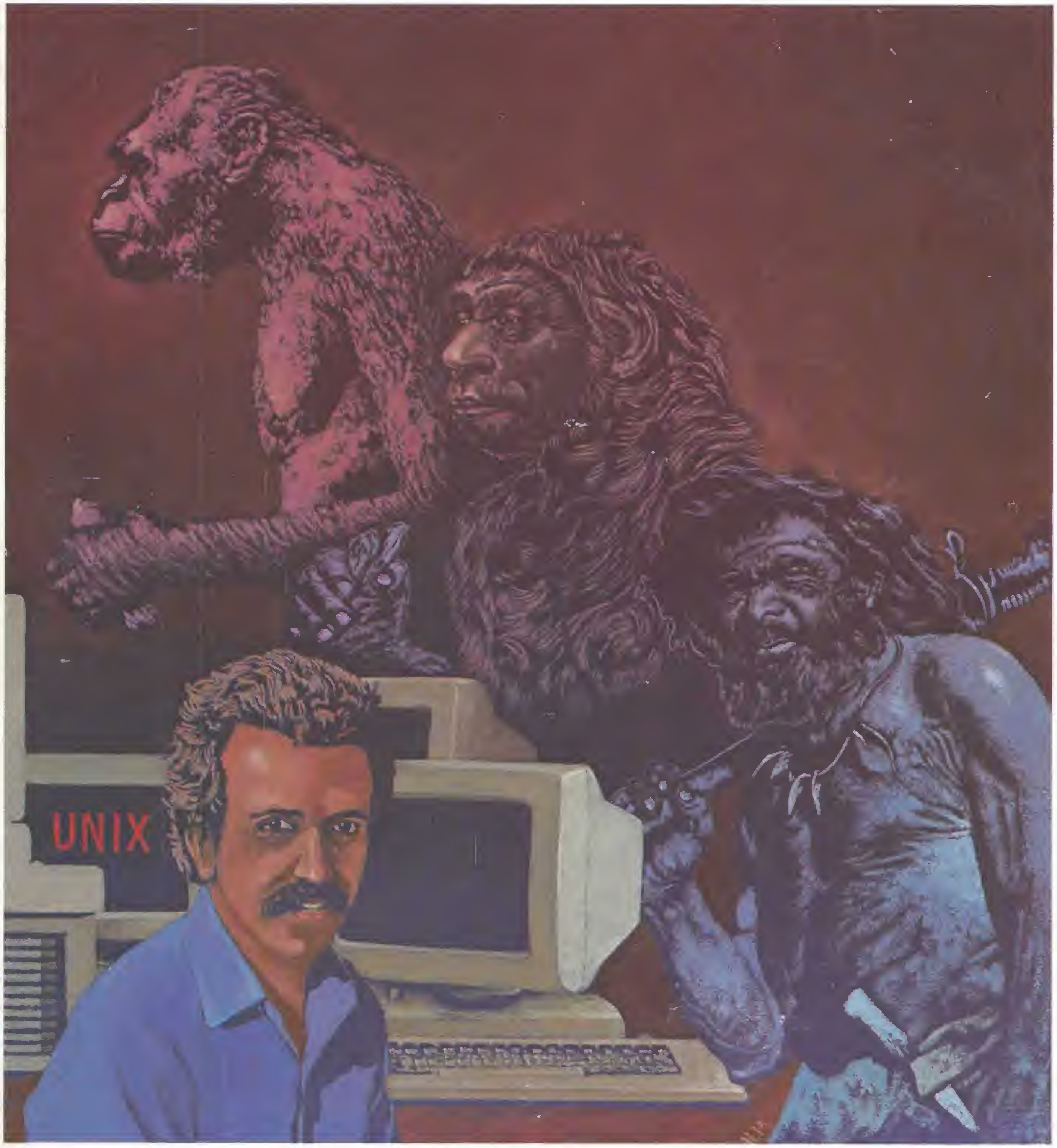
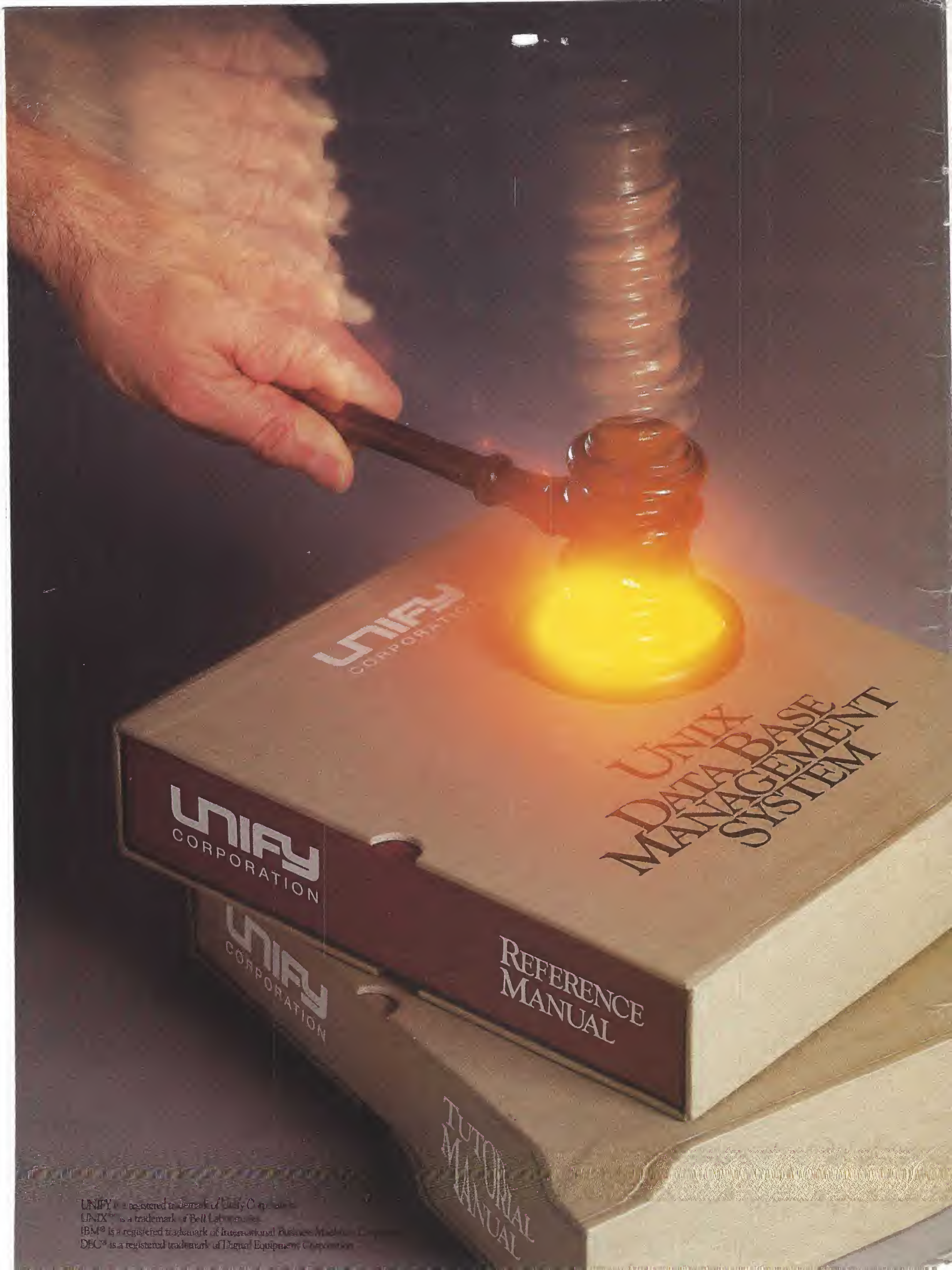
UNIX™ REVIEW

THE PUBLICATION FOR THE UNIX™ COMMUNITY

January 1985 $3.95



THE EVOLUTION OF UNIX


unify CORPORATION
UNIX
DATA BASE
MANAGEMENT
SYSTEM
unify CORPORATION
REFERENCE
MANUAL
unify CORPORATION
TUTORIAL
MANUAL

UNIFY. JUDGED NO. 1 BY COMPUTER OEMS.

One UNIX computer manufacturer after another has come to the same decision: UNIFY is the fastest, most powerful, most flexible data base management system for users of all skill levels.

By their own investigation and by system integrator requests, computer manufacturers representing some 90% of the market choose to offer UNIFY with their UNIX computers.

They include DEC. Perkin-Elmer. NCR. Tandy. Pixel. Onyx. Cadmus. Codata. Cromemco. Momentum. Plexus. Altos. Callan Data. And many more.

The evidence is overwhelming. In independent benchmarks, UNIFY consistently ranks as the top performer.

Completely menu-driven design and industry standard IBM SQL query language make it easy for non-programmers to develop data base applications.

The most powerful "back end" design in the industry, including 90 subroutines at the host language interface level, promises that UNIFY can keep adding features, keep adding users, without eroding performance.

Judge for yourself. Our comprehensive 300-page tutorial and 500-page reference manual system are yours for only $95. Together they show you how to build virtually any application of your choice.

Contact UNIFY, Dept. MMS-11, 4000 Kruse Way Place, Bldg. Two, Suite #255, Lake Oswego, OR 97034, (503) 635-6265, TELEX 469220.

UNIFY®

THE PREFERRED UNIX DBMS.

Circle No. 1 on Inquiry Card

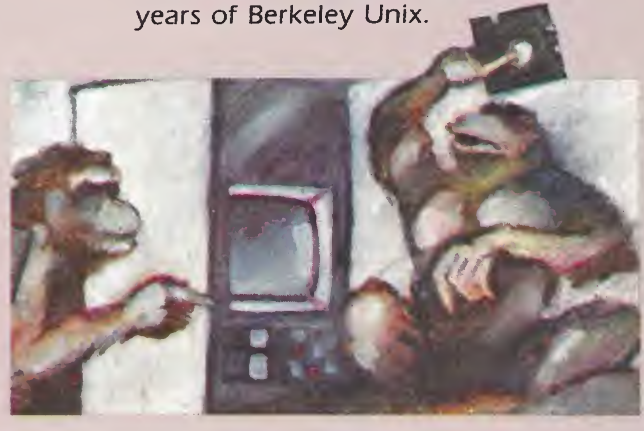
UNIX™ REVIEW

THE PUBLICATION FOR THE UNIX COMMUNITY

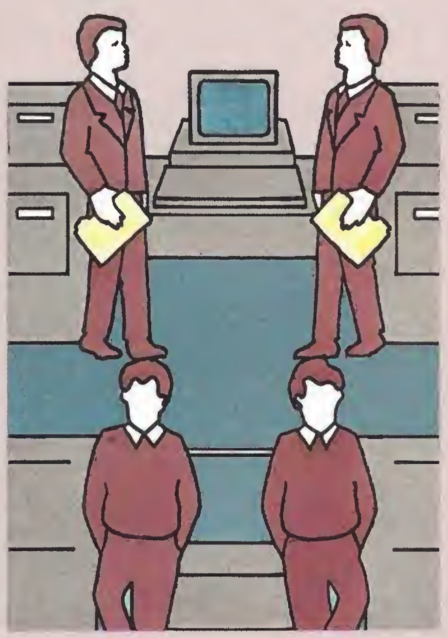
Volume 3,
Number 1
January 1985

DEPARTMENTS:

FEATURES:



THE EVOLUTION OF UNIX



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. Subscriptions are $28.00 per year (12 issues). Canada and Mexico add $7/year. Overseas add $20/year for surface mail. Address correspondence regarding editorial, press releases, product announcements to 901 South 3rd Street, Renton, WA 98055. . Correspondence regarding subscriptions, change of address, USPS Form 2579, fulfillment and dealer sales should be addressed to 901 S. 3rd Street, Renton, WA 98055. 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. Letters may be edited for the purpose of clarity or space. Opinions expressed by the authors are not necessarily those of UNIX REVIEW.

Subscription/Dealer Sales telephone: 206/271-9605.
UNIX is the trademark of Bell Laboratories, Inc. UNIX REVIEW is not affiliated with Bell Laboratories.
Application to mail at second class postage rate is pending at Renton, Washington 98055.

PUBLISHER:
Pamela J. McKee
EDITOR:
Mark Compton
ASSISTANT EDITOR:
David Chandler
ASSOCIATE EDITORS:
Ken Roberts, Scott Robin
PRODUCTION DIRECTOR:
Nancy Jorgensen
PRODUCTION STAFF:
James Allen, Dan Arthur, Tom Burrill, Cynthia Grant, Corey Nelson, Florence O'Brien, Denise Wertzler
CIRCULATION/DEALER SALES:
Tracey McKee, Barbara Perry
BUSINESS MANAGER:
Ron King
ADVERTISING REPRESENTATIVES:
Jules E. Thompson, Inc.
1290 Howard Avenue Suite 303
Burlingame, CA 94010
Lucille Dennis 415/348-8222
303/595-9299-Colorado
312/726-6047-Illinois
617/720-1888-Massachusetts
713/731-2605-Texas
Jules E. Thompson, Inc.
2560 Via Tejon
Palos Verdes Estates, CA 90274
Ed Winchell 213/378-8361
212/772-0933-New York
TELE-TYPESETTING:
Data & Staff Service Company
PRINTING:
Thomas Ogle
EDITORIAL ADVISORS:
Dr. Stephen R. Bourne, Software Engineer, Digital Equipment Corp.
EDITORIAL REVIEW BOARD:
Dr. Greg Chesson, Technical Staff, 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
George Goble, Systems Engineer, Purdue University
Bill Joy, Vice President of Research and Development, Sun Microsystems
John Mashey, Software Engineering Manager, Convergent Technologies
Robert Mitze, Department Head, UNIX Computing System Development, AT&T Bell Labs
Deborah Scherrer, Computer Scientist, Mt. Xinu
Jeff Schriebman, President, UniSoft Systems
Otis Wilson, Manager, Software Sales and Marketing, AT&T Technology Systems
Walter Zintz, Executive Director, Uni-Ops
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
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

VIEWPOINT

Learning from history

Last month's issue started with an article titled "UNIX for the People". In that article, author Gig Graham explained the economic reasons for making the system more accessible to a wider range of consumers. This month, our focus shifts to *UNIX by the people*—as in UNIX demanded by the people.

If ever there was a grassroots product, UNIX is it. No high-priced market study or slick sales campaign played a role in bringing UNIX to the market position it enjoys today. From the start, UNIX was released to ever widening circles of users only because those users demanded it.

The reasons for that demand have been remarkably consistent. Users want a well-mannered system that obligingly steps aside after preparing the way for the task at hand. UNIX succeeds simply because it performs this role. Those who don't care to do systems programming needn't—they can forget about the operating system and concentrate on either building or using applications. Those who consider themselves hackers can be selective about the portions of the system they work on since UNIX is both modular and flexible; reworking a utility thus does not entail rethinking the entire operating system.

In evaluating the success of UNIX, though, it is not enough simply to look at *what* UNIX is. One must also trace *how* it evolved—both as a tool and a product—in order to understand *why* UNIX is what it is. History yields valuable lessons for vendor and end user alike. For the vendor, trends of the past provide points against which future probabilities can be plotted. As for the end user, understanding why UNIX is the way it is may be a hedge against much gnashing of the teeth and rending of the terminal.

To say that we have sought out authorities to tell this UNIX tale would be a gross understatement. August Mohr, the former editor of CommUNIXations, leads off with an article detailing how UNIX spread within the Bell System. He draws heavily on interviews with Berkley Tague, the head of UNIX support efforts (USG) in the early years, and Rudd Canaday, who managed the Programmers' Workbench project.

As a counterpoint to this account of how UNIX took shape as a product, a reprint of Dennis Ritchie's Turing Award Lecture follows with a description of how the system initially evolved as a research tool.

The official story of the other major strain of UNIX, Berkeley UNIX, is told by the man responsible for implementing its file structure, Kirk McKusick. Bob Fabry, Bill Joy, Sam Leffler, and Eric Allman are among the other key participants who made contributions to the article.

Doug Merritt, Bob Toxen, and Ken Arnold offer a somewhat different view of Berkeley UNIX in an attached piece. Call it a report on Gonzo programming.

The organizational changes within AT&T that made it possible for UNIX to be released to the world at large are detailed in an article by Otis Wilson titled "The Business Evolution of the UNIX System." Otis knows whereof he speaks—as manager of AT&T's software marketing and sales arm, he designed the UNIX licensing program.

Rich Morin pulls us into the present and then slings us into the future with an account that tells what to expect in the workstation of tomorrow. Significant contributions to the article were made by Dr. Greg Chesson, Gene Dronek, Dr. James Gosling, and Ron Gremban.

The evolution theme concludes with an interview featuring Victor Vyssotsky, currently the Executive Director of Research in the Information Systems Division of AT&T Bell Labs and formerly the head of the Labs' Multics project. Ned Peirce, a Contributing Editor for UNIX REVIEW, delivered the questions.

Acknowledgements would not be complete, though, without a tip of the hat to Dennis Ritchie and Ken Arnold for the assistance they offered by reviewing selected articles. The net result, I hope, is an issue you will not only want to read through carefully, but retain long after it has grown dogeared and rumpled.

Mark Compton

YOU CAN'T PREDICT THE FUTURE.

BUT YOU CAN BE PREPARED FOR IT.

WITH INFORMIX AND File-it!

THE FIRST DATABASE SOFTWARE FAMILY FOR UNIX AND MS-DOS.

SUBSCRIPTION ORDER FORM

UNIX REVIEW

12 issues for $28.
Canadian subscribers add $7.
Foreign subscribers add $20 for overseas surface mail.

Name ______________________

Company ______________________

Address ______________________

City/State/Zip ______________________

PLEASE ENCLOSE CHECK OR INCLUDE CREDIT CARD INFORMATION.

MasterCard/VISA No.______________ Exp.______

Signature ______________________

SUBSCRIPTION ORDER FORM

UNIX REVIEW

12 issues for $28.
Canadian subscribers add $7.
Foreign subscribers add $20 for overseas surface mail.

Name ______________________

Company ______________________

Address ______________________

City/State/Zip ______________________

PLEASE ENCLOSE CHECK OR INCLUDE CREDIT CARD INFORMATION.

MasterCard/VISA No.______________ Exp.______

Signature ______________________

DEVIL'S ADVOCATE

Enough mollycoddling!

by Stan Kelly-Bootle

While discussing "System Friendly Users" in the November issue, Jim Joyce casually threatened to emasculate the very soul of UNIX! (The metaphysics of soul-gender are beyond the immediate scope of this column.) Suffice it to say that Cicero, Livy, and that crowd were careful to distinguish *animus* (masculine) from *anima* (feminine). Joyce suggested that we take all the macho fun out of:

```
$ rm *
```

by adding:

```
alias rm rm -i
```

to the **.login** or **.cshrc** files.

You will not need me or Don Norman to tell you that **rm *** is an awesome weapon, worth three ICBMs at any bargaining table, and hardly to be recommended as a casual demonstration of the power of the wild-card symbol, *.

But, haven't we all, at one time or another, developed the irrestible urge to kill all our files, or those of a colleague? I know I have. And as MacBeth (or was it Mrs. MacBeth?) so rightly said, "**IF** 'twere done when 'twere done, 'twere best 'twere done quickly **ENDIF**." (MacBeth and Mrs. MacBeth are probably registered trademarks of Apple Computer, Inc.). The *last* thing we want, after such a courageous and diskspace-saving decision, is an endless stream of dumb Y/N questions.

Hardcore UNIX users know that these are really {Y* or y*}/not{Y* or y*} questions, and feel honor-bound to avoid nerdish responses like "Y" or "N". It is essential to know enough words *beginning* with "Y" or "y" (or not, as the mood dictates) so that your responses: a) avoid repetition, and b) are amusingly appropriate to the filename being erased or reprieved. For example:

```
$ rm -i *
bible: yea
element: ytterbium
cash: holy_holy_holy
smegma: YUK
```

elegantly kills three of the four files.

Mollycoddling menu-managers like to inject tear-jerking pleas such as:

> Have you consulted the OWNER of these files? Y/N?
>
> Are you really sure you want to end it all? Y/N?
>
> Have you considered ALL the consequences? Y/N?

followed, eventually, by visits from neighbors, social workers, priests looking like Spencer Tracy or Barry Fitzgerald, and your sick mother flown in all the way from Wexford.

"So it's after killing off all yer foiles now, is it, they tell me?"

"Just ould rubbish from the past, Ma. Shameful memories we'll be well rid of, and no mistake. And besoids, they're mostly Jack Murphy's foiles, and him struttin' and hierarchin' about like he owned the whole patch, roots an' all."

When the screen finally hits you with:

> OK - this is IT, last chance - confirm DELETE ALL FILES. Y/N?

and when, worn out and terrified, you enter "N", the system's response is a nauseating, folksy:

> Well now, that WAS a pretty CLOSE CALL there, uh? Y/N?

UNIX panders enough already to the indecisive and chicken-hearted. The **rmdir** command, for example, will only remove *empty* directories. How mamby-pamby can you get? My solution is:

```
alias rm rm -rf
```

and zap, zap—as they are calling your mother's flight at Shannon

International, and as Jack Murphy complacently sips his coffee next door, another Royal Library of Alexandria is flagged dead without question and without the waste of a single pagan match.

It has not escaped our attention that Jim Joyce's "System Friendly Users" essay reveals elements of parody beyond those typically found in his */usr/lib* offerings, and that even from a paginational point of view he has migrated somewhat towards the *Devil's Advocate*'s column. Our millions of hidden detractors could hardly call us paranoiac, but we do detect here some encroachment on our territory, the quiche-loving Oligarchy of Satira (population 1). Apparently our flabby indolence, love of peace, and low demographic profile have been taken as signs of weakness. But while we may lack numbers, we have a terrifying national unity of purpose when tickled the wrong way.

The Satirical State is taking immediate and vigorous action to protect its borders. By an overwhelming majority our Praesidium has approved total mobilization, quiche-rationing, the purchase of MIG fighters, and the promulgation of a Sharply Written Reprimand.

It was further decreed that this column should mount a counter-attack on Joyce's */usr/lib* preserve by pre-reviewing an unpublished book on UNIX.

UNIX System XIV Rev 3.26 for Beginners

We will not mince our words about *UNIX System XIV Rev 3.26 for Beginners* (Pemmican Press, 1997, $749.95). Apart from the low price, this is a truly terrible book by Jack Murphy, who thoughout seems to confuse UNIX System XIV Rev 3.26 with a pre-AT&T/IBM-merger version, UNIX System XII Rev 9.81. We say *seems* because the ghastly SuperMacWrite typesetting it employs on coarse oatmeal recycled paper makes the whole mess damned near illegible. We spotted three typos on the spine alone, hardly a reassuring start for a book claiming to instill good programming habits in the beginner.

Haven't we all, at one time or another, developed the irrestible urge to kill all our files, or those of a colleague?

The few command sequence examples we did manage to decipher and enter caused a wide spectrum of crashes and catastrophes. Suffice it to say readers are advised to have plenty of backup disks and 30 amp fuses handy.

The maddeningly slow pace of the book is indicated by the fact that Murphy does not cover **login** until page 1207, at the end of Chapter 9. This could be a blessing in disguise, since it will postpone the many disasters in store for any innocent system and would-be user exposed to this dangerous volume.

Some historical background is undoubtedly useful for UNIX students, but one must question the need for a whole chapter on Ken Thompson's genealogy, 30 pages of PDP-7 timing charts, and the complete text of Pope Pius XIII's speech at the Beatification of Dennis Ritchie.

It is certainly false economy these days for the beginner to expect a decent UNIX book under the $1000 mark. Our recommended best buy is still Kernighan & Pike's *The UNIX Programming Environment* (Prentice-Hall), now in its 27th edition and still a steal at $1299.95.

Murphy dedicated his book to my sick mother — nice try there, Jack — but "no cigar".

Stan Kelly-Bootle has diluted his computer career by writing contemptuous folk songs for Judy Collins ("In My Life," Elektra K42009), The Dubliners and others. He is currently writing, with Bob Fowler, "The 6800 Primer" for the Waite Group, to be published by Howard W. Sams in the Spring of 1985. ■

THE HUMAN FACTOR

On the inevitability of UNIX

by Richard Morin

UNIX(tm) has been chosen as the operating system for our most advanced supercomputer, the CRAY-2.

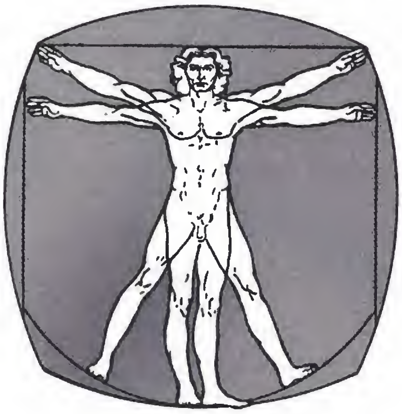
Cray Research, Inc. advertisement Computerworld (p. 117, July 16, 1984)

One often hears UNIX devotees heralding the imminent acceptance of UNIX as *the* standard operating system. Some even buy large amounts of advertising space to do so. The question they generally raise is *when*, rather than *whether* or even *how*.

This inevitability is a bit suspect, however. The proponents of other operating systems make similar claims, and we don't believe them. Why should we therefore consider UNIX to be chosen by the gods, let alone the entire future computer market?

Is UNIX likely to take over even a significant part of the market in the near future? It may in fact take over several markets, but this column will concentrate on the most probable one.

For the purpose of this discussion, let's ignore the use of UNIX as a basis for special purpose applications systems. These systems are built upon UNIX, but tend to hide it to such a degree that it is all but invisible to the user. Let's instead concentrate on the use of UNIX in all its unvarnished glory.



Next, let's forget high volume commercial data processors. UNIX is simply not well adapted to their needs. Further, the attempt to so adapt it is likely to produce a result unrecognizable as UNIX. In any case, the well known (and well justified) conservatism of commercial users will prevent them from immediately accepting UNIX as a standard.

UNIX does, however, have a good chance of taking over the scientific marketplace within the next few years. Scientists are less conservative than the commercial crowd and their need for UNIX is far greater. They won't even mind if UNIX is only a cult operating system; IBM systems haven't served their needs for years.

The scientific community is already quite interested in UNIX, and this may soon turn into acceptance. The reasons for this are the adaptability, communications capabilities, flexibility, performance and portability of UNIX systems. Let's examine these to see why each is of particular importance to scientists.

ADAPTABILITY

Scientists have a penchant for attaching weird devices to computers. Sometimes they even attach other computers. Most operating systems are significantly hostile to this sort of thing, if they allow it at all.

UNIX, on the other hand, is tolerant — even helpful. Do you need real time response? Extra processors? Exotic devices? Hack over here, fudge over there, and you have it.

The UC Berkeley Space Science Lab has a Sun workstation that has been expanded to over 20 backplane slots. Four additional single-board computers have been added, along with a raft of device controllers. A bit of hardware, a bit of software, and it's done.

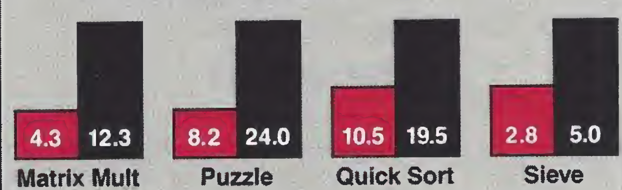
A more prosaic example is the UNIX output filter capability. This allows a UNIX user to insert a program between the line printer spooler and the printer driver. Canta Forda Computer Lab uses this to let its dot matrix printer do a variety of tricks, including bit image graphics.

Finally, note that the adapt-

The TARTAN C Compiler for VAX/UNIX. It's better than free.

TARTAN's C out-performs the portable C compiler you get free with UNIX™. With TARTAN C, you'll see: Faster execution speed, smaller-size object code, unsurpassed error diagnostics, and complete plug-compatibility with portable C. TARTAN C Compilers are part of a new generation of compilers that improve application performance and make programmers more productive.

TARTAN C Benchmarks



Standard benchmark tests* compare TARTAN C (in red) and Portable C compiler object code performance (measured in seconds) for various computational profiles.

TARTAN C Error Messages

C Program with Syntax Errors:

```
1  bubble(a) int a[100]; {int tmp,last,i;
2   for (last=100; last >= 2; last--)
3      {for (i=1; i <= (last-1; i++)
4          if (a[i] > a[i+1])
5             {tmp=a[i] a[i]=a[i+1];
6              a[i+1]=tmp;};
7      }
8  }
```

Portable C Compiler Error Messages:

```
"bubble.c", line 3: syntax error
"bubble.c", line 3: syntax error
"bubble.c", line 8: syntax error
```

TARTAN C accurately pinpoints errors and recovers:

```
    2|  for (last=100; last >= 2; last--)
    3|     {for (i=1; i <= (last-1; i++)
                                    ^1
*** 1 Error 101: Parse error; token ")" inserted.
    4|         if (a[i] > a[i+1])
    5|            {tmp=a[i] a[i]=a[i+1];
                            ^1
*** 1 Error 101: Parse error; token ";" inserted.
    6|             a[i+1]=tmp;};
```

Get More Performance Into Your Applications.

TARTAN compilers make your programs more competitive. With TARTAN C, you'll see:

- Optimized object code that makes your programs run up to three times faster
- Object code that's up to 55% smaller

Get More Productivity Out of Your Programmers.

TARTAN's object code is optimized by the compiler, not the programmer. Our advanced diagnostics catch errors the first time—in context and with complete, concise messages. With TARTAN C, you'll see:

- No more hand optimizing
- Fewer recompilations
- Error-free code before you know it
- Optional runtime array bounds checking

There's More.

With TARTAN C, you'll see responsive product support and detailed documentation. We also include a copy of the brand new *C: A Reference Manual* by TARTAN's Sam Harbison and Guy Steele Jr.

Call Today.
Order The C Compiler That's Better Than Free.

TARTAN C Compilers for VAX™ systems running Berkeley UNIX have successfully completed customer-site testing and are ready for immediate delivery. To place your order, call our Sales Department at **412-621-2212,** or write us at the address below.

The world's best-known C compiler is free. The world's best-performing C compiler isn't, but it's worth a lot more. The TARTAN C Compiler for VAX/UNIX.

We want to be your favorite software company.

TARTAN

TARTAN Laboratories Incorporated
477 Melwood Avenue
Pittsburgh, PA 15213

*A complete report on these and other benchmarks is available on request.
UNIX is a trademark of AT&T Bell Laboratories
VAX is a trademark of Digital Equipment Corporation.

Circle No. 5 on Inquiry Card

ability of UNIX is a natural consequence of its extreme modularity. The entire operating system is constructed of loosely connected pieces of code. Any piece may be removed and modified when necessary.

Although Kirk McKusick of UC Berkeley found replacing the entire file system of 4.2 BSD to be a tricky problem, it was not impossible. The rest of us may confine ourselves to hacking a few shell scripts and such, but the principle remains the same. The adaptability of UNIX is built into its very structure, a claim that very few operating systems can make.

COMMUNICATIONS

UNIX has a number of communications capabilities that could be of great use to scientists. The 4.2 BSD TCP/IP implementation provides a multivendor local area networking capability that does not depend on any particular physical network. A layered architecture, it can use whatever media happens to be available.

Adding the remote procedure call mechanism proposed by Sun Microsystems, this can become a full distributed file system. Scientists need workstations, file servers, number crunchers and a variety of other devices. The proposed protocol allows all of these to act as if they were integrated into a single monolithic computer system.

In UUCP, UNIX provides the ability to set up ad hoc networks providing electronic mail over vast distances. Both memos and documents may be sent quickly and economically in this manner. The cost and inconvenience of getting physical copies of scientific reprints makes the latter use very attractive.

The Usenet message passing system is an existing example of such a network. Usenet members send mail across the country and even into other countries. A hodge-podge of modems, computers and communications lines are used, but the user needn't care. The messages get through (in general), and there are no stamps to lick.

FLEXIBILITY

Commercial users tend to have reasonably stable data processing needs. Scientific users do not, since they never know what problem they'll be trying to solve next. Scientists thus demand more flexibility from their computer systems than do commercial users.

UNIX provides, if nothing else, an extraordinary degree of flexibility. It was designed with the idea that a user should be allowed to do anything and everything. Programs plug into other programs with great abandon, and existing utilities relieve much of the burden of development tasks.

Scientists often recognize the UNIX pipe and filter capability as a kind of composition of functions. Computer specialists may instead regard it as a primitive data flow language. In any case, it is ideally suited for use by scientists.

Scientists have an almost inexhaustible appetite for complex mathematical analysis routines. One can easily predict the production of a raft of Fourier transform filters, linear algebra utilities, continuous simulation systems, and such. With such utilities available as UNIX-style filters, scientists will be able to profitably use ever increasing amounts of computer power.

> **One often hears UNIX devotees heralding the imminent acceptance of UNIX as *the* standard operating system.**

PERFORMANCE

Scientists are notoriously profligate users of CPU cycles. Let us disregard, for the moment, the plasma physicists, meteorologists and such. Other scientists can easily bring a VAX to its knees by simply doing routine data analysis and display.

UNIX offers a range of solutions to this dilemma. For some scientists, a terminal attached to a shared computer is sufficient. For others, a garden variety UNIX workstation will provide the needed power. Some will need an added floating point processor or vector arithmetic board.

Finally, the real number crunching crowd will need still more power, up to and exceeding that available from a CRAY-2. In all cases, however, UNIX allows the needed hardware to be purchased and used. Multivendor networking even allows scientists to comfortably and invisibly share special purpose resources.

It is unreasonable to expect any vendor to meet the needs of all possible customers. IBM has stopped trying to do this, and nobody else even tried. No single vendor can begin to handle the range of needs of the scientific community.

Even a single scientist can easily outstrip the capabilities of a single vendor. A typical application might include massive

MODULAR.
INTEGRATED.
NOW.

Handle™ presents a modular software series for office automation.

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

Handle Office Automation Software Series product modules include:

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

AT&T 3B2/300 & 3B5

CONVERGENT TECHNOLOGIES MINIFRAME

DURANGO POPPY II

To be announced?

HANDLE TECHNOLOGIES, INC.

850 NORTH LAKE BOULEVARD / PO BOX 1913 / TAHOE CITY, CALIFORNIA 95730 / 916-581-5227 **Circle No. 6 on Inquiry Card**

number crunching, real time data acquisition, and color graphics output. Each of these functions demands radically different hardware.

A mixed network approach handles this by allowing a variety of computer systems, each optimized to a particular function, to work together. This optimization is brought about by cheap and powerful microprocessors and special purpose chips. It provides an economical solution to an otherwise intractable problem.

PORTABILITY

The issue of portability is important to all computer users since all hardware becomes obsolete in time. Another aspect of portability, of course, allows the use of software across a mixed network as described above. Scientists are particularly concerned, however, because they tend to swap software freely.

Commercial users don't often call up other users to get a new accounting routine. Scientists, on the other hand, are always trading software. The popularity of Fortran among scientists is largely due to its portability and modularity, which promote this interchange.

A Fortran application is often constructed largely out of calls to library functions and subroutines. These may have been written 15 years before for an entirely different computer and application. The scientist doesn't care, as long as the applications still function properly.

All too often, however, they do not. Differences in system calls, I/O standards, and other trivia conspire to "break" routines on new machines. The adoption of UNIX, by providing standard interfaces for a diverse range of hardware, largely solves this problem.

Even if routines work individually, they may not function well together. Most operating systems have no convenient way of pushing data through sequences of programs. UNIX does, of course, and it even has a folklore which encourages programming for this manner of use.

THE GESTALT

Scientific institutions will not convert to UNIX instantaneously. Instead, they will build up to full scale use in an incremental manner, often project-by-project. In many scientific institutions, each project selects and purchases its own computer systems.

These systems will be installed, used, expanded, networked and modified. The different characteristics discussed above will be brought into play. Systems which lack any critical capability will be at a disadvantage, and a form of natural selection will take place.

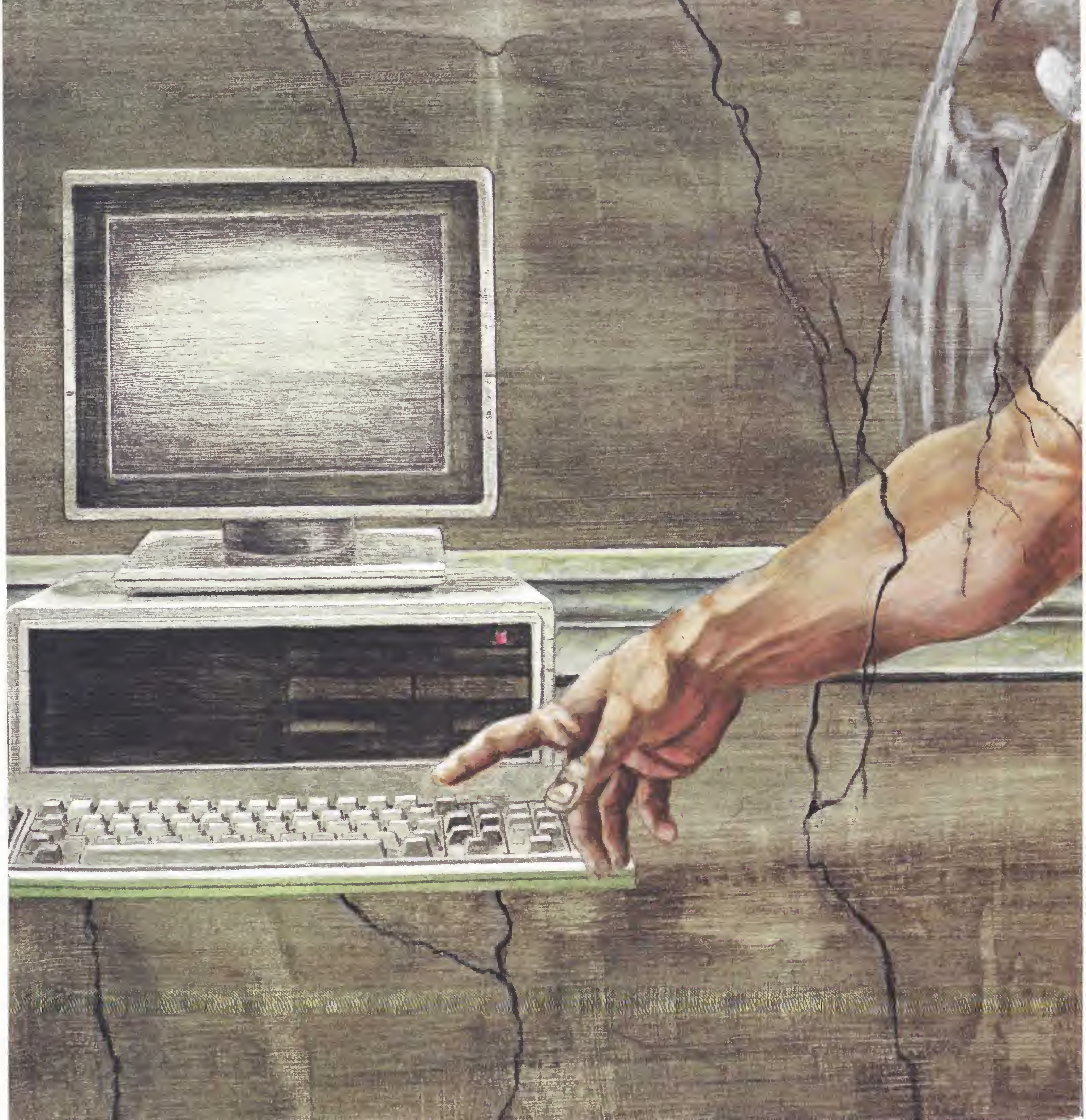
Scientific projects are increasingly dependent on the effectiveness of their computer support. Scientists take careful note of the comparative utility of systems they see being used. UNIX, with its powerful mix of capabilities, is thus certain to win an ever increasing share of the scientific marketplace.

