type conformance to the Pascal standard, locates many uninitialized variables and detects nearly 150 Pascal syntax errors.

Pascal-2 for VAX/Ultrix is now available, with license fees starting at $4500.

(The program is also available for VMS/VAXen, the PDP-11 with RSX, micro RSX, RSTS/E, UNIX, RT-11, and TSX-Plus operating systems, the Pro 350 with the RT-11 and P/OS operating system, and with MC68000-based UNIX and VERSAdos systems.)

Oregon Software, 6915 SW Macadam Ave., Portland, OR 97219, 503/245-2202.

Circle No. 49 on Inquiry Card

FAST OS FLEXES FOR 68XXX

A demand-page virtual memory version of Technical Systems Consultants' UniFlex operating system, has been written specifically for the GMX 68020 Development System from Gimix, Inc.

The release culminates the company's efforts to design software exclusively for Motorola 68XX and 68XXX microprocessors. UniFlex offers most of the features found in UNIX systems with better performance due to an assembly language implementation and its specific 68XXX design, the company says.

The UniFlex operating system can be adapted to systems ranging from ROM-based applications through single-user, multitasking workstations to multiuser, multitasking, timesharing computers. A typical kernel, without device drivers, resides in about 28K of memory. With I/O drivers, variable storage, and all static tables, the machine requires 50 to 60K. About 1.7 MB of disk storage is required for the operating system, utilities, assembler, loader, C compiler, C library, and help files. An additional 1 to 3 MB is needed for swap space.

Technical Systems Associates, 111 Providence Rd., Chapel Hill, NC 27514, 919/493-1451.

Circle No. 50 on Inquiry Card

THE MDAS TOUCH

TransEra Corp. has introduced the 7000-MDAS modular data acquisition system, designed to operate in tandem with the HP Integral. The MDAS is a 68000-based box running at 10 MHz with 2 MB of memory. It functions independently of the Integral host, captures large arrays of analog and digital data, and coordinates acquisition and control routines. Its real-time operating system supports multitasking and provides high-level communications with the attached host.

The 7000-MDAS can accommodate between one and 64 channels. Modular plug-in cards permit combinations of analog and digital I/O channels. There are more than 20 I/O modules available, for such applications as high performance differential input, frequency and pulse measurement, bridge excitation and conditioning, thermocouple input, mechanical and solid state relays, and 4 to 20-milliamp I/O.

Optional synchronized hold circuits on each input module allow simultaneous acquisition of scanned inputs. Pre-signal conditioning provides filtering, amplification, offset, and optical isolation. There are more than 60 software routines provided, to perform analog and digital I/O, signal processing, math, control, communications, hardware tests, calibration, and setups.

The TransEra 7000-MDAS handles up to 64 channels.

The 7000-MDAS weighs 12 pounds, and is available with a battery backup option. Prices start at $2000.

TransEra, 3707 North Canyon Rd., Suite No. 4, Provo, UT 84601, 801/224-6550.

Circle No. 52 on Inquiry Card

HP MID-RANGE DRIVE HAS REMOVABLE MEDIA

Hewlett-Packard's 7907A disk drive features 20.5 MB each of formatted, fixed capacity and formatted, removable capacity. It is designed for use in four primary application areas: automatic testing, data login, computer-aided engineering (CAE), and security and intelligence. This is the company's first removable drive offering in the mid-range.

The drive uses an 8-inch removable media cartridge. It can be installed by the customer in a 19-inch EIA rackmount. Its total-device average transaction time is 45 milliseconds, and an offline store and restore can be completed in two to three minutes, according to HP.

The price of the 7907A is $12,500. Extra 20.5 MB 8-inch removable media cartridges are $215 each.

Hewlett-Packard Co., 1820 Embarcadero Rd., Palo Alto, CA 94303, 208/323-6000.

Circle No. 51 on Inquiry Card

BUILD EXPERT SYSTEMS IN THE PRIVACY OF YOUR HOME

RuleMaster is the name of a software tool kit from Radian Corp. The program is designed for creation of expert systems for complex advisory, diagnostic, prediction or control applications. It is written for the IBM PC/XT and AT running Xenix or DOS 3.0, and does not require Lisp or Prolog language skills or machines.

Sample applications include fault diagnosis, online operations advice, interactive maintenance manuals, weather prediction, and chemical analysis advice.

There are two basic tools in the kit: RuleMaker, a facility for inducing rules from examples; and Radial, a high-level language for expressing rules. Radial is said by its developer to be similar to structured algorithmic programming languages such as Ada and Pascal. Users can write code directly into Radial if they prefer. It has 14 key words, and is based on backward and forward chaining.

RuleMaster can access external routines in Fortran and Pascal. It comes with a help facility that works when the user enters "Why" (wouldn't we all like to know?) at any point of a session.

The program's price is $5000 for the PC/XT, $15,000 for the AT. Purchase price includes a four-day training session.

Radian, 8501 Mo-Pac Blvd., Austin, TX 78766, 512/454-4797.

Circle No. 55 on Inquiry Card

PRECOMPILATION PROGRAM THINKS IT'S PRETTY SMART

Artificial intelligence research has gone into the development of Smart/C, a software tool from AGS Computers for use on C programs before compilation. Smart/C allows developers to create, edit, test, and debug C programs, with one common environment for both development and maintenance of programs. The company also says it can be used as a training tool for C novices because of its visual, screen-oriented user interface.

There are two sections to the program: the environment, which

Circle No. 53 on Inquiry Card

Circle No. 54 on Inquiry Card

provides syntax directed at editing and interpreting C source; and the migrator, which allows C programs not created with Smart/C to exploit the environment nevertheless. Users can create programs within the environment, invoke and suspend interpretation, re-edit, then resume interpretation at the point of suspension without having to load another tool or reload files.

Syntactically incorrect lines of codes cannot be created, according to AGS. The capability to interpret C code created in the environment before it's fully specified allows elimination of most logic errors. A module need not be fully defined before interpretation can be invoked.

Smart/C can run on many machines and operating systems—the IBM PC, AT&T 3B series, and VAX 11/780 are part of the range. It is the first in a planned series of programming research and development tools. Prices range from $500 for the IBM PC under DOS to $10,000 for the DEC VAX 11/780 running Ultrix.

AGS Computers, Inc., 1139 Spruce Dr., Mountainside, NJ 07092, 201/654-4321.

Circle No. 58 on Inquiry Card

DATA GENERAL ENTERS DESKTOP UNIX MARKET

Data General has followed AT&T with a MC68000-based desktop UNIX system, introducing the Desktop Generation Model 45. The system runs Desktop/UX, based on System V.

The Model 45 also has a Data General microECLIPSE processor in addition to the 68000. The dual architecture allows offloading of I/O from the 68000, which is dedicated to running UNIX applications. The DG microprocessor also supports peripherals.

