UNIX™ REVIEW

THE PUBLICATION FOR THE UNIX™ COMMUNITY

October 1984

$3.95

UNIX™ ON BIG IRON

Office Automation Software. . .

Selected for Distribution by AT&T.

HANDLE is the number one office automation or "integrated" software system available for the UNIX™ operating system. Selected by **AT&T** Information Services for distribution on the 3B2/300™ and 3B5™ UNIX systems, **HANDLE** is the most powerful full-featured office automation system in the market today.

Some of **HANDLE**'s unique features include:

- Dynamically integrated word processing, graphics, list management, virtual spreadsheet, and spelling correction with a powerful "document metaphor" that provides a common method for creating, printing, merging, archiving off-line, filing under multiple topics and retrieving documents.
- Full multi-user capabilities with extensive security — documents, or portions of docuents, can be marked "author only", "read only" or "general access". **HANDLE** provides its own record locking.
- User preferences — **HANDLE** conforms to each individual's preferences for such things as preferred printers and individual document catalogs. User restrictions can control such things as the ability to delete documents, access archive catalogs, etc.

The AT&T 3B2/300 computer with HANDLE on screen and integrated graphics printouts.

WORD PROCESSING
|
LIST MANAGEMENT — DATA BASE — VIRTUAL SPREADSHEET
|
BUSINESS GRAPHICS

The first Office Automation System for UNIX.

__HANDLE CORP,__ 140 MACKINAW ROAD, PO BOX 7018, TAHOE CITY, CA 95730 • 916-583-7283

Circle No. 1 on Inquiry Card

YOU CAN'T PREDICT THE FUTURE.

BUT YOU CAN BE PREPARED FOR IT.

Now OEMs and systems integrators can sleep better at night. Because one company has taken the worry out of buying the right software.

RDS.

The company that produces a family of database software designed to take on the future.

Incompatibility is a thing of the past.

INFORMIX® and File-it!™ are compatible with UNIX,™ MS™-DOS, PC-DOS,™ and PC/IX systems (over 60 micros and minis* at last count).

INFORMIX is a true relational database system designed to take full advantage of the power of UNIX. It includes the most widely used report writer on the market.

Then there's File-it! The first easy-to-use UNIX file manager. Together, they have the flexibility to accommodate novices and experts alike.

INFORMIX and File-it! are fully integrated. Users can upgrade from File-it! to INFORMIX or access data from one program or the other without re-entering data, retraining employees or reprogramming.

Applications can also be moved from MS-DOS to UNIX and vice versa without having to rewrite the application.

Simplify program development.

RDS offers C-ISAM,™ the de facto standard ISAM for UNIX. It's a library of C subroutines with a B^+-Tree based access method that stores, retrieves and modifies data from indexed files. It's embedded in INFORMIX and File-it! Or is available as a standalone product.

Software good enough for AT&T.

AT&T, inventor of UNIX, has co-labeled INFORMIX, File-it! and C-ISAM to run on their full AT&T 3B Computer line (from micros to minis).

Hewlett-Packard, Altos, Zilog, Siemens, Cromemco, Perkin-Elmer, Sydis and General Automation have selected RDS as well.

In fact, INFORMIX has an installed base of over 6,000 copies. And RDS has sold over 35,000 licenses for all their products to date.

But before you make up your mind, check the facts one more time.

There's only one database software family that's UNIX-, PC-DOS-, MS-DOS- and PC/IX-based. It runs on more than 60 systems. And it's ideal for both novice and expert.

Now it doesn't matter where the future's headed. You're already there.

*RDS products are available for the following systems:

Altos 586, 986, 8600, 68000
Apollo DN300
AT&T 3B2, 3B5, 3B20, AT&T Personal Computer
BBN C machine (all models)
Bunker Ramo Aladdin 20
Charles River Data Systems Universe 68
Convergent Technologies Miniframe and Megaframe
Corvus Systems Uniplex
Cromemco System 1
DEC 11/23, 11/34, 11/44, 11/60, 11/70, VAX 11/730, 11/750, 11/780
Dual Systems System 83
ERG Mini System (all models)
Fortune 32:16
Forward Technology 320
General Automation Zebra (all models)
Hewlett-Packard 9000 Series 200, 9000 Series 500
IBM PC, PC-XT
Intel System 86/330
Masscomp NC 500
Momentum Hawk 32
NCR Tower
Onyx C8002, C8002A
Perkin-Elmer 3210, 3220, 3240, 3250
Pixel 100/AP, 80 Supermicro
Plexus P/25, P/35, P/40, P/60
Pyramid Technologies 90x
Radio Shack Model 16
SCI Systems IN/ix
Wicat System 150WS, 160, 200
Zentec 2020
Zilog System 8000 (all models).

Demos of INFORMIX and File-it! are available. Demonstration software and complete manuals included.

RDS™
RELATIONAL DATABASE SYSTEMS, INC.

2471 East Bayshore Road, Suite 600, Palo Alto, California 94303
(415) 424-1300 TELEX 467687

See us at UNIX EXPO, October 16-18, New York, Booth No. 123

Circle No. 2 on Inquiry Card

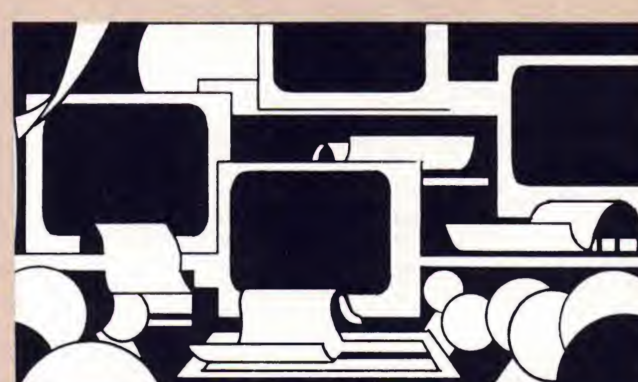
UNIX™ REVIEW
THE PUBLICATION FOR THE UNIX COMMUNITY

Volume 2,
Number 7
October 1984

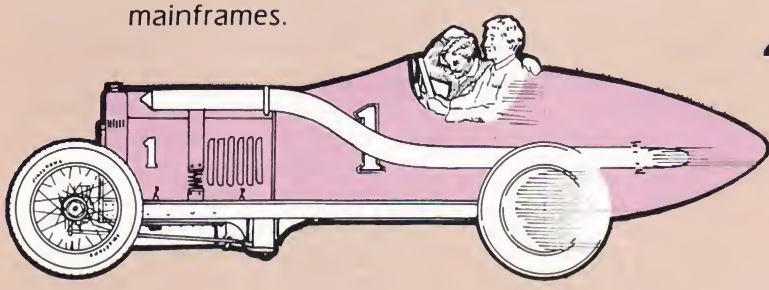
DEPARTMENTS:

Cover Art by Heda Majlessi

FEATURES:



UNIX™ ON BIG IRON



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 520 Waller Street, San Francisco, CA 94117. 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.

Editorial telephone: 415/621-6415. Subscription/Dealer Sales telephone: 206/271-9605.