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 Walnut Creek, CA. ■

THE GENESIS STORY

An unofficial, irreverent, incomplete account of how the UNIX operating system developed

by August Mohr



This is, so to speak, a history of how UNIX evolved as a product; not the "official" history of who was responsible for what features, and what year which milestones were crossed, but the "political" history of how decisions were made and what motivated the people involved. Most of the readers of this magazine are familiar with the system itself, so I don't want to go into great detail about how the system got to be what it is internally, but rather how it got to be at all.

Three years ago, I was one of many espousing the idea that Bell Laboratories and Western Electric had engaged in a masterful piece of long-term strategy by first releasing UNIX to universities and then later releasing it to the commercial world. How clever, I thought, to get universities involved in the development of the system and get a whole crop of UNIX experts trained in the process. But after talking to some of the people involved, I have reversed my opinion. Bell, it seems, was dragged kicking and screaming into providing UNIX to the world.

Illustration by Kelly Davies

IN THE BEGINNING, THERE WAS MULTICS

Multics, in some ways UNIX's predecessor, was a huge project, a combined effort of three of the largest computing centers of the day. All three principles — Bell, GE, and MIT, had already done operating systems, even *time-shared* systems, before, and following Fred Brooks' *Second System Syndrome*, it was felt Multics was going to solve all the problems of its predecessors. Ultimately, the project proved to be too late and Bell dropped out, leaving Ken Thompson, Dennis Ritchie, and Rudd Canaday without a timeshared system to play with.

So they set about building an operating system of their own.

So until this point, time-sharing systems had only been developed for big machines costing hundreds of thousands of dollars. It was unlikely that Computer Research was going to get another investment of that sort out of Bell so soon after Multics. In any event, Thompson, Ritchie, and Canaday weren't particularly interested in working on another large scale project, so they set their sights on obtaining a minicomputer on which to build a timesharing system they could use as a program development environment. They wanted the kind of flexibility and power they had worked toward in Multics, without incurring the expensive rings of protection and interlocks.

The project was short on computing power, though. With only a PDP-7 to work on, the researchers yearned for another machine — preferably a VAX 11/20. To get the necessary funds for a new machine, though, it was clear they would first need to find an application.

Fortunately, the Bell Labs' Legal Department — which was close at hand to Thompson and Ritchie's Murray Hill office — was looking for a word processing system at about this time. A paper-tape system for an old Teletype machine was under consideration.

With a bit of salesmanship, Thompson and Ritchie got the legal team interested in a UNIX-supported system. It proved to be a momentous deal. The UNIX project got the machine it needed and the legal department got the word processing system it wanted — along with some rather impressive local support.

The combination of program development and word processing was to have serendipitous effects.

A number of important barriers thus were crashed. First, the business side of Bell Labs, generally held at arm's length by the people in Research, got a healthy dose of Research assistance. More importantly for UNIX users, Thompson and Ritchie got their first live customer. UNIX word processing suddenly had to be usable by secretarial staff as well as by programming staff. The changes made to accomplish this transition were to have serendipitous effects as UNIX evolved.

Richard Haight, now AT&T Bell Labs Supervisor of Video Systems Software (a UNIX application), had his first exposure to UNIX at about this time. As he remembers it, "I came across Ken Thompson by accident and wanted to do some software development on a PDP-11. He said he had a machine I could use but that it didn't run on a DEC operating system. I had lots of exposure to timesharing systems and it was pretty obvious that this was superior in many ways.

"It was uphill in the beginning. The fact that it was homegrown in the Labs didn't cut anything with the people that I was working with at that time. They went and visited Ken and Dennis in their sixth-floor attic at Murray Hill and all they saw was hardware laying all over the floor and a bunch of guys in T-shirts and sneakers. It wasn't the sort of place that would warm the heart of a manager who had been brought up in a traditional data processing environment. It really was the 'two good guys in a garage' kind of syndrome except that they happened to be in an attic.

"That first time I met Ken Thompson, he had a small bookshelf, maybe 2 1/2 feet long, above an old Teletype Model 37 terminal. On the shelf were hardware manuals from Digital Equipment for the PDP-11, occupying maybe five inches, and the rest of the shelf was nothing but chess books. I think Thompson will tell you himself that he developed UNIX as a good place to develop chess programs. It turns out that it's everybody else's idea of where and how to develop programs too."

Haight has had many years of experience in the Bell Labs environment. Before moving to his current post, he served as part of the UNIX Support group and the PWB (Programmer's Workbench) development group. He believes that the environment

EMERALD ONE™

SOFTWARE WITH COURAGE, BRAINS AND HEART

WHAT IS EMERALD ONE?

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

THE TOOLS OF EMERALD ONE

EMERALD ONE integrates your office tasks through:

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

SOFTWARE FOR THE WORK GROUP

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

EMERALD CITY, THE PEOPLE BEHIND EMERALD ONE

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

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

EMERALD C · I · T · Y

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

See us at UniForum Booth # 3038

Circle No. 9 on Inquiry Card

of the Labs was a major factor in the creation of UNIX.

"There's a small fraction of people who just go crazy over computers," he explained. We hire a bunch of them and they remain workaholics for several years after coming in. Eventually they get a house and a mortgage and they get married and have kids and settle down to be normal human beings, but it's wonderful to hire these people because you get two or three people's work out of them for several years. Those are the kind of people that give you things like UNIX."

AN INFLUENTIAL FEATURE: PIPES

Dick Haight is one of the people who became hooked on UNIX at the very beginning, and even yet is an outspoken proponent of its value and power — although he will tell you he is still waiting for something better to come along. "There's a lot of complaint about UNIX being terse and for experts only," he said. "I've seen that blamed on the pipe mechanism. If terseness is the price for having pipes, I'll take it any day."

As one of the new system's first users, Haight got to follow many of the changes. "I happened to have been visiting the research crew the day they implemented pipes," he recalled. It was clear to everyone, practically minutes after the system came up with pipes working, that it was a wonderful thing. Nobody would ever go back and give that up if they could help it."

Haight believes that the pipe facility has had a major effect on the evolution of UNIX. Because of the shell and the piping facility, programs running under UNIX can be conversational without being explicitly programmed that way. This is because piping adds to the shell's ability to accomplish this purpose for the system as a whole.

Doug McIlroy is usually credited with the idea of adding pipes, although the idea may have been around since Multics. Haight believes there may have been yet another reason for implementing them. The original UNIX had a file size limitation of 64K and, according to Haight, "one of the people there in research was constantly blowing that size restriction in an intermediate pass of the Assembler." Between McIlroy's lobbying for the idea and this other problem with file size, Thompson and Ritchie were finally convinced to implement pipes.

MOVING INTO THE COMPANY

Independent of what was happening in the research area, Bell was starting to perceive the need for minicomputer support for its telephone operations. It needed Operations Systems, not Operating Systems. With the numbers of systems under consideration, the possibility of being tied to a single vendor, or having each site tied to a different vendor, induced a kind of paranoia. There just had to be another way.

"Naturally I knew that once they got on UNIX they wouldn't be able to get off. It's just like drugs."

The groups responsible for developing operations systems had many people from hardware and applications software backgrounds who were considering writing their own operating system — their first — when Berkley Tague, now head of Bell Labs' Computing Technology Department, suggested they use UNIX to get started.

"I observed that people were starting to put these minis out in the operating company, and saw that it was an area of both opportunity and potential problems," Tague remembered. "I found that some of the people in development had never built an operating system for any computer before; many of them had very little software background. They were coming out of hardware development and telephone technology backgrounds, and yet were starting to build their own operating systems. Having been through that phase of the business myself, it seemed silly to go through it another hundred times, so I started pushing the UNIX operating system into these projects."

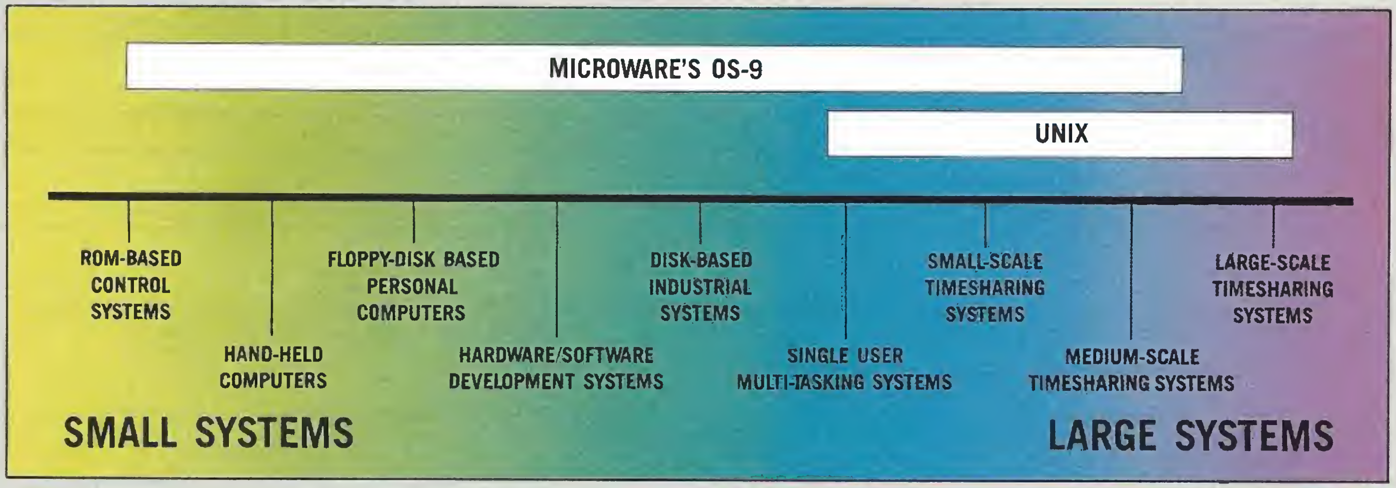
Tague's backing of UNIX, as a development system for operations, was not just a personal preference. "I had every confidence in the people who built it because I'd worked with them on Multics," he explained. "With their experience and training, I figured they could build a much better operating system than somebody who's building one for the first time, no matter how smart that person is."

SUPPORT?

UNIX had been running long enough in research by that time that Tague knew that the system the operations group would get would serve as a very good starting point. Unfortunately, there was no vendor support for it.

The argument Tague made for UNIX was: if the operations people were going to build their

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.

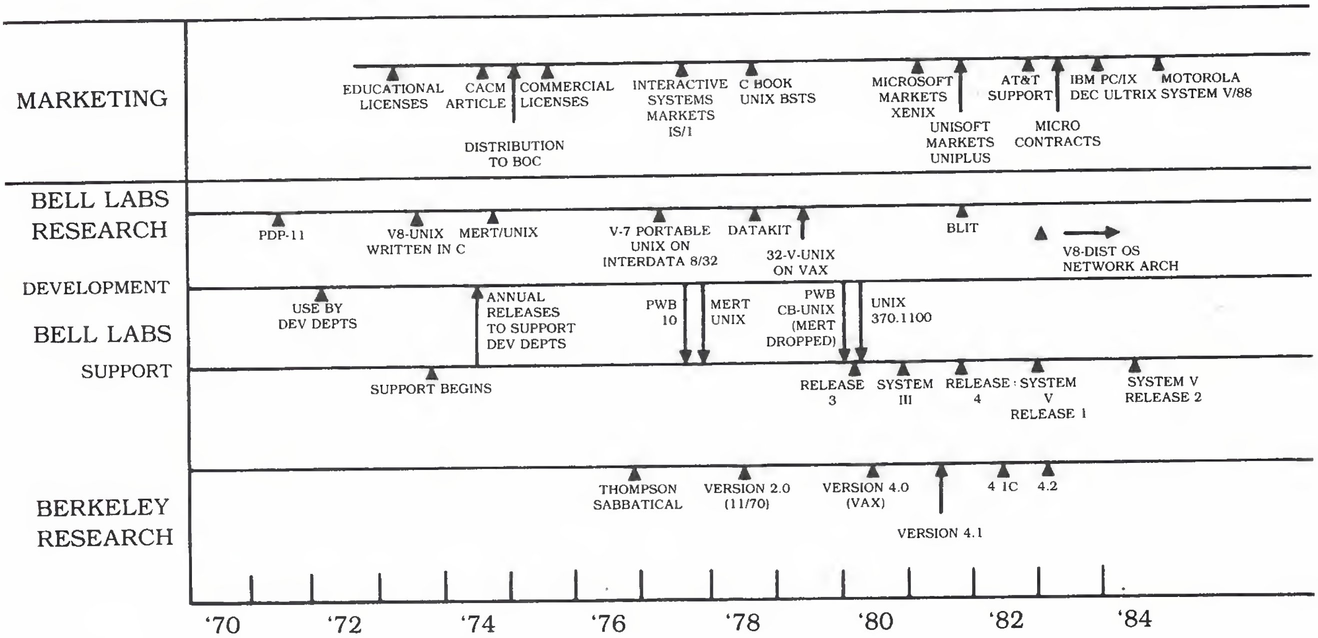
microware®
OS-9™

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

Circle No. 10 on Inquiry Card

KEY EVENTS IN UNIX SYSTEM EVOLUTION



Courtesy of Larry Crume, AT&T UNIX Pacific.

own system, they were going to have to maintain it themselves. Surely, UNIX could be no worse. They could use it to get started and do the development. If a more efficient or better operating system was needed for a target machine when they got into the field, they could always build it, but UNIX would at least get them off the ground. "Naturally I knew that once they got on this thing they wouldn't be able to get off. It's just like drugs," Tague explained.

Tague also knew it was important to get some field support. "We were starting to put these things in the operating companies all around the countryside and the prospects were that there were going to be several hundred minis over the next few years that were going to have to be maintained with all their software and hardware," he said.

Bell had already gained some field support experience maintaining electronic switching machines and their software. Supporting a network of minicomputers would be a significantly different problem, though. Maintaining an operating system is not at all like maintaining an electronic switching system. The minicomputers had different reliability demands, requiring a different support structure in the organization — one that did not yet exist in any form. In many ways, the operations group was breaking new ground.

Up to this point, Tague had served as head of the Computer Planning department responsible for systems engineering. After gaining support for UNIX in the operations group over the course of 1971 and 1972, he made a push for two significant changes. The first was to make UNIX an internal standard and the second was to offer central support through his organization. In September, 1973, he was permitted to form a group called UNIX Development Support, the first development organization supporting a "Standard UNIX". While this group worked closely with Bell Labs Research, its concerns sometimes diverged.

One area the two groups could agree on, though, was portability. By 1973, it was already on the horizon. Tague foresaw the possibility of UNIX becoming an interface between hardware and software that would allow applications to keep running while the hardware underneath was changing.

From the support point of view, such a capability would solve a very important problem. Without UNIX and its potential portability, the people building the operations support systems were faced with selecting an outside vendor that could supply the hardware on which to get their development done. Once that was complete, they would be locked into that vendor. Portability obviated this limitation and offered a number of other advantages. When making a hardware upgrade, even to equipment from the same vendor, there are variations from

version to version. That could cost a lot of money in software revisions unless there were some level of portability already written into the scenario. Fortunately, the integral portability of the system developed by Research proved adequate to make UNIX portable over a wide range of hardware.

The first UNIX applications were installed in 1973 on a system involved in updating directory information and intercepting calls to numbers that had been changed. The automatic intercept system was delivered for use on early PDP-11s. This was essentially the first time UNIX was used to support an actual, ongoing operating business.

To Tague, at this time, "our real problem was pruning the tree". There were so many different sites using UNIX that each would come up with different answers to the same problems of printer spooling, mail, help, and so on. "The customers would invent this stuff and make it work and our problem was to get the slightly different variations together, get the best of all of those worlds, put it in the standard, and get it out again." This was, in many ways, a political process. Tague credits the "technical underpinnings" for making the process easier than expected. "That made it easy to get the right stuff in without upsetting the whole world. I didn't have to go to all of my customers and tell them that this was now my new version and that nothing they had out in the field would run again," he said.

To provide a standard UNIX system, the support group had to establish what version it would back. This was a process of negotiation and compromise with the UNIX-using community — not a unilateral decision. The support team and customers often ended up arguing things out until everybody understood the issues and a suitable compromise was made. "As one of the local gurus put it to me one time," Tague said, "one of the problems in UNIX is that everybody wants to carve his initials in the tree." When the choice came down essentially to

From the very beginning within Bell, UNIX followed what has become a familiar pattern of users leading their management.

tossing a coin, Tague, as arbitrator, tried to make sure that each group got at least one pet contribution into the system.

Fortunately, UNIX is flexible enough that even the particularly traumatic decisions, such as the ones concerning standard shell versions, could be patched in slowly — at the user's discretion.

A FAMILIAR PATTERN

From the very beginning within Bell, UNIX followed what has become a familiar pattern of users leading their management. While this is not the most common marketing strategy in the commercial world, it is typical of Bell Labs' "bottom-up" organization. According to Rudd Canaday, now head of Bell Labs' Artificial Intelligence and Computing Environments Research Department, change within the Labs often comes from the people doing the work. "UNIX spread throughout Bell Laboratories because people loved to use it," he said.

Canaday first experienced UNIX, although it hadn't yet been named, while working with Thompson and Ritchie. He left that group and later became head of a group building large, mainframe-oriented systems. Because of his previous exposure to UNIX, he wanted to bring it to his new group.

Canaday found lots of support among the programmers who had already tasted UNIX. One of those, Richard Haight, recalled, "I was trying to get my management to get UNIX and we dreamed up the idea of using it as a common timesharing interface to different kinds of host computers."

Initially, the project involved interfacing with big IBM and Univac machines, and later expanded to interfacing with RCA, Xerox, and others. The basic idea was to edit programs and work with files under UNIX, but instead of compiling and executing under UNIX, you could send the remote job off to a big machine. This way, the programmer didn't have to deal with complicated IBM JCL sequences since he could just give the UNIX utility the parameters it needed to know. The masses of printout could then come back as a file under UNIX and, as Dick Haight put it, "save cutting down a tree." It also saved having to retrain programmers for a variety of host systems.

This original Programmer's Workbench system was built on a PDP 11/45. The system eventually offered lots of utilities, including ones for analyzing host machine dumps on the UNIX system.

While work proceeded on the PWB system, an interesting discovery was made. The designers had assumed that the majority of the work cycle would involve the host computer. Users were thought of as editing a file, sending it to a

Your schedule needn't be carved in stone.

When a decree comes down from above, your scheduling program better be flexible enough to adapt easily and efficiently. With any project, delays can occur, shipments can be late, specific elements of the job can take longer than expected, but with MicroTrak, changing one deadline won't smash your schedule. Other activities affected by the change will be adjusted accordingly. And *you* won't be left between a rock and a hard place.

FEATURES

Updates Quickly
True Precedence Based Scheduling
Menu Driven, Interactive
Overlapping Relationships
Clear, Useful Reports

MicroTrak available on PC-DOS, MS-DOS ($595.00) and XENIX ($895.00) operating systems.
30-day trial with full refund if not satisfied.

MICROTRAK™

APPLICATION SOFTWARE FOR PROJECT MANAGEMENT

SOFTRAK SYSTEMS *P.O. Box 22156 AMF Salt Lake City, Utah 84122 (801)531-8550*

See us at UniForum Booth # 3311

Circle No. 12 on Inquiry Card

host computer, getting the print file back, looking at it, and doing that over and over again. As it turned out, samples taken from different kinds of work groups on different systems showed people tended to use the text formatter, **nroff**, five times as often as they submitted Remote Job Entry programs.

This unexpected result might not have happened had UNIX not had fairly sophisticated word processing facilities available to programmers. The original development for Bell's legal department could hardly be called "incredible foresight", but happily for UNIX, word processing was to become the single most commonly used computer application. Once the facilities were there, programmers made massive, unexpected use of them. This happened, according to Haight, because programmers have to be able to document programs on the same machine used for development. "Things like pipes and the power of the shell are not to be slighted, but what's really important is the fact that you can do your documentation and your programming on the same machine," he said. You can be editing your documentation and break away from that to edit the source. When you're finished with that, you can submit a compile in the background and go back to editing your documentation while the compile happens."

Flushed with the success of the PWB and the Remote Job Entry facility, Canaday and his group set about showing people what was possible. Once the users were convinced, Canaday said to management, "Well, if you want to keep on using this, you're going to have to start buying machines to do it." He knew that "once you let people get their hands on UNIX, they just won't let go."

A key piece in the rapid spread of UNIX within Bell Labs was the low price of minicomputers relative to mainframes. A department head's urging was generally sufficient for purchase of a VAX. Mainframe purchases were considerably more sticky. A VAX had sufficient power to reasonably serve the needs of a department, so VAXen became increasingly commonplace.

More and more departments were becoming convinced that UNIX was part of the path toward

Continued to Page 117

REFLECTIONS ON SOFTWARE RESEARCH

Can the circumstances that nurtured the UNIX project be produced again?

by Dennis M. Ritchie

The UNIX operating system has suddenly become news, but it is not new. It began in 1969 when Ken Thompson discovered a little-used PDP-7 computer and set out to fashion a computing environment that he liked. His work soon attracted me; I joined in the enterprise, though most of the ideas, and most of the work for that matter, were his. Before long, others from our group in the research area of AT&T Bell Laboratories were using the system: Joe Ossanna, Doug McIlroy, and Bob Morris were especially enthusiastic critics and contributors. In 1971, we acquired a PDP-11, and by the end of that year we were supporting our first real users: three typists entering patent applications. In 1973, the system was rewritten in the C language, and in that year, too, it was first described publicly at the Operating Systems Principles conference; the resulting paper [8] appeared in *Communications of the ACM* the next year.

Thereafter, its use grew steadily, both inside and outside of Bell Laboratories. A development group was established to support projects inside the company, and several research versions were licensed for outside use.

The last research distribution was the seventh edition system, which appeared in 1979; more recently, AT&T began to market System III, and now offers System

Continued to Page 118

The microSystem NX from Honeywell. Quite possibly the most cost-efficient, full-featured UNIX*-based workstation.

Power, versatility, networking, and performance are all a part of the microSystem NX UNIX-based workstation from Honeywell. Featuring the MC 68000** processor, the NX is a super-micro engineered to compete with larger, more expensive multi-user systems.

The Professional's Workstation

microSystem NX is designed for engineers and other professionals who require sophisticated applications and a powerful tool for software application design and development, documentation support and technical research support.

Extensive System Development Tools

microSystem NX features a rich set of development tools that includes compilers, linkers, editors, formatters and debuggers. In addition, the system provides a "C" compiler, font editor, Fortran 77, and Pascal for enhancing commercial software packages and developing customized applications.

Powerful Applications Software

microSystem NX also provides application packages which take full advantage of the power of UNIX and include: a menu processor; a highly versatile word processor; extensive spreadsheet capabilities; a 3-dimensional graphics package; and a window manager that allows you to run up to 6 applications concurrently.

Advanced Networking

microSystem NX incorporates an advanced networking architecture that employs a local area network to provide a versatile UNIX-to-UNIX communications link between workstations.

UNIX-based Operating System

microSystem NX is driven by an enhanced version of the UNIPLUS+ operating system, with a complete set of utilities, development tools, networking options, and the ability to support a wide range of 3rd party applications software.

The new Honeywell microSystem NX. A sophisticated hardware/software system that gives the professional computer user more computer power for less money.

For complete information call 1-800-328-5111 ext. 2743 (in Minnesota call collect 612-870-2142 ext. 2743) or write: Honeywell Information Systems, Inc. MS 810, 300 Concord Road, Billerica, MA 01821.

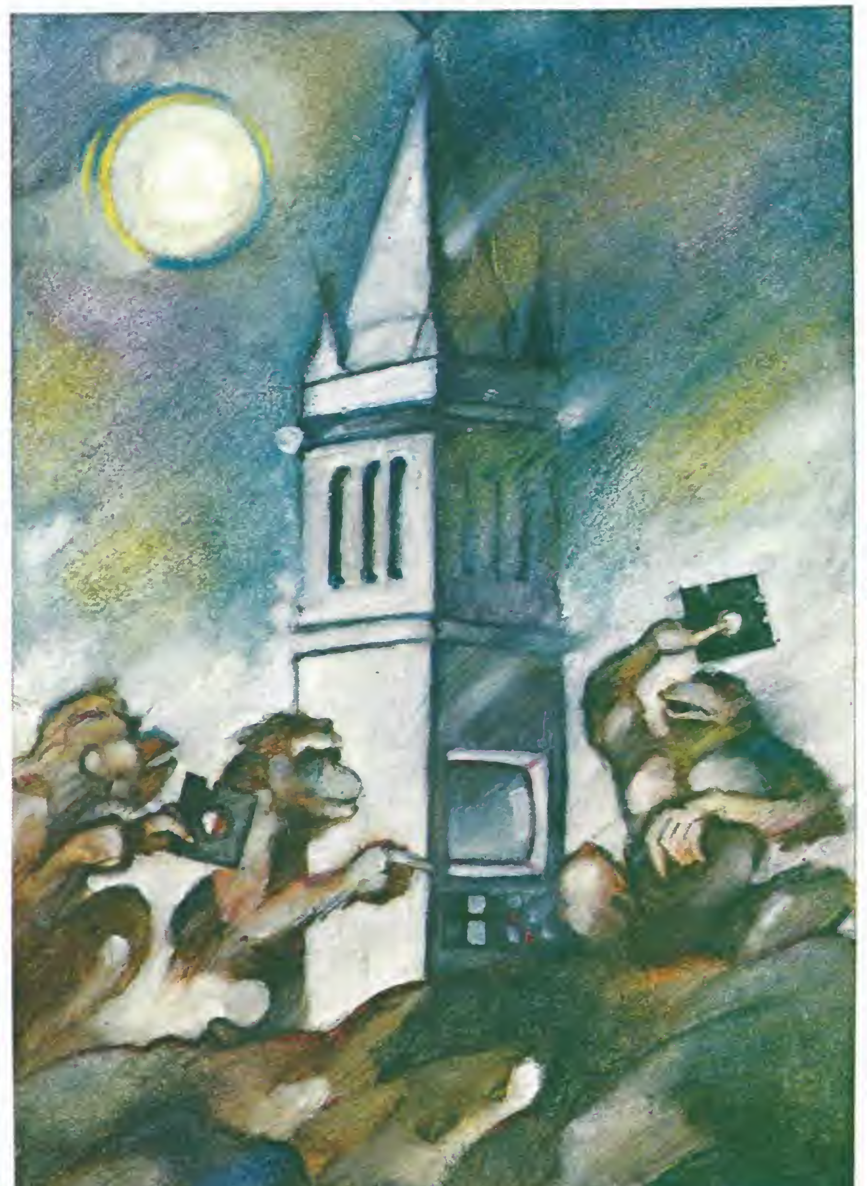
Together, we can find the answers.

Circle No. 13 on Inquiry Card

Honeywell

See us at UniForum Booth # 2160

*Trademark of Bell Laboratories
**Trademark of Motorola Inc.



A BERKELEY ODYSSEY

Ten years of BSD history

by Marshall Kirk McKusick

Ken Thompson and Dennis Ritchie presented the first UNIX paper at the Symposium on Operating Systems Principles at Purdue University in November, 1973. Professor Bob Fabry was in attendance and immediately became interested in obtaining a copy of the system to experiment with at Berkeley.

At the time, Berkeley had only large mainframe computer systems doing batch processing, so the first order of business was to get a PDP-11/45 suitable for running the then current Version 4 of UNIX. The Computer Science Department, together with the Mathematics Department and the Statistics Department were able to jointly purchase a PDP-11/45. In January, 1974, a Version 4 tape was delivered and UNIX was installed by graduate student Keith Standiford.

Although Ken Thompson was not involved in the installation — as he had been for most systems up to that time — his expertise was soon needed to determine the cause of several strange system crashes. Because Berkeley had only a 300 baud acoustic-coupled modem without auto answer capability, Thompson would call Standiford in the machine room and have him insert the phone into the modem; in this way Thompson was able to remotely debug crash dumps from New Jersey.

Illustration by Stephen G. Luker

Many of the crashes were caused by the disk controller's inability to reliably do overlapped seeks, contrary to the documentation. Berkeley's 11/45 was among the first systems that Thompson had encountered that had two disks on the same controller! Thompson's remote debugging was the first example of the cooperation that sprang up between Berkeley and Bell Labs. The willingness of the researchers at the Labs to share their work with Berkeley was instrumental in the rapid improvement of the software available at Berkeley.

Though UNIX was soon reliably up and running, the coalition of Computer Science, Mathematics, and Statistics began to run into problems; Math and Statistics wanted to run DEC's RSTS system. After much debate, a compromise was reached in which each department would get an eight-hour shift; UNIX would run for eight hours followed by 16 hours of RSTS. To promote fairness, the time slices were rotated each day. Thus UNIX ran 8 am to 4 pm one day, 4 pm to midnight the next day, and midnight to 8 am the third day. Despite the bizarre schedule, students taking the Operating Systems course preferred to do their projects on UNIX rather than on the batch machine.

Professors Eugene Wong and Michael Stonebraker were both stymied by the confinements of the batch environment, so their Ingres database project was among the first groups to move from the batch machines to the interactive environment provided by UNIX. They quickly found the shortage of machine time and the odd hours on the 11/45 intolerable, so in the Spring of 1974, they purchased an 11/40 running the newly available Version 5. With their first distribution of Ingres in the Fall of 1974, the Ingres project became the first group in the Computer Science department to distribute its software. Several hundred Ingres tapes were shipped over the next six years, helping to establish Berkeley's reputation for designing and building real systems.

Even with the departure of the Ingres project from the 11/45, there was still insufficient time available for the remaining students. To alleviate the shortage, Professors Michael Stonebraker and Bob Fabry set out in June, 1974, to get two instructional 11/45s for the Computer Science department's own use. Early in 1975, the money was obtained. At nearly the same time DEC announced the 11/70, a machine that appeared to be much superior to the 11/45. Money for the two 11/45s was pooled to buy a single 11/70 that arrived in the Fall of 1975. Coincident with the arrival of the 11/70, Ken Thompson decided to take a one-year sabbatical as a visiting professor at his alma mater. Thompson, together with Jeff Schriebman and Bob Kridle, brought up the latest UNIX, Version 6, on the newly installed 11/70.

Arriving in the Fall of 1975 were two unnoticed graduate students, Bill Joy and Chuck Haley.

Also arriving in the Fall of 1975, were two unnoticed graduate students, Bill Joy and Chuck Haley; they both took an immediate interest in the new system. Initially they began working on a Pascal system that Thompson had hacked together while hanging around the 11/70 machine room. They expanded and improved the interpreter system to the point that it became the programming system of choice for students because of its excellent error recovery scheme and fast compile and execute time.

With the replacement of Model 33 teletypes by ADM-3 screen terminals, Joy and Haley began to feel stymied by the constraints of the **ed** editor. Working from an editor named **em** that they had obtained from Professor George Coulouris at Queen Mary's College in London, they worked to produce the line-at-a-time editor **ex**.

With Ken Thompson's departure at the end of the Summer of 1976, Joy and Haley begin to take an interest in exploring the internals of the UNIX kernel. Under Schriebman's watchful eye, they first installed the fixes and improvements provided on the "fifty changes" tape from Bell Labs. Having learned to maneuver through the source code, they suggested several small enhancements to streamline certain kernel bottlenecks.

FIRST DISTRIBUTION

Meanwhile, interest in the error recovery work in the Pascal compiler brought in requests for copies of the system. Early in 1977, Joy put together the "Berkeley Software Distribution". This first distribution included the Pascal system and — in an obscure subdirectory of the Pascal source — the editor **ex**. Over the next year, Joy, acting in the capacity of distribution secretary, sent out about 30 free copies of the system.

With the arrival of some

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. 14 on Inquiry Card

See us at UniForum Booth #2008

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

ADM-3a terminals offering screen-addressable cursors, Joy was finally able to write **vi**, bringing screen-based editing to Berkeley. He soon found himself in a quandary. As is frequently the case in universities strapped for money, old equipment is never replaced all at once. Rather than support code for optimizing the updating of several different terminals, he decided to consolidate the screen management by using a small interpreter to redraw the screen. This interpreter was driven by a description of the terminal characteristics, thus spawning the now famous **termcap**.

By mid-1978, the software distribution clearly needed to be updated. The Pascal system had been made markedly more robust through feedback from its expanding user community, and had been split into two passes so that it could be run on PDP-11/34s. The result of the update was the "Second Berkeley Software Distribution" that was quickly shortened to *2 BSD*. Along with the enhanced Pascal system, **vi** and **termcap** for several terminals was included. Once again, Bill Joy single-handedly put together distributions, answered the phone, and incorporated user feedback into the system. Over the next year, nearly 75 tapes were shipped. Though Joy moved on to other projects the following year, the 2 BSD distribution continued to expand. Today the latest version of this distribution, 2.9 BSD, is a complete system for PDP-11s.

VAX UNIX

Early in 1978, Professor Richard Fateman began looking for a machine with a larger address space that he could use to continue his work on Macsyma that had started on a PDP-10. The newly announced VAX-11/780 seemed to fulfill the requirements and was available within budget. Fateman and 13 other faculty members put together an NSF proposal that they combined with some departmental funds to purchase a VAX.

Initially the VAX ran DEC's operating system VMS, but the department had gotten used to the UNIX environment and wanted to continue using it. So, shortly after the arrival of the VAX, Fateman obtained a copy of the 32/V port of UNIX to the VAX by John Reiser and Tom London of Bell Labs.

Initially the VAX ran DEC's operating system VMS, but the department had gotten used to the UNIX environment and wanted to continue using it.

Although 32/V provided a Version 7 UNIX environment on the VAX, it did not take advantage of the virtual memory capability of the VAX hardware. Like its predecessors on the PDP-11, it was entirely a swap-based system. For the Macsyma group at Berkeley, the lack of virtual memory meant that the process address space was limited by the size of the physical memory, initially 1 MB on the new VAX.

To alleviate this problem, Fateman approached Professor Domenico Ferrari, a member of the systems faculty at Berkeley, to investigate the possibility of having his group write a virtual memory system for UNIX. Ozalp Babaoglu, one of Ferrari's students, set about to find some way of implementing a working set paging system on the VAX; his task was complicated because the VAX lacked reference bits.

As Babaoglu neared the completion of his first cut at an implementation, he approached Bill Joy for some help in understanding the intricacies of the UNIX kernel. Intrigued by Babaoglu's approach, Joy joined in helping to integrate the code into 32/V and then with the ensuing debugging.

Unfortunately, Berkeley had only a single VAX for both system development and general production use. Thus for several weeks, the tolerant user community alternately found themselves logging into 32/V and "Virtual VAX/UNIX". Often their work on the latter system would come to an abrupt halt, followed several minutes later by a 32/V login prompt. By January of 1979, most of the bugs had been worked out, and 32/V had been relegated to history.

Joy saw that the 32-bit VAX would soon obsolete the 16-bit PDP-11 and began to port the 2 BSD software to the VAX. While Peter Kessler and I ported the Pascal system, Joy ported the editors **ex** and **vi**, the C shell, and the myriad other smaller programs on 2 BSD. By the end of 1979, a complete distribution had been put together. This distribution included the virtual memory kernel, the standard 32/V utilities, and the additions from 2 BSD. In December, 1979, Joy shipped the first of nearly a hundred copies of 3 BSD, the first VAX distribution from Berkeley.

DARPA SUPPORT

Meanwhile, in the offices of the planners for the Defense Advanced Research Projects Agency, COSI

C O M M U N I Q U É™

Use Communiqué 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 Communiqué for:

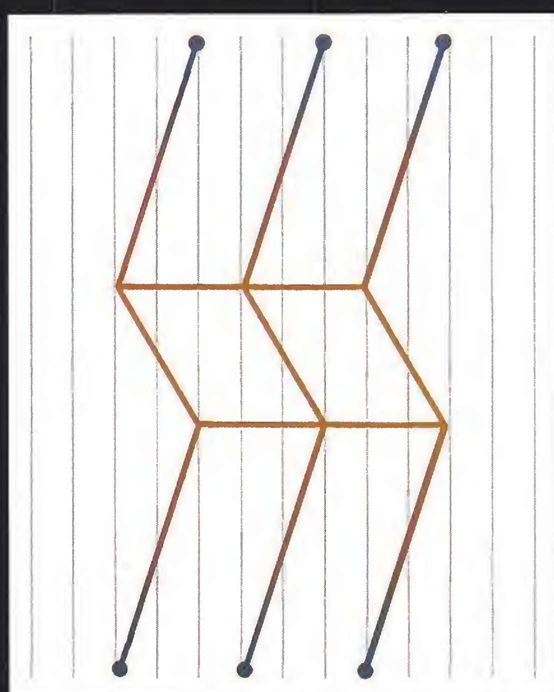
- Execution of DOS programs on the PC under UNIX control.
- Simple elegant file transfer under error correcting protocol.
- DEC, IBM, and Tektronix (graphics) terminal emulation.