A basic Model 45 configuration includes 512K of memory, a 15 MB hard disk, one floppy drive, and supports eight users at a cost of $11,845. The system can be configured with as much as 4 MB of memory and 142 MB of disk storage.

Desktop/UX is an implementation of UniSoft System's UniPlus+ operating system. It has both the Bourne and C shells, has a C compiler, and is upwardly compatible with DG/UX, Data General's 32-bit superminicomputer operating system.

Data General, 4400 Computer Drive, Westboro, MA 01580, 617/366-8911.

Circle No. 59 on Inquiry Card

CONDUCTING BUSINESS WITH STYLE

Foothill Research has introduced Style, an advanced high-level programming language and data management system, for use with systems running AT&T System V and UNIX Version 7. The stated aim is for the language to compete with COBOL for business and financial application development, and to co-exist with current COBOL applications.

Style has 33 commands and 14 keywords. Its applications are portable throughout its support range, which takes in DEC VAX/VMS; Stratus Computer's FT-200, XA-400 and XA-600; and Data General's MV Series running AOS/VS.

Single copy licenses range be-

Circle No. 56 on Inquiry Card

Circle No. 57 on Inquiry Card

tween $17,500 and $45,000.

Foothill Research, Inc., 1301 Shoreway Rd., Suite 300, Belmont, CA 94002, 415/593-6696.

Circle No. 60 on Inquiry Card

u4th: WHO WIN, PLACE AND SHOW?

Ubiquitous Systems has introduced u4th, a Forth software development system for UNIX and Xenix systems. The software is said to be fully portable, standard Forth. Features include access to UNIX systems calls, the ability to incorporate new primitives written in C, the ability to compile high-level Forth words into the load image, the use of regular UNIX files for loading, the ability to pass unrecognized words on through to UNIX, a direct-threaded interpreter, and an object-oriented Forth extension word set. The software is "largely compliant" with the Forth-83 standard, Ubiquitous says.

Systems presently supported by u4th include the IBM PC/XT (Santa Cruz Xenix 3.0), Plexus P/35, and the Sun Microsystems workstation. The company has plans for future versions for the PC/AT, NCR Tower, and DEC VAX.

Ubiquitous Systems, 13333 Bel-Red Rd. NE, Bellevue, WA 98005, 206/641-8030.

Circle No. 64 on Inquiry Card

DON'T GET NERVOUS, GET HYPER

HyperSearch is the name of a full text information retrieval processor from Gould Inc. It is a UNIX-based board and software combination designed to run on Gould's PowerNode 6000 and 9000 systems.

HyperSearch is said by its creator to be capable of reading 150,000 to 300,000 words per second. Additionally, a data compression ratio of between two and three reduces disk storage requirements. The system uses no indices or occurrence lists. Every text and function word, number and punctuation mark is encoded. Database access can be restricted through measures inherent in Gould's UTX/32 operating system.

HyperSearch is scheduled to be available in August, at a price of $60,000.

Gould, Computer Systems Div., 6901 W. Sunrise Blvd., Fort Lauderdale, FL 33310-9148, 305/587-2900.

Circle No. 65 on Inquiry Card

PEOPLE SOFTWARE

CWAY Software, Inc., has announced CWAY People and CWAY Strategy, two applications packages designed for use by managers of large and small businesses, and department managers in large corporations. Designed to run under both the PC-DOS and XENIX operating systems, these programs feature menu-driven commands with on-screen instructions.

CWAY People is designed to aid with interviewing, applicant selection, and employee appraisal. It maintains both applicant and employee files, as well as a jobs file.

The second package, CWAY Strategy, is designed to provide a blueprint for planning company or department growth. Through a series of questions organized around planning, analysis, actions, and results, this package provides a framework to be used by first-time planners or business analysts.

Further information is available from CWAY Software, Inc., 121 W. DeKalb Pike, King of Prussia, PA 19406; 215/265-4060.

Circle No. 66 on Inquiry Card

Circle No. 62 on Inquiry Card

Circle No. 61 on Inquiry Card

APPLICATIONS

Continued from Page 23

ences and Astrophysics, uses UNIX "for all the reasons we all know". The system doesn't yet provide real-time data acquisition for telescopes, but it does provide a fine programming environment, and most important, it provides UUCP. For the solar physicists, UUCP is a step towards the "kinds of communications networks that the computer scientists have had for the last five years."

Frank Hill at the National Solar Observatory at Kitt Peak said that "a lot of the solar physicists use UNIX to interconnect the different sites" (such as Boulder, Kitt Peak, and Sunspot). Hill recently used UNIX to send some numerical analysis routines (designed for the study of atmospheric turbulence) to a colleague at Sunspot. Sunspot, a rather isolated site in Arizona, in turn, routinely uses the network to communicate with the Goddard Space Flight Center, the ground base for the Solar Maximum Mission satellite. (The SMM is probably the most isolated of all the observatories. You may recall it was repaired last year by a space shuttle team.)

What exactly do solar physicists study? Recently, there's been a lot of interest in the oscillations of the sun's surface. "They probe the interior of the sun," explained Scherrer. It's just like an organ pipe; there are trapped acoustic waves inside, and the frequencies and modal structures of them are determined by the temperatures and pressures they pass through. So by measuring the frequencies of oscillations at the surface, we can infer the temperature and density profiles of the interior, as well as rotation rates."

Recently, a model of the sun's interior was run on a Cray in Boulder, and the results were sent over UUCP to Wilcox in Palo Alto for analysis. "It was handy having data that had never been touched by human hands," said Scherrer.

UNIX AND EXPERT SYSTEMS

Contrary to the predictions of Lisp-machine boosters, commercial AI is migrating to smaller machines. ACE, an interesting new system from Bell Labs, is one such system running under UNIX on the 32-bit AT&T 3B2.

Consider the problems inherent in debugging the phone system. Trouble reports that are received go onto the Cable Repair Administration System (CRAS)—a UNIX-based database system that runs on either a VAX/780 or an AT&T 3B20. Some problems can be readily fixed, but others remain elusive, so cable trouble experts study the CRAS database and try to identify correlations among the various trouble reports—correlations that point the way to more subtle and invisible malfunctions.

ACE automates the job. Running at night on a small machine networked to CRAS, it analyzes the database and tries to infer the cause of reported problems. ACE pulls data from CRAS dynamically, as it decides that it needs more detailed information. When it determines that it's found something significant, it reports that observation back to the human world along with all its supporting data and repair recommendations. The humans then get a chance to agree or disagree.

Like all expert systems, ACE has its problems. It's good at what it does, but it has a very narrow scope: within that scope, it competes very well with people, but it doesn't have any general knowledge—any common sense. As a result, it sometimes makes spurious suggestions that a human would immediately recognize as absurd. In the words of one of ACE's developers, "While it's very good at finding problems, sometimes its interpretation of the problem is a little off . . ." So while ACE is probably the first expert system of its kind, which is to say one that's able to draw data off another machine, it isn't going to replace human experts anytime soon. It *does* make a valuable assistant, however.