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, S.T. Wilson
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.
Jim Joyce, President, International Technical Seminars, Inc.
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
David Fiedler, President, InfoPro Systems
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
CONTRIBUTING EDITOR:
Ned Peirce, Systems Analyst, Bell Laboratories

VIEWPOINT

UNIX grows up

Bigger is better. Isn't that a tenet of modern life?

Perhaps, but history is certainly rich with examples to the contrary. If you don't believe it, turn the page and read Stan Kelly-Bootle's remembrances of the big, old Cambridge EDSAC.

Bigger is not always better — but it can be. This issue of *UNIX REVIEW* assesses the migration of UNIX to mainframes. After reading it, you can make your own decision about whether the movement is a thing to celebrate or mourn.

There is anything but unanimity on this point. Numerous are the advocates of small UNIX who ask whether the operating system can survive a ramping up by a factor of 10 to 100. Are big systems really that much more effective? Are they even *as* effective as an equal amount of computing power distributed over a network of mini systems?

There is a strong undercurrent in the UNIX community resisting the mainframe onslaught. UNIX, after all, was created on a VAX environment for a VAX environment by people who liked the VAX environment.

Riding the mainframe wave, though, are others who say the move to big machines legitimizes the operating system. No longer is it simply a toy limited to hackers working in academic and research institutions.

The truth, of course, lies somewhere between these two extreme views. To get at the real story, *UNIX REVIEW* sought out acknowledged experts in the field. Ian Johnstone, one the architects of UNIX for the IBM 370 at AT&T Bell Laboratories, explores the question: "Why UNIX on Big Iron?" in an article written in conjunction with Steve Rosenthal. Neal Macklin, a senior systems engineer with Amdahl, follows up that examination with a queuing theory argument for moving up to mainframes. Schemes for front-ending big machines and ensuring compatibility are presented in another piece by Ned Peirce, a systems analyst working as a consultant at AT&T Bell Laboratories.

The marketing picture for UNIX on Big Iron is clarified by August Mohr, the former editor of /usr/group's CommUNIXations newsletter. Attached to that is an eye-opening sidebar authored by Gene Dronek, architect of the Aim Benchmark suite, exploring the performance of a couple of new, large machines.

We conclude the focus on Big Iron with our second installment of interviews with UNIX security experts. This month, Dick Karpinski, manager of UNIX services at UC San Francisco, speaks with George Goble of Purdue University and Bob Chancer of AT&T Bell Laboratories — two people with security responsibility for some very large systems.

That's a lot of material, but don't expect the debate to be resolved in these pages. We just keep the waters churning.

Mark Compton

Gould . . . Innovation and Quality in Software

CELEBRATE!

New UNIX™-based software tools bring projects in on time, within budget.

Get your hands on our UNIX tool kit. Take out the Gould Software Division 4.2 UNIX with System V enhancements. You'll know it. It has a fast file system, automated system configuration, interactive debugging, and a Source Code Control System that compresses files for more efficient disk storage.

Then grab The Newcastle Connection, an ingenious tool that lets you create a distributed UNIX file system in a LAN by uniting heterogeneous UNIX versions and hardware environments.

Next, try the Unicellerator™, a high efficiency, user transparent tool that dramatically increases productivity under UNIX by offloading CPU intensive jobs from the host. The Unicellerator is available to *all* users of a system simultaneously.

The kit also contains a Cross Development System for developing and interactively debugging C language programs under UNIX that will run on Motorola® and Intel® microprocessors. The heart of the system is a C language compiler that generates compact, efficient code that reduces code space by 20–40% and increases performance by an equal amount. For programmers who want to develop Macintosh™ applications in C under UNIX, our Macintosh Cross Development System will do it four times faster than Apple®'s Lisa™-based environment.

Our UNIX tools are available individually or as a complete, well-integrated set. The sooner you get your hands on them, the faster you'll finish your development projects. And that's worth a toast any time. Contact Gould Software Division, 1101 East University Avenue, Urbana, Illinois, 61801. (800) 952-8888 or (217) 384-8500.

UNIX is a trademark of Bell Laboratories
Apple is a registered trademark of Apple Computer, Inc.
Motorola is a registered trademark of Motorola, Inc.
Intel is a registered trademark of Intel Corporation
Lisa is a trademark of Apple Computer, Inc.
Macintosh is a trademark licensed to Apple Computer, Inc.

GOULD
Electronics

DEVIL'S ADVOCATE

Mainframe days of yore

by Stan Kelly-Bootle

In my native Scouse dialect, as bespoke in Liverpool, England, a figure of speech known as the Malapudlianism intentionally confuses similar words. (See "Lern Yerself Scouse — How to Talk Proper in Liverpool," F. Spiegl, F. Shaw, S. Kelly-Bootle; Scouse Press, 1966.) In Sheridan's play, "The Rivals," Mrs. Malaprop is presumed to err by way of blissful ignorance, whereas the Scouser seeks risible effect by carefully selecting the near-miss. All of which serves as due warning to reader, editor and typesetter alike that when I indulge in moribund mainframe neuralgia, my words will be chosen with matriculate acidity. The **dispel** feature of my Comic's Workbench works good, ja? I cannot, alas, say the same for the *UNIX REVIEW* **spelchek**, which transformed an irreproachable *supersede* from my August contribution into a shocking *supercede*. A useful aide-memoire is: "Supermicro project funding needs super seeding."

My neuralgia was triggered anew while watching a Wang TV commercial (or one of that ilk) wherein well-groomed persons glide through heavenly offices, past pastel workstations, pausing only to run off rainbow pie charts for beaming Sales Managers (played by Eddie Albert Sr.). Running off gawdy pie charts is the prime application of TV computers, requiring but a casual

flick of the spacebar. I assume that the screen must normally display the message, "Hit any key to print a pie chart."

I don't know about you, but the "hit any key" prompt freezes me into prolonged abulia. I sit there for hours fearing some satanic trap. Do they *really* mean *any* key? So much freedom! I blink in the light and sing the final chorus from "Fidelio."

These TV pie charts are no mere spidery, polygonal applications, but elaborate hyper-dimensional Lucasfilm Productions that are Charlotte-moulded, glazed, fluted, rotatable feasts with detachable slices. They say that a picture is worth a thousand words. I estimate that these Carême confections eat up a megabyte at least. This wrinkled old mainframer still finds it hard to accept that behind those dainty panels serenely floats 80 MB of Winchester and 2 MB of Piiceon.

Today's neobyte hackers, pampered with menu-driven mice and mice-driven menus, can have little notion of how *tough* we had it in the early pioneering days — or nights, as was more common. Computing on the Cambridge EDSAC (See Figure 1) was not unlike stoking a coal-powered cargo steamship of dubious registration. Stripped to the waist (the men included), choking in the fumes from the mercury delay-lines, sweltering in the heat from 3000 thermionic valves, whiplashed by rum-crazed supervisors, we staggered to and fro chanting:

"Lift that Reel, Tow that Deck,
"Git a little weary and they'll
"Break your neck!"

And things got worse when the shift actually started.