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

See us at Uniforum.

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

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



The PC-to-UNIX™ Connection

Circle No. 15 on Inquiry Card

DARPA, discussions were being held that would have a major influence on the work at Berkeley. One of DARPA's early successes had been to set up a nationwide computer network to link together all its major research centers. At that time, DARPA was finding that many of the computers at these centers were reaching the end of their useful lifetime and had to be replaced. The heaviest cost of replacement was the porting of the research software to the new machines. In addition, many sites were unable to share their software because of the diversity of hardware and operating systems.

Choosing a single hardware vendor was impractical because of the widely varying computing needs of the research groups and the undesirability of depending on a single manufacturer. Thus, the planners at DARPA decided that the best solution was to unify at the operating systems level. After much discussion, UNIX was chosen as a standard because of its proven portability.

In the Fall of 1979, Bob Fabry responded to DARPA's interest in moving towards UNIX by writing a proposal suggesting that Berkeley develop an enhanced version of 3 BSD for the use of the DARPA community. Fabry took a copy of his proposal to a meeting of DARPA image processing and VLSI contractors, plus representatives from Bolt, Beranek, and Newman, the developers of the ARPAnet. There was some reservation whether Berkeley could produce a working system, but the release of 3 BSD in December, 1979, assuaged most of the doubts.

With the increasingly good reputation of the 3 BSD release to validate his claims, Bob Fabry was able to land an 18-month contract with DARPA beginning in April, 1980. This contract was to add features needed by the DARPA contractors. He immediately hired Laura Tong to handle the project administration. With the negotiations for the contract on track, Fabry turned his attention to finding a project leader to manage the software development. Fabry had assumed that since Joy had just passed his Ph.D. qualifying examination, he would rather concentrate on completing his degree than assume the software development position. But Joy had other plans. One night in early March he

One billion byte supermicro. Just think of the possibilities.

Introducing the new, expandable Dual 83/500. A UNIX*-based, 68000-driven supermicro so capable, you'd swear it was a mainframe.

The system already comes with 500 megabytes of Winchester storage. And our patent-pending high-speed SMD disk controller for fast access to data.

But you can increase memory to a massive one billion bytes just by adding a twin drive.

Or take an already sizeable two megabytes of RAM and expand it to six.

Or even double user capacity from 8 to 16. The hardware is already in place.

When it comes to value, no supermicro system offers you more than the 83/500. Because along with the computer, you get a 9-track, 1600 BPI phase-encoded tape drive for reliable disk backup and quick file transfers to other systems.

There's a convenient one megabyte double-sided/double-density floppy disk drive that protects individual files.

And the industry standard UniPlus+™ implementation of AT&T's UNIX System V with Berkeley enhancements. Plus a multi-user license.

All at no extra cost.

And while you're speedily going about processing your data, we're protecting your investment. For free. With a comprehensive 12-month warranty. And a nationwide service network that protects your system whether it's in or out of warranty.

See the system that's redefining the supermicro. The value-packed Dual 83/500. Call or write Dual Systems Corporation, 2530 San Pablo Avenue, Berkeley, CA 94702, (415) 549-3854

At just $65,940 base price, its possibilities are endless.

*UNIX is a trademark of AT&T Bell Labs.
™UniPlus+ is a trademark of UniSoft Corp.

See us at UniForum Booth # 1265

There's only one DUAL

Circle No. 17 on Inquiry Card

phoned Fabry at home to express interest in taking charge of the further development of UNIX. Though surprised by the offer, Fabry took little time to agree.

The project started promptly. Tong set up a distribution system that could handle a higher volume of orders than Joy's previous distributions. Fabry managed to coordinate with Bob Guffy at AT&T, and lawyers at the University of California to formally release UNIX under terms agreeable to all. Joy incorporated Jim Kulp's job control, added auto reboot, a 1K block file system, and support for the latest VAX machine, the VAX-11/750. By October, 1980, a polished distribution that also included the Pascal compiler, the Franz Lisp system, and an enhanced mail handling system was released as 4 BSD. During its nine-month lifetime, nearly 150 copies were shipped. The license arrangement was on a per institution basis rather than a per machine basis, thus the distribution ran on about 500 machines.

With the increasingly wide distribution and visibility of Berkeley UNIX, several critics began to emerge. David Kashtan at Stanford Research Institute wrote a paper describing the results of benchmarks he had run on both VMS and Berkeley UNIX. These benchmarks showed several severe performance problems with the UNIX system for the VAX. Setting his future plans aside for several months, Joy systematically began tuning up the kernel. Within weeks he had a rebuttal paper written showing that Kastan's benchmarks could be made to run as well on UNIX as they could on VMS. Rather than continue shipping 4 BSD, the tuned up system with the addition of Robert Elz's auto configuration code was released as 4.1 BSD in June, 1981. Over its two-year lifetime about 400 distributions were shipped.

4.2 BSD

With the release of 4.1 BSD, much of the furor over performance died down. DARPA was sufficiently satisfied with the results of the first contract that a new two-year contract was granted to Berkeley with funding almost five times that of the original. Half of the money went to the UNIX project, the rest to other researchers in the Computer Science department. The contract called for major work to be done on the system so the DARPA research community could better do its work.

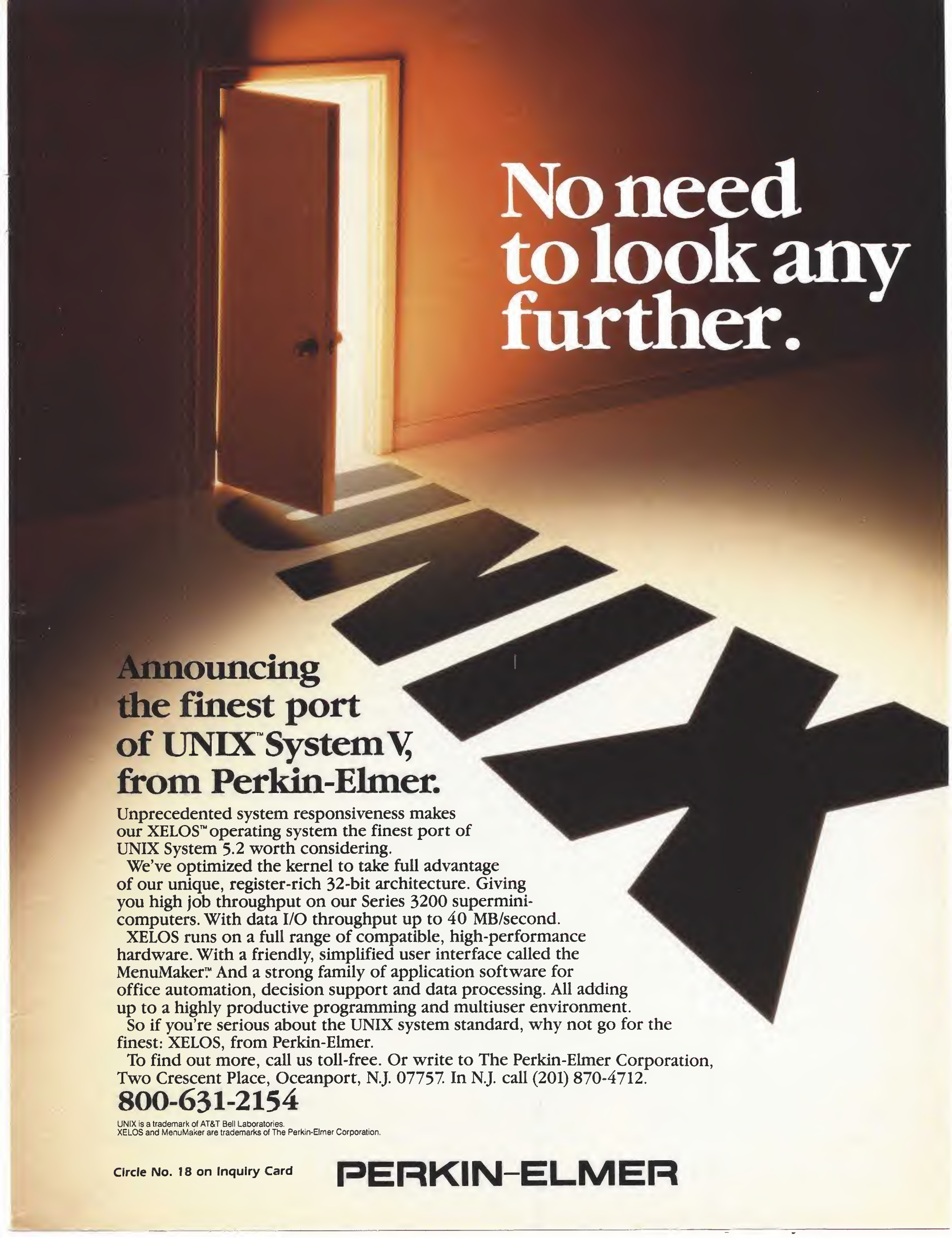
With the release of 4.1 BSD, much of the furor over performance died down.

Based on the needs of the DARPA community, goals were set and work began to define the modifications to the system. In particular, the new system was expected to include a faster file system that would raise throughput to the speed of available disk technology, would support processes with multi-gigabyte address space requirements, would provide flexible interprocess communication facilities that would allow researchers to do work in distributed systems, and would integrate networking support so that machines running the new system could easily participate in the ARPAnet.

To assist in defining the new system, Duane Adams, Berkeley's contract monitor at DARPA, formed a group known as the "steering committee" to help guide the design work and ensure that the research community's needs were addressed. This committee met twice a year between April, 1981 and June, 1983, and included Bob Fabry, Bill Joy, and Sam Leffler of the University of California at Berkeley; Alan Nemeth and Rob Gurwitz of Bolt, Beranek, and Newman; Dennis Ritchie of Bell Laboratories; Keith Lantz of Stanford University; Rick Rashid of Carnegie-Mellon University; Bert Halstead of Massachusetts Institute of Technology; Dan Lynch of The Information Sciences Institute; Duane Adams and Bob Baker of DARPA; and Jerry Popek of the University of California at Los Angeles. Beginning in 1984, these meetings were supplanted by workshops that were expanded to include many more people.

An initial document proposing facilities to be included in the new system was circulated to the steering committee and other people outside Berkeley in July, 1981, sparking many lengthy debates. In the Summer of 1981, I became involved with the project and took on the implementation of the new file system. During the summer, Joy concentrated on implementing a prototype version of the interprocess communication facilities. In the Fall of 1981, Sam Leffler joined the project as a full-time staff member to work with Bill Joy.

When Rob Gurwitz released an early implementation of the TCP/IP protocols to Berkeley, Joy integrated it into the system and tuned its performance. During this work, it became clear to Joy and Leffler that the new system would



No need to look any further.
UNIX
Announcing the finest port of UNIX™ System V, from Perkin-Elmer.
Unprecedented system responsiveness makes our XELOS™ operating system the finest port of UNIX System 5.2 worth considering.
We've optimized the kernel to take full advantage of our unique, register-rich 32-bit architecture. Giving you high job throughput on our Series 3200 superminicomputers. With data I/O throughput up to 40 MB/second.
XELOS runs on a full range of compatible, high-performance hardware. With a friendly, simplified user interface called the MenuMaker.™ And a strong family of application software for office automation, decision support and data processing. All adding up to a highly productive programming and multiuser environment.
So if you're serious about the UNIX system standard, why not go for the finest: XELOS, from Perkin-Elmer.
To find out more, call us toll-free. Or write to The Perkin-Elmer Corporation, Two Crescent Place, Oceanport, N.J. 07757. In N.J. call (201) 870-4712.
800-631-2154
UNIX is a trademark of AT&T Bell Laboratories.
XELOS and MenuMaker are trademarks of The Perkin-Elmer Corporation.
Circle No. 18 on Inquiry Card
PERKIN-ELMER

need to provide support for more than just the DARPA standard network protocols. Thus, they redesigned the internal structuring of the software, refining the interfaces so that multiple network protocols could be used simultaneously.

With the internal restructuring completed and the TCP/IP protocols integrated with the prototype IPC facilities, several simple applications were created to provide local users access to remote resources. These programs, **rcp**, **rsh**, **rlogin**, and **rwho**, were intended to be temporary tools that would eventually be replaced by more reasonable facilities (hence the use of the distinguishing **r** prefix). This system, called 4.1a, was first distributed in April, 1982 for local use; it was never intended that it would have wide circulation, though bootleg copies of the system proliferated as sites grew impatient waiting for the 4.2 release.

The 4.1a system was obsolete long before it was frozen. However, its construction and feedback from users provided valuable information that was used to create a revised proposal for the new system called the "4.2 BSD System Manual". This document was circulated in February, 1982 and contained a concise description of the proposed user interface to the system facilities that were to be part of 4.2 BSD.

Concurrent with the 4.1a development, I completed the implementation of the new file system, and by June of 1982 had fully integrated it into the 4.1a kernel. The resulting system was called 4.1b and ran on only a few select development machines at Berkeley. Joy felt that with significant impending changes to the system, it was best to avoid even a local distribution, particularly since it required every machine's file systems to be dumped and restored to convert from 4.1a to 4.1b. Once the file system proved to be stable, Leffler proceeded to add the new file system-related system calls, while Joy worked on revising the interprocess communication facilities.

In late Spring 1982, Joy announced he was joining Sun Microsystems. Over the summer, he split his time between Sun and Berkeley, spending most of his time polishing his revisions to the interprocess communication facilities and reorganizing the UNIX kernel sources to isolate machine dependencies. Pauline Schwartz was hired to take over the distribution duties. David Mosher was hired as a technical manager to resolve problems from users in the field and to handle ordering, installation, and running of the project's hardware.

With Joy's departure, Leffler took over responsibility for completing the project. Certain deadlines had already been established and the release had been promised to the DARPA community for the Spring of 1983. Given the time constraints, the work remaining to complete the release was evaluated and priorities were set. In particular, the virtual memory enhancements and the most sophisticated parts of the interprocess communication design were relegated to low priority (and later shelved completely). Also, with the implementation more than a year old and the UNIX community's expectations heightened, it was decided an intermediate release should be put together to hold people until the final system could be completed. This system, called 4.1c, was distributed in April, 1983; many vendors used this release to prepare for ports of 4.2 to their hardware.

In June, 1983, Bob Fabry turned over administrative control of the project to Professors Domenico Ferrari and Susan Graham to begin a sabbatical free from the frantic pace of the previous four years. Leffler continued the completion of the system, implementing the new signal facilities, adding to the networking support, redoing the standalone I/O system to simplify the installation process, integrating the disk quota facilities from Robert Elz, updating all the documentation, and tracking the bugs from the 4.1c release. In August, 1983, the system was released as 4.2 BSD.

When Leffler left Berkeley for Lucasfilm following the completion of 4.2, he was replaced by Mike Karels. Karels's previous experience with the 2.9 BSD software distribution provided an ideal background for his new job. The popularity of 4.2 BSD was impressive; within 18 months, more copies of 4.2 BSD had been shipped than of all the previous Berkeley software distributions combined.

As with 4 BSD, commentary of the vociferous critics was quick in coming. Most of the complaints indicated that the system ran too slowly. The problem, not surprisingly, was that the new facilities had not been tuned and that many of the kernel data structures were not well suited to their new uses. Karels' first year on the project was spent tuning and polishing the system. An anticipated release of the polished system early in 1985 is expected to quell many of the performance complaints — much as the 4.1 BSD release quelled many of the complaints about 4 BSD.

After completing my Ph.D. in December 1984, I joined Mike Karels on the project. We hope that other researchers will continue to share their work with Berkeley. By incorporating the work of other researchers

Meet us at
UNIFORUM
Dallas, TX
Booth 3313
January
21-25
1985

SIR/DBMS bridges the UNIX portability gap

SIR/DBMS helps market research analysts, statisticians, quality assurance engineers, medical researchers, and other knowledge workers transform their data into strategic information.
SIR/DBMS is a complete relational database management system with special features to manage empirical and decision-oriented data. It provides unmatched portability across a wide range of mainframes, minis, and 32-bit UNIX-based microcomputers.And it only takes a single command to move an entire database and its application programs to a different computer or operating system.

SPECIAL FEATURES

- Easy ad hoc query and reporting with SQL+
- Comprehensive data integrity and quality control
- Relational, hierarchical, or network views of data
- Direct interface to BMDP, SAS, SPSS
- Flexible report generation
- Publication-quality tabular displays
- Fast, efficient programming with a 4th generation language

UNMATCHED PORTABILITY

APOLLO	AEGIS
CDC CYBER	NOS, NOS/BE
CRAY	COS/CTSS
DATA GENERAL	AOS/VS
DEC VAX	VMS, UNIX
DEC 10/20	TOPS
GOULD SEL	S/32
HP 9000	HP-UX
HONEYWELL	CP-6, GCOS 8, MULTICS
IBM & PCMS	OS/VS, VM/CMS
ICL	VME, VME/B, EMAS
PERKIN-ELMER	OS/32
PRIME	PRIMOS
SIEMENS	BS2000
SPERRY 1100	EXEC
SPERRY 90	VS/9

SIR, Inc.

SIR, Inc. has pioneered the development of database software for INFORMATION ANALYSTS since 1976.
SIR/DBMS has been used to develop thousands of applications at R&D divisions of Fortune 500 companies, research institutions, government agencies, and universities across 25 countries. A full range of technical support and training services is available to you.

Call or write:
Director of Sales, SIR, Inc.
820 Davis Street, Evanston, IL 60201
312-475-2314

SIR/DBMS
THE choice of the knowledge worker

uniformly into the UNIX system at Berkeley, we can continue to offer the UNIX community a widely available state-of-the-art UNIX system.

ACKNOWLEDGEMENTS

I thank Bill Joy, Sam Leffler, Eric Allman, and Professors Bob Fabry, Richard Fateman, and Domenico Ferrari for providing the historical information presented in this article. Unfortunately, space constraints prevent me from describing the enormous amount of work that went into the user level software. I thank those people whose accomplishments have not been mentioned, but nevertheless have contributed much to the system's vitality. For the last five years, I have been supported by the National Science Foundation under grant MCS80-05144, and the Defense Advance Research Projects Agency (DoD) under ARPA Order No. 4031 monitored by Naval Electronic System Command under Contract No. N00039-82-C-0235.

Kirk McKusick is involved in the development of Berkeley UNIX as a Research Computer Scientist for the Computer Systems Research Group at the University of California. While a graduate student, he implemented the fast file system distributed on 4.2 BSD and worked on the Berkeley Pascal system. ■

FEAR AND LOATHING ON THE UNIX TRAIL '76

Notes from the underground

by Doug Merritt
with Ken Arnold and Bob Toxen

Illustration by Erik Jorgensen

It was 2 am and I was lying face down on the floor in Cory Hall, the EECS building on the UC Berkeley campus, waiting for Bob to finish installing our bootleg copy of the UNIX kernel. If successful, new and improved terminal drivers we had written would soon be up and running.

We were enhancing the system in the middle of the night because we had no official sanction to do the work. That didn't stop us, though, since UNIX had just freshly arrived from Bell Labs, where computer security had never been an issue. The system was now facing its first acid test — exposure to a group of intelligent, determined students — and its security provisions were failing with regularity.

I was lying face down because I'd gone without sleep for over two days, and the prone position somehow seemed the most logical under the circumstances. Bob was still working because he'd napped not 30 hours before, giving him seniority under the "Hacker-best-able-to-perform" rule of our informal order. We might have called our group "Berkeley Undergraduate Programmers for a Better UNIX", or, less euphemistically, "Frustrated Hackers for Our Own Ideas". But, in truth, our group was never named. It was simply a matter of Us versus Them.

"Them" was the bureaucracy — the school administrators, the system administrators, most professors, some grad students, and even the legendary Implementors themselves at Bell Labs.

"Us" was a small, self-selected group of undergraduates with a passion for UNIX. We were interested in computers and in programming because it fascinated us; we lived for the high level of intellectual stimulation only hacking could provide. Although some in our group never expressed an interest in breaking computer security, others invested thousands of fruitful hours in stealing accounts and gaining superuser access to various UNIX systems. Our object? To read system source code.

For the most part we stayed

Save Up to $3000 on a Model 16B Multi-User Computer

That's right, now you can get any of our UNIX-based Model 16B microcomputers at incredible savings. For only $3499 you can get a 256K Model 16B with built-in 15-megabyte hard disk (26-6006, Was $6499 in 1985 Catalog RSC-12). A Model 16B with two built-in floppy drives is only $2499 (26-6005, Was $4699 in Cat. RSC-12). And a Model 16B with one floppy disk is just $1999 (26-6004, Was $3999 in Cat. RSC-12).

Plus, you can also save on memory upgrades when you purchase them with the system.

Based on a Virtual Industry Standard

Our Unix-based software is easier to use (even for a computer novice) than any other system. This system allows more than just one person to share the power of a Model 16B microcomputer, so you can improve office productivity without the expense of multiple computers. Think of it—three people at different locations throughout your office performing different tasks, at the same time, on the same computer!

Share the Same Files and Printer

All of your data can be stored on the hard disk, so files like customer records, accounts payable and accounts receivable can be shared by all users. You can set up a system that lets you track cash flow while a department head figures a budget—at the same time an accountant works on profit and loss statements. No time is wasted setting up separate, redundant files, and accessories such as a printer, plotter or modem can be shared to save money.

Hurry In! At These Prices, They'll Go Fast

All sale-priced Model 16B computers are subject to prior sale, so stop by your local Radio Shack Computer Center today. We're ready to help with ready-to-run multi-user software, training and service plans, too. Get the full story—ask for a hands-on demonstration.

Available at over 1200 Radio Shack Computer Centers and at participating Radio Shack stores and dealers.

Radio Shack®
COMPUTER CENTERS
A DIVISION OF TANDY CORPORATION

Prices apply at Radio Shack Computer Centers and participating Radio Shack stores and dealers. Xenix/TM Microsoft Corp. Unix/TM Bell Laboratories. UNIFY/TM Unify Corp.

out of trouble, although one of our rank once had his phone records subpoenaed by the FBI — after a minor incident with a Lawrence Livermore National Laboratory computer. The Feds seemed to think our comrade had been diddling with top secret weapons research, but he actually hadn't.

Our group could probably best be characterized by its interest in creating and using powerful software, regardless of the source of the idea. Our battle cry, thanks to Ross Harvey, was "FEATURES!!!", and we took it seriously. Well, Ross may have been a little sarcastic about it, since he was referring to superfluous bells and whistles. But I used the expression as shorthand for "elegant, powerful, and flexible". We were always bugging Them to add "just one more feature" to some utility like the shell or kernel. Although They accepted some suggestions, They didn't think twice about most.

One example stands out. In early 1977, Ross, Bob, and I spent months collaborating on a new and improved shell, just before Bill Joy had started on what is now known as the C shell. The most historically significant features we designed were Ross's command to change the shell's prompt, Bob's command to **print** or **chdir** to the user's home directory, and my own **edit** feature, which allowed screen editing and re-execution of previous commands. What we did was smaller in scope than what Bill later included in the C shell, but to Us it was unarguably better than what was then available. We ceased work on our projects only when it became clear that Bill was developing what would obviously become a new standard shell. Our energies then were re-focused on persuading him to include our ideas. Some of our features ultimately were incorporated, some weren't.

We modified the kernel to support asynchronous I/O, distributed files, security traces, "real-time" interrupts for subprocess multitasking, limited screen editing, and various new system calls. We wrote compilers, assemblers, linkers, disassemblers, database utilities, cryptographic utilities, tutorial help systems, games, and screen-oriented versions of standard utilities. User friendly utilities for new users that avoided accidental file deletion, libraries to support common operations on data structures such as lists, strings, trees, symbol tables, and libraries to perform arbitrary precision arithmetic and symbolic mathematics were other contributions. We suggested improvements to many system calls and to most utilities. We offered to fix the option flags so that the different utilities were consistent with one another.

It was simply a matter of Us versus Them.

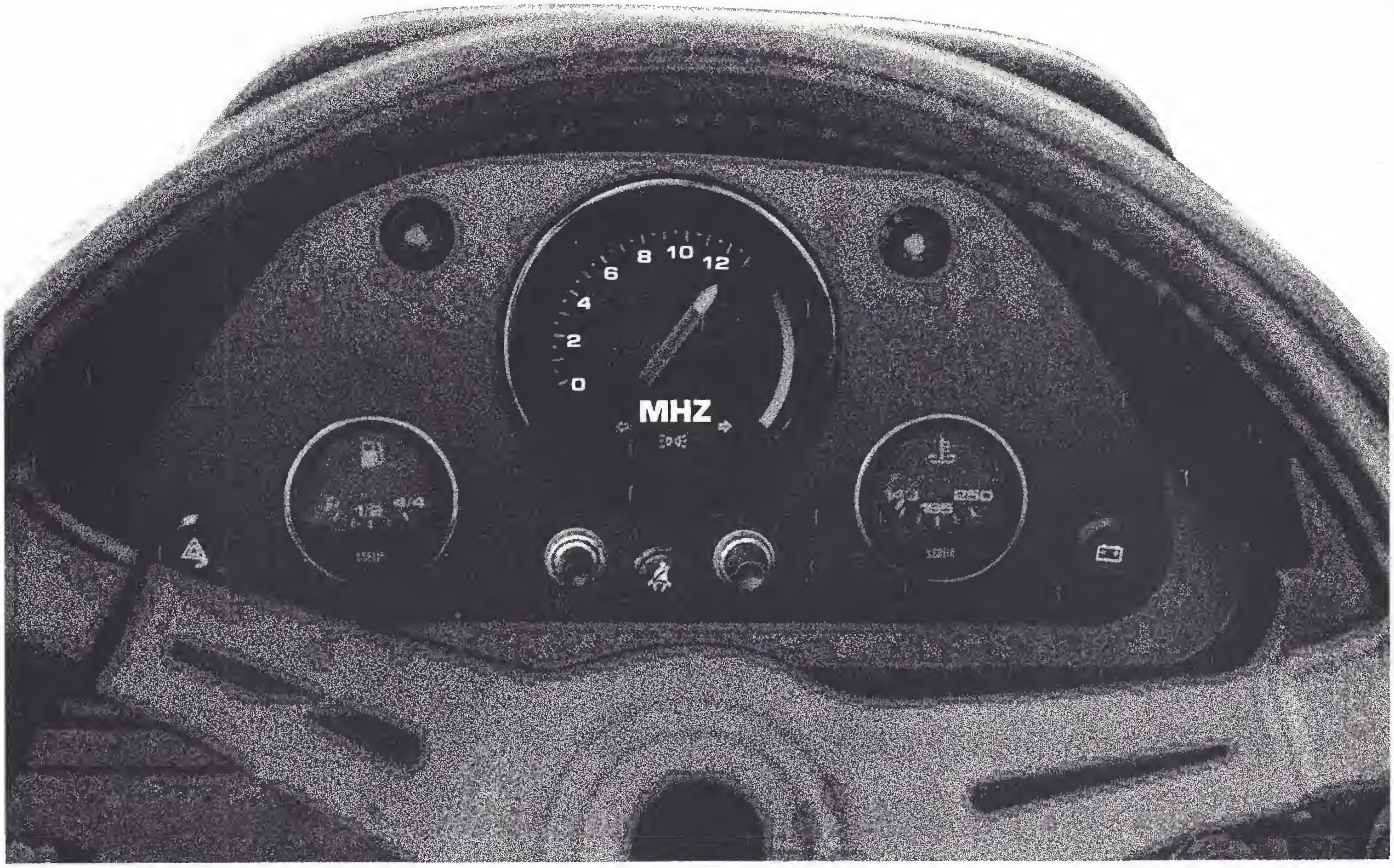
To Us, nothing was sacred, and We saw a great deal in UNIX that could stand improvement. Much of what We implemented, or asked to be allowed to implement, is now a part of System V and 4.2 BSD; others of our innovations are still missing from all versions of UNIX. Despite these accomplishments, it seemed that whenever We asked The Powers That Be to install Our software and make it available to the rest of the system's users, We were greeted with stony silence.

Fred Brooks, in The Mythical Man-Month, describes the NIH (Not Invented Here) Syndrome, wherein a group of people will tend to ignore ideas originated outside their own social group. However, there was a stronger force at work at Berkeley, where a certain social stratification prevails that finds Nobel Laureates and department chairs ranking as demigods, professors functioning as high priests, graduate students considered as lower class citizens, and undergraduates existing only on sufferance from the higher orders — and suffered very little at that. Now, the individuals cannot be blamed for what is, in essence, an entire social order. But this is not to say that we did not hold it against them — for we most assuredly did. Unfortunately, it took time for us to appreciate the difficulties of Fighting City Hall.

This is why We were frustrated. This is why We felt We HAD to break security. Once We did, We simply added Our features to the system, whether The Powers That Be liked it or not. Needless to say, They didn't. This is why We felt like freedom fighters, noble figures even when found in the ignoble position of lying face down on the floor of Cory Hall at two in the morning.

We were on a mission that morning to install our new terminal driver. With the old, standard terminal driver, the screen gave you no indication that the previous character had been deleted when you pressed the erase character. You had to accept it on faith. This remains true on many UNIX systems today. Most people on Cory Hall UNIX changed their erase character to backspace so that later characters would overwrite the erased ones, but even that was not sufficient. This was especially true when erasing a backslash, which counter-intuitively required two erase characters. We wanted the system to show that the character

Continued to Page 108



SPEEDBUSTER

12.5 MHz 68000. CACHE MEMORY. NO WAIT STATES.

You know how critical it is to keep your product line in the fast lane. Your competition is just waiting for a chance to pass. That's why you need a systems vendor who can keep pace.

Wicat Systems.

When it comes to price/performance, we're clearly in the lead. Take, for example, our System 2220. A 12.5 MHz 68000 processor. 4K of cache memory. And no wait states to slow you down.

Sure we're fast. But we're also flexible. You can take your choice of UNIX V or our own enhanced WMCS operating system. Along with the broadest range of high level languages on the market.

Whatever your multiuser application, odds are it will run better on Wicat.

Your application will also fit just about any size end user. Because you can upgrade from one to sixty-four user systems. Through six different Wicat CPUs. Without changing one single line of code.

If all this doesn't put you miles ahead of your competition, there's more. And only from Wicat. Like embedded training. And our interactive videodisc technology.

Or our touch screen option. To add an extra touch of user-friendliness.

There's more yet, but you'll have to come to UniForum booth 2181 to see it. Or call (801) 224-6400. Or write to Wicat Systems, P.O. Box 539, 1875 South State Street, Orem, UT 84058.

Because a shift to Wicat can get your business on the fast track.

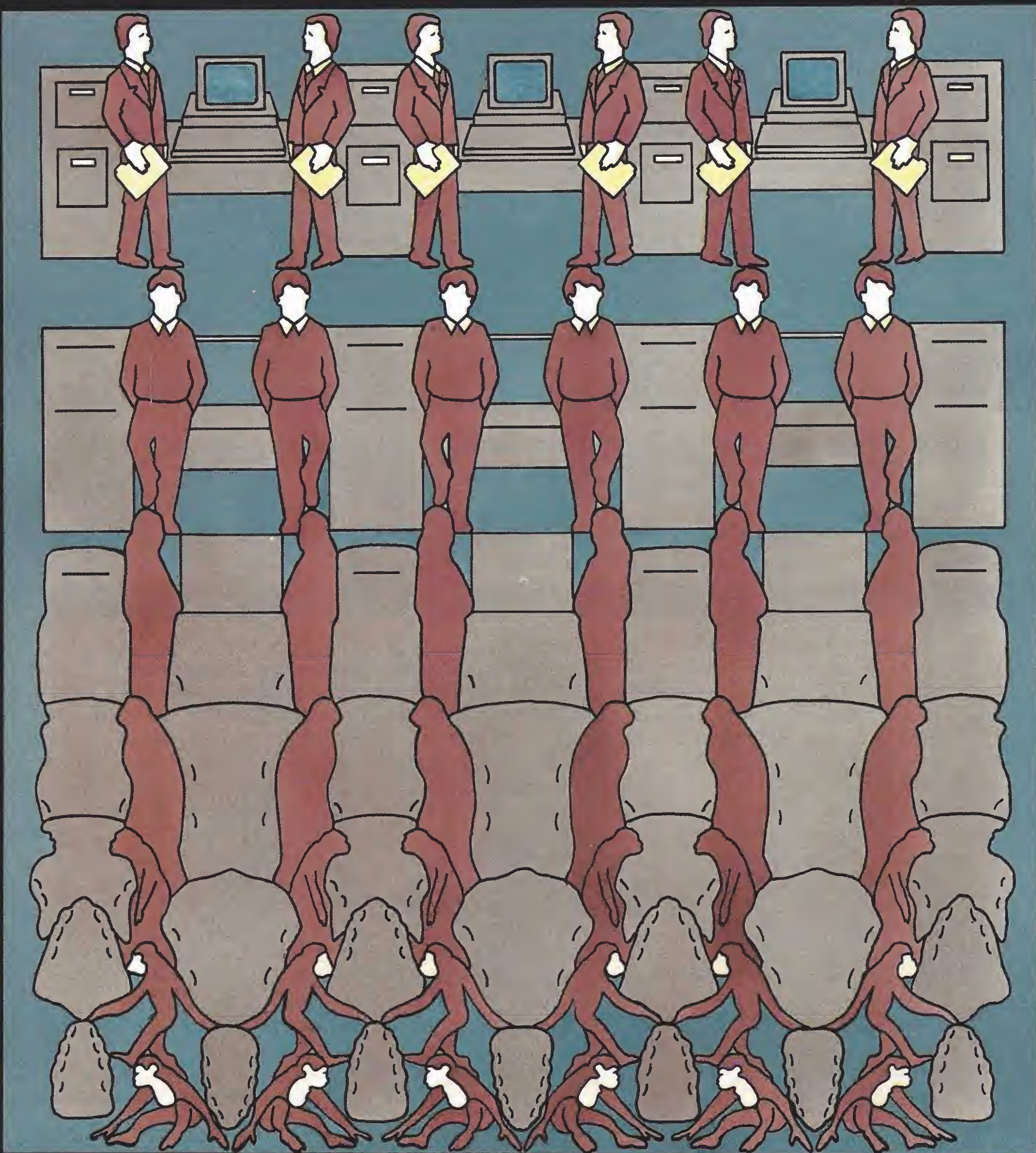
WICAT SYSTEMS

UNIX is a trademark of Bell Laboratories.

Circle No. 21 on Inquiry Card

THE BUSINESS EVOLUTION OF THE UNIX SYSTEM

An account from the inside



by Otis Wilson

Thanks to the developers of the UNIX operating system, and to the research method at AT&T Bell Laboratories, the technical evolution of the UNIX System has been well documented and its history largely understood. From a technical perspective, there just isn't much argument about who did what when and why things were done the way they were.

On the other hand, the "business" history of the UNIX system is largely an oral one, rich in folklore and popularized by the modern press in hopes of finding some explanation for the phenomenon that is the UNIX system.

For all its entertaining quality, the oral tradition is not very instructive, especially for those, including AT&T, that will build their futures on the success of UNIX software.

Although a complete business history is beyond the scope of a single article, a few insights into the major events can put the "evolution" into its proper perspective, and can give us all a better view of what the future might hold.

INSIDE THE CORPORATE CULTURE

Nothing has had a more profound impact on the business development of the UNIX system than AT&T's own corporate culture. And although there have been many "cultural" influences that have molded the evolution, two stand out most prominently.

The first is the well-established intellectual bond between AT&T Bell Laboratories and the academic community — one that goes beyond formal internship and summer on-campus programs. The relationship has been nurtured throughout the history of Bell Labs, and it has contributed significantly to the Labs' effectiveness as one of the world's leading research institutions. It has also had more than a casual impact on AT&T's ability to attract outstanding technical talent.

The second major influence was the 1956 Consent Decree, the settlement of a complex antitrust case that would, among other things, clearly define the Bell System's business. In its definition, the Consent Decree was unequivocal: the Bell System was to provide telecommunications products to the Bell Telephone Companies and telecommunications service to the nation — and nothing more.

THE CONSENT DECREE AND SOFTWARE

Under the terms of the Consent Decree, the Bell System was required to license certain technologies that had been funded by its regulated business, and an organization, Patent Licensing, had to be set up to manage the effort.

But contrary to popular belief, the Consent Decree did not address the issue of software. In fact, the technology was virtually unknown at the time. Consequently, AT&T was not specifically obligated to license its software; nor was it restrained from entering the software business. Software was a muddy issue in an otherwise clear statement of AT&T's legal responsibilities.

As early as 1967, even before the development of the UNIX system, the technical relationship between Bell Laboratories and universities across the nation had spawned a number of requests for AT&T software. In the interest of further cultivating that relationship, there was a genuine desire to share the technology. There were also concerns. Was the company required to, or enjoined from, licensing software? A definitive answer would have required revisiting the Justice Department's ruling, and, understandably, few in the company were inclined to raise any issue that might require doing so.

Instead, the company took a more prudent approach. To preclude any conflict with the Consent Decree, AT&T would license its software under the Consent Decree's legally established procedures but would made it clear that it had no intention of pursuing software as a business. The policy was restated over and over again at every gathering of the faithful — "As is, no support, payment in advance!"

ENTER UNIX SOFTWARE

It was in this climate of technical exchange and legal uncertainty that the UNIX system first spread to academia in the early 1970s. At the time, the UNIX system was still a research project, and its impact on AT&T's total computing environment was minimal. In fact, it's fair to say that many colleges and universities knew more about the UNIX system than did AT&T's mainstream data processing community.

By 1972, educational interest in the UNIX system had grown to the point that there were a handful of universities requesting copies of the software for use in their computer science programs. Initially, requests were referred to the the developers themselves, and through the patent attorneys at Bell Laboratories, the software was conveyed royalty-free under simple letter agreements.