LOTS OF OTHERS

In addition to the uses suggested by these seven examples, UNIX has been used to analyze hieroglyphics, to forecast the weather, to automate factories, to navigate supertankers, to build flight simulators, to generate graphics for the evening news—even to write love letters. It may not be entirely general purpose, but it does come tantalizingly close.

Tom Athanasiou is a UNIX programmer evolving into a science writer. He lives and works in San Francisco. ■

TAGUE INTERVIEW

Continued from Page 65

on it. Then, we'd all be kept up late at night sometime trying to help the guy get his payroll out. As it turns out, Research knew its market because things haven't turned out the way I feared they would. The first paid commercial license was issued to the Rand Corporation [it was acutally sent to Peter Wegner, who later went on to form Interactive Systems], and from there, it went on to universities and think tanks that knew exactly what to do with the system. Notice, the institution did not blunder in this case, but actually came to the right decision.

> The reliability and availability of UNIX was very good right from the start.

REVIEW: *And properly ignored you?*

TAGUE: Right. On the other hand, I was absolutely correct in saying I didn't want any part of that support. We weren't prepared to do it. I had my hands full with internal support at that time, and I knew it. I was right about that. In my part of the business, we offered support when we sold systems. We put UNIX out without support to the research community where everything worked very well. If that was a blunder, then it was a blunder that worked just fine. ■

CALENDAR

EVENTS

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.

June 12-14 Calhners Exposition Group, Anaheim, CA: "INFO: Information Management Exposition and Conference". Contact: CEG Client Services, 999 Summer St., Stamford, CT 06905. 203/964-8287.

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

JUNE

June 3 Gray Strayton International, New York: "Technology Issues in Networking". Contact: Dana Eckert, Gray Strayton International, 800 South St., Waltham, MA 02154. 617/443-7311.

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, IL 60604. 800/323-UNIX.

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

June 3-7 Interactive Systems Corp., Santa Monica, CA: "The C Programming Language". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.

June 3-7 Computer Technology Group, London: "Berkeley Fundamentals and **csh** Shell". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 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, IL 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, IL 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, IL 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, IL 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.

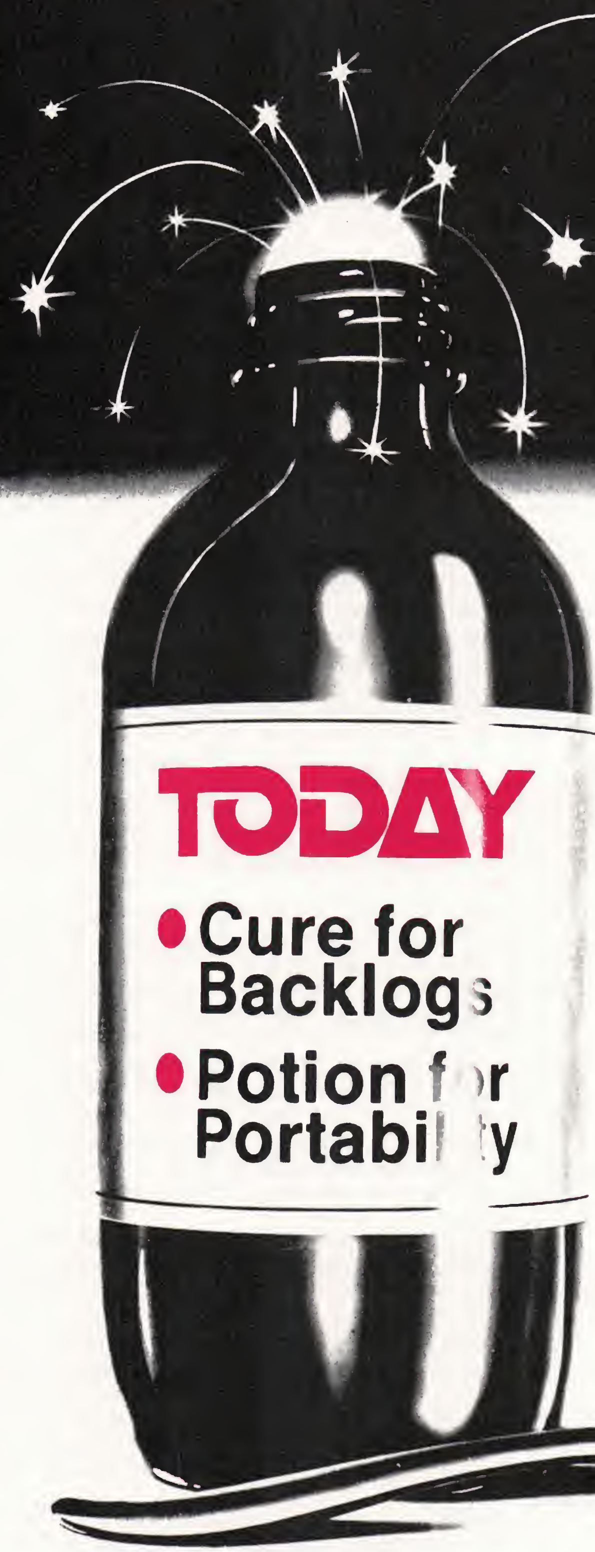
June 10 NCR Corp., Minneapolis: "UNIX Operating System". Contact: NCR Corp., CASE-Special Orders, 101 W. Schantz Ave., Dayton, OH 45479. 800/845-2273 or 800/841-2273.

June 10-11 Interactive Systems Corp., Santa Monica, CA: "Advanced C Topics for Programmers". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.

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

UNIX™ & TODAY

The Magical Mixture



Are you dizzy with software application backlogs? You need a straightforward cure—**TODAY**, the **UNIX** fourth-generation language that is breaking time zones in software development.

TODAY runs under **UNIX** to provide application portability across a wide range of machines and with a range of databases. Your application developers will enjoy a COMPLETE application building environment.