Programs were programs then; *software* and such nonsense came much later. Our routines were heroic deeds etched in blood and binary on five-channel paper tape — and you supplied your own ADD subroutine! Output, on a fairish day, was at a dizzy 7 1/2 cps on a teleprinter. The good news was that a word had 36 bits (eat your heart out, Motorola!); the incredibly bad news was that we had only 512 of the little darlings. I'm talking about a *total* memory of 512 words. No drums, no backing store, no mag tape, no ROM lurking under *our* keytops, no sir.

Cambridge University EDSAC Specifications in May, 1949	
Chief, Design Team	Maurice V. Wilkes, F.R.S.
Word length, bits	36
Instruction length	18
Instruction format	1-address
Instruction set	18 ops
Storage size, words	512
Storage type	mercury delay-line (32 tanks each 16 words)
Add time (average)	1.4ms
Multiply time (average)	5.4 ms
Input medium	paper-tape reader 5ch
Output medium	paper tape punch 5ch Teleprinter
Digit period	2 usecs
Main valve type	EF 54
Approx number of valves	3000
First program run	6 May 1949
Shutdown	July 1958

Figure 1 — *Cambridge University EDSAC Specifications (as of May,1949).*
(Adapted from "Early British Computers" Simon Lavington, Digital Press, 1980)

Yet we were happier in those days (fade in violin obligato).

At the time I honestly do not recall thinking of 512 words in terms of "small" or "limited" memory. Such things, of course, are relative, and my only base for comparison was my double-barreled, hand-cranked Brunsviga calculator with no words at all (my children taught me the meaning of volatility by playing with the Brunsviga's levers at critical moments).

Furthermore, EDSAC words were undoubtedly *physically* imposing, residing in 32 tanks of mercury, each five feet long. Such storage certainly discouraged sloppy coding, and — on a serious note — we all owe an enormous debt to Wilkes, Wheeler and Gill for developing the techniques for building libraries of efficient subroutines with such frugal resources. When we did occasionally exhaust our 512 words, we resorted to a primitive form of "virtual memory" by punching out intermediate results on paper-tape and looping them back in the reader. I gather that similar manuevers are forced upon some Macintosh users today while they wait for their 512K chips to arrive. Is there not a Law stating that code expands to fill all available RAM?

Many of the current computer scientific preoccupations were, Dieu soit loué, absent from our daily chores. Portability, for instance, was seldom discussed. Not only was EDSAC enormously heavy and bulky, but airplane seats were much smaller then. Running *our* programs on *our* machine was challenge enough, leaving little time to ponder whether they might work on some *other* unmentionably Mancunian machine 200 miles to the north. Piracy, I believe, never raised its ugly head — mainly because the accurate copying of paper-tape proved to be a task of daunting complexity.

The evil that hides in the hearts of all genders did, however, find gratification in other directions. EDSAC had a single slow batch access, calling for devious plots to abort jobs that were running where your job ought to be. One canny ploy was to flick the paper-tape reader tension-arm while yelling "I'm next, I'm next."

There are some true myths and some mythical myths stemming from that period. It *was* widely believed that three or four EDSACs could cope with all of the UK's scientific computing needs: a typical example of failing to see that inventions generate applications rather than vice versa. This misconception was modified later, in the satirical environment, when it was suggested that the origin of the name "National Cash Register" laid in the general consensus of 1890 that *one* cash register per nation would suffice to

meet all retailing requirements.

Looking back over my 30 years of diverse computrefaction, the most surprising facet of the whole sordid mess is the apparent revocation, chiefly by the miracles of MOS technology, of Grosch's law ("The Devil's DP Dictionary" Stan Kelly-Bootle, p.60; McGraw-Hill, 1981). Herbert R. J. Grosch (or Hoib the Grosch, as he was known in the Bronx), a legendary guru of early computing and the only man ever to be fired twice by IBM and twice by Univac (he once told me this himself), convinced us all that computing power is proportional to the square of system price. It seemed reassuring to us that irrefutable economic forces would lead to bigger and better jobs for the mainframe priesthood.

Yet, today, I receive a duplicated letter from Byte Magazine which opens: "Dear Computer Enthusiast: You're enthused about small computers, as well you ought to be!" This

> **Our routines were heroic deeds etched in blood and binary on five-channel paper-tape.**

reminds me that I'm no longer sure of what "small" means anymore. The MC68000 in my AlphaMicro costs less than one of the 3000 thermionic valves that triggered the old EDSAC. Moreover, with the mercury in each tank of that old machine I could now buy the whole Wang TV soundstage, including the heavenly clouds. HOLD THE PIE!

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 68000 Primer" for the Waite Group, to be published by Howard W. Sams in the Spring of 1985. ■

THE HUMAN FACTOR

Both sides now: BIG UNIX and little unix

by Richard Morin

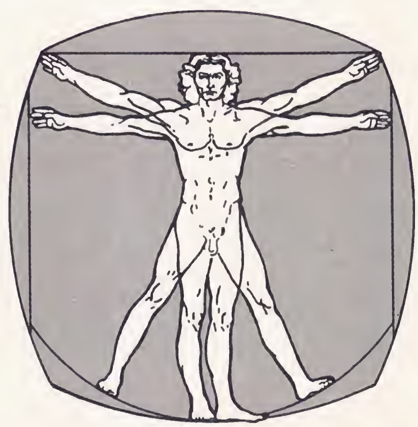
A battle rages on around us. It is sometimes seen as a battle between Bell Labs and UC Berkeley, but its roots extend far beyond those two institutions. On the Bell Labs side we find Stuart Feldman, Ken Thompson, Rob Pike and a host of other savants. These folks want UNIX to be small, elegant and efficient. For lack of a better term, let's call this the "little unix" position.

On the other side we find scores of nameless software developers, each intent on "improving" UNIX by adding a few more commands or features. Let's call this the "BIG UNIX" position.

little unix

The little unix folks don't like the style of UNIX practiced at Berkeley. Ken Thompson mutters about how he used to be able to read a copy of UNIX at home over the weekend. Rob Pike, in his talk, "UNIX Style, or cat -v Considered Harmful" (offered at the Toronto USENIX conference, Summer 1983), gave a number of examples of the "cancerous growth" and "creeping featurism" afflicting UCB's version of UNIX. Quoting from his abstract:

> ***cat** isn't for printing files with line numbers, it isn't for compressing multiple blank lines, it's not for looking at non-printing ASCII characters, it's for concatenating files.*



Stuart Feldman's keynote address at the Salt Lake City USENIX conference (Summer 1984) traversed "An Architecture History of the UNIX System" by tracing the development of a number of architectural styles and comparing them to various versions of UNIX. Version 6 was compared to High Gothic, while 4.2 BSD was likened to Rococo:

> *...there seems to be a tendency to start with a cautious design. After some experimentation, a basic design philosophy dominates, and there are buildings that make elegantly simple use of that approach to make an aesthetic statement. As designers become more comfortable with the techniques, they then make daring and ill-conceived extensions which are dominated by eccentricity and ornamentation.*

The little unix position is steeped in tradition. The first point on style in Ritchie and Thompson's article "The UNIX Time-Sharing System" (Bell System Technical Journal, July-August, 1978) lays out the ground rules:

> *Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new "features."*

This distaste for new features also includes a more generalized lack of appreciation for the hundreds of new commands contributed by the BIG UNIX crowd. Fundamentally, the little unix position is an attempt to convince the UNIX community to return to elegance and simplicity.