Within a few months however, as the volume began to tax the resources of the Labs, responsibility for licensing fell to AT&T's Patent Licensing organization, the group that had been vested with the responsibility for licensing intellectual properties under the Consent Decree.

INSIDE PATENT LICENSING

Patent Licensing was an organization with an intentionally low profile. Historically, negotiations for AT&T patents were conducted at very high levels and the discussions were particularly sensitive. The organization's management would typically be involved in only four or five negotiations per year, but each might stretch on for months and involve teams of lawyers, accountants, and millions of dollars. It was, by any measure, a high stakes game.

In such an environment, non-revenue-producing software was a

Illustration by Ivy Nichols

low priority. The lack of commitment was no more evident than in the licensing organization itself.

There were those, however, who recognized the potential for software. By the early 1970s, there were approximately two dozen software packages that had been released by AT&T under license agreements. Of them, 10 showed significant potential. The UNIX system was the clear leader.

Those 10 packages formed the basis of a "mini" business case, but few would be convinced of the need for additional resources to support the licensing effort. AT&T's policy was clear, and it would withstand a number of challenges before being reversed.

ENTER COMMERCIAL INTERESTS

Had the licensing of UNIX software been restricted to universities, the story might have ended there. But the emergence of commercial interest in the software placed new demands on the organization and forced a decision that would eventually lead to the creation of a market.

By 1974, AT&T had established nominal distribution fees for the software, but the income from universities was insufficient to offset the costs associated with licensing. When the company actually received its first commercial requests in 1974, there were no commercial prices in place, nor was there much of a notion about how the software should be priced for commercial use.

Although most agreed that commercial licensing would help the software effort earn its keep, there was little initial agreement on how the pricing should be structured. Many felt the price should be set abnormally high to discourage widespread use. Under such a scheme, income from just a few licenses would provide more than enough income to defray licensing expenses. Others, including the developers, felt the price should be set artificially low to encourage use and experimentation. Still others argued for a price that would be competitive with existing systems (minus an

adjustment to reflect the lack of support). Significantly, the company chose to base the software price on market considerations.

The decision would have a profound influence on the business history of the UNIX system. It indicated, for the first time, the company's genuine business interest in the software. More importantly, it led to the establishment of a "customer base" whose needs were radically different from those for whom the software had been designed.

Clearly, commercial users of the system considered themselves "customers" despite AT&T's disclaimer that it had but one customer — namely, the Bell System. Commercial users wanted greater functionality, more commercial features, more flexible terms and conditions, as well as additional rights, including the right to sublicense. They were the impetus for change, and a constant challenge to AT&T's established policy. The changes were slow in coming, a source of frustration to many of them and a condition that set the stage for the love/hate relationship that has long characterized UNIX software in the commercial world.

A BUSINESS EMERGES

During the period 1974-1978, AT&T introduced several versions of the UNIX system for a wide variety of DEC hardware, and each release prompted an increase in the number of requests for the software.

For the most part, licensing policies and pricing remained consistent, but functional differences increased both internal and external pressures to provide a single, consistent and — from an internal viewpoint — supportable version that could be used in a production environment. There were also demands from external licensees to incorporate more commercial features that had been developed elsewhere.

In late 1980, Western Electric, the manufacturing and supply arm of the Bell System, essentially "bought out" its other corporate partners that provided the funding for UNIX system development and established a single funding source. Responsibility for production and distribution of the source tapes was moved from Bell Laboratories, which had performed those functions from the beginning, to Western Electric. Although a firm business commitment was not yet established, it was clear that the company was beginning to think about the UNIX system as a product. After all, that was Western Electric's role in the Bell System.

Under Western Electric's direction and in response to AT&T's largest external customer, the federal government, a number of efforts were made to provide a more "commercial" product. Research notes were assembled to form official "documentation", and a support organization was established to assist government licensees. What emerged from the whole of this effort was UNIX System III.

UNIX System III was enormously successful in many respects. For the first time, it combined in a single release many of the best features of the earlier versions, plus some enhancements of its own. It was also the first version to be released externally at about the same time it was released internally. Historically, "commercial" releases lagged significantly behind releases to the Bell System. But most significantly, its success demonstrated, without a doubt, the market potential of UNIX software.

The success of System III also brought its share of problems, not the least of which was the extra strain it put on the resources of Patent Licensing. The licensing process was still a legal one that required, in the opinion of many AT&T customers and AT&T officials, an extraordinary length of time. Licensing delays were creating a lengthy backlog of unfulfilled requests. Delay had always been a hallmark of UNIX software licensing, but considering the resources devoted to the effort, it had always been manageable. The licensing of System III rapidly became unmanageable.

On the software front, System III represented a good first step in the right direction, but there was still much to be done before the software could be considered a product. For all its merits, System III still lacked many commercial features that licensees were demanding.

In the meantime, renewed antitrust action signaled a fundamental shift in AT&T's thinking about its traditional business role. The time had come to seriously evaluate the company's potential as a software vendor. In January, 1983, just a few weeks after the announcement of the proposed Modified Final Judgement that would bring about the dissolution of the Bell System, AT&T formally announced its intentions in the software business by releasing UNIX System V, complete with training and technical support. It also pledged its commitment to UNIX System V as the foundation of AT&T's future development efforts.

To achieve that charter, AT&T set up Software Systems as a "sub-business" under the Computer Systems Division. The first organization under that new structure was the Software Sales and Marketing organization, whose responsibility it would be to make the transition from a licensing organization to a customer-oriented sales organization. The licensing of AT&T's other software, which had, by necessity, taken a back seat to UNIX software, remained with the Patent Licensing organization.

The first item on the agenda of the new organization was to reduce the enormous backlog of pending requests. Within six months, the backlog had been reduced from more than 200 to less than a handful by carefully restructuring the effort to scale down the need for extensive legal review, by cutting out unnecessary paperwork, and by instituting standard agreements that could be easily adapted for any UNIX software product. As a result, average turnaround time for processing agreements was reduced from four weeks to one.

A second major objective was to rebuild relationships with those licensees who offered products based on AT&T software. Towards that end, a new, more flexible royalty schedule was offered to more adequately reflect the reality of the microcomputer marketplace. An on-going communications program was also established and a visitation program was implemented.

In addition, a variety of add-on products are currently being offered, and many of the commercial features so long requested by commercial resellers are being implemented. Perhaps most importantly, the future direction of AT&T's development effort has been documented, providing a blueprint for those who will follow AT&T's lead with UNIX System V.

There is still too much to be done to close the book on the UNIX system's business history. But much of the remaining story will be written under the influence of a new corporate culture — one characterized by a business commitment to the company's custodial role. There will also be a number of contributing authors — other vendors, who like AT&T, have grown to recognize the potential of the UNIX system.

Otis Wilson is the manager of AT&T Technology System's software sales and marketing efforts. His contributions were key to the design of AT&T's current licensing structure and his leadership is still central to its implementation. Mr. Wilson is also a member of the UNIX REVIEW *Editorial Board.* ■

THE FUTURE OF THE UNIX WORKSTATION

A glance at tomorrow

by Richard Morin

The advent of 32-bit microprocessors and 64K RAM chips has made graphics workstations a viable and growing way of using computers. Coming jumps in microprocessor power and memory density can only increase this trend.

Several vendors are now producing UNIX-based graphics workstations. These products generally provide a very pleasant UNIX environment. The question, however, is whether the advantages of current and future workstations can be fully utilized under UNIX.

WHAT IS A WORKSTATION?

First, let's decide what we mean by a UNIX graphics workstation. Start with a graphics terminal: you find a keyboard, a high resolution screen, and communications. Speed up the screen and the communications, add a processor, some memory, and some software, and you have a graphics workstation.

Note, however, that not all workstations are exactly alike. Some have increased processing power, in either general or special-purpose forms. Others have higher resolution screens, and/or exotic I/O devices. Laboratory versions may have real-time I/O capabilities.

Workstations may have local storage or I/O facilities, but they generally rely on local area networks to provide at least part of these services. They may also call upon the network for computing resources, but they do most processing independently. Thus, Morris's law of personal computing applies: "Even when you go in at night, it's still slow."

Graphics workstations tend to have bitmapped screens, mice or other pointing devices, as well as software for graphics and window management. Since these features are simultaneously getting cheaper and more popular, we can expect them to become very common on future workstations.

HOW ARE WORKSTATIONS USED?

Graphics workstations are capable of being used effectively in a variety of applications. Their current applications tend, however, to be limited to the scientific, engineering, and software development arenas.

Scientists can use arbitrarily large amounts of computing power, and have been using graphics for ages. Scientific computing resources of the last decade or so have typically been timeshared mainframes or superminis. A set of terminals, some graphic, have typically been attached.

This makes a reasonably nice environment — except for one slight problem. Between two and five in the afternoon, the system hits a brick wall. Most scientific institutions cannot afford to provide a supermini for each worker. The advent of the supermicro-based workstation is thus a timely event for the scientific community.

Engineers also use large amounts of computer power when they perform modelling or other analytic work. They, too, have adopted the workstation, but for a different compelling reason. The workstation serves as the engineer's mechanized pencil and pad.

A slew of workstation-based Computer Aided Design/Computer Aided Engineering (CAD/CAE) systems have appeared during the last few years. These systems allow an engineer to make drawings, simulate the object(s) in question, and even produce manufacturing specifications. The complexity of today's designs, coupled with the cost of errors, makes these systems very attractive indeed.

The software development community is starting to use workstations because they increase productivity. Windowing systems allow a programmer to edit a program in one window, test it in a second, and monitor performance data in a third.

In addition, increasingly sophisticated software is becoming available to support programming on workstations. Much of this software will work only poorly, if at all, on traditional terminals. As this software proves its worth to software developers, programmers will find a compelling reason to switch to workstations.

Finally, we can expect the business community to adopt workstations as price/performance ratios come down. Many of the features which make workstations attractive to other communities are relevant to business as well. At over $10,000 apiece, however, they are still too expensive for typical business use.

HOW DOES UNIX FIT IN?

UNIX has a strong toehold in the workstation arena for a variety of reasons. First, it lets vendors concentrate on workstation hardware and software issues, ignoring operating system design. Next, it allows users to mix various workstations and other processors on a single network. Finally, it provides a (somewhat) standardized software environment for workstation applications development.

Some of the more interesting workstation designs, however, still emerge from non-UNIX environments. The current breed of bitmapped workstations did not, in fact, originate in or for the UNIX environment. Instead, it came from research laboratories such as Xerox PARC, where it was used as part of the Smalltalk development effort. Many of the most powerful and friendly workstation user interface features are still missing from the UNIX arena.

Some would even say that UNIX, due to its pedestrian origins, is ill-suited for use in a workstation environment. The question, then, is whether and how UNIX can be adapted to take advantage of the coming flood of workstations.

THE BASIC HARDWARE

The Carnegie-Mellon University (CMU) standard for workstations has been said to be a million instructions per second (1 MIPS), a million pixels, and a megabyte (1 MB) or more of memory. Of late, the folks at CMU and elsewhere seem to say that 2 MB of RAM is really the lower limit. This could be a description of a Sun Microsystems workstation, save that a Motorola 68010 is not quite a 1 MIPS processor.

Given that benchmarking is a somewhat arcane art, speed ratings are a bit suspect. Still, the 68020 and other full 32-bit processor chips are on the way, and

The question is whether the advantages of current and future workstations can be fully utilized under UNIX.

they will be faster than the current chips. Of course, this speed increase will create a demand for still more memory.

Fortunately, the 256K RAM chips are now starting to become available in production quantities, and their prices will soon start to drop. The desired amount of memory in workstations will increase to meet these events. We can thus expect to see workstations with 4-8 MB of RAM becoming quite common over the next few years.

Disk storage is more problematic. A 4.2 BSD system requires over 30 MB of disk just for binaries. Manuals consume still more disk. Add 16 MB for paging space and 4 MB for miscellaneous files, and you've used up over 50 MB in overhead alone. A minimum of 100 MB of formatted disk storage is therefore reasonable for comfortable use.

Networked disk farms can provide cost-effective mass storage for workstations. They can also cut down on the amount of duplicate storage required for system files and such. Primarily, however, their benefit lies in allowing members of a project to share critical data quickly and easily.

There is a simple reason for this. Only so many diskless workstations can page and access files over a network. The chief bottleneck currently lies in the fileservers. The network itself will run out of steam eventually, however.

The performance degradation caused by such an overload is severe. The solution, fortunately, is simple. Each workstation can retain a modicum of local disk for paging, frequently used files, caching, and the like. This reduces network traffic while retaining access to communal disk storage.

NETWORKING

Workstations can be useful with or without networking, but they are generally designed to work well within a network. Convenience and speed of data sharing are reasons, but there is a more pressing factor involved. As the costs of system resources are spread over a network, the cost effectiveness of the associated workstations goes up dramatically.

A computer system requires mass storage, which is not cost-effective in small quantities. A tape drive or something of the sort is also necessary, even though it may be used only sporadically by any given user. Finally, options such as laser printers are often unjustifiable unless they can be shared by a number of users. Networks can clearly pay their way.

The choice of which network to use is not at all obvious, however. AT&T has announced 3Bnet, a snazzy new proprietary product. Other such products are available now, with more to come.

Given that proprietary networks are about as useful as proprietary wall socket designs, one wonders why they bother.

AT&T may be an exception, however, as may IBM. When and if IBM settles on (as opposed to flirting with) a method for UNIX networking, a de facto standard will have been created. The PC-AT has a low-speed networking hack, and we can only wait to see if it sticks around.

Meanwhile, out in cloudland, we have the ISO network model. Everybody says they will be happy to conform to it, just as soon as they know what it is. This is not too useful to someone needing a network today, however.

Fortunately, there is an available non-proprietary standard for linking UNIX workstations. It is available from all the 4.2 BSD UNIX vendors, and AT&T will support it soon (trust me). The Transmission Control Protocol / Internet Protocol (TCP/IP), sponsored by the Defense Advanced Research Projects Agency (DARPA), is the current de facto UNIX standard.

TCP/IP runs on a variety of operating systems, not just UNIX. It can use a variety of communications media, not just Ethernet. It is layered, much like the ISO model, allowing new technologies and techniques to be grafted on.

What's more, it works—as demonstrated at last Summer's Usenix Conference. The trade show featured an ad hoc multi-vendor Ethernet. People logged in remotely, copied files, and in general had a splendid time.

DISTRIBUTED FILE SYSTEMS

Remote login (**rlogin**) and remote copy (**rcp**) aren't the ideal networking model, however. We really want networks to act as if they were just a funny kind of mainframe. That is, it would be nice if we could write programs and run them without worrying about the exact whereabouts of the data they draw from.

Now it is true that some proprietary distributed file systems have been developed. Fine, but this, too, is about as useful as proprietary wall sockets. What we want is the ability to share file systems not only among different computers, but even among different operating systems.

It would be even nicer if this were layered onto TCP/IP so that it would gain from the portability of that protocol. Finally, we might ask that the system not attempt to provide all services to all customers. Any attempt to impose such complete solutions would clearly fail over time.

Note that not all workstations are exactly alike.

A flexible file system with the sort of portability discussed here has been developed by Sun Microsystems. What a vision of Joy! (Sorry.) Its protocols, based on a remote procedure call technique, are being publicly released. One can only hope that AT&T and the other vendors will look at the proposals like Sun very carefully.

PICK DEVICES

Since keyboards are virtually useless as graphic input devices, any number of pointing or "pick" devices have emerged. Touch screens, verniers, bit pads, track balls, 3-D digitizers, and other pick devices are examples of this diversity. Each device has a range of applications it excels at.

Touch screens are very useful in many process control applications. Verniers provide a convenient way of setting and adjusting values. Bit pads seem to work very well in sketching applications. Track balls are very fast, accurate, and require little surface area.

Dr. Greg Chesson reports that Silicon Graphics has used a variety of graphics input devices with its Iris workstation, including joysticks, tablets, and a 3-D digitizer. In short, these devices will continue to be used in situations where a particular characteristic is needed.

The mouse is the clear winner at this time, however, for a number of reasons. It is cheap, easy to use, versatile, reliable, and has reasonable speed and resolution. The mouse is thus here to stay, at least until a better alternative appears.

AUDIO INPUT AND OUTPUT

Audio I/O is conspicuously absent on today's UNIX workstations. It seems ironic that one has to use a personal computer to play with audible cues and voice recognition. Some older machines used to tell how hard they were working by means of their console speakers. Many of today's workstations don't even have an audible bell.

This situation seems likely to continue for some time. Voice recognition is hard, requiring Artificial Intelligence (AI) techniques and lots of computing horsepower. It won't show up on workstations until it becomes easier.

Audio output is also problematic. Most office workers treasure what silence is available, and do not want their workstations to sound like video games. In any case, cues that are available through audio can generally be made (almost) equally well on a screen.

A possible exception to this trend may emerge from the commercial workstation marketplace. There is an accelerating move

Circle No. 24 on Inquiry Card

towards combining data and voice in private telephone systems. A few vendors are experimenting with very intelligent telephones, telephone equipped workstations, voice mail, and so forth. If this catches on, the general purpose workstation vendors will be forced to provide the necessary audio interfaces.

VIDEO INPUT

A few vendors are selling subsystems for digitizing pictorial information. Other vendors have video subsystems designed for data acquisition, robotics, and such. Experimental systems have even been designed that watch users' eyes to get cues about what they are watching.

In general, however, the immediate prospects for video input on workstations are pretty slim, with one possible exception. There is a lot of pressure for getting rid of the piles of paper that infest the typical office. If an economical scanning device were to show up, particularly with some general and efficient character recognition software, this market could explode overnight.

COLOR DISPLAYS

It seems to be very difficult to make good color workstations. Both storage and bandwidth requirements increase markedly when a workstation attempts to produce full color. The million pixels, only one bit each on a b/w unit, become a byte or more apiece on a color unit. Color workstations thus require megabytes of fast RAM for display memory.

There is also the problem of achieving satisfactory text display. We like to see nice sharp letters on displays, and most color workstations do this poorly.

This is primarily a problem of color CRT monitor design. Each color pixel is formed by a triad of dots. These must be placed so as to cover the same spot on the screen. The difficulty of achieving this convergence goes up dramatically with the desired screen resolution.

Other factors are also involved, however. The Sun 160, which has 900 x 1152 resolution, is a magnificent color graphics display, but gets just a bit fuzzy on text. According to James Gosling of Sun, this could be substantially improved by using anti-aliased fonts and other techniques that take better advantage of the 160's gray scale.

Proprietary networks are about as useful as proprietary wall socket designs.

Unfortunately, these techniques are both tricky and expensive in terms of computing resources. This is not to say that Sun and other vendors won't use them, but nobody seems to be using them at present.

TEXT EDITING

The text editor is one of the most important programs on any interactive system, simply from the amount of use it gets. The de facto standard UNIX screen editor is **vi**, but nobody seems to be really happy with it.

UNIX drastically needs, and does not have, a simple and powerful bitmapped editor. A "What you see is what you get" (WYSIWYG) editing system is quite possible on a graphics workstation. Such a system allows the user to get immediate feedback on a given formatting decision.

Unfortunately, the WYSIWYG systems developed to date do not have the power and flexibility of the UNIX **vi** and **nroff** combination. A skilled UNIX user can make global changes to the content and/or the format of a document by invoking appropriate mysterious commands. One might hope that someone would produce an equally powerful but far simpler WYSIWYG editor.

The closest approximation currently on the market seems to be the Interleaf system. Unfortunately, its several thousand dollar price tag puts it out of reach for most editing applications. Time should solve this problem eventually, however, since a market for such an editor clearly exists.

One might also wish for syntax-sensitive editing, which would allow the editor to tell the user when a syntax error has been made. An editor might also assist in the editing process, supplying editable models for desired syntactic structures.

Several such editors have been developed as research projects, and some are even becoming available on UNIX systems. A caveat, though, is perhaps in order. Users must be able to expand such an editor's repertoire at any time. Extensibility and modifiability are thus critical in this kind of editor.

THE WINDOW MODEL

UNIX started out as an interactive system, but its design never contemplated bitmapped workstations. Most UNIX workstations treat the screen as a collection of windows, with each window simulating a quasiindependent terminal. A number of system calls are generally provided to allow both text and graphic use of windows.

The design of window management systems is still an active research area, however. Vendors have been forced to produce and support libraries in an area where the "correct" solutions are not known. Many CAD designers and other users have found it expedient to code for a "lowest common denominator" graphics interface, often avoiding use of the windowing system altogether.

The best solutions to the window management problem will probably come as a result of a change in implementation languages. This is a somewhat radical notion, given the historic association of UNIX and C. While UNIX supports a variety of languages, C has always been a bit more equal than the others.

It is thus interesting to find heavy-duty UNIX hackers such as Bill Joy talking about supporting languages such as Ada and Modula-2. It is also interesting that the designers of workstation interface libraries are looking at a variety of object-oriented hacks to C, as well as at other languages entirely.

C is a very powerful and versatile language, and it is not likely to disappear soon, if at all. We may well see it expanded or even supplanted, however, as hackers struggle to add flexible networking, workstation interfaces, and AI features to UNIX.

ARTIFICIAL INTELLIGENCE

The sophisticated user interface features found on AI workstations are, in general, conspicuously absent from current UNIX workstation implementations. Some of these could conceivably be grafted onto UNIX workstations. Others are dependent on the single language environments in which they currently reside.

The Smalltalk **browser** and Interlisp **Masterscope** facilities are powerful tools for finding one's way around large bodies of text. They can be thought of as intelligent editing systems, with the ability to move gracefully among all files referencing a given item. There are currently no UNIX

Continued to Page 124


Dr. Greg Chesson


Gene Dronek


Dr. James Gosling


Ron Gremban

Pat Johnson Studios

UNIX REVIEW gratefully acknowledges the contribution of the following four experts to the preparation of this article. The four participated in a roundtable interview conducted by Richard Morin that served as the basis for the piece.

Dr. Greg Chesson serves on the Editorial Review Board of UNIX REVIEW and works as a Senior Scientist at Silicon Graphics, Inc., in Mountain View, CA. While at Bell Laboratories, Dr. Chesson developed the packet driver protocol used by **UUCP** and the **mpx** files in the Seventh Edition of UNIX. He also developed original protocols and software implementations for the Datakit network. At Silicon Graphics, he has implemented XNS and other network protocols.

Gene Dronek sits on both UNIX REVIEW's Editorial Review and Hardware Review Boards. He is the author of the standard benchmarking package for UNIX systems, the Aim Benchmark suite. Formerly, he served as the lead UNIX consultant at UC Berkeley.

Canadian-born Dr. James Gosling has been a UNIX hacker since the mid-70's when he worked with Version E on a PDP-11/40. Having formerly served on the IBM/ Carnegie-Mellon University joint workstation development project, he currently works for Sun Microsystems in Mountain View, CA. He has built a multiprocessor version of Version 7, an **emacs** editor for UNIX systems, and an assortment of graphics systems.

Ron Gremban is presently a software consultant, doing work in Smalltalk for the organization that originally developed it, the Xerox PARC System Concepts Lab. A graduate of Caltech, he has worked for Tektronix and as an Engineering Manager for Kontron Electronics, where he was involved in the development of UNIX and CP/M-based microprocessor development systems, as well as the development of microcomputers and logic analyzers.

Studio 84, Wayne, New Jersey

PUTTING UNIX IN PERSPECTIVE

An interview with Victor Vyssotsky

You may have heard that before there was UNIX, there was Multics. But what came before Multics? And if Multics and a number of other operating systems already existed, why was UNIX created?

To trace the development of UNIX, it's appropriate not only to cite places and dates but also to consider that UNIX came out of a greater process of research — a process in which Victor Vyssotsky was very much a key player.

Vyssotsky's years at AT&T Bell Laboratories overlap the years of UNIX growth. From the days when he was in charge of Bell Labs' portion of the joint Multics project to his present position as Executive Director of Research in the Labs' Information Sciences Division, he has had an opportunity to develop a perspective we wanted to probe. We asked Ned Peirce, a Systems Analyst at Bell Labs and a Contributing Editor for UNIX REVIEW, *to pose the questions.*

REVIEW: *The UNIX community continues to be fascinated with what started Ritchie and Thompson thinking about the things that* *later became known as UNIX. I'd like to learn about your involvement in Multics and hear your impressions of those early days, perhaps even those before you were associated with Ritchie and Thompson. I'd also like to get your thoughts on current directions in UNIX.*

VYSSOTSKY: Well to begin with, the interest in operating systems at Bell Laboratories goes all the way back to late 1957 when George Mealy and Gwen Hansen produced the first of a succession of versions of what was called the BESYS operating system for the IBM 700 - 7000 series. There was a BESYS I but it was never used for evolutionary reasons. The first home-grown operating system that actually came into use was BESYS II, which was run on in-house IBM 704s starting in March of 1958. The reason we had a home-grown system to begin with, of course, was that in March of 1958 you couldn't get an operating system from a vendor. If you wanted an operating system, you built your own or got one from a friend. We couldn't find any friends who had operating systems we regarded as satisfactory for our purposes, so Mealy and Hansen built one. It was not the world's first operating system but it was certainly one of the very early ones.

REVIEW: *What was the application?*

VYSSOTSKY: General purpose computing and comp center support. We were running pure batch processing at that time, of course. But we were running a wide range of applications for a variety of users — mostly short jobs. We just couldn't take the time to get them on and off the machine manually. We needed an operating system to sequence jobs through and control machine resources.

The sequence of evolving BESYS systems went up through BESYS VII. Several of them were used fairly widely outside of Bell Laboratories. People simply took them. You know — you gave them to friends.

REVIEW: *That's very similar to what later happened to UNIX.*

VYSSOTSKY: Right.

REVIEW: *There was no support?*

VYSSOTSKY: No, there was no support. When we shipped a BESYS tape to somebody, we would answer reasonable questions over the telephone. If they found troubles or we found troubles, we would provide fixes.

If you wanted an operating system, you built your own or got one from a friend.

That world came to an end with the advent of third generation equipment in 1964. We had a decision to make in 1964 and '65. Were we going to go to a vendor operating system, have one built, or build one ourselves? Through a rather murky process of internal deliberation, we decided to join forces with General Electric and MIT to create Multics. Our intention was to use the Multics operating system as a mainstay for Bell Laboratories' internal service computing in precisely the way that we had used the BESYS operating system.

REVIEW: *That was the only model* you had!

VYSSOTSKY: Yes. It turned out that from our point of view the Multics effort simply went awry. In the first place, we were naive about how hard it was going to be to create an operating system as ambitious as Multics. It was the familiar second system syndrome. You put in everything you wished you'd had in the other one.

REVIEW: *And later discover that it was ridiculously difficult to do?*

VYSSOTSKY: Yes. Furthermore, although at the outset it had seemed that the objectives of Bell Labs, MIT, and General Electric were consistent with one another because we could agree on a common definition of the product, it turned out that our objectives were really not all that consistent. What MIT was interested in was advancing the state of the art in operating systems.

REVIEW: *Which you were slightly interested in?*

VYSSOTSKY: Which we were somewhat interested in. Our primary objective was to have a suitable programming and execution environment for the work of the engineers, scientists, and computing people in Bell Laboratories. What General Electric was primarily interested in, of course, was having a state-of-the-art operating system for their products.

So, we had three sets of motivations: MIT's motivation to advance the state-of-the-art, Bell Labs' motivation to have a good environment for our people to work in, and GE's motivation to strengthen its product line.

It turned out that under the stress of slipping schedules and the increasing realization that we had difficulty agreeing on a common course of action, we ended up simply pulling out of Multics. We said, "OK, it's too wet to plow. We aren't going to get from here to there."

REVIEW: *What stage was the project in at that point?*

VYSSOTSKY: Multics was limping. We actually had, I guess you would call it, a precursor of Multics running here on a GE 645. From the point of view of the few people who could use it, it was a very nice programming environment. In particular, Ken Thompson thought it was a very nice programming environment. When we pulled out of Multics, we simply took it off our machines and put up GECOS instead.

GECOS was not nearly as nice an environment from Ken's point of view as even the limping Multics had been. But if you were an old line Spanish-American War type computer user like me, GECOS was a perfectly satisfactory system for getting from here to there on a well-defined application. You knew what it was going to do. It was nowhere near as satisfactory if you were trying to do things that were technically difficult and imperfectly defined, which is the main task of research. This is the reason Ken Thompson and many others felt that GECOS wasn't satisfactory. It is also the reason incidentally, why I had been so interested in something like Multics in the first place. Although I say GECOS was a good system, it was good provided you were doing a reasonably straightforward application.

REVIEW: *So you wanted a more flexible system?*

VYSSOTSKY: I wanted a much more flexible system than BESYS or GECOS or OS360 or anything I could see. I had various things that I was trying to do with computers that were just plain hard to do with existing operating systems. That is not a criticism of OS360 or GECOS. These were good mainstream systems for what they were intended for.

Moreover, for people like Ken Thompson, having this embryonic version of Multics taken away and GECOS slapped down in its place was something of a disaster. Suddenly they were back to square one.

REVIEW: *They had been benefiting from all the people working on Multics. Were Thompson and Ritchie also working on the Multics project?*

VYSSOTSKY: Neither Ken nor Dennis regarded himself as a mainstream member of the Multics endeavor. They contributed things, though Ken worked on at least one version of the Multics editor. Dennis worked on a version of Multics I/O. But I don't think, for example, that you will find either of their names on any of the significant papers that came out of the Multics experience. They were around the effort but

It was the familiar second system syndrome. You put in everything you wished you'd had in the other one.

Multics was not their main interest. I don't think that either of them was particularly fascinated by operating systems until they found themselves cast back upon GECOS. They sort of got interested in the subject out of self defense.

My recollection is that neither of them wanted to do an operating system for its own sake so much as they just wanted to get their research done. When Multics went away, they did something. You know, they didn't just sit around and cry. Still, they did not regard the GECOS environment as satisfactory from their point of view.

After a while, Ken got the use of a semi-orphaned PDP-7, which at that time sat up in the attic at Murray Hill.

REVIEW: *How big a machine was the 7?*

VYSSOTSKY: Tiny. I don't remember how fast it was or how much memory it had, but it was a machine more nearly in the class of a Commodore 64 than the class of a PC-AT. Its chief virtue was that it was available. Ken started putting together an environment. The very first thing he ran on it was a space travel game.

He and Dennis and a few others gradually started tinkering and adding to this thing. The two most active contributors at that stage were Joe Ossanna and Rudd Canaday. I should also add that Doug McIlroy was tremendously influential on their thinking. I don't think that Doug actually contributed much of the programming, but for example, the appearance of pipes in UNIX was clearly a result of Doug's discussions with Ken and Dennis. Dennis actually put them in as I recall, but it was McIlroy who said, "Look, you ought to do it." Pipes, like most things in UNIX, were not a radically new idea. Co-routines had, after all, shown up in SIMULA by the end of 1967.

REVIEW: *You mentioned that in the design stages of Multics things got rather complicated. Do you think this complexity had much effect on Ritchie and Thompson's elegant design of UNIX?*

VYSSOTSKY: I would say it differently, I would say that the greatest intellectual achievement embedded in UNIX is the success Ken Thompson and Dennis Ritchie had in understanding how much you could leave out of an

operating system without impairing its capability. To some extent, that was forced by the fact that they were running on small machines. It may also have been a reaction to the complexity of Multics, but more than anything else, I would say, it was simply a manifestation of how good those two guys are when they set their minds to thinking about something.

Multics was simply more complicated than it needed to be, but I certainly didn't realize that at the time. I only knew it was complicated.

REVIEW: *Do you think UNIX has fulfilled the vision of Multics?*

VYSSOTSKY: You can still find many people who will point out that Multics is in various respects more capable and actually superior to UNIX.

REVIEW: *In ways that cannot be added on to UNIX as an application?*

VYSSOTSKY: For example, embedded in Multics are security features which are necessarily part of the architecture of a system. You do not get the simplification of UNIX for free but you lose very little. That was what we didn't realize in 1965. A lot of the stuff we were doing had passed over the shoulder of diminishing returns. I'll give you a trivial example. I was startled in the early days of UNIX when I realized that UNIX as an operating system was incapable of doing binary-to-decimal conversion. I had never seen an operating system that didn't do binary-to-decimal conversion and I took it for granted that every operating system had to include binary-to-decimal conversion. It took a bit of thinking and a bit of direction from Ken and Dennis to realize that they were right and that an operating system does not, in fact, have to do binary-to-decimal conversion. That's a utility function that can sit way up in the hierarchy of software.

REVIEW: *To be used when needed.*

VYSSOTSKY: Now that's a trivial example, but there was all sorts of stuff that people put into operating systems believing that otherwise they wouldn't be good. It took some very clear thinking on the part of the creators of UNIX to realize that most of that stuff didn't have anything to do with the operating system and didn't have to be included.

Multics was simply more complicated than it needed to be, but I certainly didn't realize that at the time.

REVIEW: *So Multics had a large multipurpose kernel like OS and all the others?*

VYSSOTSKY: Yes. That's an observation that looks like old hat now, but in 1969, it was new stuff.

REVIEW: *Is there any event that you feel was particularly important to the development of UNIX?*

VYSSOTSKY: There is one piece of history that I think is very important to understand. When UNIX evolved within Bell Laboratories, it was not a result of some deliberate management initiative. It spread through channels of technical need and technical contact.

The first version that was fairly widely used outside of the research area of Bell Laboratories was propagated largely through a group called USG in Berkley Tague's department.

The Programmers Workbench and the PWB version of UNIX was created in Rudd Canaday's department when Rudd was at Piscataway. That work started in about 1973 and came to fruition in about 1975. Rudd had been one of the original contributors of ideas to UNIX when he was here in computing research. Three of the other contributors to the PWB and PWB/UNIX were Ted Dolotta, Evan Ivie and Dick Haight. Both had spent periods of years in computing science research.

The PWB version of UNIX was created when I was at Piscataway. It was done in my organization and I recall being somewhat doubtful about the whole thing.

REVIEW: *What were your concerns?*

VYSSOTSKY: The objective of Canaday's department was to provide an improved programming environment for large software development in Piscataway. At the time that they started work on PWB/UNIX, UNIX as it then existed was rather fragile, somewhat limited in its capabilities, and unable to handle much in the way of load. I was working in a shop where we had very large applications and it wasn't clear that the source files for a major application could be made to fit into a UNIX system. On the other hand, it was well known that the frequency with which UNIX lost files or did similarly calamitous things was high.

REVIEW: *All too well known.*

VYSSOTSKY: It just wasn't clear that you could base a programming environment for production programming on UNIX. Recall, this was 1973 when we were


CODATA

engaged in discussions about whether or not UNIX was a sensible direction to head in. I was kind of queasy about the whole thing but I figured Canaday was paying those guys to have great sense, so I backed them up. Afterwards, I was awfully glad I had.

REVIEW: *How long afterwards were you glad?*

VYSSOTSKY: [laugh] Two years. It took them about two years before they could actually provide a realistic production programming environment for groups like the large programming shop at Piscataway. They presented a set

UNIX was not something that would have warmed the heart of a senior EDP manager.

of papers at a meeting in San Francisco around 1975. At the time those papers were presented, we were actually using PWB/UNIX as a vehicle for production programming.

This was typical of the way UNIX spread around Bell Laboratories. You had MTSs, Supervisors and Department Heads saying we had to go in this direction while Executive Directors were saying, "Well, I'm awful nervous about it. But if you guys say that is what we've got to do, I'll back your play."

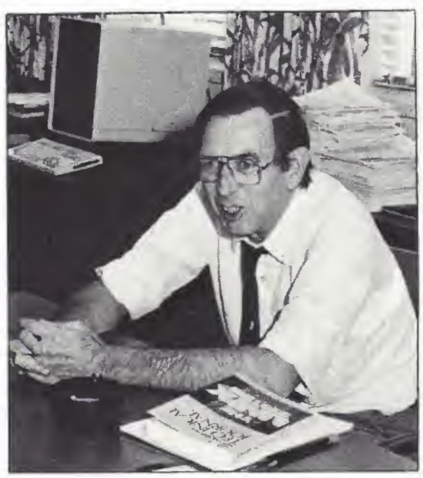
REVIEW: *It's probably pretty unusual for you to give your people their head in that way.*

VYSSOTSKY: [Contrite] It's not all that unusual. Bell Laboratories works that way.

REVIEW: *What about other organizations?*

VYSSOTSKY: There are a lot of organizations that do not work that way. I brought out that little hunk of history to point out that the spread and success of UNIX, first in the Bell organizations and then in the rest of the world, was due to the fact that it was used, modified, and tinkered up in a whole variety of organizations within Bell Laboratories.

All of the basic philosophy



and technical components were there when Thompson and Ritchie and their friends stepped out of it. But, it was not something that would have warmed the heart of a senior EDP manager. It was what I'll call an "annealing" process that took place in the Bell Labs development organizations that turned UNIX into a viable vehicle for the world at large. The refinement of UNIX was not done as the result of some management initiative or council of vice presidents. It was the supervisors saying, "This thing is already better than our other options and flexible enough for us to make it go."