Features:

- Non-procedural specification with powerful recursive logic and Decision Tables
- Synonyms, Menus, Prompts, Helps and Defaults for streamlined definitions
- Tailoring control allowing multi versions of a base application
- 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

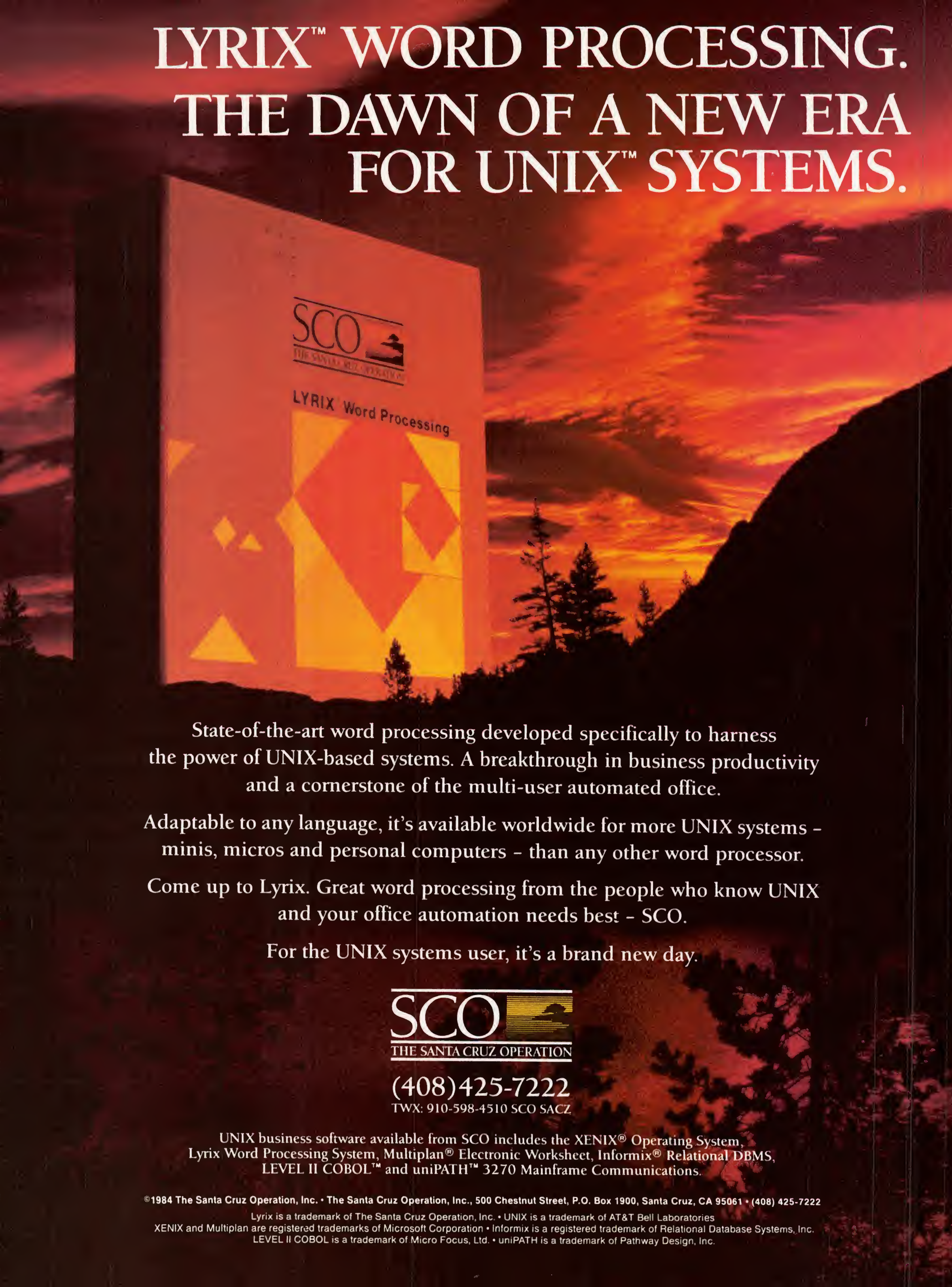
Visit our booth—No. 2144—at the NCC 85, Chicago, Illinois, July 15-18.

Call **TODAY —(408) 727-4464**

Or write: **bbj Computer Services Inc.**
2946 Scott Boulevard
Santa Clara, CA 95054

Circle No. 69 on Inquiry Card

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, IL 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, IL 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, IL 60604. 800/323-UNIX.
June 10-14 Computer Technology Group, New York: "C Language Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 11 Computer Technology Group, London: "UNIX Overview". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 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.
June 12-14 Computer Technology Group, London: "UNIX Fundamentals for Non-Programmers". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 13-14 Interactive Systems Corp., Santa Monica, CA: "Advanced C Programming Under UNIX". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.
June 17 NCR Corp., Vandalia-Dayton: "UNIX System Administration". Contact: NCR Corp., CASE-Special Orders, 101 W. Schantz Ave., Dayton, OH 45479. 800/845-2273 or 800/841-2273.
June 17 NCR Corp., Minneapolis: "UNIX System Administration". Contact: NCR Corp., CASE-Special Orders, 101 W. Schantz Ave., Dayton, OH 45479. 800/845-2273 or 800/841-2273.
June 17 NCR Corp., New York: "C Programming". Contact: NCR Corp., CASE-Special Orders, 101 W. Schantz Ave., Dayton, OH 45479. 800/845-2273 or 800/841-2273.
June 17-18 Computer Technology Group, Washington, D.C.: "Shell Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 17-18 Computer Technology Group, New York: "Shell Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 17-19 Interactive Systems Corp., Santa Monica, CA: "UNIX Fundamentals". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.
June 17-19 Interactive Systems Corp., Santa Monica, CA: "UNIX Fundamentals". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404-9989. 213/453-8649.
June 17-19 Computer Technology Group, London: "UNIX Fundamentals for Programmers". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 17-28 Information Technology Development Corp., Cincinatti: "Advanced C Programming" and "UNIX System Administration". Contact: ITD, 9952 Pebbleknoll Dr., Cincinatti, OH 45247. 513/741-8968.
June 17-28 Information Technology Development Corporation, Cincinnati: "UNIX for Application Developers". Contact: ITDC, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.
June 18 Computer Technology Group, San Francisco: "UNIX Overview". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 18 Computer Technology Group, Dallas: "UNIX Overview". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 18-21 Integrated Computer Systems, Boston: "UNIX: A Hands-on Introduction". Contact: Integrated Computer Systems, 6305 Arizona Pl., P.O Box 45405, Los Angeles, CA 90045. 213/417-8888.
June 18-21 Integrated Computer Systems, Toronto: "UNIX: A Hands-on Introduction". Contact: Integrated Computer Systems, 6305 Arizona Pl., P.O Box 45405, Los Angeles, CA 90045. 213/417-8888.
June 19-21 Digital Equipment Corp., New York: "Comprehensive Overview of the UNIX Operating System". Contact: Digital Education Resources, 12 Crosby Drive, Bedford, MA 01730. 617/276-4949.
June 19-21 Center for Advanced Professional Education, Plymouth, MI: "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.
June 19-21 Computer Technology Group, Dallas: "UNIX Fundamentals for Non-Programmers". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 19-21 Computer Technology Group, San Francisco: "UNIX Fundamentals for Non-Programmers". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 19-21 Computer Technology Group, Washington, D.C.: "Using Advanced UNIX Commands". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 19-21 Computer Technology Group, New York: "Using Advanced UNIX Commands". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 20-21 Interactive Systems Corp., Santa Monica, CA: "Using the Shell". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.
June 20-21 Computer Technology Group, London: "Shell as Command Language". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 24 Interactive Systems Corp., Santa Monica, CA: "Advanced Commands for Programmers". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.
June 24-25 Interactive Systems Corp., Santa Monica, CA: "Advanced C for Programming". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404-9989. 213/453-8649.
June 24-26 Computer Technology Group, Dallas: "UNIX Fundamentals for Programmers". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 24-26 Computer Technology Group, San Francisco: "UNIX Fundamentals for Programmers". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 24-28 Structured Methods Inc., New York: "Prolog Programming Workshop". Contact: Keith Eisenstark, 7 West 18th St., New York, NY 10011. 212/741-7720.
June 24-28 Bunker Ramo Information Systems, Trumbull, CT:



LYRIX™ WORD PROCESSING.
THE DAWN OF A NEW ERA
FOR UNIX™ SYSTEMS.
SCO
LYRIX Word Processing
State-of-the-art word processing developed specifically to harness the power of UNIX-based systems. A breakthrough in business productivity and a cornerstone of the multi-user automated office.
Adaptable to any language, it's available worldwide for more UNIX systems – minis, micros and personal computers – than any other word processor.
Come up to Lyrix. Great word processing from the people who know UNIX and your office automation needs best – SCO.
For the UNIX systems user, it's a brand new day.
SCO
THE SANTA CRUZ OPERATION
(408) 425-7222
TWX: 910-598-4510 SCO SACZ
UNIX business software available from SCO includes the XENIX® Operating System, Lyrix Word Processing System, Multiplan® Electronic Worksheet, Informix® Relational DBMS, LEVEL II COBOL™ and uniPATH™ 3270 Mainframe Communications.
©1984 The Santa Cruz Operation, Inc. • The Santa Cruz Operation, Inc., 500 Chestnut Street, P.O. Box 1900, Santa Cruz, CA 95061 • (408) 425-7222
Lyrix is a trademark of The Santa Cruz Operation, Inc. • UNIX is a trademark of AT&T Bell Laboratories
XENIX and Multiplan are registered trademarks of Microsoft Corporation • Informix is a registered trademark of Relational Database Systems, Inc.
LEVEL II COBOL is a trademark of Micro Focus, Ltd. • uniPATH is a trademark of Pathway Design, Inc.

"Advanced C". Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.
June 24-28 Computer Technology Group, Washington, D.C.: "UNIX Internals". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 24-28 Computer Technology Group, New York: "UNIX Internals". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 24-28 Computer Technology Group, London: "C Language Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 25-26 Interactive Systems Corp., Santa Monica, CA: "UNIX Architecture—A Conceptual Overview". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.
June 25-26 Zilog Education and Training Department, Campbell, CA: "UNIX for Non-Technical Users". Contact: Kay Ferrell, 1315 Dell Ave., Campbell, CA 95008. 408/370-8000.
June 25-28 Integrated Computer Systems, Boston: "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 26 Specialized System Consultants, Seattle: "UNIX for Managers". Contact: SSC, P.O. Box 7, Northgate Station, Seattle, WA 98125-0007. 206/367-UNIX.
June 26-28 Interactive Systems Corp., Santa Monica, CA: "UNIX Architecture: A Conceptual Overview". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404-9989. 213/453-8649.
June 27-28 Interactive Systems Corp., Santa Monica, CA: "System Administrator's Overview". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.
June 27-28 Computer Technology Group, Dallas: "Shell as a Command Language". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
June 27-28 Computer Technology Group, San Francisco: "Shell as a Command Language". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

JULY

July 1-2 Computer Technology Group, London: "Shell Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 1-3 Interactive Systems Corp., Santa Monica, CA: "Interactive Networking Tools". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.
July 3-5 Computer Technology Group, London: "Using Advanced UNIX Commands". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 8-9 Interactive Systems Corp., Santa Monica, CA: "Using

- Passage, Uniq's TCP/IP Solution to UNIX SYSTEM V. 2 Networking.
- Ready for immediate delivery on Digital's VAX Processors.
- Local Area Network and DDN ARPANET Support.
- Uniq is committed to UNIX... UNIX Networking...UNIX Applications ...UNIX Consulting...We make it Work.

uniq

312/879-1008
DIGITAL TECHNOLOGIES
28 S. Water Street, Batavia, IL 60510

Los Angeles	Washington, D.C.	Boston
213/277-6288	703/448-8508	603/883-4860

*Unix and System V are trademarks of AT&T Bell Laboratories.

Ten/Plus". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.
July 8-12 Computer Technology Group, San Francisco: "C Language Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 8-12 Computer Technology Group, London: "UNIX Internals". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 8-12 Information Technology Development Corporation, Cincinnati: "INFORMIX Relational Data Base". Contact: ITDC, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.
July 8-12 Computer Technology Group, Dallas: "C Language Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 8-12 Information Technology Development Corporation, Cincinnati: "INFORMIX Relational Data Base". Contact: ITDC, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.
July 9-11 Computer Technology Group, Washington, D.C.: "UNIX Administration". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 9-11 Computer Technology Group, New York: "UNIX Administration". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 9-12 Integrated Computer Systems, Los Angeles: "UNIX: A Hands-on Introduction". Contact: Integrated Computer Systems, 6305 Arizona Pl., P.O Box 45405, Los Angeles, CA 90045. 213/417-8888.
July 10 Bunker Ramo Information Systems, Trumbull, CT: "UNIX Marketing". Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.
July 10-12 Interactive Systems Corp., Santa Monica, CA: "Customizing Ten/Plus". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.
July 15-16 Computer Technology Group, Washington, D.C.: "Advanced C Programming Workshop". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 15-16 Computer Technology Group, San Francisco: "Shell Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 15-16 Computer Technology Group, New York: "Advanced C Programming Workshop". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 15-16 Computer Technology Group, Dallas: "Shell Programming". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 15-17 Computer Technology Group, London: "UNIX Administration". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 15-19 Interactive Systems Corp., Santa Monica, CA: "Helper Writer Workshop". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.
July 15-19 Bunker Ramo Information Systems, Trumbull, CT: "Introduction to UNIX". Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.
July 15-19 Zilog Education and Training Department, Campbell, CA: "UNIX Users Seminar". Contact: Kay Ferrell, 1315 Dell Ave., Campbell, CA 95008. 408/370-8000.
July 15-19 Information Technology Development Corporation, Cincinnati: "UNIX for End Users". Contact: ITDC, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.
July 15-19 Information Technology Development Corporation, Cincinnati: "UNIX for End Users". Contact: ITDC, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.
July 16-19 Integrated Computer Systems, Los Angeles: "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.
July 17-19 Computer Technology Group, San Francisco: "Using Advanced UNIX Commands". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 17-19 Computer Technology Group, Washington, D.C.: "Advanced C Programming Under UNIX". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 17-19 Computer Technology Group, Dallas: "Using Advanced UNIX Commands". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 17-19 Computer Technology Group, New York: "Advanced C Programming Under UNIX". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 17-19 Digital Equipment Corp., Boston: "Comprehensive Overview of the UNIX Operating System". Contact: Digital Education Resources, 12 Crosby Drive, Bedford, MA 01730. 617/276-4949.
July 22-23 Computer Technology Group, London: "Advanced C Programming Workshop". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 22-24 Interactive Systems Corp., Santa Monica, CA: "UNIX Fundamentals". Contact: Claire Donahue, 2401 Colorado Ave., 3rd floor, Santa Monica, CA 90404. 213/453-8649.
July 22-24 Zilog Education and Training Department, Campbell, CA: "UNIX System Administrator". Contact: Kay Ferrell, 1315 Dell Ave., Campbell, CA 95008. 408/370-8000.
July 22-26 Bunker Ramo Information Systems, Trumbull, CT: "Advanced UNIX". Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.
July 22-26 Computer Technology Group, Washington, D.C.: "Berkeley Fundamentals and **csh** Shell". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 22-26 Computer Technology Group, San Francisco: "UNIX Internals". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 22-26 Computer Technology Group, Dallas: "UNIX Internals". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 22-26 Computer Technology Group, New York: "Berkeley Fundamentals and **csh** Shell". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.
July 22-August 2 Information Technology Development Corporation, Cincinnati: "C Programming Language". Contact: ITDC, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.
July 22-August 10 Rocky Mountain Institute of Software Engineering, Colorado Springs and Aspen, CO: "1985 Summer Tutorial Program". Contact: RMISE, P.O. Box 3521, Boulder, CO 80303. 303/499-4782.
July 24-26 Computer Technology Group, London: "Advanced C Programming Under UNIX". Contact: Computer Technology Group, 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX.