BIG UNIX

The opposing position is less often articulated, but this might be expected. BIG UNIX advocates are free to produce and use whatever features they desire without needing to convince anyone of anything. So what if 4.2 BSD only fits on machines with

virtual memory and piles of disk storage? The added features are useful and computer resources are cheap. (One can only wonder what features will be available on 8.4 BSD, designed to run only on networked CRAY IVs.)

A justification for the BIG UNIX position is nevertheless available, even though it comes by way of a little unix source. The preface to *The UNIX Programming Environment* (Kernighan & Pike) contains this statement:

> *...the power of a system comes more from the relationships among programs than from the programs themselves. Many UNIX programs do quite trivial tasks in isolation, but, combined with other programs, become general and useful tools.*

A BIG UNIX aficionado couldn't have said it better. The power of UNIX comes from the ability to combine tools in useful ways. The more tools we have, the more combinations we can make. If a feature "fits" in an old tool, why not put it there? The name of a tool is just a mnemonic device to the user, and if it is easier to remember a flag on an existing program name than a whole new name, then so be it.

AT THE FRONT

Adding to the intensity of the debate, of course, are the assorted efforts currently being made at standardization. Polarization is occurring rapidly. AT&T's current advertising blitz is ubiquitous — some might even say ominous. Meanwhile, BIG UNIX advocates are muttering, "From now on, consider it sub-standard" and wearing buttons reading "4.2 > V". This is most unfortunate.

System V, which AT&T is vigorously promoting, is finding a great deal of support among commercial users and vendors. Meanwhile, 4.2 BSD is finding wide acceptance within research and university crowds, and among the vendors that supply those institutions. The 4.2/System V debate should be kept distinct from the BIG/little UNIX debate. BSD is, of course, huge, but System V isn't so little itself. Neither one will exist in five years, however, and most of us will still be trying to use whatever "UNIX" is then available.

The 4.2/System V debate should be kept distinct from the BIG/little UNIX debate.

The discussion is too important to be left to the slogan hackers, whether from Madison or Telegraph Avenue. Let's see if we can ignore the slogans for a while and proceed.

First, one must recognize that more than one argument is going on. Stuart Feldman's complaints have to do with the number of commands in the system, the size of the kernel and other unjustifiable accretions to UNIX. Rob Pike dislikes the proliferation of features within existing commands. Ken Thompson is upset at the "improvements" his product has suffered. His problem is probably insoluble, though, since UNIX is very unlikely to shrink back to the size of a weekend's reading matter. Even the most fanatical of the little unix crowd would shy away from that kind of surgery.

Pike's complaint has demonstrable merit and addresses a soluble problem. When features are stacked on top of other features, the code of a program becomes more complicated and unwieldy. Convincing studies show that the error rate in code rises non-linearly with respect to average module size. Further, if a command tries to do too many unrelated things, it may lose its ability to do any of them well.

Finally, since UNIX is so receptive to the combining of commands, we can generally build scripts and such to perform special functions. As an example, your author is logged in at all times, and thus needs to check his mail manually. Since he also likes to clear the screen and find out what is happening fairly regularly, he has installed the following alias:

```
alias cll 'clear; from; pwd; ls -aFq'
```

This is very handy and it works just fine as an alias. But submerging it into some command as a set of flags would serve no useful purpose, and would cause needless complication.

Stuart Feldman's complaint, insofar as it differs from Rob Pike's, can be handled very simply. Invisible kernel modifications are the province of kernel hackers, and we mere mortals should ignore them. The hackers should tell each other about their favorite hacks, and the better ones will presumably spread.

Visible kernel modifications are subject to debate and discussion. The kernel interface needs to stay relatively constant, lest we be constantly fixing our applications programs. Since /usr/group is actively working in this area, let's

support its efforts and hope for the best.

Finally, we get to the number of commands on the system, where the solution is the simplest of all. If you don't want to use a tool, you don't have to. Ignore it, throw it away or don't get it in the first place. But let those of us who like it have access to it.

You see, there *will* be an 8.4 BSD or the equivalent. It *will* have scads of commands, and the first section of the manual will no doubt fit comfortably in three binders. With luck, however, some reasonable indexing will have appeared by then.

People will use 8.4 BSD to do any number of silly things, wasting enormous amounts of computer resources in the process. But why not? The UNIX tools are very efficient at letting people use the power of whatever machine they have. Given the proper tools and the right hardware, one can do quite a bit, and productivity, after all, is the bottom line.

UNIX is very unlikely to shrink back to the size of a weekend's reading matter.

ONE SIZE FITS ALL

Regardless of the amount of gingerbread one desires on one's own dwelling, it isn't necessary to demand that all dwellings come so equipped. If a "standard" UNIX is to be developed, it should be as sparse as is consistent with providing needed functionality. Commands which duplicate functions of other commands should be purged. Flags which perform unessential or unrelated functions should also go away.

Before the screaming gets out

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.

PERKIN–ELMER

Circle No. 11 on Inquiry Card

of control, remember that this is only a restriction on "standard" UNIX. It defines the platform upon which an application may safely be built. Users are then encouraged to add features and options to taste. UNIX is, of course, uniquely open to this sort of tailoring.

As the UNIX community continues to grow, new commands will suggest themselves to programmers. Those that pass a rigorous screening should be allowed into the standard set. The rest should still be available to anyone who wishes to acquire them, with the warning that their availability is not guaranteed on standard UNIX.

The world's software libraries are full of routines capable of doing any number of things. Unfortunately, none of this software works together very well, and its portability is (cough) limited. Enter UNIX and the prospect of truly portable and interconnecting utilities.

If you don't want to use a tool, you don't have to.

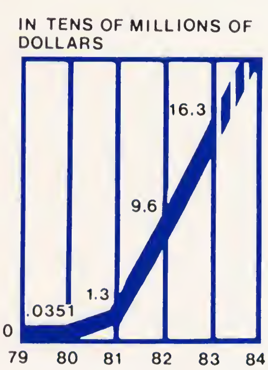
Any number of new tools and retrofitted old tools are going to appear. With luck, most of them will be portable, will work reasonably well and will be built as useful UNIX building blocks. The possible combinations are staggering. I don't know about you, but I can hardly wait.

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 Best of Both Worlds.