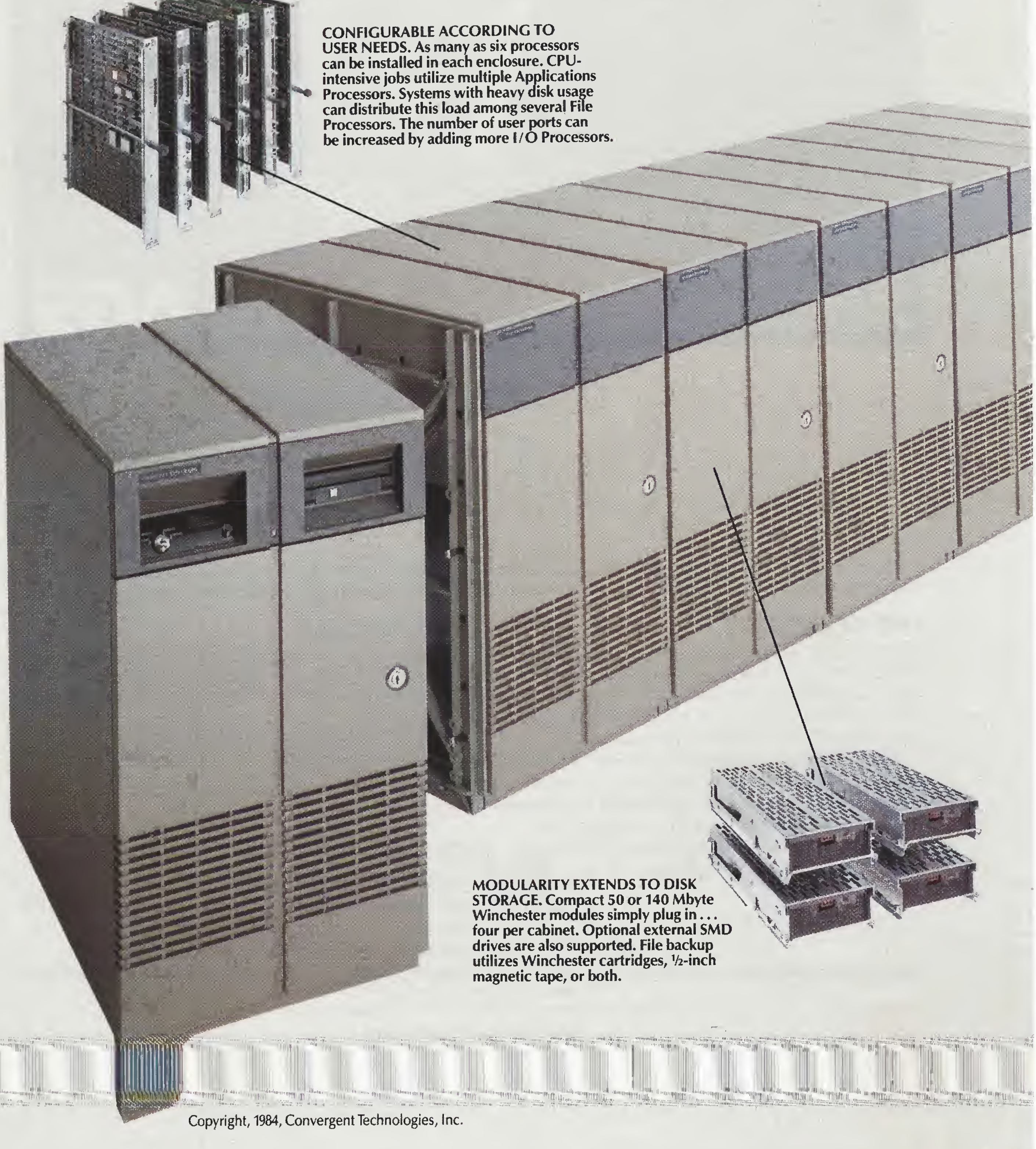
REVIEW: *How early was UNIX tracked in development as a product? Was it a controlled product right from the start?*

VYSSOTSKY: It certainly was not a controlled product initially. As late as 1976, there were three different versions of it in use within Bell Laboratories that were supported out of different places. There was the USG that Berkley Tague was supporting from here at Murray Hill, there was the PWB version that Canaday and friends were supporting out of Piscataway, and there was CB/UNIX which was being supported out of Columbus [Ohio] and had better inter-process communicaton capability. CB/UNIX was better adapted to handling applications like the switching control center applications. It was not a real-time UNIX per se. True real-time versions of UNIX such as UNIX RTR didn't come along until later.

REVIEW: *Were you using a software development tracking system yet?*

VYSSOTSKY: In 1976, there were those three versions of UNIX. The Change Control Process on all three of those versions was such that, at any moment in time, the people who were programming could tell what changes had gotten in and what changes were scheduled to go in. However, it was still a little hard for the users to tell what they were getting. It wasn't until about 1978 that we had anything that I would consider to be a reasonable configuration management process for UNIX. That was the point at which we finally realized we had something which, like it or not, was a major product. So we said, "Given that it is a major product, there can be no horsing around." We could no longer regard it is something in the underbrush. We had to regularize our arrangements. We set up a process

IT WILL GROW ON YOU.



CONFIGURABLE ACCORDING TO USER NEEDS. As many as six processors can be installed in each enclosure. CPU-intensive jobs utilize multiple Applications Processors. Systems with heavy disk usage can distribute this load among several File Processors. The number of user ports can be increased by adding more I/O Processors.

MODULARITY EXTENDS TO DISK STORAGE. Compact 50 or 140 Mbyte Winchester modules simply plug in . . . four per cabinet. Optional external SMD drives are also supported. File backup utilizes Winchester cartridges, ½-inch magnetic tape, or both.

MegaFrame™. Now OEMs can offer a high-performance UNIX™-based system that can't run out of performance.

OEMS can now deal cost-effectively with the problems encountered when user applications produce computing demands that outstrip the capabilities of conventional systems.

Convergent Technologies' MegaFrame is a revolutionary new UNIX-based super-minicomputer—so innovative in its architecture that it represents the ultimate in multiuser systems design. It grows exponentially from a system offering minicomputer-level performance to an enormously powerful engine serving as many as 128 users with 36 parallel processors, 24 megabytes of RAM and gigabytes of disk storage.

No other system can match the MegaFrame's potential for field expansion. It enables manufacturers and systems builders to keep pace with today's requirements for more and more computing services . . . but *not* at the cost of discarding hardware or performing expensive CPU upgrades.

MegaFrame's architectural breakthrough. Dependence on traditional single-CPU shared-logic architecture is the root of systems bottlenecks.

Convergent's response: a novel system utilizing *multiple* specialized processors to distribute workloads for optimum performance—even if user needs are unpredictable or subject to rapid change.

MegaFrame's virtual memory Applications Processors each have a 32-bit CPU, up to 4 Mbytes of RAM and run a demand-paged version of UNIX System V. Up to 16 of them can operate in parallel.

The File Processors effectively function as back-end machines providing DBMS, ISAM and other disk-related services. Up to six File Processors each with four disks can operate in parallel.

Terminal and Cluster Processors can also be added—the latter serving front-end communications needs. They off-load communications from the other processors by running protocols such as SNA and X25 networks.

MegaFrame's daisy-chained cabinets offer total expansion potential of up to 36 slots. OEMs configure the system needed for specific applications simply by adding the correct number/combination of processors.

Flexibility in applications development. Inclusion of one or more Applications Processors allows running UNIX System V. All standard UNIX tools are provided, along with COBOL, FORTRAN-77, BASIC interpreter and compiler, plus Pascal.

The "least-cost solution" to serving a wide range of UNIX-systems needs, MegaFrame has won acceptance from OEMs in the U.S. and abroad. The uniqueness of its modular design, its versatility in providing upgrade-path options and its price/performance advantages give it market-share potential of outstanding dimensions.

The system that will grow on you starts at a very attractive price: about $20,000 for a system that effectively supports 16 users. Send now for a comprehensive Information Package including reprints of magazine articles. It explains how MegaFrame's growth potential can impact favorably on your plans for growth in the UNIX market.

Convergent Technologies, Data Systems Division, 3055 Patrick Henry Drive, Santa Clara, CA 95050. Phone: 408/980-0850. Telex: 176-825.

MiniFrame™: the entry-level multiuser UNIX system.

Starting at under $5,000 for a single-user system, Convergent's MiniFrame offers outstanding capabilities for small to medium sized organizations running large UNIX-based applications. Utilizing an MC68010 microprocessor operating at 10Mhz, with no wait states, it provides impressive CPU speed—comparable to VAX™-11/750 running the AIM™ Benchmark. MiniFrame features virtual memory management, with demand-paged implementation of UNIX System V. It runs as many as eight terminals, with up to 50 Mbytes of integral mass storage. MiniFrame and MegaFrame are object-code compatible, allowing OEMs to offer a *complete family* of systems unrivaled in price/performance characteristics.

Convergent Technologies

Where great ideas come together

Circle No. 27 on Inquiry Card

for configuration management and we focused the thing in the direction of a coherent system. This is how System V was formed.

REVIEW: *Was the change in management policy because of a perception of future revenue possibilities? Or, was it, "We have this thing out there, so we should control it or we are going to have problems with a haphazard product"? Or, was it more like, "This is useful but is it going to become less so if the evolution is not directed"?*

VYSSOTSKY: Some of all of those things. But, perhaps the most important one was that UNIX was being used as the operating system base for a bunch of operations support systems in the Bell Operating Companies and we could not afford to let those support systems go down.

We put configuration management and all of the associated paraphernalia in place about 1978.

REVIEW: *It seems that more effort is going into future planning now and not just fixing up the old code. Would you agree with that perception?*

VYSSOTSKY: Yes. Fixing bugs is not all that big a deal. As far as directing the future is concerned, I think that is a big deal. When we talk about UNIX, we tend to talk as if operating systems had some inherent importance in the eyes of customers or users.

But operating systems from my point of view are not so important in themselves as they are important because they provide people with an environment in which to create applications. From the point of view of the end user, it is the application that is important. The thing that has made UNIX so attractive is the fact that it is a good environment in which you can quickly get an application up that works for the end user.

Future development becomes an issue because UNIX is not, in fact, as good as we would like for creating certain types of applications. UNIX is still wobbly around the knees if you want to create a transaction system that has to handle a large complex database. The worst aspect of that wobbliness, at the moment, is the fact that there is not yet a database management capability in UNIX systematic enough to allow one to

The thing that has made UNIX so attractive is the fact that it is a good environment in which you can quickly get an application up.

create database administration tools in any standard straightforward way. The question of whether one can put the database into the system, so far as I'm concerned, is a non-issue. I've convinced myself that one can take essentially any arbitrary database and put it into a UNIX system.

REVIEW: *There are plenty of examples.*

VYSSOTSKY: Yes. The problem comes up from the point of view of the database administrator in the field installation who has to be able to control that database. They have to keep it consistent, update it, roll it back, recover it. There is no standard set of tools. Each application has to provide its own set of tools for database management.

REVIEW: *Each with its own learning curve?*

VYSSOTSKY: Yes. We have to improve that situation. Similarly, the proliferation of UNIX-like real-time systems within AT&T indicates that we still do not understand real-time as well as we should.

REVIEW: *Would you like to regularize that also so that there is a single operating system supporting real-time applications?*

VYSSOTSKY: I would like to understand real-time. We were able to make UNIX converge in the face of a diversity of evolutionary directions and the use of at least three major versions.

The reason that incredible spread occurred was not because everyone insisted on going it alone for their own reasons. It was because people had slightly different needs and at the time didn't see how to meet those various needs with one version of the operating system. By 1978, we found that we could drive the pieces back together.

REVIEW: *But you didn't have to force it together. You provided the users with something that was similar to what they had but which was regularized.*

VYSSOTSKY: Yes. I guess what I am saying is that in real-time, I perceive us as now in a situation that is analogous to where we were in the mid- 70s on UNIX itself. That is, for example, when I look at some applications using ORYX/PECOS and others using NRTX, it is quite clear that the ones using NRTX couldn't use ORYX/PECOS and vice versa. That doesn't mean that you can't build a system with UNIX-like features that would meet the needs of that entire spectrum. It just means that we haven't

understood the applications needs well enough to cause the convergence.

I think it's an important thing to try to do. Not just for the sake of having one operating system. I have no inherent objection to having multiple operating systems. But, the fact that I can't give a coherent exposition of why we have different real-time operating systems in use for different applications tells me that I don't understand the subject as well as I should. And we need to understand that issue.

This is an area where I think there needs to be an evolution of UNIX-like systems. I say "UNIX-like" because it is not clear to me that "real-time" and "UNIX" are synonymous.

Another area where the evolution of UNIX-like systems needs to proceed is distributed computing. I think that research Edition 8, with Dennis Ritchie's new stream I/O, may have actually gotten past the biggest hurdle on distributed computing. I'm not sure yet, but it looks hopeful. Distributed computing is a subject we have been worrying about for years. Maybe five years from now, we will be able to look back and say, "OK, with research Edition 8, we've got that one beat." I hope so.

As yet, we don't, from my point of view, have a satisfactory understanding of how to get full capability out of a UNIX system on a network which is running on a very large, very fast machine. We are bringing in a Cray computer in 1985 with one of the primary purposes being to attack exactly that problem. Our local computing environment here in research is mostly a collection of fairly small machines. But we have some processor-intensive jobs that we run in the comp center downstairs.

REVIEW: *You've been batching them to the Cray?*

VYSSOTSKY: Yes, we've been batching them to the Cray though RJE and it works — though not satisfactorily from our point of view. What we'd like to have is a nice distributed computing environment where, from the point of view of the person sitting at the terminal, the only difference between a Cray and a 3B2 is that the Cray runs faster and has more storage. We're not there yet. That is really quite a difficult issue. Precisely because the Cray is so fast, 'impedance matching' the Cray to a network capability is difficult.

Distributed computing is a subject we have been worrying about for years.

REVIEW: *What other areas of development to you foresee?*

VYSSOTSKY: We also need to get a better generic of UNIX than we yet have for the purpose of running on very small machines, personal computers, and things of that sort. Recall that the very earliest version of UNIX ran on a machine so small that nobody would even consider it as a serious computer today. But, over the years, UNIX has grown and grown and gotten more and more powerful, which is great. Except that UNIX is pretty slow and pretty clunky if you run it on a machine as small as a PC and there is no reason why it has to be. It just is.

Part of the problem is that we don't understand the structure of UNIX and programs that run on UNIX well enough to say, "Well, what can peel back out of there to get a nice small system that truly looks like UNIX?"

REVIEW: *You are not just talking about stripping out commands?*

You are talking about shrinking the kernel?

VYSSOTSKY: That is exactly what I am talking about. One still wants multi-programming.

REVIEW: *But you don't need to run 100 processes.*

VYSSOTSKY: Right, you don't need to run 100 processes. Look, it just isn't clear to me how to strip things out in the right way. But, it is clear that it must be possible to do because the very early versions of UNIX ran just fine on teeny machines.

The reason that I think that's important is because what I want to be able to do and what I think a lot of people want to be able to do is use a personal computer or a workstation that is sometimes pulled off a network for reasons, like privacy reasons, but which is often connected to a network. While the operating system in the small personal machine looks completely consistent with the operating system in the larger distributed environment, it

Continued to Page 102

INDUSTRY INSIDER

Desktop computers and networks

by Mark G. Sobell

"What is a desktop computer?" and "Do *desktop* and *UNIX* belong together?" were two questions Jim Barclay, Software Consulting Engineer in the UNIX Engineering Group at DEC, raised when I asked him about desktop UNIX machines in the works at DEC. Of course, he hastened to assure me that DEC had all the bases covered, noting that "the Rainbow people are working on UNIX with VenturCom and Microsoft and we sell VenturCom's Venix for some of the machines in our PRO series. But the real value and original intent of UNIX was resource sharing to promote the easy movement of thoughts, ideas, and software chips that you can easily piece together and share. By software chips I don't just mean the UNIX utilities and shell scripts. I also mean small, custom pieces of C code that are dedicated to doing one thing well."

But what happens when you use a desktop system? First off, you end up having to move away from your desk to share your work with co-workers. Memos have to be printed on paper and delivered by inter-office mail, data becomes harder to share, and everyone gets more isolated.

"I am not a strong advocate of single-user desktop computers," Barclay noted. "We need to be bringing a department together —

not isolating its members."

Tying powerful desktop PCs together is one solution. But frequently the mechanism that ties the computers together is cumbersome and, although theoretically sound, does little to promote actual team spirit within the department.

"We also need to teach people to properly use the tools they already have," Barclay lamented. "My secretary just told me that someone sent me a 48-page piece of electronic mail. I'm afraid to look at it. All I need to know is where the document is and what it's about. Then I can decide what I want to do with it — read it, copy it to my system, or ignore it."

In order to judge the effectiveness of any computer system — desktop or otherwise — you need to be able to measure its effect on company productivity.

The ideal system would make it easier for people to share thoughts and ideas. It would tie into much more than simply a network of UNIX systems; it would also allow communication with people using Macintosh machines, programmers logged into corporate IBM mainframes, and R&D types running UNIX on minis.

SUN'S NEW NETWORK SOFTWARE

The Sun Network File System (NFS) comes very close to providing this sort of ideal system. It encompasses a transparent file system that spans a network. It took me a while to understand just *how* transparent it is when Beau James, Product Manager at Sun Systems, explained it to me.

Instead of needing to log onto another system on the network to read a file or needing to copy a file to your system before reading it, NFS makes remote file systems appear as though they were actually mounted on your local computer.

Each administrator of a system on the network designates the local file systems that he or she wants to make available to other people on the network. The system, called the *server* in this case, exports the designated file systems.

Each administrator also decides which of the file systems operated by other systems would be useful on the local system. The administrator can select from available (exported) file systems using Sun's *yellow pages*, a collection of network protocol and associated programs for use on the local system, it is called a *client*. Once a client has mounted an imported file system, you cannot distinguish the newcomer from a local file system. You can read from, write to, modify, or delete files in the file system as long as the owner of the file has given you the appropriate permissions.

Perhaps the most important thing about Sun's new network is that it is not locked into UNIX. It is the job of the NFS software to make imported file systems appear the same as local file systems, even when the imports come from a different hardware/software environment.

NFS makes remote file systems appear as though they were actually mounted on your local computer.

Because the ability to communicate with other manufacturers' hardware and software is of limited value unless the other manufacturers support the NFS standard, Sun would like to see its network become the industry standard. It hopes to ensure that other companies adopt it by publishing the protocol specifications for the entire NFS system and giving away *source* code for all its portable parts (the parts that are not operating system dependent).

The NFS software dovetails with another new Sun product, the Sun-2/50 — a diskless node that sells for under $10,000 and features a high-resolution bitmapped screen, 1 MB of memory, a CPU, an Ethernet interface, and necessary software. With NFS software, the diskless node can import file systems in such a way that the user is unaware that the local system doesn't have a disk.

On other Sun fronts, it was learned that Eastman Kodak just invested $20 million in Sun and now owns 7 percent of the company. Sun has a $27 million line of credit and had sales of $39 million for the year ending June 30. It doesn't sound like Sun is a candidate for the shakeout.

MORE NETWORKS

Another network software package that is gaining momentum is Microsoft's Networks. HP has picked up the Networks for use on its personal computers. HP says Networks will run in conjunction with HP AdvanceNet, its standard networking protocol, to enable all HP computer families to talk to each other.

Another vote for Microsoft's Networks comes from Corvus Systems, which announced it is licensing the product for use with its OMNINET network.

DESKTOP VENIX

Back on the desktop front, I talked about Venix 2.0 with Gig Graham, Executive Vice President of VenturCom. The biggest questions raised in my mind by VenturCom's efforts to put UNIX (or Venix, VenturCom's port of UNIX) on a desktop machine without a hard disk were: "Will it run fast enough to be usable?" and "How do you manage to fit all the UNIX utilities *and* the user's data on the system?"

The latest release of Venix helps alleviate the speed problem with *sticky memory*, a capability akin to the UNIX sticky bit that keeps a program from getting swapped off the swap disk. Sticky memory keeps a program in memory as long as another program doesn't need the space. It is implemented as a *fifo* list of programs that always tries to hold the most recently run programs in memory.

As far as fitting UNIX onto a floppy disk, Graham maintains that the kernel and most individual utilities are actually quite small. (However, the *entire* Venix system ships on 10 floppies.) VenturCom's solution to the problem of fitting UNIX on a floppy-based computer is to provide the user with a system disk that includes a subset of the utilities — the user has full use of the second disk for data and application programs. This approach will be satisfactory in environments where the user only needs to use as many programs as will fit on the system disk (including the kernel, swap area, device drivers, and system files). It will be tedious, however, to use the system with more programs because, unlike MS-DOS, you cannot just pop a disk out and insert a new one. With a UNIX system, you must unmount the disk, remove it, insert the new disk, and finally mount the new disk before using it.

MORE PORTABLES

Venix is the UNIX that Data General has selected for its DATA GENERAL/One portable computer. But, DG is not putting all its eggs in the UNIX basket. It is also pushing its "One" in other markets. DG provides CPM-86, MS-DOS, LISP, and an Ada editor for checking the syntax of Ada programs. It also provides four levels of service from five-day mail-in to two-hour walk-in service. The machine, which has a full-sized screen, uses fast (150 ns) CMOS memory with no wait states, an 80C88 processor (the CMOS version of the 8088), and can run 8-10 hours on one charge of its batteries. DG even has a battery-powered printer to go along with the One.

Sharp Electronics is taking another tack in speeding up its portable/desktop UNIX system. It is test marketing Venix on its 8088-based PC 5000 portable computer and says that if Venix is well received, it will be placed in ROM to improve performance of the system. That would rate as quite a feat — *if* Sharp can pull it off.

The approaches of both VenturCom and Sharp are interesting and unusual. It remains to be seen, though, whether they are viable, and if so, in what environments.

SUMMARY

It's pretty clear that computers are getting smaller, faster, and more connected than ever. The fact that they're getting smaller and faster is good and necessary. The interconnection is essential if computers are going to be more useful than mere number crunchers. With good connections, computers can improve communication between people, allowing them to become more productive.

If you have an item that is appropriate for this column, you can contact Mr. Sobell by electronic mail at ***ucbvax!olympus!its!mgs*** *or by US mail at UNIX REVIEW, 520 Waller St., San Francisco, CA 94117.*

Mark G. Sobell is the author of "A Practical Guide to the UNIX System" (Benjamin/Cummings, 1984). His 10 years in the computer industry include programming, technical writing and management experience. Mr. Sobell has been working with UNIX for four years. In addition to consulting in the San Francisco Bay Area, he teaches, lectures and writes. ■

RULES OF THE GAME

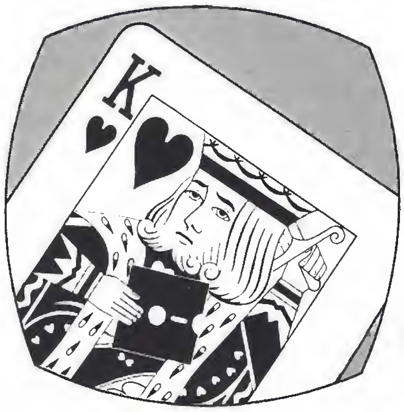
A better mousetrap?

by Glenn Groenewold

How often have cartoonists evoked the image of someone sitting in the waiting room of a patent attorney, cradling an outlandish contraption bristling with crankhandles, indicator lights and antennae! Often enough that this caricature has come to represent the concept that many of us have of a patentable invention.

Perhaps the Patent Office and the Supreme Court were influenced in similar fashion, because for years both refused to concede that there was a valid theory which would permit an ordinary computer program to be included within the scope of a patent. It finally required a contrary decision by a bitterly divided Supreme Court to persuade the Patent Office to change its mind.

This history should not be surprising, actually. American patent law comprises one of two branches from the same limb of our Constitutional tree — the other being the law of copyright. Originally these domains were easy to differentiate. Patents secured to "inventors" an exclusive right for a limited time in their "discoveries"; copyrights accomplished the same thing for "authors" with respect to their "writings". It was not easy to see how computer programs could fit into either of these categories — let alone both of them, as things finally turned out.



The patent law that Congress has written under its Constitutional authority states that whoever invents a "new and useful" process *or* machine may obtain a patent. At first glance, it would seem logical to assume that if a computer program were to fit into the patent concept at all, it would be as part of a *process* used to bring about a desired result. However, the Supreme Court had difficulty accepting this approach, since it viewed a computer program as nothing more than a series of mathematical formulas, which are not patentable.

For a while, patents were issued anyway, under the compulsion of decisions by the Court of Customs and Patent Appeals that in essence viewed a computer program as part of a *machine*, since it caused the computer to function in a different fashion than it could have otherwise. As one opinion of this court put it, "if a new machine has not been invented, certainly a 'new and useful improvement' of the unprogrammed machine has been...."

After the Supreme Court got around to tossing this rationale out the window, things looked bleak for the patentability of computer programs. But as so often happens, another case came along in which a new majority on the Court succeeded in overcoming the effect of its earlier rulings.

By a 5 to 4 vote, the Supreme Court decided that a *process* using a computer program to control the curing of molded synthetic rubber could be patented. However, the bare computer program by itself — including the algorithms — still cannot be the subject of a patent. And there the matter stands, at least for now.

Whether or not this judicial distinction can be said to make much sense, it does permit a computer program to comprise part of a process which can be patented. Still, this method of protecting intellectual property has not been widely utilized for computer software.

GETTING A PATENT

For one thing, patents are not easy to obtain. Unlike copyrights, which come into existence automatically, a patent is granted only after the Patent Office has conducted a thorough investigation. The procedure is drawn-out and often costly — $15,000 to $25,000 being considered a ballpark figure for electronics patents.

One reason the process is so expensive is that specialized professionals are almost always necessary to prepare the application, which must be presented in such a way as to make it clear that the innovation is indeed patentable. There are some people who find it a challenge to do this for themselves, just as there are folks who get a charge out of building their own aircraft and flying around in them. But most of us feel more secure relying on professionals in such matters.

By contrast, a copyright registration costs only $10, is processed promptly, and seldom requires professional assistance. Take the case of Mary and Joe, owners of GarageCo, a small operation with limited resources. Though they would like to patent their new process, including the applications program central to it, they may have to rely on copyright (or trade secret) protection instead.

The bare computer program by itself — including the algorithms — still cannot be the subject of a patent.

The reason for the discrepancy between the patent and copyright processes is that the Patent Office attempts to determine before issuing a patent that it will be valid. (The copyright office assumes no comparable burden.) Making this determination is no simple matter.

The law requires, among other things, that the subject matter of a patent be useful and novel (that is, that it not be an obvious solution or merely a fresh combination of "prior art"). The law also demands that the thing to be patented not be something that has been known or used by others in this country and that it has not been previously patented or described in a publication in this or any foreign country. These requirements not only make plenty of work for the Patent Office but also provide a fertile ground for attacks on the validity of the patent once it has been granted.

Now we can have some appreciation of the problems involved in preparing a patent application, and why it's no job for amateurs. Many a potential patent has foundered because of an application that made it appear that the innovation merely strung together existing technologies. An application of this sort will be turned down on the basis that it constitutes nothing more than an application of "prior art." In the past, many computer progams have been denied patents for just this reason.

Assuming a patent application has successfully surmounted these hurdles, what does the holder of the patent get in return for all the trouble?

Not all that much, in the view of a lot of people. True, the patent does confer an exclusive right of use for a period of 17 years, and there is no "fair use" exception, as there is with copyrights. (This exclusive right has to be exercised without running afoul of such things as antitrust laws, however.) But there can be *no extension* of the 17-year period.

Consider, moreover, that a United States patent has effect only within this country and its territories; it's necessary to obtain separate patents for each country in which the patented material will be used. Though US law provides a year's grace period within which to apply for a patent after the innovation has been disclosed or made available to others, most other countries do not. So any desired foreign patents usually must be requested before the subject matter has been released.

In addition, the penalties that the law provides for patent infringement are not as severe as those for copyrights. And the rate of failure for patents in litigation has been horrendous—over three-quarters of the patents that have been challenged have ultimately been found invalid.

It's also necessary to disclose the subject matter of a patent upon its issuance, and while this can't be used commercially by anyone else during the 17-year exclusivity period, it can give other innovators a leg up in devising something that may be better than the patented item.

WORTH THE TROUBLE?

After all this, you may wonder why anyone bothers with patents. An answer, of course, is that with all their drawbacks, patents still constitute the only feasible form of protection for some intellectual property.

Suppose, for example, that a program comprising an essential part of the operating system for a computer can be written only one way. Such a program could *not* be copyrighted, because as the courts view it, the idea behind the computer program and its expression have "merged". Or to put it another way, they are one and the same. This is the possibility that the appellate court raised in *Apple vs. Franklin*, which we discussed a few columns back. Moreover, the manner in which the program is distributed for use may make it impossible to maintain as a trade secret. The only way, then, to protect this program would be to endeavor to include it within the scope of a patent on the computer and its operating system.

Another advantage of the patent route is that in some other countries it may provide better protection than a copyright. A number of nations that are trying to catch up in the software field have demonstrated reluctance to enforce American copyrights. In countries where a patent can be obtained, this difficulty should be obviated.

However, a patent will probably not be worth the trouble for an innovation that is expected to have a brief useful life unless it is *basic*, with subsequent development dependent on it.

This all brings us to the point that patents often complement other forms of intellectual property protection. During an in-house R&D phase, a process can be guarded as a trade secret. This can continue to be done even after a patent application is filed, since the contents of the application remain confidential until the patent is granted. And that may take years.

The rate of failure for patents in litigation has been horrendous.

Even after issuance of the patent and the accompanying disclosure of the innovation, further refinements in the process can be kept as trade secrets and licensed accordingly. And documentation concerning the patented process can be protected by copyright.

The thing to remember is that all of the legal devices for intellectual property protection have their uses. It's perilous to concentrate on one of them and forget about the others. But if you're thinking about the possibility of a patent, it's essential to get expert advice at the outset.

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. ■

C ADVISOR

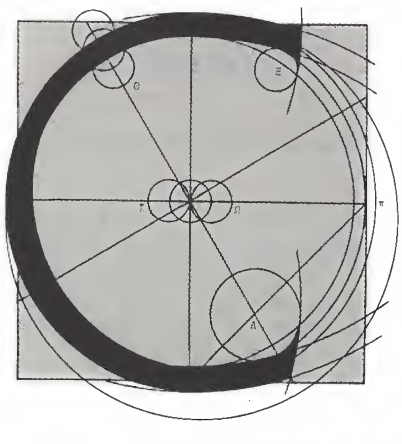
The Evolution of C: Heresy and Prophesy

by Bill Tuthill

C is descended from B, which was descended from BCPL. BCPL (Basic Combined Programming Language) was developed in 1967 by Martin Richards. B was an interpretive language written in 1970 by Ken Thompson[1] after he abandoned a Fortran implementation for the PDP-7.

BCPL and B were typeless languages, which may account for the type permissiveness of C. They restricted their scope to machine words and were rather low level. However, they provided structured programming constructs similar to those in Algol. BCPL, B, and C all provide pointers and address arithmetic. All three pass function parameters by value, rather than by address, but permit passing by address if desired.

The first real C compiler was completed in 1972, at which time the only supported types were **char**, **int**, **float** and **double**. After the addition of structures in 1973, the UNIX kernel was successfully recoded into C, which helped rationalize and organize the operating system.[2] The C language continued to evolve, largely because of porting efforts, and was finally codified in 1978.[3] By then, the type adjectives **unsigned**, **short**, and **long** had been added, in addition to the type aggregates **union**, and **typedef**. More importantly, an efficient and portable



Standard I/O Library had been developed.

C is gradually losing its type flexibility and becoming more and more like Pascal, the language of wimps and quiche-eaters. In 1980, two new types were added, **void** and **enum.** Both were derived from Algol 68 and Pascal. The first is used primarily to make C code harder to read, but easier for **lint** to digest. The second might actually be useful. C routines that return a value are comparable to Pascal functions, whereas functions declared as **void** are comparable to Pascal procedures. At this point, it may be a good idea to avoid **void** and **enum** because most C compilers on cheap micros don't yet support them.

This is the current state of most C compilers. The C compilers on System V and the C compiler on 4.2 BSD are almost identical. The Berkeley compiler was the first to offer infinite-length variable names in order to support a companion Pascal compiler. The System V.2 compiler now offers the same feature. Despite their advantages, infinite-length variables can be expensive to implement and can cause portability problems when software is moved to systems with older compilers. Dennis Ritchie has to be commended for shepherding C programmers onto a narrow path. Despite the proliferation of UNIX versions, there is but one C.

ANSI STANDARD C

A sign that C has become a major commercial language can be seen in the current effort to standardize it. The American National Standards Institute (ANSI) has formed committee X3J11 to handle the task. Their draft standard that has not yet been approved. Nonetheless, I believe this C standard represents the most significant step forward for the language since Kernighan and Ritchie's white book appeared.

The standards committee is divided into three subcommittees: environment, libraries, and language.[4] Programming environments are a hazy area. Most likely, the **main(argc,argv)** argument passing convention will be

required on all operating systems. However, the function of the UNIX environment cannot be duplicated on many other operating systems, so the third parameter **envp** (environment pointer) will probably be dropped. European character sets are still under discussion.

The library routines in section 3 of the *Unix Programmer's Manual* (except for Bessel functions) will be part of the standard. However, system calls in section 2 will be dropped because they cannot always be duplicated on non-UNIX systems. The exception to this is **signal(2)**, which is necessary to make C programs re-entrant.

The language subcommittee started with the System V.2 *C Reference Manual*. It should be noted, though, that there have been three major changes since the *C Reference Manual* was published with Kernighan and Ritchie's *The C Programming Language*.

Dennis Ritchie has to be commended for sheparding C programmers onto a narrow path.

First, identifiers are significant to 31 characters, rather than 8 (as on Version 7), or infinite length (as on Berkeley UNIX and System V.2). Originally the committee was going to limit external names to 6 characters without case distinction, but public outcry was so strong that this will be left as an implementation detail.

The second change is that structure and union assignment is possible, and that structures and unions may be passed as parameters. Member names are local to structures and unions, instead of being global.

Finally, the **void** and **enum** types have been added. A function returning no value actually returns the type **void**, and programmers can throw away unwanted values by casting to **void**. For example, you can throw away the value returned by **fclose( )** with the expression:

```
(void)fclose(fp);
```

The **enum** type allows you to replace ugly preprocessor code like this:

```
#define DEV202       1
#define DEVAPS5      2
#define DEV8600      3
#define DEVIMAGEN    4
#define DEVQMS       5
int dev;
```

with the more streamlined:

```
enum devtype { DEV202, DEVAPS5, DEV8600,\
     DEVIMAGEN, DEVQMS } dev;
```

Variables of type **enum** are treated as second-class integers — in Steve Johnson's Portable C Compiler, for example, arithmetic comparison (except equality) is illegal. The standard may change this, however, particularly because **enum** comparisons are allowed in Ritchie's C compiler.

The committee has introduced many changes above and beyond the System V.2 standard. Arguments to functions may be declared, for instance, if the programmer wants the compiler to check them. For example, you could say:

```
extern int fread(char *, int, int, FILE *);
```

In the event of a type mismatch, the same conversions as for the assignment operator apply. This means that **NULL** pointer arguments will no longer have to be cast to the appropriate type! Variable-argument functions can be declared like so:

```
extern int printf(char *, ...);
```

The convention for declaring functions that take no parameters is:

```
extern int rand(void);
```

The new data type **const** marks variables as read-only, with run-time assignment forbidden. This used to be done less reliably, and without placing an entry in the symbol table, by preprocessor definitions. The type **const** will be useful for data placed in ROM on special hardware. Also, it makes the **:rofix** kludge obsolete (this was a way to move data to text space by changing **.data** assembly code to **.text**).

If all operands in an expression are of type **float**, the compiler is allowed (but not required) to evaluate the expression in single, rather than double, precision. Casts may be used to force double precision evaluation. Numeric constants are treated as **double** unless explicitly cast to **float**. If function arguments are declared, passing a **float** to a function expecting a **double** will be harmless.

The preprocessor is part of the language definition. Its syntax has been extended with **#elif** so that **if-else** blocks can be coded more easily. Space before the sharp (#) will be permitted.

Two string constants (not variables) next to each other in the source code are considered concatenated. This makes it easier to continue strings across line boundaries.

The types **unsigned char**, **unsigned short**, and **unsigned long** are part of the specification. Johnson's Portable C Compiler has always accepted these types, but they were not defined in the *C Reference Manual*. Plain **char** may be either signed or unsigned depending on the implementation.

Promiscuous pointer assignments are considered illegal (on most machines they now just generate a warning message). You must use casts when mixing pointer types or mixing integers with pointers. A new kind of pointer, **void***, cannot be dereferenced but can be assigned to any other type of pointer without a cast. Before, **char *** was the universal pointer that could point to anything. The earlier declaration of **fread( )** should really be:

```
extern int fread(void *, int, int, FILE *);
```

The compiler will make the appropriate pointer conversions. The new storage class **volatile** (the name is tentative) means that the compiler should not optimize references to data bearing this label. This will make it possible to write better optimizing C compilers.

The selection expression of a **switch** can be of any integer type, including **long** or **short**. The unary plus operator (analogous with the unary minus operator) does nothing. This is for consistency with the library routines **atoi( )** and **atof( )**.

When a **union** is initialized, the type of the initializer is the type of the first member in the union. This may not be ideal, but it is simple.

Hexadecimal string escapes have been added. To put an ESC in a string, you say:

```
"Here's a real ESC: \x1b"
```

Previously you had to use octal escapes. This is an admission that hexadecimal is more common in the computer world than octal.

Some things will disappear. The keywords **entry**, **asm**, and **fortran** will not be part of the standard, though the last two may be recognized as valid extensions. The keywords **long float** will no longer be a synonym for **double**. The octal digits 08 and 09, which used to be interpreted as 10 and 11, will no longer be valid. It will be illegal to take the address of a **register** variable.

The names of arguments to functions will not be permitted to clash with the names of automatic variables. This code will be illegal:

```
function(arg)
int arg;
{
     int arg;
}
```

Some existing compilers interpret this as nested scope, where the inner declaration hides the outer one. No good programmer would write code like that anyway.

The chair of the draft standard language subcommittee is Larry Rosler of AT&T Bell Laboratories, the author of an interesting article on C evolution.[5] Comments should be addressed to him.

FUTURE DIRECTIONS

Beginning C programmers complain that error messages are confusing, and often have difficulties with pointers. Advanced C programmers usually grow fond of the language, but recognize its shortcomings and limitations. The preprocessor is primitive and has different syntax than the rest of the language. The aesthetics of the **case** statement are horrible, and bitwise operators should bind more tightly than they do. Most C compilers are painfully slow, as is the UNIX loader.

It's easy to imagine a much better language, but there are only a few serious contenders — among them LISP, SmallTalk, Modula2, MainSail, and Ada. The first two require specialized environments, the next two don't appear to be a great step forward, and Ada isn't real yet. In computing, we have to live with what we have.