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.

Making UNIX Easier with TEN/PLUS

Easier to Learn

Get your users started on UNIX by teaching them the standard keyboard commands and ten special commands. In the TEN/PLUS environment, that is enough to perform most common tasks and to invoke any application. As they gain experience, your users can employ more powerful TEN/PLUS commands and all UNIX commands.

Easier to Use

The procedure for using TEN/PLUS is simple: point at data with a cursor and then use a TEN/PLUS command. If your users need some prompting, they can ask for a menu. If they are confused, they can ask for a HELP message. If they are processing several files, they can open a window on each file.

Easier to Support

The TEN/PLUS environment eliminates the errors your users make when different applications require different sets of commands. Designed around a full-screen editor, TEN/PLUS allows users to manipulate all data with the text editing commands that they use most often. This means that users can use already familiar techniques to process both text and data. In addition, the TEN/PLUS system provides self-explanatory error messages.

Here's another way to reduce support costs: adopt TEN/PLUS as a standard user environment on a variety of computers—from personal computers to mainframes. Then provide TEN/PLUS to all kinds of users: clerks, managers, and engineers. A common user environment means your computer staff will have fewer products to support.

Easier to Network

We're porting the TEN/PLUS environment to most versions of UNIX and to VMS. We'll help you port it to other systems.

To link systems running our software, we're offering electronic messaging (INmail) and a network manager (INnet) as TEN/PLUS options. These packages are already a part of IX/370, the IBM mainframe UNIX system, and are available as an option on PC/IX, the IBM UNIX system for PCs. Thus, your TEN/PLUS system can readily participate in a network with IX/370 and PC/IX.

Easier to Expand

We're also offering a set of development tools that helps expand the TEN/PLUS environment. One kit provides your programmers with utilities and languages for defining and using screen forms, a user-friendly interface to the C compiler, and subroutine libraries. Another kit provides forms design and programming capabilities for your end users.

Call today for more information about the TEN/PLUS environment: (213) 453-UNIX.

Software Tools for System Builders.

INTERACTIVE
SYSTEMS CORPORATION

2401 Colorado Ave., 3rd Floor
Santa Monica, CA 90404
TWX 910-343-6255; Telex 18-2030
Telephone (213) 453-UNIX

Circle No. 75 on Inquiry Card

THE LAST WORD

Letters to the Editor

A NIT TO PICK

Dear UNIX REVIEW,

I have just received my first issue of UNIX REVIEW and I have found it to be both stimulating and informative.

I would also like to comment on Jack Cohen's letter in the April, 1985 issue, because I think that understanding why something fails is probably more important than noting the fact that it does fail—since it is then possible to avoid the same kind of error later.

The application:

```
pick * | rm
```

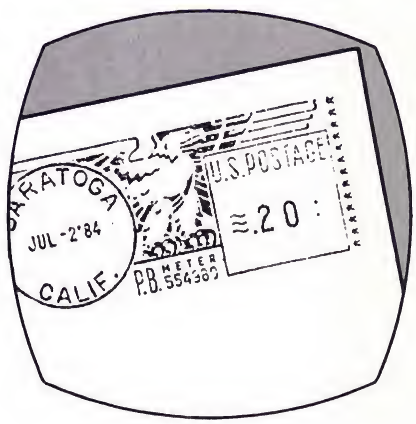
fails because **rm** does *not* read the standard input in any implementation with which I am familiar. Yet the above format does redirection of standard output to standard input. The **rm** utility expects filenames as arguments, which requires the second format. However, as printed in Mr. Cohen's letter, it will not work correctly either because the apostrophes result in **rm** getting a filename that looks like *pick-space-asterisk*. The proper character is the grave accent, which produces the desired action. The correct form is then:

```
rm `pick*`
```

which causes **pick** output to be given to **rm** as arguments.

Bob McGowan
Sunnyvale, CA

Many thanks for the observation, but I'm afraid that UNIX REVIEW *will have to take the rap for not presenting the accents in the grave orientation. Chalk another one up to the joys of computer typesetting.* *Editor*



DIGGING INTO THE ARCHIVES

Dear UNIX REVIEW,

There is a regrettable error in the January 1985 */usr/lib* department that deserves correction. In it, Jim Joyce writes that "HP-UX runs on top of BASIC for Hewlett-Packard ports." This statement is entirely false.

There are two series of HP 9000 machines which have distinct origins and run HP-UX. The Series 500 is based upon a proprietary 32-bit multiprocessor piece of hardware and a proprietary operating system with virtual memory, networking, and multiprocessing support. The Series 200 is based upon the Motorola 68000 family of processors and is a System III port. The newly introduced Hewlett-Packard Integral Personal Computer is also a System III port. HP-UX provides System III functionality and semantics with selected Berkeley and Hewlett-Packard extensions.