If you think that you can't have your cake and eat it too, then you better take a closer look at what Convergent Technologies has to offer. At Convergent Technologies, we offer computer professionals the "Best of Both Worlds." We know that computer professionals want the freedom to turn ideas into products that get quickly shipped out of the door, so generally, our products ship less than 1 year from conception. We also know that computer professionals do not want the uncertainty of seeing development money drying up, projects getting scrapped or the continuing industry shake-out to start affecting them, so that is why we provide qualified people with all of the necessary responsibility and resources to continue to build technically aggressive products. In the 5 years since our founding, we have brought 8 major products to the market and that makes us one of the fastest growing publicly held companies in California . . . but our results speak for themselves. So if you want the "Best of Both Worlds", consider the following opportunities:

Communications — SNA, BSC, LU2.1, 6.2, DIA/DCA **Intelligent Terminal Software** — Intel 80186 or 8088, graphics, communications; bit map display **Compiler Writers, Toolsmiths** **Operating Systems/Utilities (UNIX or CTOS)** **Networking** — TCP/IP; XNS; distributed file systems **Human Interface** — (graphics; system administration) **Test Automation** **Performance Testing**



Most available positions call for a minimum of 2 years of software development experience and a mini/micro background. BS/CS or MS/CS preferred. In addition to a competitive compensation package, we offer significant project and equity participation in a fast product turnaround environment, plus a 13 week sabbatical after just 5 years. Call Joy Mar at (408) 945-6791, or send your resumé to Convergent Technologies, Dept.ur1, 30 East Plumeria, San Jose 95134. We are an Equal Opportunity Employer. Prinicipals Only, Please.

Convergent Technologies

UNIX ON BIG IRON

The whys and wherefores

by Ian Johnstone and Steve Rosenthal

If your application requires a large system and you are worried about having to give up your favorite operating system, take heart. The UNIX system turns out to be robust and flexible enough to implement on large mainframe computers, and ports to such machines have proven to be feasible. The result is a system uniquely suited to large development projects.

Although UNIX was originally designed for machines that would today be considered small in computing power and storage, the operating system continues to spread to larger systems as experienced UNIX alumni go on to positions in research, and university and commercial computer centers. Like any other software system, UNIX has its detractors as well as its proponents. But due to a strong grassroots movement, it is becoming a *de facto* standard, with a large body of trained users. The question seems now not to be

Illustration by Daniel Hanson

whether UNIX will be implemented on most big machines, but *how soon* and *by whom*.

The advantage of running UNIX on a large mainframe as opposed to a configuration of several VAX-class machines lies principally in the faster processing speed, larger main memory and bigger I/O capacity of large machines. In a networked configuration, overhead consumes a large part of the machine's processing power and an equally high proportion of user time.

The exact point at which a large machine becomes more cost-effective depends on the application. A big mainframe requires less floor space, operator time, hardware servicing and air conditioning tonnage than an equivalent configuration of smaller machines (measured in terms of computation power). The mainframe will also use less resources, meaning that fewer storage, backup and output devices need to be purchased.

One of the projects that prompted the UNIX port to the IBM 370 at AT&T Bell Laboratories involved the development of software for a telephone switch consisting of hundreds of thousands of lines of code. The job clearly demanded mainframe power.

A large software system such as that developed for the No.5 ESS telephone switch consists of more than one module built in parallel. There are conditions, however, where it is more appropriate to work serially and synchronously. Under these circumstances, a single fast CPU would be the better choice since it allows the inherent multiprocessing capability of UNIX to be used to advantage.

Another impetus for bringing UNIX over to large machines is that UNIX is currently the operating system of choice at many research labs and universities. This is also where most of the work is being done to develop big systems.

Porting UNIX to a large machine is something a true UNIX guru might consider a trivial task — meaning it might take a year or so of development time. This, of course, depends on several assumptions.

First, you must have the right people. The best course is to start with a small team of UNIX experts. This is better than working with a team of engineers that has expertise with the new hardware, though of course this expertise is also helpful. For example, an IBM guru will be more oriented toward the IBM method of accomplishing tasks than toward the equivalent UNIX approach. A good UNIX guru need not be restricted in this way because UNIX isn't limited to specific hardware. This is demonstrated by the variety of hardware to which UNIX has already been ported.

A UNIX guru, in this context, must be someone who also has a good understanding of hardware. This person would need to know about bus structures, major hardware components, memory organization schemes and other hardware-specific concepts.

The machine itself has to be "reasonable" with respect to UNIX functionality. Among its desirable attributes would be a notion of 8-bit characters and a reasonable method of memory management. With the current trend toward larger programs and cheaper memory, a 32-bit address is also almost mandatory. A stack machine is nice but not necessary since you can always emulate a stack. A reasonable I/O interface, providing high disk throughput, and support for asynchronous terminals are two other desirable attributes.

Some machines, of course, are more esoteric than others. It shouldn't come as a surprise that these make for the most difficult ports. During the course of the Bell Labs port to the IBM 370, limitations of the 370 architecture raised their ugly heads on occasion, but they never got in the way to any overwhelming extent.

DOING THE PORT

This discussion of porting UNIX assumes a single CPU system. But if you've got a multiprocessor system, you should be aware that there are other problems and issues to consider. UNIX 370, for example, is a multiprocessor system running on 3033 and 3081 APs. It presented the Bell Labs team with a number of special problems, but, like other multiprocessor systems, it also offered the potential for building a powerful single box system out of less powerful (and cheaper) CPUs, an option ripe with price/performance gains.

Regardless of whether you are working with a single or multiprocessor system, though, work can't proceed until you have the people and the machine in place. Some of the initial tasks you'll then face include: gaining an understanding of the architecture of the target machine; charting an overview of the entire system; mapping the UNIX view of the world onto both the CPU and its various devices; and segmenting the port into areas of specialization (such as process management and memory management) and assigning tasks to appropriate specialists.

One rule of thumb is that it is preferable to develop native ports. This is far better than running UNIX as a task under another operating system. If UNIX is in control of the machine, it can make best use of the hardware and the developers can match UNIX functionality directly to the architecture. Otherwise, UNIX internals might need to be significantly altered to adapt to another operating system's conventions.

On the IBM 370 project, for instance, the operating system had to be accommodated. The focus of the work was to implement UNIX as a task running under TSS (Time-Sharing System). The TSS supervisor handled all the scheduling and the port used a preemptive scheme quite unlike the way UNIX normally works. Under this scheme, there was no guarantee that the kernel could keep a hold on the CPU — even if its priority were raised. As a result, a large amount of UNIX code had to be modified.

Implementing UNIX as a standalone operating system offers a number of advantages. It's easier to map UNIX primitives directly to the hardware than it is to map the primitives of most other operating systems, which typically don't support many of the more unique features of UNIX. One example is forking: there simply aren't many other operating systems that will allow processes to duplicate themselves.

Another advantage of a native port is that it tends to run faster than a port interfaced through a virtual machine provided by another operating system. The UNIX system can optimize the actual attributes of the new hardware instead of simply making use of another operating system's view of the hardware. This reduces the amount of UNIX tuning necessary and allows for the best advantages of the hardware to be realized. It also makes tuning simpler because it allows the porting team to focus exclusively on UNIX and the hardware — without a second operating system to muddy the waters.

A cross environment is necessary to get started, but once a minimum system is established on the target machine, the developers should switch to it and start building experience in the new environment. The speed at which this step is accomplished is highly variable.