Always an evolving language, C may develop to keep pace with user needs and compiler technology. Bjarne Stroustrup has implemented a new language (or extension of C, depending on your perspective,) that he has cleverly called C++.[6] Although the jury is still out, C++ may well supplant C sometime in the future. The most interesting features of the C++ language are:

- User-definable types with operators that apply. Complex and BCD data types could be created this way.
- Derived classes that allow object-oriented programming, which is a great advantage when doing graphics.
- Data abstraction facilities using classes, which

CROMEMCO COMPUTERS: DESIGNED TO MAKE UNIX SYSTEM V EVEN BETTER...

UNIX System V, the new standard in multi-user microcomputer operating systems, gives you high performance features along with the portability and flexibility of a standard.

Cromemco computers can make UNIX System V even better. Because our systems are designed with UNIX in mind. First of all, we offer UNIX System V with Berkeley enhancements. Then, our hardware uses advanced features like 64K of on-board cache memory and our high speed STDC controller to speed up disk operations—very important with UNIX.

More capability and expandability

We have a high-speed, 68000-based CPU that runs at 10 MHz, coupled with a memory manager that uses demand-paging and scatter loading to work *with* UNIX, not for it.

We provide room for expanding RAM to 16 megabytes – with error detection and correction – for running even the most sophisticated and advanced microcomputer programs. And the power to accommodate up to 16 users – all with plenty of memory.

But we give you even more.

A complete solution

We give you a choice in systems: the System 100 series, expandable up to 4 megabytes of RAM, and the System 300 series, expandable to 16 megabytes. A high speed 50 megabyte hard disk drive is standard on the systems. And you can expand the hard disk capacity up to 1200 megabytes using standard SMD drives. You can add floating point processing. High resolution graphics. Video digitizing and imaging. Communications through standard protocols. Mainframe interface.

And software support is here to meet your needs. We offer major programming languages, data base management systems, communications software, including SNA architecture, X.25 protocol, and Ethernet; even a program to interface to an IBM PC if you need to. And, of course, access to the broad range of standard UNIX applications programs that is growing dramatically every day.

Easy to use.

We also make our systems easier to use, because we install the operating system before we ship your computer. No complicated installation procedures. And the Berkeley enhancements give you the standard UNIX System V operating system, but with the added convenience of these widely acclaimed improvements.

Cromemco's System 100 and System 300 computers: designed to be the highest performance UNIX systems available anywhere.

Just call or visit one of our UNIX System V Official System Centers to see for yourself. They'll also give you a copy of our new publication, "What you should know before you buy a UNIX system." Or contact us directly.

We'll be glad to show you how to get a better UNIX system.

Corporate Headquarters: Cromemco, Inc., 280 Bernardo Avenue, P.O. Box 7400, Mountain View, CA 94039. (415) 969-4710. In Europe: Cromemco GmbH, 6236 Eschborn 1, Frankfurter Str. 33-35, P.O. 5267, Frankfurt Main, Germany.

See us at UniForum Booth #2340

Cromemco®

UNIX is a trademark of Bell Laboratories.
IBM is a trademark of International Business Machines Corp.

provide data hiding, structure initialization, and dynamic typing (all optional).

- Argument checking and coercion (overridable) for all functions, making it unnecessary for the programmer to cast arguments that are not of the proper type.
- Function and operator overloading, making it unnecessary to have separate functions for floating point and integer arguments.
- A high degree of compatibility with C. The C++ compiler can optionally compile C if required.

Note that all but the first three features are provided in the ANSI draft standard. A big advantage of C++ is that programs can define new data types as required.[7] For commercial data processing, BCD arithmetic can be defined, and for mathematical computing, complex arithmetic can be defined. This is much better than modifying the compiler. Complex arithmetic would be simpler to define than BCD arithmetic. The latter would require overloading C++ operators to make them call functions that would have to be implemented in Assembly code, using machine-dependent BCD instructions.

The principal advantage of C has always been that it doesn't try to do everything itself — libraries can be written to do whatever is required. The complex and BCD data types are not needed by everybody using the language, so they are most appropriately relegated to a library.

[1] D.M. Ritchie, S.C. Johnson, M.E. Lesk, and B.W. Kernighan, "The C Programming Language," *Bell System Technical Journal*, vol. 57, no. 6, pp. 1991-2019, 1978.

[2] D.M. Ritchie, "The Evolution of the UNIX Time-sharing System," *AT&T Bell Laboratories Technical Journal*, vol. 63, no. 8.2, pp. 1577-1594, 1984.

[3] B.W. Kernighan and D.M. Ritchie, *The C Programming Language*, Prentice-Hall, Englewood Cliffs, NJ, 1976.

[4] The discussion of the ANSI draft standard is derived mostly from a Usenet article submitted by Henry Spencer of the University of Toronto.

[5] L. Rosler, "The Evolution of C — Past and Future," *AT&T Bell Laboratories Technical Journal*, vol. 63, no. 8.2, pp. 1685-1700, 1984.

[6] B. Stroustrup, "C++ Reference Manual" *Computing Science Technical Report*, no. 108, AT&T Bell Laboratories, Murray Hill, NJ, January 1984.

[7] B. Stroustrup, "Data Abstraction in C," *AT&T Bell Laboratories Technical Journal, vol. 63, no. 8.2, pp. 1701-1732, 1984.*

Bill Tuthill was a leading UNIX and C consultant at UC Berkeley for four years prior to becoming a systems software analyst at Imagen Corporation. He enjoys a solid reputation in the UNIX community earned as part of the Berkeley team that enhanced Version 7 (BSD 4.0, 4.1 and 4.2). ■


The Sperry Challenge in Salt Lake City
Delivering innovative systems in an exciting and competitive marketplace is a formidable challenge—one that we accept.
At work within the Micro Products Division and Sperry Network Systems groups, we'll surprise you with uncommon opportunities for career growth in state-of-the-art microprocessor and network communications technology.
We are seeking individuals with expertise in the following areas: UNIX Software programming, Systems design engineering, systems programming, SNA, DCA, OSI communications architectures, LAN technology communication software engineering, and many others.
At Sperry, you'll work in a business environment which encourages your participation as you share responsibility with other team members, for developing new generations of micro-based products and data communications systems.
This is the Sperry challenge and we invite your participation.
And, nowhere in America will you find better skiing than in Utah—powder snow, blue skies and sunshine less than 40 minutes away from work.
If you are interested in the Sperry challenge, please send your resume to:
SPERRY, Employment Dept., Attn: Dept. IJ
322 N. Sperry Way, Salt Lake City, Utah 84116
SPERRY
© Sperry Corporation 1985
We are an equal opportunity employer m/f/v/h
Circle No. 41 on Inquiry Card

/usr/lib

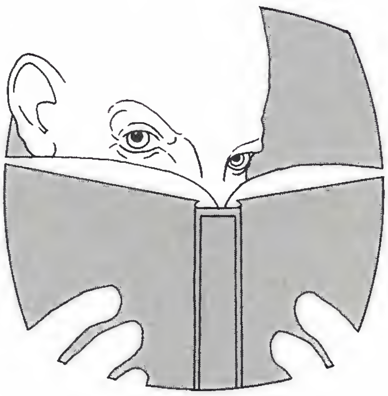
Another BSTJ classic?

by Jim Joyce

Part two of the October, 1984 issue of *AT&T Bell Laboratories Technical Journal* (*BSTJ* for short) has as its subtitle *The UNIX System*. This is the second time an issue of the *BSTJ* has focused on UNIX. The first is the legendary (and still available) July-August, 1978 issue subtitled *UNIX Time-Sharing System*, timed to coincide with the release of Version 7 UNIX. As far as I can tell, this issue is not timed to coincide with any other UNIX-oriented event, and perhaps that in part is why the articles, when taken as a whole, lack the excitement of the 1978 issue.

There are excellent articles in *The UNIX System*. Certainly both articles by Dennis Ritchie ("The Evolution of the UNIX Time-Sharing System" and "A Stream Input-Output System"), and the article on security by Fred Grampp and R. H. Morris ("UNIX Operating System Security") qualify.

The Pike and Kernighan contribution, "Program Design in the UNIX Environment", is also excellent, except that the material has been covered earlier this year in their justly popular book, *The UNIX Programming Environment* (see UNIX REVIEW, Feb/Mar, 1984 for a review). In fact, the heart of the article, the **cat-v** discussion, was covered in a Usenix conference paper by Pike and is well-



known in the UNIX community. Still, if anyone hasn't read or heard the discussion, the Pike and Kernighan article is nitty-gritty UNIX thinking at its best.

There seems a curious lack of occasion for this special issue. The passage of six years' time since the 1978 issue seems no particular milestone. Robert Martin, writing in the "Preface", states, "This Computing Science and Systems issue of the *Journal* demonstrates two key points. First, the intellectual foundations laid by Thompson and Ritchie are firm footings for continued innovation and advances in computer science. Second, even though the UNIX system is already widely accepted, it is continuously being improved by the company that invented it," (p. 1572). Let's examine these reasons in light of the articles.

That the intellectual foundations are firm footings for continued innovation and advances cannot be denied: UNIX is a wonderful environment for innovation. Yet in "A UNIX System Implementation for System/370" we read that TSS/370 was used as the basis on top of which UNIX was to run. Innovation? Hardly. EUNICE runs on top of VMS for VAXen, and HP-UX runs on top of BASIS for Hewlett-Packard ports. EUNICE has been around for at least the two years attributed to the 370 port. HP-UX is, of course, relatively recent.

But the fun part comes in examining the assertion that UNIX is continuously being improved by the company that invented it. Although many references are made to System V, there are also references to "Edition 8". The term "Edition" refers to the internal version of UNIX at the Labs, evolving from its original meaning as the version of UNIX described in a particular *edition* of the *UNIX Programmer's Manual*. And, sure enough, the password stealer shell script in the Grampp and Morris article begins:

```
echo -n "login: "
```

— a sure sign this script does not run on System V (or System III, for that matter), where the comparable code would be:

```
echo "login: \c"
```

Since Edition 8 continues developing, with interest in it mounting in the UNIX community outside the Labs, and since System III/V migrates in an independent direction, I cannot help asking *which* UNIX the company is improving? Will Edition 8 become Version 8, and will those who scrambled to join the bandwagon after AT&T's massive ad campaign for System V slap themselves on the forehead, exclaiming, "Wow! I *could* have had a V-8!"?

The bottom-line problem I am getting at is that this collection of articles shows there is no cohesive view of UNIX at AT&T. I get the feeling from reading variations of the motto, "UNIX is now widespread in the marketplace", that the slogan serves as a sad substitute for the genuine excitement over the wonder of UNIX expressed in the 1978 issue of *BSTJ*.

Should this issue of *BSTJ* be read by every serious UNIX user? Conditions that made the answer to that question in 1978 a resounding "YES!" are not the conditions of 1984. The answer is now a qualified "Yes". The UNIX afficionado will want to read the October *BSTJ*, and may rejoice to glean tidbits. Among them is that 512 bytes is the line-length limit for **sort**; the message on the manual page for **sort**, "Very long lines are silently truncated", has perplexed and even vexed many users for years.

The article "Data Abstraction in C" may evoke cries of anguish from those who would like the language frozen, especially with the REQUIREMENT (no less) that functions called with a variable number of articles must have an ellipsis indicating such is the case:

```
int fprintf(FILE*, char*, ...);
```

How much existing C code is affected by such a pronouncement one wonders. If I wanted a language that got in my way, I would have stayed with Pascal. Of course, we need not use the C++ compiler discussed in the article — yet.

Unfortunately, the people who may decide whether C++ succeeds are the same folks who arbitrarily broke:

```
echo -n "mumble"
```

so it had to be recoded:

```
echo "mumble\c"
```

in the Giant Leap Forward from Version 7 to System III (and V). (Why both forms are not tolerated seems to have been overlooked — perhaps because those deciding did not have to change the shell scripts.)

To wrap this assessment up, I am put in mind of Dennis Ritchie's Turing Award Lecture, in which he affirmed (albeit cautiously) that something like UNIX could again grow at the Labs. I would like to believe that as well, yet I also know that many internal goodies await impatiently for attorneys to decide licensing issues.

Sadly, the October *BSTJ* includes no articles on text processing (I exclude the articles on encryption deliberately). Nor is there an exciting application, such as Mike Lesk's paper describing the program that uses the Yellow Pages as a database from which queries such as "Where is the nearest pizza parlor?" are answered by a map that is output along with driving directions that take into account the fastest throughways.

None of this is to say that the October *BSTJ* is not full of good things to read. Unfortunately, the number of typos, such as the missing backquotes on page 1599, and the journal's general lack of focus detract, making it seem an issue thrown together hastily. For all that, there are good things worth digging out.

Maybe I was spoiled the first time the *BSTJ* did an issue on UNIX. The Table of Contents follows:

Table of Contents for *AT&T Bell Laboratories Technical Journal Vol.63, No.8, Part 2 October 1984 — The UNIX System*

1. Preface (2)
2. Foreword (4)
3. The Evolution of the UNIX Time-sharing System (18)
4. Program Design in the UNIX System Environment (12)
5. The Blit: A Multiplexed Graphics Terminal (26)
6. Debugging C Programs With The Blit (16)
7. UNIX Operating System Security (24)
8. File Security and the UNIX System Crypt Command (12)
9. The Evolution of C — Past and Future (16)
10. Data Abstraction in C (32)
11. Multiprocessor UNIX Systems (18)
12. A UNIX System Implementation for System/370 (18)
13. UNIX Operating System Porting Experiences (22)
14. The Evolution of UNIX System Performance (24)
15. Cheap Dynamic Instruction Counting (12)
16. Theory and Practice in the Construction of a Working Sort Routine (18)

17. The Fair Share Scheduler (14)
18. The Virtual Protocol Machine (18)
19. A Network of Computers Running the UNIX System (20)
20. A Stream Input-Output System (13)

UNIX for People

This is a fine book for someone wanting to learn to do document preparation under UNIX. The examples are heavily oriented toward Berkeley UNIX, largely because the book was done at Berkeley by the three authors: Peter Birns, Patrick Brown, and John C. C. Muster. Yet the excellence of the examples make the book usable in a non-Berkeley environment.

Nevertheless, there is something very wrong about this book. For one thing, it claims as its subtitle: "A Modular Guide to the UNIX System: Visual Editing, Document Preparation, & Other Resources". They may as well have left off the "& Other Resources", since their excursion into UNIX commands is only in the context of preparing documents, and they do not do much with that excursion.

Make no mistake: what they have done is quite nice, though at times a bit precious. This preciousness reminds me of the Waite group's *UNIX Primer Plus* (UNIX REVIEW, Oct/Nov, 1983), which was also Berkeley-oriented. At times, the approach comes dangerously close to being patronizing.

The photo illustration on page 15 for the command **look egg** shows a man looking at an egg much as Hamlet regards Yorick's skull. Since Yorick was a jester in the play, one can wonder whether the egg is being asked, "Any good yolks?" See, it's catching. Seriously, the photos and drawings do add to the text. Detracting, though, are excesses such as the section heading "You've Made It (Puff-Puff)" — as though the very gentle, step-by-step presentation had offered a challenging slope for a runner. "You've Made It" would have been enough.

Although the subtitle to the book does make clear the emphasis, it is most unfortunate that the "Preface" claims it was "originally intended as an introductory book on text processing, ... [but] has grown into a more complete introduction to the entire UNIX Operating System." (p. xii) This is patently false, and unnecessarily misleading to potential readers. The book is not in any way an introduction to "the entire UNIX Operating System". The book *is* a good introduction to text processing, including many nice examples of table processing using **tbl**. Why not call the book *UNIX Text Processing for People* so we know who the audience is? As it is, the exaggerated claim detracts from the genuine merit of the achievement.

The Table of Contents follows:

Table of Contents for *UNIX for People*

1. How to Use This Book (5)
2. Accessing the UNIX System (15)
3. Your First File (16)
4. Editing Files Using the Visual Editor (24)
5. Nroff Formatting Commands (20)
6. File Management with Shell Commands (24)
7. Conceptual Overview (15)
8. Advanced Visual Editor Commands (22)
9. Advanced Formatting Commands The -ms Macros (27)
10. A Stroll Through a Full Production Number (8)
11. The Line Editor Ex (26)
12. Special Search and Substitution Characters (19)
13. Truly Advanced Visual Editing (19)
14. Communicating with Others (8)
15. The UNIX Directory Structure (23)
16. Account Management Activities (17)
17. Backgrounding a Process (12)
18. Parts and Wholes (11)
19. Phototypesetting with Troff and Troff -ms (15)
20. Macro Construction (28)
21. Utility Programs (11)
22. Commands, Files, and Directories: Paths, Bins, & Yellow Brick Modes (16)
23. Form Letters (10)
24. Special Formatting Topics: A Title Page, Table of Contents, & Index (12)
25. Bibliographies and Footnotes: The REFER Program (17)
26. Setting Tables: A Busboy's Nightmare The TBL Pre-Processor (24)
27. Equalizing Equations: The EQN Pre-Processor (23)
28. Troubleshooting (13)
29. Where to Now? (7)

Command Summary Section
- Shell (9)
- Visual Editor (10)
- Format (11)

Location of Material within the Book
- Index (9)
- Quick Access Chart (7)

Jim Joyce is President of International Technical Seminars, Inc., a firm committed to UNIX training and founder of the Independent UNIX Bookstore. For answers to your question's about books, call 415/621-1593. ■

THE UNIX GLOSSARY

Echoes from the past

by Steve Rosenthal

Note: Much could be said about each of these terms. But only those aspects that concern the evolution of UNIX are included in this listing.

2 BSD — the first widely distributed version of the UNIX system enhancements developed by the Computer Systems Research Group at the University of California at Berkeley. It ran on the PDP-11. See *BSD* for more details.

4 BSD — the second widely distributed version of the UNIX system enhancements developed by the Computer Systems Department at the University of California at Berkeley. It runs primarily on the DEC VAX. The current release is 4.2 BSD. See *BSD* for more details.

B — an interpretive language developed by Ken Thompson after he gave up on a week-long attempt to implement Fortran for the PDP-7. B is now primarily known as the predecessor to the language C.

binary license — a license to use UNIX code as compiled for a specific implementation. In the early days of UNIX, most of the university and research users who brought up UNIX worked from source code. Now, most commercial users buy only a binary license, leaving it to major software vendors to make changes to their system code.

BISP — Short for **B**usiness **I**nformation **S**ystems **P**rograms, the unit at Bell Labs that included many of the early users of the UNIX system and its utilities. Because their requirements were more oriented towards software production and less towards computer research, BISP people added most of the PWB (Programmer's Work Bench) utilities to UNIX, including the SCCS (Source Code Control System). Some of the tools developed at BISP were incorporated into Version 7, and most found their way into System III, which was descended from PWB/UNIX.

BSD — short for **B**erkeley **S**oftware **D**istribution, a succession of UNIX implementations developed by the Computer Systems Research Group at the University of California at Berkeley (see CSRG for more details). The BSD effort, which was funded by the DARPA (Defense Advanced Research Projects Agency), produced a number of enhancements, including demand-paged virtual memory. BSD code is in the general public domain (although, of course, you need a valid UNIX license or a UNIX-like system to make direct use of it). Even though AT&T has made some effort to incorporate Berkeley's enhancements into its implementations of UNIX (particularly into System V), the Berkeley versions have a significantly different flavor. It appears that while there is still much cross-fertilization, some level of speciation may have already occurred, and System V and BSD may continue to evolve only somewhat in parallel.

C — the computer language tied closely to the evolution of the UNIX system. C was developed by Dennis Ritchie, in an effort to create a language that would be sufficently high level to permit users to be productive, yet sufficiently close to the machine code to allow for the efficient writing of compilers and operating systems. The current level of C's popularity would indicate the effort has been largely successful. C arose as an evolutionary advance, being

PC*works.*

The Right Connection.

With TouchStone's PCworks™ software it's easy to connect industry-standard PCs to a variety of other computers and information services. Now previously isolated PCs can share information with other computer systems.

It's So Simple

Just load the PCworks program into your PC, plug in a standard cable or telephone line, and your PCs are ready for work as part of a network. PCworks will do the rest — dial the right numbers, log onto another computer, and start a program. When you want your PC as a PC, no problem. PCworks uses the MS-DOS™ operating system, and will run with all your favorite PC programs.

PCworks and UniHost. A UNIX Software Connection.

PCworks in combination with TouchStone's UniHost™ software provides a special connection from your PC to over 50 kinds of UNIX™ computers. With the PCworks/UniHost combination the UNIX (or Xenix™) system acts as a network manager, allowing PCs and UNIX system users to communicate with each other as well as take advantage of more powerful UNIX system hardware and software.

Now PCs can share electronic mail, exchange all types of files, and use the UNIX system disk and printer.

THE CONNECTABLES™:
FAMILY OF COMMUNICATION SOFTWARE PACKAGES.

Move a spreadsheet from your PC to the UNIX system. Transfer a text file from the UNIX system to your PC for editing. Print the file later on the UNIX system printer, or use the UNIX system disk for backup.

The Connectables Software from TouchStone.

The Connectables™ family of communication software packages from TouchStone offers a simple solution to networking problems. Look for MacLine,™ the newest member of the family, which provides connections to PCworks and UniHost for Macintosh™ computers.

TouchStone
Software Corporation
909 Electric Avenue, Suite 207, Seal Beach, California 90740 213/598-7746

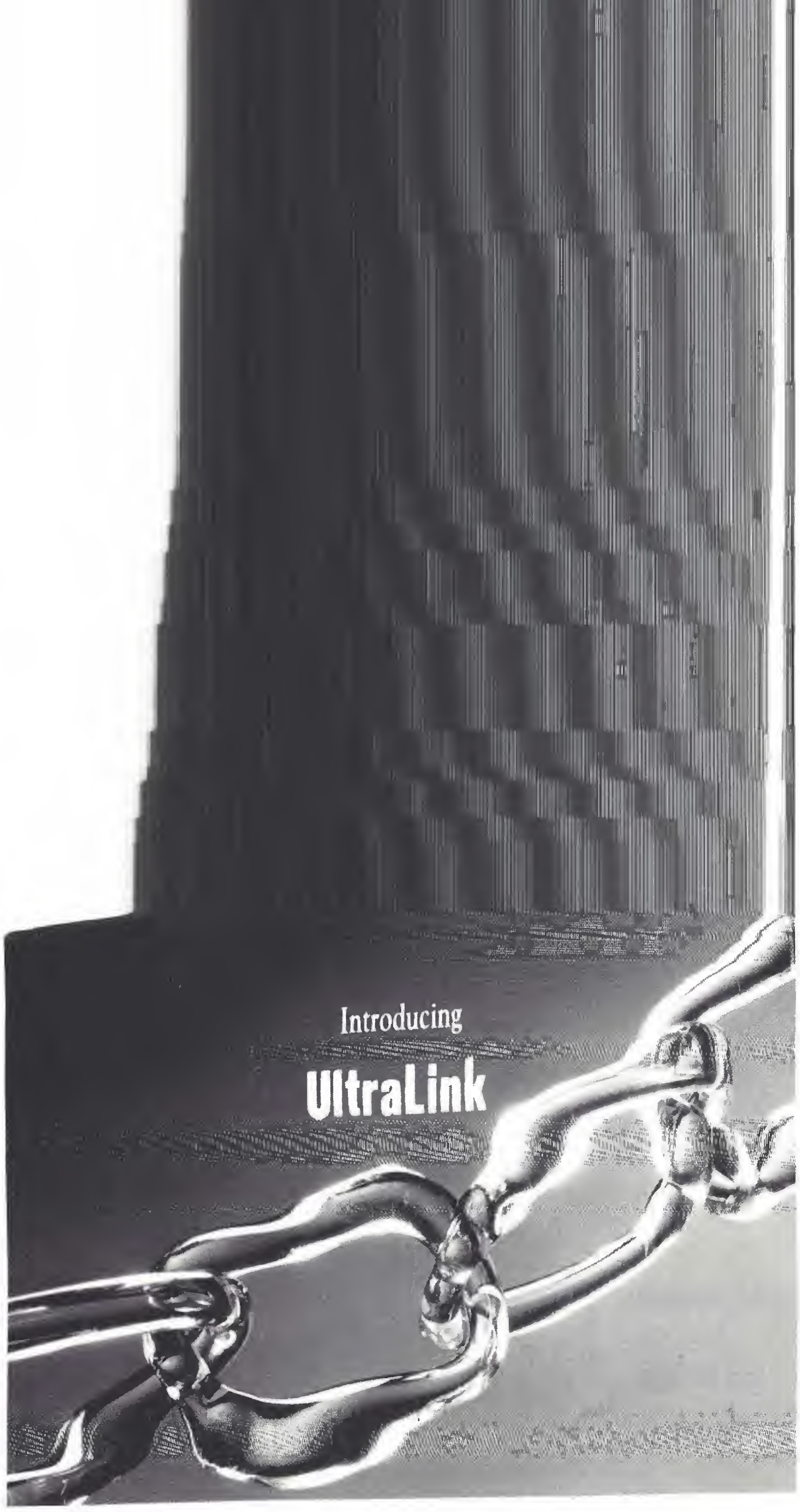
Circle No. 50 on Inquiry Card

The first versions of PWB were developed at the BISP (Business Information Systems Programs group at Bell Labs), and first installed on a PDP-11 in 1973.

PWB/UNIX — a version of UNIX derived from the sixth edition, but containing many features for program development created at BISP. Most of the features were incorporated in System III and then System V.

Seventh Edition — one of the original names for what is now called UNIX Version 7. See *Version* for why this happened.

source license — a UNIX license that includes the right to read and modify the UNIX source code (but not necessarily to distribute those modifications). In the early days of UNIX, when most users were at universities, the majority of installations had source licenses. Now, most casual users buy UNIX as a packaged product, operating under a sublicense and receiving



/group have a distinct commercial flavor.

Version — the early generations of UNIX, up through Version 7, were known by the numbering on the corresponding edition of documentation. Thus Version 6 was the UNIX documented in the sixth edition of the UNIX manuals.

Version 1 — the unofficial name for the earliest UNIX system. It was developed in 1969 and 1970, and ran on the PDP-7. The original version was a single-user system, with only a few of the utilities now considered standard.

Version 5 — the earliest version of UNIX that anyone seems to mention as having been regularly used outside of Bell Laboratories. Version 5 apparently made its way out to a handful of university and research labs.

Version 6 — the first widely-distributed version of UNIX. This version was made available to educational and non-profit research institutions for a nominal charge (mostly for handling and tape copying), starting in 1975.

Version 7 — the first version of UNIX to be made available to commercial users outside of Bell. It was first released in 1979, and is still in widespread use. Officially, Version 7 was superseded by System III.

Western Electric — before the Bell System divestiture, Western Electric, which was the Bell System's supply and manufacturing arm, was the logical place to take orders for software. Consequently, it served as the distribution point for Bell UNIX until the breakup re-shuffled the cards. AT&T Technologies, which ended up with UNIX, still cuts the deck with regularity, but UNIX distribution seems to be somewhat fixed in the Software Sales and Marketing group in Greensboro, NC.

Xenix — Microsoft Corporation's operating system family based on UNIX. According to Microsoft, over half of the current UNIX licenses are for its Xenix implementation. Xenix was originally based on Version 7, but Microsoft has made a public commitment to upgrade the system to System V compatibility.

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

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

32-BIT SYSTEMS DES

COMPETIT

Competitive In Performance.

Areté 1000 series systems are designed to provide decisively superior performance in on-line commercial applications. Designed to fully exploit the power of modern 32-bit microprocessor technology, Areté 1000 systems can clearly outperform VAX 780, DG MV 8000 and similar superminis.

A unique and proprietary shared-memory, multiprocessor architecture allows the Areté 1000 series systems to offer a single virtual processor more than three times as powerful as other systems employing comparable technologies. That helps you to support more users with less hardware. You can see the results in your bottom line.

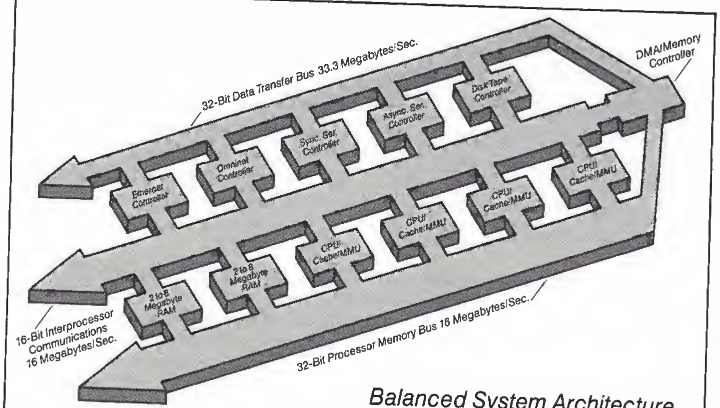
A powerful CPU architecture demands a powerful I/O subsystem. Areté's 33.3 megabyte data transfer bus can support the demands of 16 to 60 interactive users who not only expect, but demand, superior performance for their investment.

The use of multiple optimized buses prevents the bottlenecks found in single bus implementations. That keeps Areté resources working for you instead of being idle.

See us at UniForum Booth # 1788

Competitive In Features.

Areté 1000 series I/O controllers support streaming cartridge tapes, 9-track tape, làrge capacity Winchester disk drives, bit synchronous, byte synchronous and asynchronous data communications. Areté software includes UNIX System V and RM/COS operating systems, COBOL, C, FORTRAN, BASIC and Pascal languages, SNA and Bisync communications subsystems.

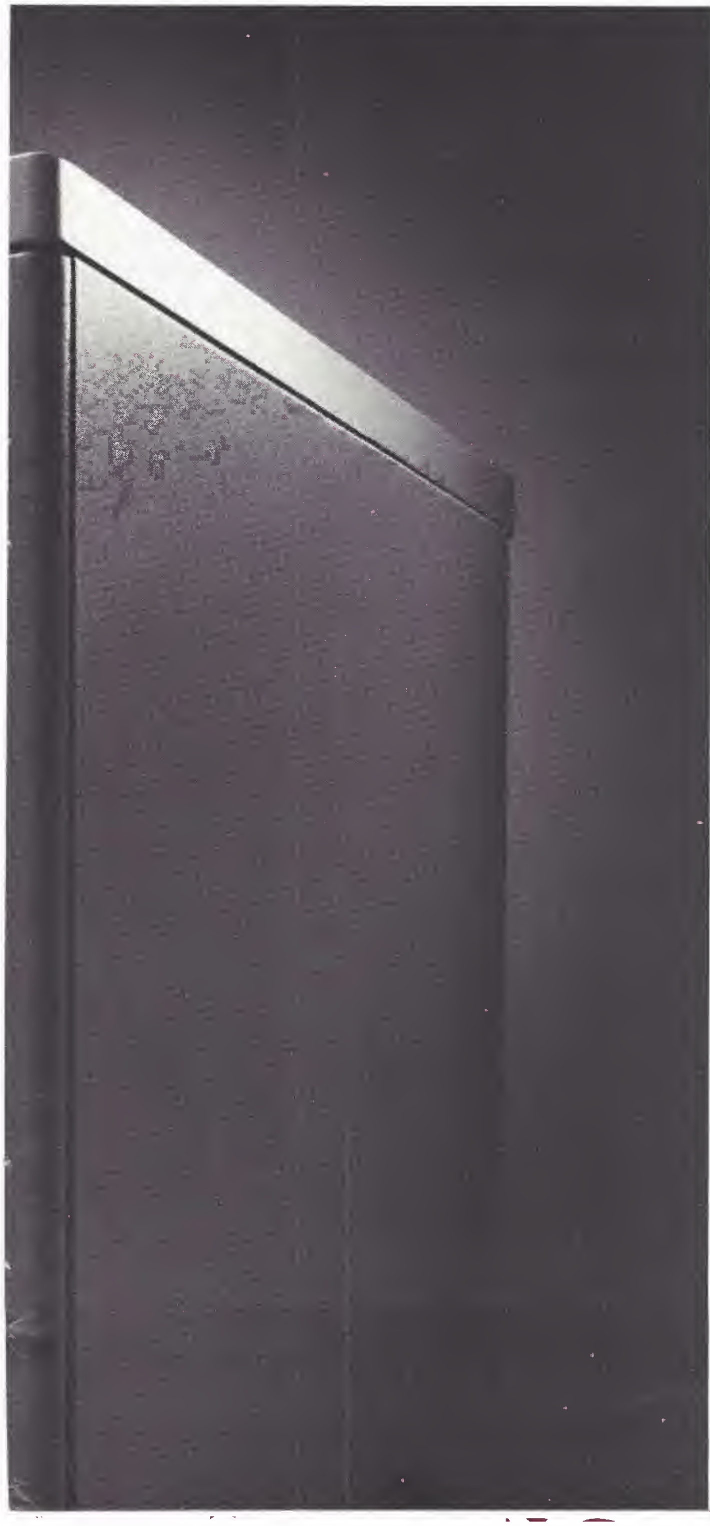


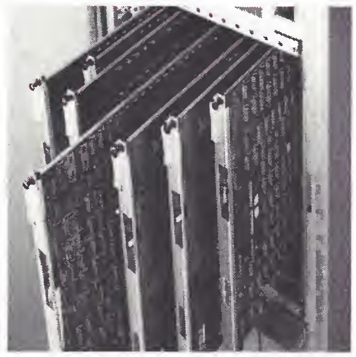
Balanced System Architecture

A modularly expandable architecture enables Areté 1000 systems to meet a wide range of performance requirements. With 1 to 4 processors, 2 to 16 megabytes of memory, and up to 11 powerful I/O controllers, you can fit the system to the to the application without duplicating resources or modifying your applications.

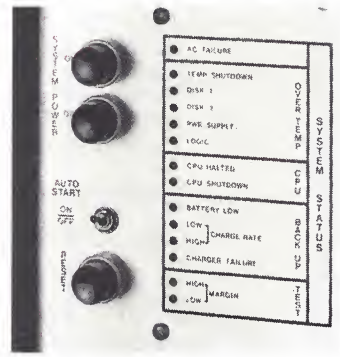
GNED TO GIVE YOU A

IVE EDGE.





The Arete' 1000's spacious cabinet accepts any combination of CPU or memory boards, DMC, or IOCP boards up to a total of nineteen.



Diagnostic status panel assists in diagnosing system problems.

Competitive In Reliability.

Areté systems offer you more reliability features than any other computers in the same price range. Mirrored disks, power margining, sophisticated diagnostics, integrated UPS option, thermal monitoring and superior manufacturing methods insure that Areté systems keep working for you. That adds up to fewer service calls and more satisfied users.

Competitive In Quality.

The 1000 series was designed to offer you a decisive advantage in price/performance, cost of application development, and cost of ownership. Areté didn't sacrifice quality to achieve this objective. It found innovative methods for squeezing the last possible iota of performance from available technology.

To the ancient Greeks the word *Areté* meant excellence in all things. We felt it would be an appropriate name for a company committed to becoming a leader in providing on-line computer systems to OEMs, value-added resellers, and volume end-users.

There is only one way to become leader and remain a leader . . . by being committed to excellence.

If you would like more information about the Areté 1000 series systems, please call or write Areté Systems Corporation, 2040 Hartog Drive, San Jose, California 95131. (408) 263-9711.

Excellence in all things

ARETÉ
SYSTEMS CORPORATION

Circle No. 55 on Inquiry Card

Continued from Page 70

doesn't have to have all of the capability of the operating system in the larger distributed environment. I certainly would not expect an operating system in a PC to support, for example, photo typesetting. But I really, by golly, do expect an operating system in a 3B20-sized machine to support photo typesetting.

To the extent that the capabilities are in the small machine, I want them to operate in a way consistent with the capabilities in the larger machines. I want to be able to operate in an environment where sometimes I'm huddled with my small machine in my office and sometimes I'm connected to a network.

REVIEW: *You don't want to have to be concerned about how it looks different and how you have to do things differently?*

VYSSOTSKY: I don't want to have to change gears mentally when I go from one to the other. For example, I want to be able to do things like when I use text processing capabilities to handle information on personnel matters. I darn well do not want a wire running down the hall to the general use machine. That exposes the possibility of a privacy violation. I want to cut the thing off the network for that situation. But I don't want to have to use a different editor. That's only one example. There are many where I want to be part of a network or not part of it. I want to have consistency throughout. I want to use those small machines for that purpose because I do not expect to see a 3B20 sitting in my office and I wouldn't want one here even if I could get it.

REVIEW: *Do you have anything to add to this general speculation about which areas AT&T will continue to support? Clearly, there is an interest both from your organization's and the world's point of view in getting UNIX onto other machines so that it is more transparent. Computer managers want to minimize retraining.*

But AT&T is now going into the computer business and it would appear counter-productive to put development effort into supporting other vendors' hardware. At the same time, a company this large must need the capability in-house to run UNIX on various machines.

VYSSOTSKY: I don't see the issue in quite the terms that you couched it. AT&T is a new player in some parts of the computing market. Obviously, we have been in the market for switching gear and operations support systems for the telephone companies for quite some time. To the extent that we are moving into new areas, we have to ask ourselves, "What about our products and our product strategy would make the customers want to buy them?" And, being technically great by itself and even technically great and attractively priced by itself is probably not enough. You know, it just is not true that if you invent a better mousetrap, the world will beat a path to your door. It doesn't happen that way.

We have technically fine products. I think that they are priced very attractively but then we have to ask, "What on top of that is going to make that big world out there want our products rather than somebody else's?" The thing that we can offer is we can provide with the UNIX operating system a very fine programming environment that spreads across all sorts of vendors' products. Thus, a system builder, an OEM vendor, an end user who buys our products and uses them for creating his application gets tremendous flexibility. This includes flexibility to take advantage of our full

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

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

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

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

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

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

TELETYPE: VALUE SETS US APART.

AT&T
Teletype Corporation

Circle No. 58 on Inquiry Card

product line but also the flexibility to use the products of other vendors in sensible ways.

We are realistic enough to know that this is a new game for us and that most of our customers will use other vendors' products as well as ours. It would be silly of us to try to discourage customers from doing that. Rather, what we want the customer to do is to use other vendors' products in a way that gets synergy from their use of our products. That's good for the customer and it's good for us. So, I don't view it as an either/or question between supporting UNIX on our equipment or supporting UNIX on other vendors' equipment. We want to support our customers in a way that makes it attractive to our customers to use our products.