Both HP 9000 machines offer totally separate standalone Basic workstation products. In addition, the Series 200 offers a Pascal workstation product. In no way whatsoever does HP-UX on either machine run "on top of" the Basic workstation product.

Bruce P. Rodean
Hewlett-Packard Company
Ft. Collins System Division
Ft. Collins, CO

TO GOTO OR NOT TO GOTO

Dear UNIX REVIEW,

I was shocked to see within the *C Advisor* (March, 1985) a passionate defense of the **goto** statement. While Mr. Tuthill pleads for a multilevel break command in order to permit more structured code, he then proceeds to extoll the virtues of the **goto** statement in error handling. I have extracted a fragment of his exemplary code [listed in Figure 1], having changed it only to put the '{ }' characters into positions more reasonable for the example.

This code is intended to demonstrate the difficulty in dealing with abnormal input without using the **goto** statement. Mr. Tuthill then provides an accompanying example of "so-called structured code"

```
name:
        fputs("Your name please: ",stdout);
        if(!fgets(s1,sizeof(s1),stdin))
                exit(1);
        if(*s1 == '\n')
                goto name;
        for(c=s1;*c != '\n'; c++)
                {
                if(!isalpha(*c) && !isokname(*c))
                        {
                        fputs(" invalid characters\n", stdout);
                        goto name;
                        }
                }
```

Figure 1 — *An example of Bill Tuthill's* **goto** *transgressions.*

```
for(answered=FALSE; !answered;)
        {
        fputs("Your name please: ",stdout);
        if(!fgets(s1,sizeof(s1),stdin))
                exit(1);
        for(c=s1;*c != '\n'; c++,answered++)
                if(!isalpha(*c) && !isokname(*c))
                        {
                        fputs(" invalid characters\n", stdout);
                        answered= FALSE;
                        break;
                        }
        }
```

Figure 2 — *An example of the virtues of structured code.*

that is so poorly written that it generates longer and slower object. He argues that attempting to add structure results in inefficiencies.

The fragment [listed in Figure 2] is comparable in function to the fragment [in Figure 1] and is, in fact, structured. This fragment produces assembly code on a PDP-11 that is four statements longer than the fragment [in Figure 1] but performs as efficiently as the code generated by Mr. Tuthill's example. The resulting object modules are comparable in size. It may be of interest to note that the code generated by the break statement is identical to the code generated by the **goto** statement, some of the **compare** operations were not present, and the additional code consists of the moderately efficient instructions *increment* and *test* (**inc** and **tst**). In my estimation, the structured code is more lucid as well.

Mr. Tuthill avoids the fundamental issues surrounding the **goto** statement. Those issues are:

1) The **goto** statement makes it easier to induce bugs into code.
2) Use of the **goto** statement is an expedient that relieves the programmer of the need to understand the actual problem at hand.

Induced bugs typically occur when code is added to the "gone to" segment of the program. Since there is no indication in the code fragment being "gone to" concerning where the program may have come from, it is both possible and common to introduce unwanted side effects for one or more paths.

The issue of expediency is of much more concern to me, however. The purpose of the exemplary code fragment is to ask the question until it has been answered properly by the user. The manner in which it was originally coded, however, would suggest that the purpose is to reject incorrect answers. Admittedly, there is little functional difference in the example; however, in more complex coding situations, there is often a substantial difference between the two forms of the issue.

By structuring the code, a programmer is virtually forced to consider the actual issues rather than the one that first comes to mind. In my experience, the chronically good code is separated from the chronically bad code by whether the programmer was thinking about the real issues or not. The density of **go to** statements per pound of code has historically shown itself to be an accurate indicator of the programmer's thinking.

In all fairness to Donald Knuth, it would appear that he has not brought the controversy to an end. It would also be unfair to characterize Mr. Knuth as an apologist or advocate of the **go to** statement. I suspect that Mr. Knuth would like to see languages in which the construct is not necessary, while Mr. Dijkstra would prefer that programmers simply delete it from their lexicon. I see no functional difference between the two positions, only differences of perspective.

John Cornelius
Western Scientific
San Diego, CA

```
#include        <stdio.h>
#include        <ctype.h>
#define         isNchar(c)      (isalpha(c)||((c)==' ')||((c)=='-')||((c)==',')||((c)=='.'))
#define         isPchar(c)      (isdigit(c) || ((c) == '-'))

main() /* getname - validate user's name and phone number */
{
        char    name[BUFSIZ/2];
        char    phone[BUFSIZ/2];
        char  * lastc;  /* last valid character in name */
        int     length = 0;

        do {
                fputs("Your name please: ", stdout );
                if (!fgets(name, sizeof(name), stdin ))
                        exit(1);

                /* check for valid name */
                for (lastc=name;((*lastc)!='\n') && isNchar(*lastc);lastc++) {
                }

                if ( (*lastc) != '\n' ) {
                        fputs(" invalid characters\n", stdout );
                }

        } while ((*name == '\n') || (*lastc != '\n' ));

        do {
                fputs("Your phone number: ", stdout );
                if ( !fgets(phone, sizeof(phone),stdin ) )
                        exit(1);

                /* check for valid phone number */
                for (lastc=phone; ((*lastc)!='\n') && isPchar(*lastc); lastc++) {
                        if ( isdigit(*lastc) )
                                length++;
                }
                if ( (*lastc) != '\n' ) {
                        fputs(" invalid number (digits and -)\n",stdout );
                }
                else if ( (length != 7) && (length != 10) ) {
                        fputs(" incorrect length (7 or 10)\n",stdout );
                        length = 0;
                }
        } while (( *lastc != '\n' ) || (length == 0) );

        printf("\nName:\t%s", name );
        printf("Phone:\t%s", phone );
}
```