If the port starts with hardware that's known to work, an adequate C compiler, a workable loader and a few useful support utilities, the first phase of the porting process might take something on the order of six months. Of course, many times the new machine is only a prototype, ensuring that the port team will encounter numerous surprises along the way. Needless to say, this will cause the port to take longer.

After "getting UNIX up on the new machine," the next task is to tune and refine the system. In this phase, the system is tested in a more rigorous fashion. Sometimes, it becomes evident that some earlier decisions were wrong: perhaps memory management needs to be re-implemented or a driver needs to be rewritten.

The reworking phase is the part that determines the performance and quality of the system. You can spend as much time tuning the system as is available, but

Implementing UNIX as a standalone operating system offers a number of advantages.

a reasonable figure for a project of this sort might be 12 months. This makes for a total of about 18 months for doing a port: six months to bring up the system, and up to 12 to tune and test it sufficiently for public release.

PORTABILITY ISSUES

The ability to make use of existing utilities and applications software has always been a UNIX strong point. By the time UNIX 370 came along, for example, there was already a strong movement within Bell Laboratories to write portable code. With UNIX running on 11/70s, VAXs, 3Bs and an Interdata 832 at the Labs, such zeal is understandable.

C code can be written to be very portable. It can, however, also be written in a very non-portable, machine-specific way that will not allow code to be shared among heterogenous UNIX systems.

Writing code with portability in mind still has to take into account any differences in the hardware. When the IBM 370 port was done, it was necessary to modify the **tty** system, and some programs required modification in order to work with the IBM 370 **tty** environment. Some programs needed to be rewritten because the base registers on the IBM 370 only addressed 4096 bytes, limiting both local variable and stack spaces. Most programs, however, ran without change. C compiler or C language compatibility will not be discussed here since the current effort to create an ANSI C standard should address this.

Suffice it to say, it would be highly desirable if, at the end of a port, it could be guaranteed that everything was done right. Unfortunately, this is fine in theory but difficult in practice. There is no absolute way of telling whether a port is compatible with previous UNIX implementations because, effectively, there are no standards. But one obvious test is to run a set of programs on machines on which UNIX is already running and then run the same set on the machine you're doing a port for — thus allowing you to compare the results. If the results are essentially the same, congratulations: you have no apparent problems. But if the results are different, it's obvious that the two implementations are not equivalent. Bear in mind, though, that complete compatibility cannot be proven.

The amount of compatibility/certification testing that must be done depends in part on the application. Some applications need to be tested down to extreme

WESTERN ELECTRIC • PEACHTREE SOFTWARE INC. • CRAY RESEARCH • UPJOHN • RCA • DYSAN CORP. • WESTINGHOUSE • AMERICAN BELL •
SOFTWARE BANC • JOHNSON & JOHNSON • HOFFMAN LaROCHE • AT&T • TIME LIFE • BALLY MANUFACTURING • APPLE COMPUTER •

AZTEC C ...the most portable and comprehensive C software development system available

Manx Software Systems with over 6,000 licensed users is one of the leading suppliers of "C" compilers and cross compilers for the world's most popular micro computers. The MANX AZTEC C compiler is available as a cross compiler or native compiler for the following environments:

AZTEC C86	**AZTEC C II**	**AZTEC C65**
PC DOS	CP/M-80	APPLE DOS
MS DOS	TRSDOS	COMMODORE 64 (12/83)
CP/M-86		ProDOS (2/84)

The AZTEC C product is a complete development system. In addition to a full v7 "C" compiler, the basic product includes assemblers, linkage editors, development utilities, and full run time libraries. New products to be released in late 1983 and early 1984 include graphics development tools, data base managers, program editors, screen management systems, and other development tools that work in conjunction with the AZTEC C system.

CROSS DEVELOPMENT SYSTEMS

The AZTEC C cross development systems include all of the utilities and library support routines available with the native versions including a cross assembler. The binary image created in the host environment is downloaded and tested in the target environment. MANX has been using its own cross compilers on a daily basis since 1980.

HOSTS

PDP-11	**68000**	**8086/8088**	**8080/8085/Z80**
UNIX	UNIX ports	UNIX ports	CP/M-80
		PC DOS	
		MS DOS	
		CP/M-86	

TARGETS

8086/8088	**8080/8085/Z80**	**6502/65xx**
PC DOS	CP/M-80	APPLE DOS
MS DOS	TRSDOS (12/83)	APPLE ProDOS (2/84)
CP/M-86	LDOS (12/83)	COMMODORE 64
	DOSPLUS (12/83)	

Other host and target environments will be released in early 1984.

NATIVE COMPILERS

The AZTEC C native 8080 compiler was first released in 1981. Since that time it has been acquired by more than 300 colleges and universities, thousands of corporations, small business, and government agencies. The compiler has been ported to the 6502 and 8086. Plans for future ports include the 68000, 16032, and IBM 370. All native versions are source compatible. Source developed in one environment can therefore be transferred to another environment and compiled, linked and executed.

For prices and information call:
***800-221-0440** (outside NJ)*
201-780-4004 (inside NJ)

*Or write to: **MANX SOFTWARE SYSTEMS***
P.O. BOX 55 • SHREWSBURY, NJ 07701

MANX®
software systems

Circle No. 13 on Inquiry Card

U.S. STEEL • XEROX • NORTH STAR COMPUTERS • FORD MOTOR CO. • MICROSOFT • SPERRY UNIVAC • KAYPRO CORP. • BELL LABS • MODULAR COMPUTER • VIDEX • MICROPRO • NATIONAL SEMICONDUCTOR • DRI • SOFT
SOUTHWESTERN BELL • MOTOROLA • CITIBANK • BAUSCH & LOMB • COMPAQ COMPUTER CORP. • BUNKER RAMO • CONTROL DATA CORP. • DEC • TYMAK • IBM • ROCHE BIO-MEDICAL • 3M • BELL & HOWELL • BENDIX •
WARE PUBLISHING • FRANKLIN • TELEVIDEO SYSTEMS INC • PERKIN ELMER • NCR • SIERRA-ON-LINE • NORTON CO. • HEWLETT-PACKARD • U.S. ROBOTICS • TRW • G.T.E. • SO
FLORIDA COMPUTER GRAPHICS • WESTERN UNION • MERRILL LYNCH • LOTUS DEVELOPMENT CORP. • MONOLITHIC SYSTEMS CORP. • HUGHES AIRCRAFT • ASHTON-
FTWARE BANC • JOHNSON & JOHNSON • HOFFMAN LaROCHE • AT&T • TIME LIFE • BALLY MANUFACTURING • APPLE COMPUTER • AMDAHL CORP. • B.O.A.C. SOUTHWESTERN BELL • MOTOROLA • CITIBANK • BAUSCH & LOMB • MIT •
TATE INC. • LOCKHEED MISSILES & SPACE CO. • ZENITH • TIME INC. • MATTEL • WANG LABS • ROCKWELL INTERNATIONAL • EAGLE COMPUTER INC. • CORVUS • PITNEY BOWES • GENERAL ELECTRIC • VISICORP • MIT • STANFORD •

levels. Others allow more latitude in the discrepancy between new systems and existing ones. By and large,the market for systems will determine how much testing and certification is appropriate.