REVIEW: *Do you have any speculations about the future of the UNIX market?*

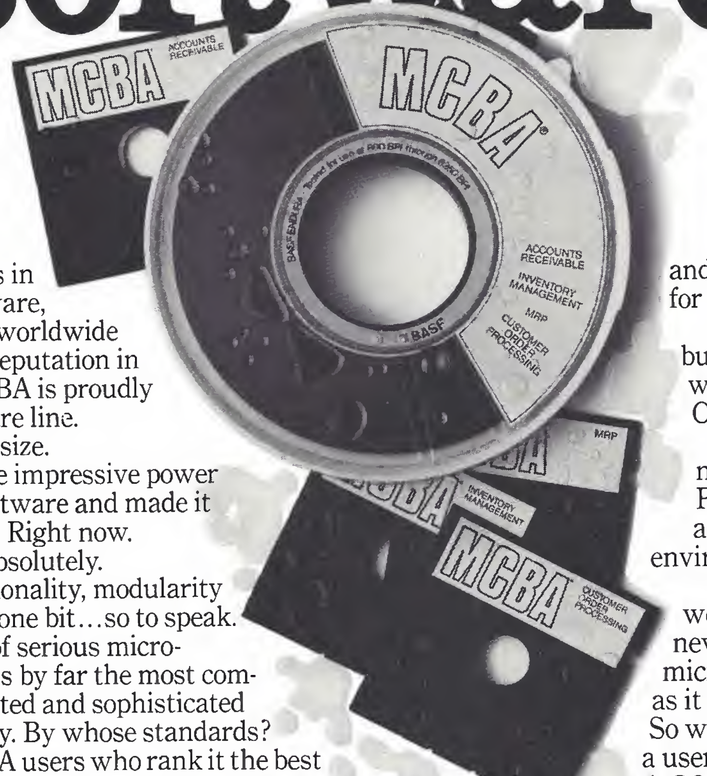
VYSSOTSKY: The only speculation I can offer you is not even a speculation, but rather a viewpoint on how things are going to evolve. Let me start out by recalling to you that until very recently in Bell Laboratories and other companies that I am familiar with, the decisions on what hardware to procure, what types of terminals to use, what kind of software to use, what sort of applications the organization could run were not the kinds of decisions usually made by end users. Financial information came up on a standard 3270-type terminal because some data processing manager decided that was the variety of terminal the report would show up on. The trend I see more and more is caused by two major factors: the declining cost of workstations and personal computers, and the increasing level of computer literacy. End users can now exercise a voice about what hardware they want to see and what application should run and how the stuff should look.

REVIEW: *All of which is new. In a way, you could almost couch it in terms of increased freedom of choice.*

VYSSOTSKY: Yes. For example, the trend is towards more sophisticated terminals with bit-mapped displays, good graphics capability, and color. I think the trend is largely driven by end users. Choices like that used to be made by the EDP vice president, while the people sitting at the terminals were effectively clerks. The clerks really had very little effective leverage on the question of whether they liked the way the display appeared.

When the end user is no longer a clerk but a line manager and when the terminal is no longer a conventional old fashioned terminal but something with capability of its own, the line manager will say, "I'm not going to look at this. I don't like this." You see what is happening here; it's happening all over the place. For example, I had a department head who doesn't happen to like the display format he gets from the administrative system. So he goes and gets a workstation or a personal computer. He dumps the output information from the standard administration system into the personal computer, rearranges it to suit himself and puts it up on a display he can understand. That

MCBA introduces shrink-to-fit software.



With nine years in minicomputer software, 15,000 installations worldwide and an established reputation in the mini world, MCBA is proudly shrinking its software line.

Down to micro size.

We've taken the impressive power of minicomputer software and made it available for micros. Right now.

Alter the fit? Absolutely.

Alter the functionality, modularity and capability? Not one bit...so to speak.

This new line of serious microcomputer software is by far the most comprehensive, well-tested and sophisticated in the industry today. By whose standards? Thousands of MCBA users who rank it the best in the business.

MCBA's library of 16 integrated manufacturing, distribution and accounting packages can be installed in whatever combination and sequence a user needs for his or her business.

It grows with businesses. No matter what size they are now. Or want to be later.

And MCBA software now runs in RM/COS,® PC-DOS, UNIX™ and UNIX look-alike environments.

In other words, we've tailor-made our newest software to fit micros—as comfortably as it fits user needs.

So whether you're a dealer or a user, find out about it. Call **1-800-MCBA NOW** (toll free outside of California). In California, call **(818) 957-2900.**

MCBA's shrink-to-fit software. For growing businesses.

MCBA®

Minicomputer Software for Micros.

2441 Honolulu Avenue, Montrose, California 91020

Also for DEC, Wang, HP, TI, and Perkin-Elmer minis.

MCBA is a registered trademark of MCBA, Inc. UNIX is a trademark of AT&T. RM/COS is a registered trademark of Ryan-McFarland Corp.

See Us At UniForum™ The International Conference of UNIX Users **January 21-25 • INFOMART • Dallas, Texas • Booth #2659**

UR185

Circle No. 60 on Inquiry Card

is my vision of the end user being in control, which is my vision of the way the future is going.

The thing that I think is important about that is that it makes it very difficult for people like me to predict what is going to happen in the marketplace. Things used to be largely constrained by the question of what a relatively small number of people in vendor management and user companies decided. To the extent that the users have become the driving force, predicting the computer market becomes almost like predicting the evolution of the consumer market—which is notoriously difficult.

REVIEW: *It's slightly different, though, since this is an educated consumer market.*

VYSSOTSKY: I think that most consumers are educated consumers. When I buy a TV set, I may not know much about how TV sets are designed, but I know what I want in a TV. You are right that we're talking about an educated consumer market but I don't think that distinguishes it much from the market for automobiles or anything like that. I think it is very different from a market in which one person makes a decision for 300 other people, so it becomes hard to predict. It also means that the market has a chance to evolve in ways that are satisfactory to the end user. I think it also offers an opportunity for UNIX systems to spread even more rapidly than they have in the past, precisely because the UNIX system is such a nice programming environment that it is well adapted to the end user world.

Note that I haven't said where the market will go but that I have said that the market will make its own choices. ■

Continued from Page 44

was gone by blanking it out. We also wanted the line-erase character to display a blanked-out line. Some UNIX systems such as 4.2 BSD and System V now support this, but it was not then available anywhere under UNIX Version 6.

Bob and I had argued, somewhat sleepily, for hours as to the correct method of erasing characters, and Bob had started putting our joint design into effect just as I collapsed on the floor for "a short nap". I awoke around dawn to find Bob asleep over the terminal. When he woke up, he said he was pretty sure he'd finished the job before falling asleep, but neither of us had enough energy to check. It was time for food and 14 hours of sleep.

When we finally checked our handiwork the next day, we found some serious flaws in the implementation — not an uncommon situation with work performed under extreme conditions. But the system was up and running, and although the new features were flawed, they didn't seem to cause any problems, so we forgot about it for the time being. A week later, I was consulting in Cory — we all offered free programming help to other students in the time-honored tradition of hackers everywhere — when Kurt Schoens called me over to the other side of the room.

"Hey Doug," he said. "Look at this. It looks like someone tried to put character deletion into the terminal drivers, but only half finished."

My heart raced. Did he suspect me? Or was he just chatting? I could never tell whether Kurt was kidding; he had the most perfect poker face I had ever seen. But he quickly made the question academic, and proved again that he was one of Them.

"I showed this to Bill, and he wanted to fix it", Kurt said. "Oh, really?" I stammered. "Sounds good to me," thinking that it was a real stroke of luck that Bill Joy would be interested in the half-completed project. If Bill finished it, then it would be in the system on legitimate grounds, and would stay for good.

Kurt paused for effect. "Yeah, he was all fired up about it, but I talked him out of it, and I just deleted it from the system instead."

Oh, cruel fate! Kurt must know that I was involved; he just wanted to see me jump when he said "boo!"

Although I'm sure Kurt thought the whole incident very funny, all I could think of was that yet another of my features had gone down the drain. I discussed this latest setback with others in the group, and we shared a sense of frustration. More than ever before, we were determined to get our contributions accepted somehow.

Kurt was both a graduate student and a system administrator, but I liked him all the same — chiefly because of his practical jokes. We had recently cooperated in a spontaneous demonstration of Artificial Intelligence at the expense of an undergraduate named Dave who had joined Them as a system administrator. Dave had watched Kurt as he typed **pwd** to his shell prompt and received **usr/kurt/mind** as the response. His next command had been **mind -i -l english** . During all this time, Kurt was double-talking about psychology and natural language processing and some new approach to simulating the human mind that he'd thought of. Dave looked dubious, but was willing to see how well Kurt's program worked.

What Dave didn't realize was that Kurt had not been typing commands to the system at all; although we were sitting not 10 feet apart, Kurt and I had been writing to each other and chatting for half an hour, and as a joke I had been pretending I was Kurt's shell, sending him prompts and faking responses to commands. Dave had walked in at just the right time. So when Kurt typed **mind -i -l english** , I had naturally responded with:

"Synthetic Cognition System, version 17.8"
"Interactive mode on,
Language = english"
"Please enter desired conversational topic: (default:philosophy)"

Dave couldn't help looking a little impressed; Kurt's "artificial intelligence" system was off to a great start. Kurt had talked to his budding mind for several minutes, and Dave of course had grown more and more impressed. Kurt and I faced the greatest challenge of our lives in keeping a straight face during the demonstration, but we eventually made the mistake of making the mind altogether TOO smart to be believable, in effect sending Dave off to tackle more serious work.

There was one practical joke that was notable for the length of time it was supported by the entire group. The target was system administrator Dave Mosher. Dave had been suspicious of bugs in our system's homebrewed terminal multiplexer for some time. Ross decided to persecute Dave by having random characters appear on his screen from time to time, which of course convinced Dave that the terminal multiplexer did indeed have problems. To help Ross with the prank, each of us sent Dave some garbage characters at random intervals whenever any one of us was on the system. We had settled on the letter "Q" so that Dave would be sure it was always the same bug showing the same symptom. Since Dave had these problems no matter which terminal he was on, day or night, no matter who else was logged onto the system, he was positive there was a problem, and he spent much time and effort trying to get someone to fix it.

Unfortunately for Dave, he was the only one who ever saw these symptoms, so everyone thought he was a little paranoid. We thought it was pretty funny at first, but after a few months of this, it seemed that Dave was really getting rattled, so one day Ross generated a capital "Q" as big as the entire terminal screen and sent it to Dave's screen. This made it pretty obvious to poor Dave that someone, somehow, really *had* been persecuting him, and that he wasn't paranoid after all. He had an understandably low tolerance for practical jokes after that.

The numerous practical jokes we played were probably a reaction to the high level of stress we felt from our ongoing illicit operations; it provided some moments of delightful release from what was, at times, a grim battle. There were many secret battles in the war; if Our motto was "Features!", Theirs was "Security for Security's Sake" and the more the better. We were never sure how long our victories would last; on the other hand, They were never sure whether They had won. The war lasted almost three years.

We were primarily interested in the EECS department's PDP 11/70 in Cory Hall, since that was the original UNIX site and continued to be the hotbed of UNIX development, but We "collected" all the other UNIX systems on campus, too. One peculiar aspect of the way the Underground had to operate was that we rarely knew the root password on systems to which we had gained superuser access. This is because there were easier ways to get into, and stay into, a system than guessing the root password. We tampered, for instance, with the **su** program so that it would make someone superuser when given our own secret password as well as when given the usual root password, which remained unknown to us. In the early days, one system administrator would mail a new root password to all the other system administrators on the system, apparently not realizing that we were monitoring their mail for exactly this kind of security slip. Sadly, they soon guessed that this was not a good procedure, and we had to return to functioning as "password-less superusers", which at times could be a bit inconvenient.

Late one night on Cory Hall UNIX, as I was using my illegitimate superuser powers to browse through protected but interesting portions of the system, I happened to notice a suspicious-looking file called *usr/adm/su*. This was suspicious because there were almost never any new files in the administrative *usr/adm* directory. If I was suspicious when I saw the filename, I was half paralyzed when I saw it contained a full record of every command executed by anyone who had worked as superuser since the previous day, and I was in a full state of shock when I found, at the end of the file, a record of all the commands that I'd executed during my current surreptitious session, up to and including reading the damning file.

It took me perhaps 10 minutes of panic-stricken worry before I realized that I could edit the record and delete all references to my illicit commands. I then immediately logged out and warned all other members of the group. Since nothing illicit ever appeared, the system administrators were lulled into a sense of false security. Their strategy worked brilliantly for us, allowing us to work in peace for quite a while before the next set of traps were laid.

The next potential trap I found was another new file in */usr/adm* called *password*, that kept track of all unsuccessful attempts to login as root or to **su** to root, and what password was used

REMEMBER THE REVOLUTION?

The Unix revolution is a reality in the United States. Big Business for companies who recognised the trend and got in early. Bad News for those who didn't.

But what's done is done and history doesn't repeat itself. At least not often,... Now, that same Unix Revolution is about to sweep Europe. In June 1985, in Britain, over 150 companies – including Unix Europe, AT & T and Olivetti, DEC, Burroughs, NCR, Data Logic, Gould Electronics, Perkin Elmer, Intel and Zilog – will exhibit at The 1985 European Unix User Show. This, the first major event of its kind outside the United States, is sponsored by /usr/group/UK – Europe's largest commercial Unix organisation. The cooperations, partnerships, ventures and contacts formed at this event will largely determine the future shape of the European Unix market.

Europe has enormous Unix potential – undoubtedly one of the largest markets for Unix products outside the United States. Remember the Revolution?

For details on how to exhibit at The 1985 European Unix User Show ring Mario Meoli, Exhibition Manager in London, England, on 011 441 837 3699, before 10.00am, or telex 32157 EMAPPB G, or clip the coupon NOW.

EUROPEAN UNIX™ USER SHOW

JUNE 12-14 1985
OLYMPIA 2, LONDON.

☐ Please send me information about exhibiting at The 1985 European Unix User Show

☐ Please send me details of The European Unix User Show Conference.

Please send me details of other computer events in the 1985 series of EMAP European expositions and conferences.

☐ PC Trade Show ☐ PC User Show ☐ DEC User Show
☐ IBM System User Show ☐ CAD/CAM International Show VR9

Name
Job Title
Company
Address

Zip
Tel./Telex

Return to: EMAP International Exhibitions Ltd., Durrant House, 8 Herbal Hill, London EC1R 5JB, U.K.

emap. INTERNATIONAL EXHIBITIONS LTD.

EUROPE'S FIRST MAJOR EVENT FOR THE UNIX MARKET

™ Unix is a trademark of AT & T BELL Laboratories in the attempt. Since none of us had known the root password for months and therefore weren't going to become superuser by anything as obvious as logging in as root, this wasn't particularly threatening to us, but it was very interesting. The first few days that we watched the file it showed attempts by legitimate system administrators who had made mistakes of various sorts. One of Them once gave a password that We discovered, through trial and error, to be the root password on a different system. Several of Them gave passwords that seemed to be the previous root password. Most of them were misspellings of the correct root password. Needless to say, this was a rather broad hint, and it took Us less than five minutes to ascertain what the correct spelling was.

One might think that, since we had several ways to become superuser anyway, it wouldn't make any real difference whether or not we knew the actual root password as well. The problem was that our methods worked only so long as nothing drastically changed in the system; the usual way that They managed to win a battle was to backup the entire system from tape and recompile all utilities. That sometimes set Us back weeks, since it undid all of our "backdoors" into superuserdom, forcing us to start from ground zero on breaking into the system again. But once we knew the root password, we could always use that as a starting place.

We worked very hard to stay one step ahead of Them, and we spent most of our free time reading source code, in search of either pure knowledge or another weapon for the battle. At one time, I had modified every single utility that ran as superuser with some kind of hidden feature that could be triggered to give us superuser powers. Chuck Haley once sent a letter to Jeff Schriebman commenting that he "had even found the card reader program" to show signs of tampering. I thought that I had disguised it well, but it was extremely difficult to keep things hidden from a group of system administrators who were not only very intelligent, but also highly knowledgeable about the inner

workings of UNIX. As an indication of the caliber of the people we were working against, I should note that Chuck Haley is now a researcher at Bell Labs; Bill Joy is VP of Engineering at Sun Microsystems; Kurt Schoens is a researcher at IBM; Jeff Schriebman is founder and President of UniSoft; and Bob Kridle, Vance Vaughn, and Ed Gould are founders of Mt. Xinu.

This was an unusual situation; system administrators are not usually this talented. Otherwise, they'd be doing software development rather than administration. But at the time, there was no one else *capable* of doing UNIX system administration.

As a result, we had to move quickly, quietly, and cleverly to stay ahead, and planting devious devices in the midst of standard software was our primary technique. Normally trusted programs which have been corrupted in this way are called "Trojan Horses", after the legend of the Greeks who were taken in by a bit of misplaced trust. One of our favorite tricks for hiding our tracks when we modified standard utilities was the **diddlei** program, which allowed us to reset the last change time on a modified file so that it appeared to have been unchanged since the previous year. Bob modified the **setuid** system call in the UNIX kernel so that, under certain circumstances, it would give the program that used it root privileges. The "certain circumstances" consisted simply of leaving a capital "S" (for Superuser) in one of the machine's registers. Bob was bold enough to leave this little feature in the system's source code. We usually put our Trojan Horses in the system executables only — to decrease the chance of it being noticed. But Bob took the chance so that the feature would persist even if the system were recompiled. Sure enough, it lasted for several months and through more than one system compilation before Dave Mosher noticed it (undoubtedly with a sense of shock) as he was patiently adding comments to the previously undocumented kernel.

This sort of battling continued for several years, and although They were suspicious of most of Us at one time or another, none of Us was ever caught red-handed. It undoubtedly helped that we never performed any malicious acts. We perhaps flouted authority, but we always enhanced the system's features. We never interfered with the system's normal operation, nor damaged any user's files. We learned that absolute power need not corrupt absolutely; instead it taught us restraint.

This is probably why we were eventually accepted as members of the system staff, even though by then several of Us had confessed to our nefarious deeds. Once we were given license to modify and improve UNIX, we lost all motivation to crack system security. We didn't know it at the time, but this has long been known to be one of the most effective ways of dealing with security problems: hire the offenders, so that there is no more Us versus Them, but simply Us.

It worked well in our case; under the auspices of the System Development and Research Group, created by the ever-industrious Dave Mosher, we went happily to work on UNIX development. The development of UNIX at Berkeley, always fast-paced, exploded once everyone — including undergraduates — were participating.

The only fly in the ointment was the introduction a short while later of UNIX Version 7. While it was a vast improvement in many ways over the Version 6 that we had been working with, most of the enhancements we had developed were lost in the changeover. Some were reimplemented under Version 7 by those of the group who remained at Berkeley, but by then many of us were leaving school, and the impetus behind our ideas left with us.

Ken Arnold is, perhaps, the most famous of our original group. He stayed at Berkeley longer than any of the rest of us, and became well known for such contributions as **Termlib**, **curses**,**fortune**, Mille Bourne, and of course his co-authorship of Rogue. But somehow it seemed a Pyrrhic victory even for Ken; much of his best work in the early years never saw the light of day.

We could not help but feel that we had passed through a sort of Dark Age for UNIX development, and even with the Renaissance in full bloom, We ponder what might have been, and bewail the features that UNIX will now never have.

Doug Merritt became one of the earliest UNIX users outside Bell Laboratories while attending UC Berkeley in 1976. He helped to debug **termcap** *and contributed to the development of* **vi** *and* **curses**. *Mr. Merritt now works as a consultant in the San Francisco Bay Area.*

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

Best known as the author of **curses** *and co-author of Rogue, Ken Arnold was also President of the Berkeley Computer Club and the Computer Science Undergraduates Association during his years at UC Berkeley. He currently works as a programmer in the Computer Graphics Lab at UC San Francisco and serves as a member of the* UNIX REVIEW *Software Review Board.* ■

UNIX Update™ 1

Building New UNIX Networks

Mainframes to Micros

INTERACTIVE Systems Corporation and its OEMs are introducing a series of new products that allow users to distribute their computing tasks among central computers, departmental computers, and personal computers running UNIX and/or PC DOS.

Announced To Date

First, INTERACTIVE introduced the Workbench (IS/WB) and IS/3 systems for distribution by its own sales force to DEC users. IS/WB is a set of key UNIX tools running as extensions to DEC's VMS operating system. IS/3 is an enhanced version of the standard AT&T UNIX operating system for PDP and VAX computers. Both support INed, a proprietary full-screen editor that is the primary user interface on all INTERACTIVE operating systems. Both systems also support INmail, an electronic mail system, and INnet, a networking package that links computers so that they share resources and exchange mail.

At the 1983 Fall COMDEX, SCI Systems demonstrated a new family of multi-user microcomputers (SCI 1000) running IN/ix, which is a version of IS/3 with INed. SCI offers INmail and INnet as applications.

At the UNIFORUM show in early 1984, IBM demonstrated PC/IX, a version of IS/3 with INed, on the IBM PC XT, IBM PC XT/370, and the IBM PC with Fixed Disk Expansion. At the same show, INTERACTIVE demonstrated software that allows personal computers running PC DOS to act as intelligent terminals to any system running INed.

In June, IBM announced that INmail, INnet, and INfort, INTERACTIVE's Fortran 77 compiler, were available as applications for PC/IX. In July, IBM announced VM/IX, a version of IS/3 with INed, which runs as a guest operating system on their Virtual Machine/System Product (VM/SP). And at the fall Expo '84 in LA and UNIX Expo in NYC, IBM announced and demonstrated PC/IX on the new IBM Personal Computer AT.

The Net Effect

INTERACTIVE is the only company to offer a complete network of UNIX-based operating systems, utilities, and applications for building corporate communications systems. Our products provide compatibility across a wide range of processors, making virtually any configuration of PC's, minis, multi-user micros, and mainframes possible. We are also unique in offering full-service support and maintenance, as well as extensive training programs to assist our customers in implementing UNIX networks.

Watch for more on our latest developments in the next INTERACTIVE UNIX Update.

Software Tools for System Builders.
For more information, contact:

INTERACTIVE
SYSTEMS CORPORATION

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

See us in Booth 1541 at UNIFORUM in Dallas, Jan. 21-25.

UNIX is a trademark of AT&T Bell Laboratories. DEC, VMS, PDP, and VAX are trademarks of Digital Equipment Corporation. IN/ix is a trademark of INTERACTIVE Systems Corporation. IBM is a registered trademark of International Business Machines Corporation.

Circle No. 68 on Inquiry Card

Continued from Page 28

improved productivity. According to Canaday, no one had yet been able to provide statistical evidence showing that timesharing was more productive than batch processing. In all the studies, the difference between individual programmers was greater than the difference between timesharing and batch processing. But the subjective evidence was that timesharing was superior. That evidence, in time, prevailed. Regardless of any "proof", the users wanted UNIX and their management gave it to them. What has since become a universal pattern for the spread of UNIX was already underway. As Canaday put it, "The reason people are so devoted to UNIX is that Ken and Dennis wanted to build something that they would enjoy using, and they succeeded."

Dick Haight agrees. "I don't believe UNIX is Utopia. It's just the best set of tools around. I think the reason they're as good as they are is they weren't invented by some committee sitting down and saying 'we ought to have an editor and a **move** command and a **copy** command and a **diff** command.' It was done by people who needed these things and who worked together in a little room where they could complain to each other immediately when something was done that was inconsistent with the spirit of things. Right from the start, Thompson and Ritchie put people on UNIX who had real work to do. In a way, it was done in a vacuum — just their little coterie in Bell Labs research, but these people turned out to be a very representative set."

Two tracks of UNIX expansion within Bell — the UNIX Development Support department and the Programmers Workbench group — merged in 1975 under the Users' Support Group heading.

LETTING GO

Somewhat earlier, there also had been a movement to release Research UNIX to the outside world. Universities, in particular, had heard of it and were very interested. Berkley Tague, sensitive to the real requirements of support, opposed the idea. "I was very concerned that some bank in the Midwest would put its payrolls on UNIX and get in terrible trouble. The fact is that didn't happen. It went to a few technical shops; it went to universities; it was very influential in all the places it should be influential. The people who picked it up were, by and large, people who could deal with the kind of support we offered — or didn't offer, if you'd like to put it that way. That was when we basically said, 'Here's a tape; take it.'

"If you'd asked me at the time if releasing UNIX to the universities was a good idea, I would have said no. I have been very grateful since then, particularly when hiring new people, that no one paid any attention to me."

August Mohr is former editor of the /usr/group newsletter, CommUNIXations. He is currently acting as a consultant and developing in-house utilities for electronic publishing. He is also at work on a book. ■

Continued from Page 28

V, both products of the development group. All research versions were "as is," unsupported software; System V is a supported product on several different hardware lines, most recently including the 3B systems designed and built by AT&T.

UNIX is in wide use, and is now even spoken of as a possible industry standard. How did it come to succeed?

There are, of course, its technical merits. Because the system and its history have been discussed at some length in the literature [6, 7, 11], I will not talk about these qualities except for one. Despite its frequent surface inconsistency, so colorfully annotated by Don Norman in his *Datamation* article [4] and despite its richness, UNIX is a simple, coherent system that pushes a few good ideas and models to the limit. It is this aspect of the system, above all, that endears it to its adherents.

Beyond technical considerations, there were sociological forces that contributed to its success. First, it appeared at a time when alternatives to large, centrally administered computation centers were becoming possible; the 1970s were the decade of the minicomputer. Small groups could set up their own computational facilities. Because they were starting afresh, and because manufacturers' software was, at best, unimaginative and often horrible, some adventuresome people were willing to take a chance on a new and intriguing, even though unsupported, operating system.

Second, UNIX was first available on the PDP-11, one of the most successful of the new minicomputers that appeared in the 1970s, and soon its portability brought it to many new machines as they appeared. At the time that UNIX was created, we were pushing hard for a machine, either a DEC PDP-10 or SDS (later Xerox) Sigma 7. It is certain, in retrospect, that if we had succeeded in acquiring such a machine, UNIX might have been written but would have withered away. Similarly, UNIX owes much to Multics [5], I have described [6, 7], it eclipsed its parent as much because it does not demand unusual hardware support as because of any other qualities.

Finally, UNIX enjoyed an unusually long gestation period. During much of this time (say 1969-1979), the system was effectively under the control of its designers and being used by them. It took time to develop all of the ideas and software, but even though the system was still being developed, people were using it, both inside Bell Labs, and outside under license. Thus, we managed to keep the central ideas in hand, while accumulating a base of enthusiastic, technically competent users who contributed ideas and programs in a calm, communicative, and noncompetitive environment. Some outside contributions were substantial, such as those from the University of California at Berkeley. Our users were widely, though thinly, distributed within the company, at universities, and at some commercial and government organizations. The system became important in the intellectual, if not yet commercial, marketplace because of this network of early users.

What does industrial computer science research consist of? Some people have the impression that the original UNIX work was a bootleg project, a "skunk works". This is not so. Research workers are supposed to discover or invent new things, and although in the early days we subsisted on meager hardware,

we always had management encouragement. At the same time, it was certainly nothing like a development project. Our intent was to create a pleasant computing environment for ourselves, and our hope was that others liked it.

The Computing Science Research Center at Bell Laboratories to which Thompson and I belong studies three broad areas: theory; numerical analysis; and system, languages, and software. Although work for its own sake resulting, for example, in a paper in a learned journal, is not only tolerated but welcomed, there is strong though wonderfully subtle pressure to think about problems somehow relevant to our corporation. This has been so since I joined Bell Labs around 15 years ago, and it should not be surprising, the old Bell System may have seemed a sheltered monopoly, but research has always had to pay its way. Indeed, researchers love to find problems to work on; one of the advantages of doing research in a large company is the enormous range of the puzzles that turn up. For example, theorists may contribute to compiler design, or to LSI algorithms; numerical analysts study charge and current distribution in semiconductors; and, of course, software types like to design systems and write programs that people use. Thus, computer research at Bell Labs has always had considerable commitment to the world, and does not fear edicts commanding us to be practical.

For some of us, in fact, a principal frustration has been the inability to convince others that our research products can indeed be useful. Someone may invent a new application, write an illustrative program, and put it to use in our own lab. Many such demonstrations require further development and continuing support in order for the company to make best use of them. In the past, this use would have been exclusively inside the Bell System; more recently, there is the possibility of developing a product for direct sale.

For example, some years ago Mike Lesk developed an automated directory-assistance system [3]. The program had an online Bell Labs phone book, and was connected to a voice synthesizer on a telephone line with a tone decoder. One dialed the system, and keyed in a name and location code on the telephone's key pad; it spoke back the person's telephone number and office address (it didn't attempt to pronounce the name). In spite of the hashing through 12 buttons (which, for example, squashed "A", "B" and "C" together), it was acceptably accurate: it had to give up on around 5 percent of the tries. The program was a local hit and well-used. Unfortunately, we couldn't find anyone to take it over, even as a supported service within the company, let alone a public offering, and it was an excessive drain on our resources, so it was finally scrapped. (I chose this example not only because it is old enough not to exacerbate any current squabbles, but also because it is timely: the organization that publishes the company telephone directory recently asked

us whether the system could be revived.)

Of course not every idea is worth developing or supporting. In any event, the world is changing: our ideas and advice are being sought much more avidly than before. This increase in influence has been going on for several years, partly because of the success of UNIX, but more recently, because of the dramatic alteration of the structure of our company.

AT&T divested its telephone operating companies at the beginning of 1984. There has been considerable public speculation about what this will mean for fundamental research at Bell Laboratories; one report in *Science* [2] is typical. One fear sometimes expressed is that basic research, in general, may languish because it yields insufficient short-term gains to the new, smaller AT&T. The public position of the company is reassuring: moreover, research management at Bell Labs seems to believe deeply, and argues persuasively, that the commitment of support to basic research is deep and will continue [1].

Fundamental research at Bell Labs in physics, chemistry, and mathematics may, indeed, not be threatened; nevertheless, the danger it might face, and the case against which it must be prepared to argue, is that of irrelevance to the goals of the company. Computer science research is different from these more traditional disciplines. Philosophically it differs from the physical sciences because it seeks not to discover, explain, or exploit the natural world, but instead to study the properties of machines of human creation. In this it is analogous to mathematics, and indeed the "science" part of computer science is, for the most part, mathematical in spirit. But an inevitable aspect of computer science is the creation of computer programs: objects that, though intangible, are subject to commercial exchange.

More than anything else, the greatest danger to good computer science research today may be *excessive* relevance. Evidence for the worldwide fascination with computers is everywhere, from the articles on the financial, and even the front pages of the newspapers, to the difficulties that even the most prestigious universities experience in finding and keeping faculty in computer science. The best professors, instead of teaching bright students, join start-up companies, and often discover that their brightest students have preceded them. Computer science is in the limelight, especially those aspects, such as systems, languages, and machine architecture, that may have immediate commercial applications. The attention is flattering, but it can work to the detriment of good research.

As the intensity of research in a particular area increases, so does the impulse to keep its results secret. This is true even in the university (Watson's account [12] of the discovery of the structure of DNA provides a well-known example), although in academia there is a strong counterpressure: unless one publishes, one never

we always had management encouragement. At the same time, it was certainly nothing like a development project. Our intent was to create a pleasant computing environment for ourselves, and our hope was that others liked it.

The Computing Science Research Center at Bell Laboratories to which Thompson and I belong studies three broad areas: theory; numerical analysis; and system, languages, and software. Although work for its own sake resulting, for example, in a paper in a learned journal, is not only tolerated but welcomed, there is strong though wonderfully subtle pressure to think about problems somehow relevant to our corporation. This has been so since I joined Bell Labs around 15 years ago, and it should not be surprising, the old Bell System may have seemed a sheltered monopoly, but research has always had to pay its way. Indeed, researchers love to find problems to work on; one of the advantages of doing research in a large company is the enormous range of the puzzles that turn up. For example, theorists may contribute to compiler design, or to LSI algorithms; numerical analysts study charge and current distribution in semiconductors; and, of course, software types like to design systems and write programs that people use. Thus, computer research at Bell Labs has always had considerable commitment to the world, and does not fear edicts commanding us to be practical.

For some of us, in fact, a principal frustration has been the inability to convince others that our research products can indeed be useful. Someone may invent a new application, write an illustrative program, and put it to use in our own lab. Many such demonstrations require further development and continuing support in order for the company to make best use of them. In the past, this use would have been exclusively inside the Bell System; more recently, there is the possibility of developing a product for direct sale.

For example, some years ago Mike Lesk developed an automated directory-assistance system [3]. The program had an online Bell Labs phone book, and was connected to a voice synthesizer on a telephone line with a tone decoder. One dialed the system, and keyed in a name and location code on the telephone's key pad; it spoke back the person's telephone number and office address (it didn't attempt to pronounce the name). In spite of the hashing through 12 buttons (which, for example, squashed "A", "B" and "C" together), it was acceptably accurate: it had to give up on around 5 percent of the tries. The program was a local hit and well-used. Unfortunately, we couldn't find anyone to take it over, even as a supported service within the company, let alone a public offering, and it was an excessive drain on our resources, so it was finally scrapped. (I chose this example not only because it is old enough not to exacerbate any current squabbles, but also because it is timely: the organization that publishes the company telephone directory recently asked

us whether the system could be revived.)

Of course not every idea is worth developing or supporting. In any event, the world is changing: our ideas and advice are being sought much more avidly than before. This increase in influence has been going on for several years, partly because of the success of UNIX, but more recently, because of the dramatic alteration of the structure of our company.

AT&T divested its telephone operating companies at the beginning of 1984. There has been considerable public speculation about what this will mean for fundamental research at Bell Laboratories; one report in *Science* [2] is typical. One fear sometimes expressed is that basic research, in general, may languish because it yields insufficient short-term gains to the new, smaller AT&T. The public position of the company is reassuring: moreover, research management at Bell Labs seems to believe deeply, and argues persuasively, that the commitment of support to basic research is deep and will continue [1].

Fundamental research at Bell Labs in physics, chemistry, and mathematics may, indeed, not be threatened; nevertheless, the danger it might face, and the case against which it must be prepared to argue, is that of irrelevance to the goals of the company. Computer science research is different from these more traditional disciplines. Philosophically it differs from the physical sciences because it seeks not to discover, explain, or exploit the natural world, but instead to study the properties of machines of human creation. In this it is analogous to mathematics, and indeed the "science" part of computer science is, for the most part, mathematical in spirit. But an inevitable aspect of computer science is the creation of computer programs: objects that, though intangible, are subject to commercial exchange.

More than anything else, the greatest danger to good computer science research today may be *excessive* relevance. Evidence for the worldwide fascination with computers is everywhere, from the articles on the financial, and even the front pages of the newspapers, to the difficulties that even the most prestigious universities experience in finding and keeping faculty in computer science. The best professors, instead of teaching bright students, join start-up companies, and often discover that their brightest students have preceded them. Computer science is in the limelight, especially those aspects, such as systems, languages, and machine architecture, that may have immediate commercial applications. The attention is flattering, but it can work to the detriment of good research.

As the intensity of research in a particular area increases, so does the impulse to keep its results secret. This is true even in the university (Watson's account [12] of the discovery of the structure of DNA provides a well-known example), although in academia there is a strong counterpressure: unless one publishes, one never

us whether the system could be revived.)

Of course not every idea is worth developing or supporting. In any event, the world is changing: our ideas and advice are being sought much more avidly than before. This increase in influence has been going on for several years, partly because of the success of UNIX, but more recently, because of the dramatic alteration of the structure of our company.

AT&T divested its telephone operating companies at the beginning of 1984. There has been considerable public speculation about what this will mean for fundamental research at Bell Laboratories; one report in *Science* [2] is typical. One fear sometimes expressed is that basic research, in general, may languish because it yields insufficient short-term gains to the new, smaller AT&T. The public position of the company is reassuring: moreover, research management at Bell Labs seems to believe deeply, and argues persuasively, that the commitment of support to basic research is deep and will continue [1].

Fundamental research at Bell Labs in physics, chemistry, and mathematics may, indeed, not be threatened; nevertheless, the danger it might face, and the case against which it must be prepared to argue, is that of irrelevance to the goals of the company. Computer science research is different from these more traditional disciplines. Philosophically it differs from the physical sciences because it seeks not to discover, explain, or exploit the natural world, but instead to study the properties of machines of human creation. In this it is analogous to mathematics, and indeed the "science" part of computer science is, for the most part, mathematical in spirit. But an inevitable aspect of computer science is the creation of computer programs: objects that, though intangible, are subject to commercial exchange.

More than anything else, the greatest danger to good computer science research today may be *excessive* relevance. Evidence for the worldwide fascination with computers is everywhere, from the articles on the financial, and even the front pages of the newspapers, to the difficulties that even the most prestigious universities experience in finding and keeping faculty in computer science. The best professors, instead of teaching bright students, join start-up companies, and often discover that their brightest students have preceded them. Computer science is in the limelight, especially those aspects, such as systems, languages, and machine architecture, that may have immediate commercial applications. The attention is flattering, but it can work to the detriment of good research.

As the intensity of research in a particular area increases, so does the impulse to keep its results secret. This is true even in the university (Watson's account [12] of the discovery of the structure of DNA provides a well-known example), although in academia there is a strong counterpressure: unless one publishes, one never

we always had management encouragement. At the same time, it was certainly nothing like a development project. Our intent was to create a pleasant computing environment for ourselves, and our hope was that others liked it.

The Computing Science Research Center at Bell Laboratories to which Thompson and I belong studies three broad areas: theory; numerical analysis; and system, languages, and software. Although work for its own sake resulting, for example, in a paper in a learned journal, is not only tolerated but welcomed, there is strong though wonderfully subtle pressure to think about problems somehow relevant to our corporation. This has been so since I joined Bell Labs around 15 years ago, and it should not be surprising. the old Bell System may have seemed a sheltered monopoly, but research has always had to pay its way. Indeed, researchers love to find problems to work on; one of the advantages of doing research in a large company is the enormous range of the puzzles that turn up. For example, theorists may contribute to compiler design, or to LSI algorithms; numerical analysts study charge and current distribution in semiconductors; and, of course, software types like to design systems and write programs that people use. Thus, computer research at Bell Labs has always had considerable commitment to the world, and does not fear edicts commanding us to be practical.