Figure 3 — *A transformed version of C Advisor code—sans* ***gotos****.*

Dear UNIX REVIEW,

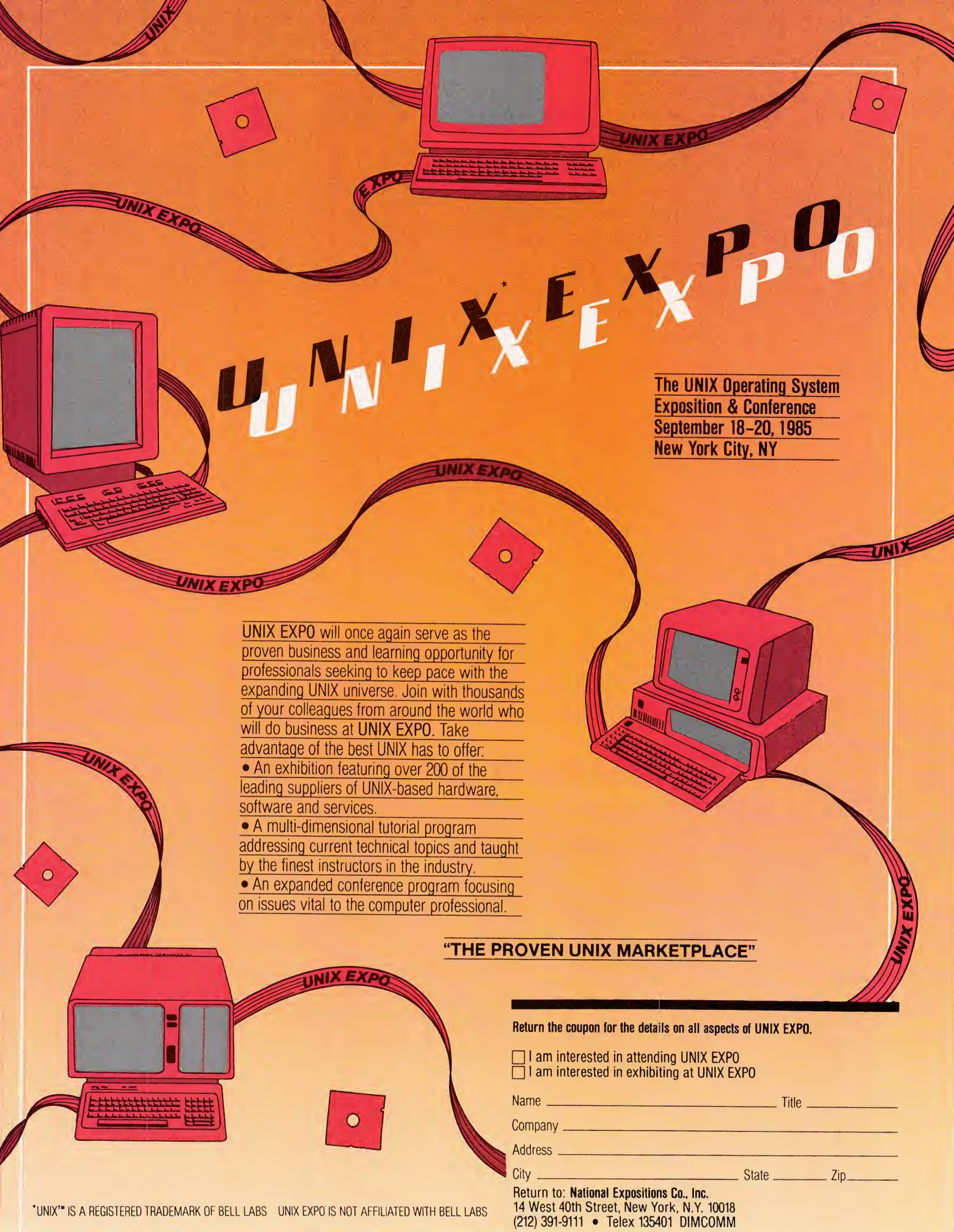
In a recent column (*C Advisor*, March 1985), Bill Tuthill presents an example program using several **goto** statements and then claims, "transforming this into a program without **goto**s requires two extra Boolean variables, at least four extra initializations, and at least four extra comparisons." Mr. Tuthill should learn to program without using **goto**s before making such statements. It is, in fact, easy and natural to transform his example into a structured version that is both clear and that costs no extra variables or initializations. [An example is shown in Figure 3.]

The structured version requires four extra comparisons, and the object file is 28 bytes longer than Mr. Tuthill's version, using the 4.2BSD C compiler. But I believe that the benefits of structured programming are well worth this small cost.

I have frequently found that structured programming forces me to understand algorithms better. For example, the **while** statement at the end of the name input loop clearly indicates that the loop will repeat as long as either a zero-length string or an invalid string is input. These conditions are not as clear in Mr. Tuthill's code, where the reader must search the entire distance between the two labels to find all occurrences of the command "**goto name**". In fact, the labels can be jumped to from anywhere within the procedure.

Given its benefits, structured code should be used as much as possible, especially to write large programs. Once a program has been written, it should be thoroughly analyzed to find bottlenecks that justify the use of non-structured programming techniques. Such code should be especially well documented and as short as possible.

George Entenman
Microelectronics Center
of North Carolina
Research Triangle Park, NC



Circle No. 2 on Inquiry Card

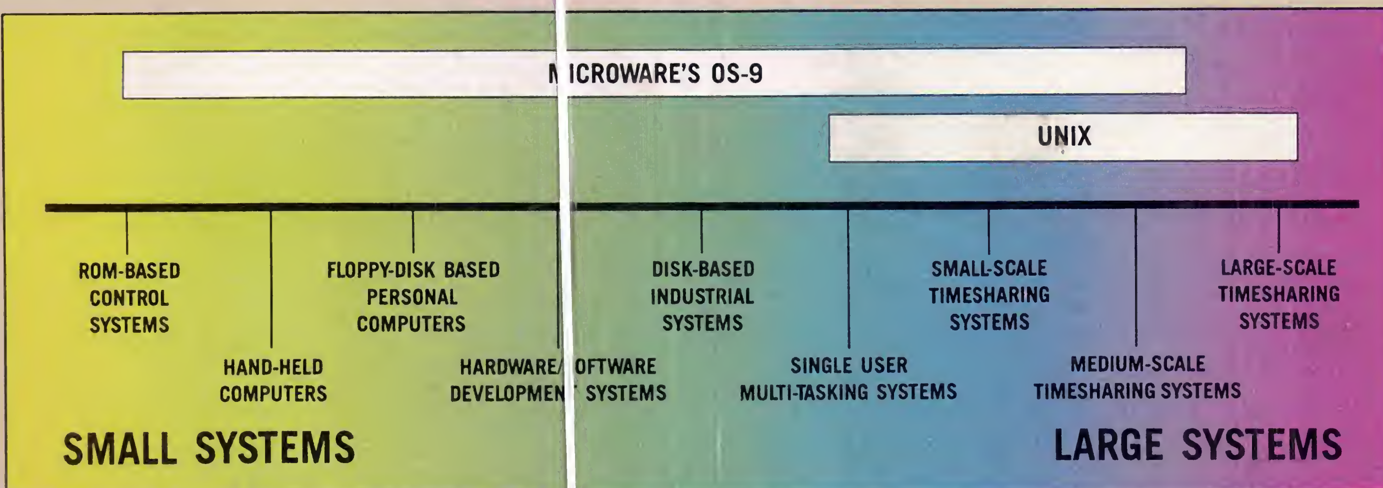
ADVERTISER'S INDEX

COMING UP IN JULY

Office Automation

- **Historical Perspectives on the UNIX Role**
- **Do-It-Yourself Applications**
- **The Role Office Mail Plays**
- **Directions in Technology**
- **How Output Capabilities are Shaping Up**

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

 See us at NCC '85, Booth #1444

Circle No. 12 on Inquiry Card