How big a system UNIX can be ported to is not yet clear. There is probably a limit on the extent to which UNIX can be scaled up, but at present there are no machines that approach the limit where UNIX won't be practical.

UNIX has two factors in its favor. First, it is written in a high-level language, which makes it amenable to change in a way that other operating systems haven't proven to be. Secondly, the kernel is largely the same for all machines. Of course, there are differences between the kernels of a micro-X and a Maxi-Y, but overall the basic code is the same. Most of the differences come in the changes that must be made to tune systems. User-level code should be the same, assuming that the ports in question have been done "correctly."

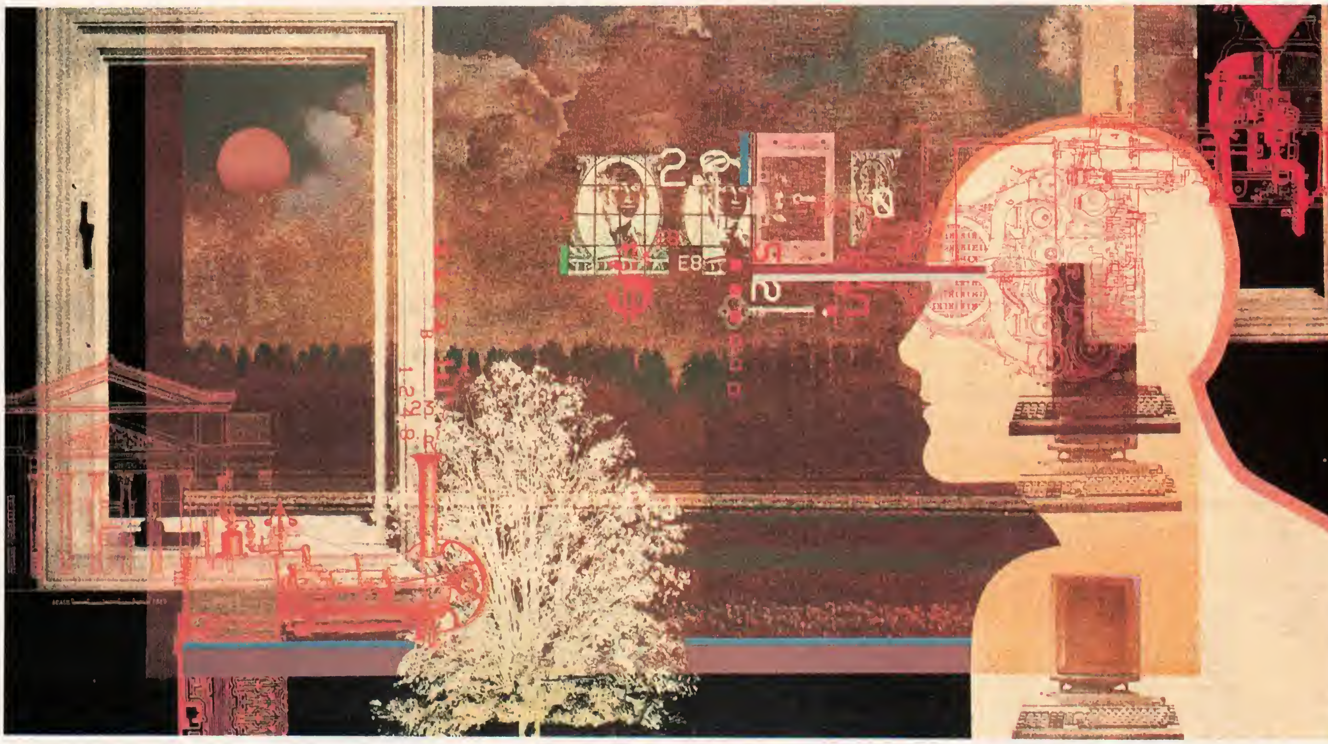
In fact, most of the tweaks needed to expand UNIX from smaller machines to larger ones consist of working around assumptions valid for smaller machines but not true for larger ones. This includes changing the methods for looking at systems tables such as the process table and the number of open file descriptors. These tables, of course, are substantially larger on big systems than on smaller ones.

All in all, though, there seems to be no reason to suspect that UNIX won't be able to keep up with the foreseeable growth of hardware. Considering that today's big system often turns into tomorrow's desktop, that bodes well for all UNIX users — on big and small systems alike.

Ian Johnstone managed the completion of the AT&T Bell Labs port of UNIX to the IBM 370 computer. He also led the group that did the port to the AT&T 3B20A multiprocessor system. Prior to his UNIX work at Bell Labs, Mr. Johnstone spent several years with the Australian Atomic Energy Commission, including 2 1/2 years as a system programmer on large IBM 360 machines. He is now with Sequent Computer Systems in Portland, OR.

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

WHY BUY A DATABASE SYSTEM?



TO BUILD APPLICATIONS!

PROGRESS™ is the high-performance database system and fourth-generation application development language for multi-user microcomputers.

PROGRESS gives you all the flexibility and control of structured programming languages, but uses plain English syntax for simplicity and ease of customization. You and your staff will learn to use PROGRESS quickly and benefit immediately from its state-of-the-art features.

Why build business applications in C, BASIC, or COBOL when PROGRESS will increase your productivity 10 to 40 times? PROGRESS makes this possible by integrating these powerful features:

- English-like application language
- Flexible relational DBMS
- Integrated data dictionary
- Automatic screen and report formatter
- Full-screen syntax-checking editor with on-line help

Once your application is built, PROGRESS offers:

- High performance
- Comprehensive multi-user capabilities
- Automatic database recovery
- Easy modification and customization
- Ad hoc query and report-writing capability
- Portability to a wide range of machines

PROGRESS is available today on major UNIX™-based super-microcomputers.

Power. Performance. Ease of Use . . . Now, that's PROGRESS! You can't afford to build applications without it.

DATA LANGUAGE CORPORATION
5 Andover Road, Billerica, Ma. 01821
617-663-5000

PROGRESS

A NEW APPLICATION ENVIRONMENT

Want to make PROGRESS in a hurry?
Call Area Code 617-663-5000 today.

Or just fill out and return the coupon below.

DATA LANGUAGE CORPORATION
5 Andover Road, Billerica, MA 01821
(617) 663-5000

☐ Yes! Please send me a demonstration disk and documentation for PROGRESS. I enclose $295.

☐ Please send more information about PROGRESS.

☐ I am interested in PROGRESS seminars.