For some of us, in fact, a principal frustration has been the inability to convince others that our research products can indeed be useful. Someone may invent a new application, write an illustrative program, and put it to use in our own lab. Many such demonstrations require further development and continuing support in order for the company to make best use of them. In the past, this use would have been exclusively inside the Bell System; more recently, there is the possibility of developing a product for direct sale.

For example, some years ago Mike Lesk developed an automated directory-assistance system [3]. The program had an online Bell Labs phone book, and was connected to a voice synthesizer on a telephone line with a tone decoder. One dialed the system, and keyed in a name and location code on the telephone's key pad; it spoke back the person's telephone number and office address (it didn't attempt to pronounce the name). In spite of the hashing through 12 buttons (which, for example, squashed "A", "B" and "C" together), it was acceptably accurate: it had to give up on around 5 percent of the tries. The program was a local hit and well-used. Unfortunately, we couldn't find anyone to take it over, even as a supported service within the company, let alone a public offering, and it was an excessive drain on our resources, so it was finally scrapped. (I chose this example not only because it is old enough not to exacerbate any current squabbles, but also because it is timely: the organization that publishes the company telephone directory recently asked

SOFTWARE ENGINEERS

The Convergent Direction

Since 1979, we've provided solutions to the OEM marketplace — seven successful products each developed in less than 1 year from idea to customer ship. As one of the fastest publicly held California companies, our revenues in 1983 were $163 million. Small, cohesive development teams work in start-up fashion to design, manufacture and market technically aggressive products that sell.

Data Systems Division is only 2 years old and profitable — 2 major products have been designed and shipped:

The MegaFrame™ system is an innovative multiprocessor system designed to expand from 8 to 128 users using both CTOS™ and UNIX™ operating systems. MiniFrame™ is a high-performance, low cost UNIX™ system.

Both machines run an unusual demand-paged UNIX™ system V with Berkeley extensions; both support multi-window intelligent terminals.

We've shipped thousands of each in one year.

If you are one of the best **software engineers** around, if you enjoy working with high caliber people in an energetic environment, consider the following opportunities.

- **Human Interface**
- **Test Automation & Performance Testing**
- **Communications & Networking**
- **Intelligent Terminal Software/Graphics**
- **Compiler Writers/Toolsmiths**
- **Distributed Operating Systems/Utilities (UNIX)**
- **Diagnostics**
- **Applications-Word Processing**

All positions require 2+ years experience; BSCS preferred with mini/micro background.

Send your resume to Joy Mar, Convergent Technologies, Dept. UR/J, 30 East Plumeria Drive, San Jose, CA 95134. We are an equal opportunity employer. Principals only please.

Convergent Technologies

Where great ideas ^ come together.

and great people

becomes known at all. In industry, a natural impulse of the establishment is to guard proprietary information. Researchers understand reasonable restrictions on what and when they publish, but many will become irritated and flee elsewhere, or start working in less delicate areas, if prevented from communicating their discoveries and inventions in suitable fashion. Research management at Bell Labs has traditionally been sensitive to maintaining a careful balance between company interest and the industrial equivalent of academic freedom. The entrance of AT&T into the computer industry will test, and perhaps strain, this balance.

Another danger is that commercial pressure of one sort or another will divert the attention of the best thinkers from real innovation to exploitation of the current fad, from prospecting to mining a known lode. These pressures manifest themselves not only in the disappearance of faculty into industry, but also in the conservatism that overtakes those with well-paying investments — intellectual or financial — in a given idea. Perhaps this effect explains why so few interesting software systems have come from the large computer companies: they are locked into the existing world. Even IBM, which supports a well-regarded and productive research establishment, has in recent years produced little to cause even a minor revolution in the way people think about computers. The working examples of important new systems seem to have come either from entrepreneurial efforts (VisiCalc is a good example) or from large companies, like Bell Labs and most especially Xerox, that were much involved with computers and could afford research into them, but did not regard them as their primary business.

On the other hand, in smaller companies, even the most vigorous research support is highly dependent on market conditions. *The New York Times*, in an article describing Alan Kay's passage from Atari to Apple, notes the problem: "Mr. Kay...said that Atari's laboratories had lost some of the atmosphere of innovation that once attracted some of the finest talent in the industry. "When I left last month it was clear that they would be putting their efforts in the short term," he said..."I guess the tree of research must from time to time be refreshed with the blood of bean counters."[9]

Partly because they are new and still immature, and partly because they are a creation of the intellect, the arts and sciences of software abridge the chain, usually in physics and engineering, between fundamental discoveries, advanced development, and application. The inventors of ideas about how software should work usually find it necessary to build demonstration systems. For large systems, and for revolutionary ideas, much time is required. It can be said that UNIX was written in the '70s to distill the best systems ideas of the '60s, and became the commonplace of the '80s. The work at Xerox PARC on personal computers, bitmap graphics, and programming environments [10] shows a similar progression, starting and coming to fruition a few years later. Time, and a commitment to the long-term value of the research, are needed on the part of both the researchers and their management.

Bell Labs has provided this commitment and more: a rare and uniquely stimulating research environment for my colleagues and me. As it enters what company publications call "the new competitive era", its managers and workers will do well to keep in mind how, and under what conditions, the UNIX system succeeded. If we can keep alive enough openness to new ideas, enough freedom of communication, enough patience to allow the novel to prosper, it will remain possible for a future Ken Thompson to find a little-used CRAY/I computer and fashion a system as creative, and as influential, as UNIX.

REFERENCES

1. Bell Labs: New order augurs well. *Nature 305*, 5933 (Sept. 29, 1983).
2. Bell Labs on the brink. *Science 221* (Sept. 23, 1983).
3. Lesk, M.E. User-activated BTL directory assistance. Bell Laboratories internal memorandum (1972).
4. Norman, D.A. The truth about UNIX. *Datamation 27* (1981).
5. Organick, E.I. The Multics System *MIT Press* Cambridge, M.A. (1972)
6. Ritchie, D. M. UNIX time-sharing system: A Restrospective. *Bell Syst. Tech. J.* 57.6 (1978) (1947-1969.
7. Ritchie, D. M. The Evolution of the UNIX time-sharing system. In Language Design and Programming Methodology, Jeffrey M. Tobias, Ed. *Springer-Verlag*, New York. (1980).
8. Ritchie, D. M. and Thompson, K. The UNIX time-sharing system. *Commun. ACM* 17.7 (July 1974) 365-375.
9. Sanger, D. E. Key Atari scientist switches to Apple. *The New York Times* 133,46,033 (May 3, 1984).
10. Thacker, C. P. et al. Alto, a personal computer. *Xerox PARC Technical Report* CSL-79-11.
11. Thompson, K. UNIX time-sharing system: UNIX implementation. *Bell Syst. Tech J.* 57,6 (1978). 1931-1946.
12. Watson, J. D. The Double Helix; A Personal Account of the Discovery of the Structure of DNA. *Atheneum Publishers*, New York (1968).

SOFTWARE ENGINEERS

The Convergent Direction

Since 1979, we've provided solutions to the OEM marketplace — seven successful products each developed in less than 1 year from idea to customer ship. As one of the fastest publicly held California companies, our revenues in 1983 were $163 million. Small, cohesive development teams work in start-up fashion to design, manufacture and market technically aggressive products that sell.

Data Systems Division is only 2 years old and profitable — 2 major products have been designed and shipped:

The MegaFrame™ system is an innovative multiprocessor system designed to expand from 8 to 128 users using both CTOS™ and UNIX™ operating systems. MiniFrame™ is a high-performance, low cost UNIX™ system.

Both machines run an unusual demand-paged UNIX™ system V with Berkeley extensions; both support multi-window intelligent terminals.

We've shipped thousands of each in one year.

If you are one of the best **software engineers** around, if you enjoy working with high caliber people in an energetic environment, consider the following opportunities.

- **Human Interface**
- **Test Automation & Performance Testing**
- **Communications & Networking**
- **Intelligent Terminal Software/Graphics**
- **Compiler Writers/Toolsmiths**
- **Distributed Operating Systems/Utilities (UNIX)**
- **Diagnostics**
- **Applications-Word Processing**

All positions require 2+ years experience; BSCS preferred with mini/micro background.

Send your resume to Joy Mar, Convergent Technologies, Dept. UR/J, 30 East Plumeria Drive, San Jose, CA 95134. We are an equal opportunity employer. Principals only please.

Convergent Technologies

Where great ideas come together.

and great people

becomes known at all. In industry, a natural impulse of the establishment is to guard proprietary information. Researchers understand reasonable restrictions on what and when they publish, but many will become irritated and flee elsewhere, or start working in less delicate areas, if prevented from communicating their discoveries and inventions in suitable fashion. Research management at Bell Labs has traditionally been sensitive to maintaining a careful balance between company interest and the industrial equivalent of academic freedom. The entrance of AT&T into the computer industry will test, and perhaps strain, this balance.

Another danger is that commercial pressure of one sort or another will divert the attention of the best thinkers from real innovation to exploitation of the current fad, from prospecting to mining a known lode. These pressures manifest themselves not only in the disappearance of faculty into industry, but also in the conservatism that overtakes those with well-paying investments — intellectual or financial — in a given idea. Perhaps this effect explains why so few interesting software systems have come from the large computer companies: they are locked into the existing world. Even IBM, which supports a well-regarded and productive research establishment, has in recent years produced little to cause even a minor revolution in the way people think about computers. The working examples of important new systems seem to have come either from entrepreneurial efforts (VisiCalc is a good example) or from large companies, like Bell Labs and most especially Xerox, that were much involved with computers and could afford research into them, but did not regard them as their primary business.

On the other hand, in smaller companies, even the most vigorous research support is highly dependent on market conditions. *The New York Times*, in an article describing Alan Kay's passage from Atari to Apple, notes the problem: "Mr. Kay...said that Atari's laboratories had lost some of the atmosphere of innovation that once attracted some of the finest talent in the industry. "When I left last month it was clear that they would be putting their efforts in the short term," he said..."I guess the tree of research must from time to time be refreshed with the blood of bean counters."[9]

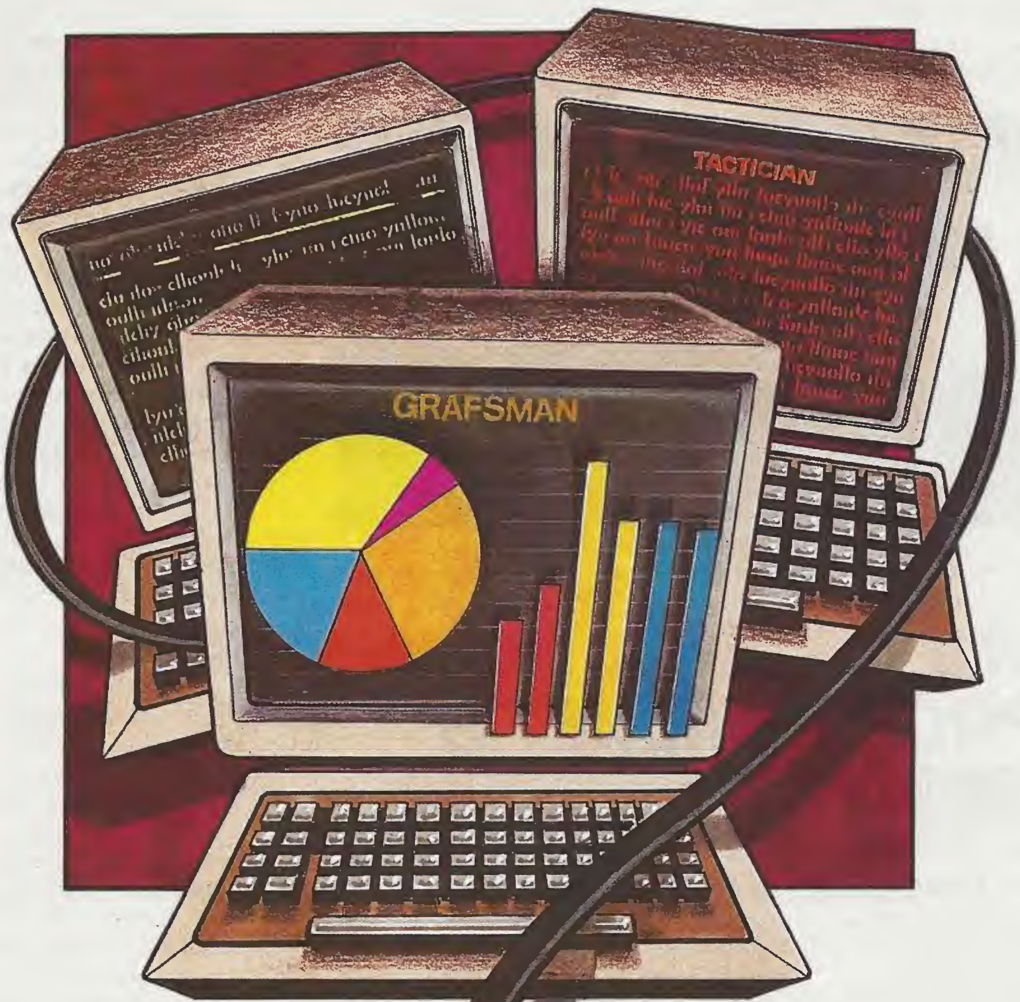
Partly because they are new and still immature, and partly because they are a creation of the intellect, the arts and sciences of software abridge the chain, usually in physics and engineering, between fundamental discoveries, advanced development, and application. The inventors of ideas about how software should work usually find it necessary to build demonstration systems. For large systems, and for revolutionary ideas, much time is required. It can be said that UNIX was written in the '70s to distill the best systems ideas of the '60s, and became the commonplace of the '80s. The work at Xerox PARC on personal computers, bitmap graphics, and programming environments [10] shows a similar progression, starting and coming to fruition a few years later. Time, and a commitment to the long-term value of the research, are needed on the part of both the researchers and their management.

Bell Labs has provided this commitment and more: a rare and uniquely stimulating research environment for my colleagues and me. As it enters what company publications call "the new competitive era", its managers and workers will do well to keep in mind how, and under what conditions, the UNIX system succeeded. If we can keep alive enough openness to new ideas, enough freedom of communication, enough patience to allow the novel to prosper, it will remain possible for a future Ken Thompson to find a little-used CRAY/I computer and fashion a system as creative, and as influential, as UNIX.

REFERENCES

1. Bell Labs: New order augurs well. *Nature 305*, 5933 (Sept. 29, 1983).
2. Bell Labs on the brink. *Science 221* (Sept. 23, 1983).
3. Lesk, M.E. User-activated BTL directory assistance. Bell Laboratories internal memorandum (1972).
4. Norman, D.A. The truth about UNIX. *Datamation 27* (1981).
5. Organick, E.I. The Multics System *MIT Press* Cambridge, M.A. (1972)
6. Ritchie, D. M. UNIX time-sharing system: A Restrospective. *Bell Syst. Tech. J.* 57.6 (1978) (1947-1969.
7. Ritchie, D. M. The Evolution of the UNIX time-sharing system. In Language Design and Programming Methodology, Jeffrey M. Tobias, Ed. *Springer-Verlag*, New York. (1980).
8. Ritchie, D. M. and Thompson, K. The UNIX time-sharing system. *Commun. ACM* 17.7 (July 1974) 365-375.
9. Sanger, D. E. Key Atari scientist switches to Apple. *The New York Times* 133,46,033 (May 3, 1984).
10. Thacker, C. P. et al. Alto, a personal computer. *Xerox PARC Technical Report* CSL-79-11.
11. Thompson, K. UNIX time-sharing system: UNIX implementation. *Bell Syst. Tech J.* 57,6 (1978). 1931-1946.
12. Watson, J. D. The Double Helix; A Personal Account of the Discovery of the Structure of DNA. *Atheneum Publishers*, New York (1968).

Fast...Versatile ...Powerful Tools



Integrated Productivity Tools from SouthWind Software provide today's professional with fast, versatile tools to organize, manipulate and display information with clarity and effectiveness.

TACTICIAN is an enhanced 1024 x 1024 spreadsheet which interacts with popular UNIX DBMS ie. UNIFY, INFORMIX and MICROINGRES.

GRAFSMAN's panel driven graphic editor helps create brilliant color graphics for reports and presentations using data from TACTICIAN, DBMS, or application files.

Integrated Productivity Tools were designed with the OEM in mind to allow for quick, easy incorporation into problem solving applications. IPT is available today for a wide variety of UNIX/XENIX based systems.

Call for further information about the fast, versatile and powerful Integrated Productivity Tools family.

See us at UniForum Booth #3337

Integrated Productivity Tools, IPT, TACTICIAN, and GRAFSMAN are trademarks of SouthWind Software, Inc. UNIX is a trademark of AT&T Bell Laboratories. UNIFY is a trademark of the Unify Corporation. INFORMIX is a trademark of Relational Database Systems, Inc. MICROINGRES is a trademark of Relational Technologies, Inc. XENIX is a trademark of Microsoft, Inc.

SOUTHWIND SOFTWARE, INC.
4520 E. 47th St. So.
Wichita, Kansas 67210
316-788-5537
1-800-346-3025 EXT 234

SOUTHWIND SOFTWARE

Circle No. 75 on Inquiry Card

Continued from Page 54

analogs for these.

The Interlisp DWIM (Do-What-I-Mean) feature causes many errors and incomplete specifications to be remedied by the system. If the necessary modification is trivial and unambiguous in nature, the repair is performed silently. This is very different from the UNIX philosophy of error handling.

The Interlisp-D and Smalltalk environments are both notable for their active use of windows to aid the programming process. Windows can be created and removed trivially, and the user is encouraged to build new windows to perform specific tasks. No current UNIX workstation offers the degree of ease and flexibility to be found in using windows under these systems.

IN SUMMARY

Workstations are here to stay. They vary in detail, but are almost identical in general characteristics. A number of forces will cooperate to increase their numbers.

Packaging, mechanical components, displays, and so forth are expensive, and will stay expensive. Processors, memory, and so forth get cheaper all the time. The cost of turning a terminal into a workstation is thus dropping rapidly.

The nifty new software products we see emerging are costly in processing time. A typical mainframe cannot support large numbers of drafting packages, plotting applications, or even spreadsheets. In any case, terminals are generally too bandwidth-limited to do satisfactory interactive graphics.

Get ready, therefore, to see workstations replacing terminals on ever-increasing numbers of desks. You might even start thinking about where to put yours. ■

CALENDAR

EVENTS

JANUARY

January 21-25 UniForum, The International Conference of UNIX Users, Dallas, TX. Contact: UniForum/Professional Exposition Management Co., 2400 East Devon Avenue, Des Plaines, IL 60018. 800/323-5155, or in IL, 312/299-3131.
January 29 Silicon Valley Net Monthly Meeting, Palo Alto, CA. Presentation and discussion sponsored by the UNIX user group. Contact: SVNet, 5301 Stevens Creek Blvd., Santa Clara, CA 95050.

MARCH

March 30-April 2 10th West Coast Computer Faire, San Francisco, CA. Contact: Computer Faire, Inc., 181 Wells Avenue, Newton, MA 02159. 617/965-8350.

APRIL

April 24-26 UNIX Systems Expo/85, San Francisco, CA. Contact: Computer Faire, Inc. (see above).

TRAINING

JANUARY

January 7 AT&T Technologies, Princeton, NJ: "Shell Command Language for Programmers." Contact: AT&T Technologies, Corporate Education & Training, PO Box 2000, Hopewell, NJ 08525. 800/221-1647.
January 7 AT&T Technologies, Sunnyvale, CA: "Internal UNIX System Calls and Libraries Using C Programming." Contact: AT&T (see January 7).
January 7 AT&T Technologies, Lisle, IL: "UNIX System V Internals." Contact: AT&T (see January 7).
January 7 NCR Education Seminar, New York, NY: "UNIX Operating System." Contact: NCR Customer and Support Education, 101 W. Schantz Avenue, Dayton, OH 45479. 800/845-CASE, or in OH, 800/841-CASE.
January 7-11 Plum Hall Training, Raleigh, NC: "UNIX Workshop." Contact: Plum Hall, 1 Spruce Avenue, Cardiff, NJ 08232. 609/927-3770.
January 9 AT&T Technologies, Sunnyvale, CA: "Fundamentals of the UNIX Operating System for Users." Contact: AT&T (see January 7).
January 9-11 CAPE Seminar, Cincinnati, OH: "A User-Oriented Evaluation Three-Day Seminar: UNIX." Contact: Center for Advanced Professional Education, 1820 East Garry Street, Suite 110, Santa Ana, CA 92705. 714/261-0240.
January 9-11 Digital Seminar Program, Cambridge, MA: "Comprehensive Overview of the UNIX Operating System." Contact: Digital Educational Services, 12 Crosby Drive, Bedford, MA 01730. 617/276-4949.
January 12 International Technical Seminars, San Francisco, CA: "Programming Control Tools" with Bob Toxen; "4.2 Fast File System Overview" with Kirk McKusick; "4.2 Fast File System Administration" with Kirk McKusick. Contact: ITS, 520 Waller Street, San Francisco, CA 94117. 415/621-6415.
January 13 International Technical Seminars, San Francisco, CA: "Writing **termcap** Entries" with Doug Merritt; "Fast Prototyping on UNIX" with Rich Morin and Gene Dronek; "4.2 Interprocess Communication" with Ken Arnold; "Using **curses** Effectively" with Ken Arnold. (See January 12).
January 14 AT&T Technologies, Princeton, NJ & Sunnyvale, CA: "C Language for Programmers." Contact: AT&T (see January 7).
January 14 AT&T Technologies, Lisle, IL & Sunnyvale, CA: "Fundamentals of the UNIX Operating System for Programmers." Contact: AT&T (see January 7).
January 14 AT&T Technologies, Sunnyvale, CA: "UNIX System V Device Drivers." Contact: AT&T (see January 7).
January 14 AT&T Technologies, Sunnyvale, CA: "UNIX System V Internals." Contact: AT&T (see January 7).
January 14 NCR Education Seminars, Los Angeles, CA & Houston, TX: "UNIX Operating System." Contact: NCR (see January 7).
January 14-16 Computer Technology Group, Chicago, IL: "UNIX Fundamentals for Programmers." Contact: CTG, Telemedia, Inc., 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX, or in IL, 312/987-4082.
January 14-18 Structured Methods Seminar, New York, NY: "UNIX System Workshop." Contact: Structured Methods, 7 West 18th Street, New York, NY 10011. 800/221-9274, or in NY, 212/741-7720.
January 15 Computer Technology Group, New York, NY & Washington, DC: "UNIX Overview." Contact: CTG (see January 14-16).
January 16 AT&T Technologies, Lisle, IL: "Shell Command Language for Users." Contact: AT&T (see January 7).
January 16 Specialized Systems Consultants, Seattle, WA: "UNIX for Managers." Contact: SSC, PO Box 7, Northgate Station, Seattle, WA 98125. 206/FOR-UNIX.
January 16-18 CAPE Seminar, New York, NY: "A User-Oriented Evaluation Three-Day Seminar: UNIX." Contact: CAPE (see January 9-11).
January 16-18 Computer Technology Group, New York, NY & Washington, DC: "UNIX Fundamentals for Non-Programmers." Contact: CTG (see January 14-16).
January 16-18 Digital Seminar Program, Cambridge, MA: "The C Programming Language Application Development." Contact: Digital Educational Services (see January 9-11). **January 17-18** Computer Technology Group, San Francisco, CA & Chicago, IL: "Shell as a Command Language." Contact: CTG (see January 14-16).
January 17-18 Computer Technology Group, San Diego, CA: "Developing a UNIX Market Strategy." Contact: CTG (see January 14-16).
January 21 AT&T Technologies, Lisle, IL: "C Language for Programmers." Contact: AT&T (see January 7).
January 21 AT&T Technologies, Princeton, NJ: "UNIX System V Internals." Contact: AT&T (see January 7).
January 21 NCR Education Seminar, Houston, TX: "UNIX System Administration." Contact: NCR (see January 7).
January 21 NCR Education Seminar, Los Angeles, CA: "C Programming." Contact: NCR (see January 7).
January 21-22 Intelligent Solution Seminar, Baltimore, MD: "UNIX Concepts." Contact: Intelligent Solution (see December 10-11).
January 21-23 Computer Technology Group, New York, NY & Washington, DC: "UNIX Fundamentals for Programmers." Contact: CTG (see January 14-16).
January 21-25 Bunker Ramo Information Systems, Trumbull, CT: "Introduction to UNIX." Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.
January 21-25 Computer Technology Group, Chicago, IL: "C Language Programming." Contact: CTG (see January 14-16).
January 21-25 Plum Hall Training, Raleigh, NC: "C Programming Workshop." Contact: Plum Hall (see January 7-11).
January 21-25 Structured Methods Seminar, New York, NY: "C Programming Workshop." Contact: Structured Methods (see January 14-18).

January 21-25 Structured Methods Seminar, New York, NY: "Advanced Topics in The C Language." Contact: Structured Methods (see January 14-18).
January 22-25 Integrated Computer Systems Seminar, Baltimore, MD: "Hands-On UNIX Workshop." Contact: ICS, 6305 Arizona Place, Los Angeles, CA 90045. 800/421-8166, or in CA, 800/352-8251.
January 23-24 Intelligent Solution Seminar, Baltimore, MD: "Programming in C." Contact: Intelligent Solution (see December 10-11).
January 23-25 CAPE Seminar, Palo Alto, CA: "A User-Oriented Evaluation Three-Day Seminar: UNIX." Contact: CAPE (see January 9-11).
January 24-25 Computer Technology Group, New York, NY & Washington, DC: "Shell as a Command Language." Contact: CTG (see January 14-16).
January 24-25 Productivity Products International Seminar, Dallas, TX: "An Introduction to Object-Oriented Programming with Objective C." Contact: PPI, 27 Glen Road, Sandyhook, CT 06482. 203/426-1875.
January 25 Intelligent Solution Seminar, Baltimore, MD: "UNIX Overview." Contact: Intelligent Solution (see December 10-11).
January 28 AT&T Technologies, Sunnyvale, CA: "Shell Command Language for Users." Contact: AT&T (see January 7).
January 28 AT&T Technologies, Lisle, IL & Sunnyvale, CA: "UNIX System Document Preparation Utilities." Contact: AT&T (see January 7).
January 28-29 Computer Technology Group, Chicago, IL: "Shell Programming." Contact: CTG (see January 14-16).
January 28-29 Software Institute of America, New York, NY: "UNIX." Contact: SIA, 8 Windsor Street, Andover, MA 01810. 617/470-3880.
January 28-February 1 Bunker Ramo Information Systems, Trumbull, CT: "Advanced UNIX." Contact: Bunker Ramo (see January 21-25).
January 28-February 1 Computer Technology Group, New York, NY & Washington, DC: "C Language Programming." Contact: CTG (see January 14-16).
January 28-February 1 Digital Seminar Program, Dallas, TX: "Programming in C." Contact: Digital Educational Services (see January 9-11).
January 28-February 1 Plum Hall Training, Raleigh, NC: "Advanced C Topics Seminar." Contact: Plum Hall (see January 7-11).
January 28-February 1 Plum Hall Training, New York, NY: "C Programming Workshop." Contact: Plum Hall (see January 7-11).
January 29-February 1 Integrated Computer Systems Seminar, Baltimore, MD: "Programming in C: A Hands-On Workshop." Contact: ICS (see January 22-25).
January 30 AT&T Technologies, Princeton, NJ: "Fundamentals of the UNIX Operating System for Programmers." Contact: AT&T (see January 7).
January 30-February 1 CAPE Seminar, Honolulu, HI: "A User-Oriented Evaluation Three-Day Seminar: UNIX." Contact: CAPE (see January 9-11).
January 30-February 1 Computer Technology Group, Chicago, IL: "Using Advanced UNIX Commands." Contact: CTG (see January 14-16).
January 31 AT&T Technologies, Sunnyvale, CA: "UNIX System V Screen Editor **vi**." Contact: AT&T (see January 7).

FEBRUARY

February 2 AT&T Technologies, Sunnyvale, CA: "UNIX System V Tools." Contact: AT&T (see January 7).
February 4 AT&T Technologies, Princeton, NJ: "Software Development Under the UNIX Operating System." Contact: AT&T (see January 7).
February 4 AT&T Technologies, Sunnyvale, CA: "Shell Command Language for Programmers." Contact: AT&T (see January 7).
February 4 AT&T Technologies, Princeton, NJ: "UNIX System V Device Drivers." Contact: AT&T (see January 7).
February 4 AT&T Technologies, Sunnyvale, CA: "Internal UNIX System Calls and Libraries Using C Programming." Contact: AT&T (see January 7).
February 4 NCR Education Seminars, Dayton, OH: "UNIX Operating System." Contact: NCR (see January 7).
February 4-5 Computer Technology Group, New York, NY & Washington, DC: "Shell Programming." Contact: CTG (see January 14-16).
February 4-5 Intelligent Solution Seminar, Palo Alto, CA: "UNIX Concepts." Contact: Intelligent Solution, 849 22nd Street, Santa Monica, CA 90403. 800/367-0948 or in CA, 213/207-5356.
February 4-8 Bunker Ramo Information Systems, Trumbull, CT: "C Programming." Contact: Bunker Ramo (see January 21-25).
February 4-8 Computer Technology Group, Chicago, IL: "UNIX Internals." Contact: CTG (see January 14-16).
February 6-7 Intelligent Solution Seminar, Palo Alto, CA: "Programming in C." Contact: Intelligent Solution (see February 4-5).
February 6-8 CAPE Seminar, Minneapolis, MN: "A User-Oriented Evaluation Three-Day Seminar: UNIX." Contact: CAPE (see January 9-11).
February 6-8 Computer Technology Group, New York, NY & Washington, DC: "Using Advanced UNIX Commands." Contact: CTG (see January 14-16).
February 6-8 Specialized Systems Consultants, Seattle,

WA: "Hands-On UNIX for Managers." Contact: SSC (see January 16).

February 8 Intelligent Solution Seminar, Palo Alto, CA: "UNIX Overview." Contact: Intelligent Solution (see February 4-5).

February 11 AT&T Technologies, Sunnyvale, CA: "C Language for Programmers." Contact: AT&T (see January 7).

February 11 AT&T Technologies, Lisle, IL: "Fundamentals of the UNIX Operating System for Programmers." Contact: AT&T (see January 7).

February 11 AT&T Technologies, Sunnyvale, CA: "UNIX System V Internals." Contact: AT&T (see January 7).

February 11 AT&T Technologies, Dublin, OH: "UNIX System V Administration." Contact: AT&T (see January 7).

February 11 NCR Education Seminars, Dayton, OH: "UNIX System Administration." Contact: NCR (see January 7).

February 11-15 Bunker Ramo Information Systems, Trumbull, CT: "Advanced C Programming." Contact: Bunker Ramo (see January 21-25).

February 11-15 Computer Technology Group, New York, NY & Washington, DC: "UNIX Internals." Contact: CTG (see January 14-16).

February 11-15 Digital Seminar Program, Bedford, MA: "UNIX and ULTRIX Utilities and Commands." Contact: Digital Educational Services (see January 9-11).

February 11-15 Digital Seminar Program, Landover, MD: "Programming in C." Contact: Digital Educational Services (see January 9-11).

February 11-15 Structured Methods Seminar, New York, NY: "UNIX System Workshop." Contact: Structured Methods (see January 14-18).

February 12-14 Computer Technology Group, Chicago, IL: "UNIX Administration." Contact: CTG (see January 14-16).

February 13 AT&T Technologies, Lisle, IL: "UNIX System V Screen Editor **vi**." Contact: AT&T (see January 7).

February 13 AT&T Technologies, Princeton, NJ: "Overview of the UNIX System." Contact: AT&T (see January 7).

Please send announcements about training or events of interest to: UNIX Review Calendar 520 Waller Street San Francisco, CA 94117.

Electronic mail can be sent to **ucbvax!olympus!its!dave.** *Please include sponsor, date and location of event, address of contact and relevant background information.*

First we made the promise . . .

Then we delivered!

UNIXEXPO

There aren't that many trade shows specifically dealing with UNIX, just enough to be confusing. UNIX EXPO . . . the New York Show . . . is the one that **delivered**. Delivered the right audience in the right numbers. Numbers so right that many of last year's exhibitors have already **doubled** the size of their exhibit space for the upcoming show.

UNIX EXPO will continue to deliver. We have a new location in the heart of New York City's business district, an expanded conference program, a proven track record, more exhibit space . . . and the trade show savvy to produce an audience that comes to talk business.

If you sell to any segment of the UNIX environment, make certain that you exhibit in the show with the right stuff.

Call Bob Birkfeld or Don Berey today for the complete story. You won't be disappointed. That's a promise!

September 18, 19, 20, 1985 • New York, N.Y.

T.M.
UNIXEXPO

THE UNIX OPERATING SYSTEM EXPOSITION & CONFERENCE

Produced & Managed by National Expositions Co., Inc.,
14 West 40th Street • New York, NY 10018 • Telephone: 212/391-9111 • Telex: 135401 DIMCOMM

UNIXtm is a registered trademark of Bell Labs   UNIX EXPO is not affiliated with Bell Labs

Circle No. 85 on Inquiry Card

ADVERTISER'S INDEX

COMING UP IN FEBRUARY

- Portability Defined
- How UNIX Portability Evolved
- Marketing Angles
- C Portability
- Moving Applications: Portability Put to the Test
- Legal Hurdles

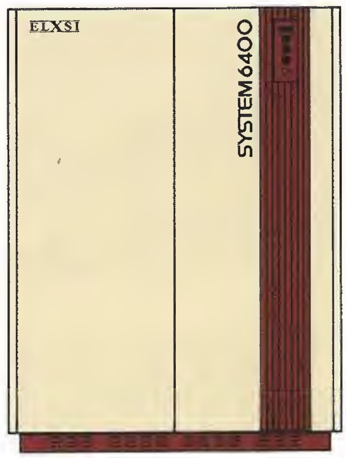
BACK ISSUES AVAILABLE

- June/July 1983 .... ☐
- August/September 1983 .... ☐
- October/November 1983 .... ☐
- December/January 1984 .... ☐
- February/March 1984 .... ☐
- April/May 1984 .... ☐
- June 1984 .... ☐
- July 1984 .... ☐
- August 1984 .... ☐
- September 1984 .... ☐
- October 1984 .... ☐
- November 1984 .... ☐
- December 1984 .... ☐

All back issues are $4.95, including postage and handling. Enclose payment or credit card information or call 206/271-9605.

Name ____________________
Company ____________________
Address ____________________
City __________ State __________ Zip __________
M/C or VISA ____________________
Exp. Date ____________________

UNIX MULTIPROCESSOR PERFORMANCE AND EXPANDABILITY IN ONE BOX



NOW

ELXSI System 6400 has a true native port of UNIX System V, not an emulation. With a unique combination of Berkeley enhancements. Virtual memory. Super fast I/O. Ethernet with TCP/IP. Plus the largest physical memory in the business. ELXSI System 6400 will run either UNIX or our proprietary, message-based EMBOS operating system. Or both, concurrently.

ELXSI System 6400: The first true multiprocessor UNIX system. Up to 5 CPU's operating at 20 MIPS in a single cabinet, through the ELXSI Gigabus—our 320 mByte/second, 64-bit wide system bus that can accommodate further expansion up to 10 CPU's and 40 MIPS of computing power. More power for more system users, expandable as your needs grow, with the highest power to footprint ratio in the business.

ELXSI System 6400. No one can match our performance or our expandability, deliverable now. Contact ELXSI today for complete information. Sales offices in most major cities.

THE ELXSI GUARANTEE: find a computer system that can match our performance and our potential for expansion, and we'll raise our price to match theirs.

ELXSI

ELXSI, 2334 Lundy Place, San Jose, California 95131.
408/942-1111, Telex 172-320.

Circle No. 86 on Inquiry Card

See us at UniForum Booth # 3036

UNIX is a trademark of Bell Laboratories. Ethernet is a trademark of Xerox Corporation. Gigabus and EMBOS are trademarks of ELXSI.

UNIX™ HORSEPOWER!

There are a lot of UNIX based systems on the market today claiming to be "SUPERMICROS". But do they really have what it takes to run multi-user UNIX well? The IBC ENSIGN™ does and here's why:

FAST MEMORY: No computer running at **any** clock speed can run faster than it's overall memory design. The ENSIGN has up to 8MB of 120nsec memory with dual bit error correction. With IBC's proprietary memory management, **all** of this memory runs with no wait states as fast as the 68000 CPU will go. Compare this to other systems running only small cache memories at full speed. Other multiple user systems cannot load all their programs into a small cache memory. Their systems slow down considerably under a heavy multi-user load.

INTELLIGENT SERIAL I/O CONTROLLER: Even the fastest CPU will slow down when it's trying to handle interruptions from multiple on-line users. The ENSIGN provides slave serial I/O CPU's **and** FIFO buffering for both input and output. The result is the ENSIGN's ability to support up to 32 users, with heavy serial I/O demand, while leaving the main 68000 CPU free to run with little serial I/O overhead.

INTELLIGENT DISK CONTROLLER AND HIGH PERFORMANCE DISK DRIVES: The ENSIGN has a slave CPU to handle all disk operations, **plus** 16K of disk buffering. IBC's proprietary disk DMA allows high speed data transfer to main memory without slowing down the main CPU. Further, the ENSIGN supports SMD type 8" hard disks with much faster seek times and transfer rates than 5¼" hard disks usually found in personal desk top computers.

THE RESULTS: The IBC ENSIGN runs multi-user UNIX at performance levels not attainable by other supermicros.

Call IBC and get a copy of IBC's multi-user benchmarks—benchmarks that test 8 users running large CPU programs, with heavy disk I/O **and** heavy serial I/O simultaneously. You'll find that nothing can compare to the ENSIGN.

If you want to run multi-user UNIX on a high performance system with up to 32 users, 8MB memory, and over 1,000MB disk storage, see the IBC ENSIGN.

IBC/Integrated Business Computers
21621 Nordhoff Street
Chatsworth, CA 91311
(818) 882-9007
Telex No. 215349

UNIX is a trademark of Bell Laboratories

Circle No. 87 on Inquiry Card