Name__________ Title__________

Company__________

Address__________

City__________ State_____ Zip_____

Telephone (_____) __________

4

Circle No. 67 on Inquiry Card

AIRPORT '84: A PARABLE

A queuing argument for Big UNIX

by Neal Macklin

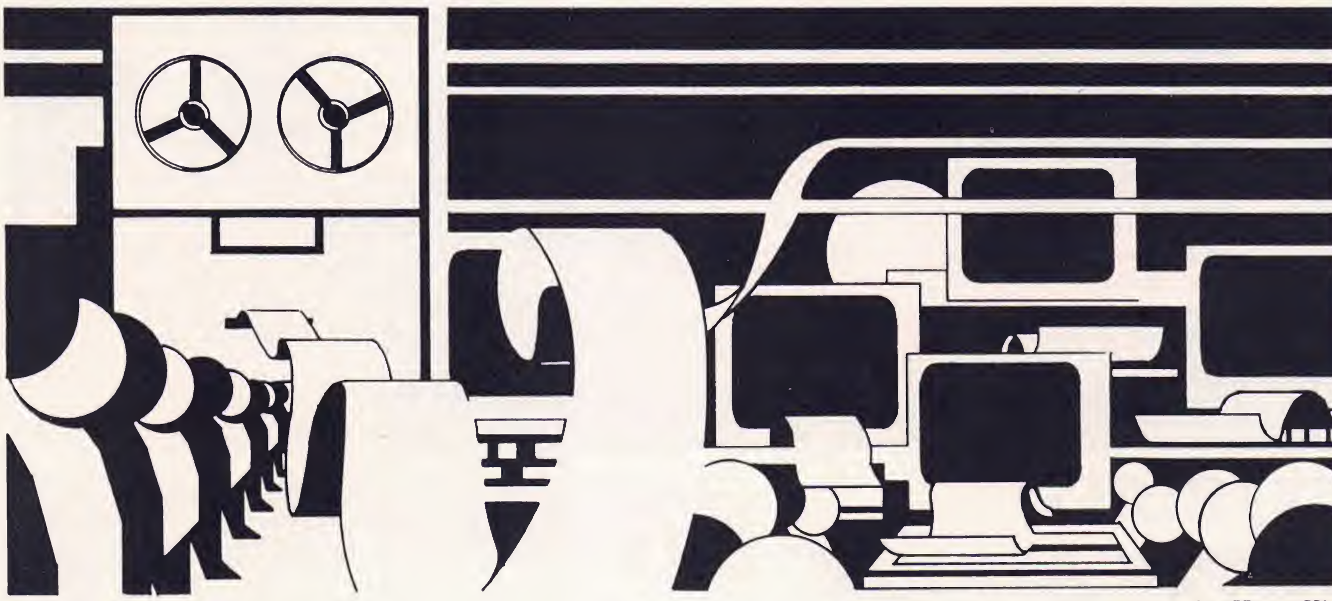


Illustration by Hyon Kim

Screeching to a halt, the taxi driver deposits you at curbside baggage check-in. Unfortunately, though, the skycap won't check your luggage since you changed your flight and don't have a current ticket. Frantically you tear into the terminal, looking for the "Distributed Airlines" ticket counter. Ah...the benefits of deregulation. There's your counter — all the way at the *other* end of the terminal. You take off toward it, ignoring the man with the "Nuke the Whales" badge smiling at you from behind the Three Mile Island poster.

Meanwhile, 1000 miles away, your spouse equivalent is putting the final touches on the dinner that is about to go into the microwave. The timer is set.....one last check on the candles on the dinner table and the artificial log in the fireplace...and it's off to the airport.....

Breathing hard, bags flailing behind you, you do a Tom Cruise style skid-to-a-stop in front of Distributed Airlines, and much to your relief, the lines don't look too bad. All you really need to do is check your bag and get a new sticker on your ticket. You pick the shortest "Baggage Check-in" line. But the "Purchase Tickets" lines look shorter, so you change your mind, throw your bags over your shoulder, and hike over to Position 14 — where there are only four people in line.

Whew! Now you can catch your breath. There's 25 minutes till flight time and only four people to go...four minutes per person. You have plenty of time!

"Outsmarted time again," you think. "Boy, I've got this down to a science!"

You look smugly at the long snaking line behind "Central Airlines" and count your blessings that you didn't have to wait in *that*. Your mind begins to drift to thoughts of the evening..... you've been in a class learning to write non-functional kernel mods for the last week and you can't *wait* to get home to see your loved one.

The thoughts of this evening have taken your mind off the fact that your line still is not moving and you have only 22 minutes till flight time. Anxiously you glance up at the Distributed Airlines monitor. *!Rats! Your flight is boarding! "Oh well — still have plenty of time," you think.

Just then somebody skids up behind you. At least you're not last anymore. You again survey the scene back at Central Airlines and notice that the man in the rumpled suit who shared the cab with you is stuck halfway back in line. "Poor guy," you think, "he'll never make it. I wonder how he managed to cut into line?"

You smile and wave at him from across the terminal. He waves back and motions for you to look at the woman in front of him who is busy applying makeup with a portable mirror. It's funny because every few seconds the line moves up and she is forced to stop what she is doing, pick up her bags, move them forward a few feet, and then drop them and begin the cycle again. You watch this for a few iterations and nod knowingly to your friend. Some people!

Your line *still* hasn't moved and now you're beginning to sweat. Only 19 minutes till flight time! The guy in front of the line seems to be arguing that he doesn't have any room left on his ticket for his "frequent flyer" sticker and refuses to get on the plane without his mileage credit. Why can't they have separate lines for "Jerks" and "Others" at this airline? "I never fail to pick the slowest line," you think. The line to your right has moved up a couple of places, so without thinking clearly you pick up your bags and join the end of that line. The man who was behind you looks relieved and slides his bags forward with his knee, closing up the space you had occupied. There is now someone behind him...

You have just violated one of the unwritten laws of the universe — though you remember it dimly from the class on statistics you took at MIT. The first law is: whichever way you walk in New York City, the wind blows in your face. The second law is: don't ever switch lines.

You wipe the perspiration from your brow and look back at Central's ticket counter. That line is still quite long but your friend and Miss Revlon have now rounded the second bend and are quite close to the front of the line. "That's odd," you think, "both airlines have the same number of agents — it's just that Central has its passengers feeding from one queue and Distributed has individual queues for each small group. In theory that should equalize, shouldn't it?" Too bad you got stuck behind the guy that wanted a super saver to Albania. "But wait, what would happen in that case at Central?"

"Well, one agent would get tied up wading through the Albanian vaccination laws, but since there's only one line, the *next* people in line would not be stuck waiting for the bottlenecked agent to finish. They would just go to whichever agent finished next.

"Yes, but each of the Distributed lines had only four or five people in them," you reason. But, *statistically* in the case of Distributed, you are limited by the speed of your server and it is more probable for an *individual* queue to get bottled up than it is for *all* the agents at Central to be *simultaneously* bottled up.

You can now see your candlelit dinner going up in smoke.

"Hmmm," you think. "Maybe Central is the way to go...?" Too bad your company only flies Distributed.

Your new line is still moving up quite fast, though — two people have been processed. There's still 13 minutes till departure and you're second in line. But wait...*What's this!!?* An officious looking women in a white jacket emblazoned with Distributed's logo is talking with the agent processing tickets for your line. "Probably some union rule against working too fast," you think. *OH NO!!* Your agent reaches up to the sign overhead and turns it so it says "closed."

With eyes wide and mouth agape, you are too stunned to notice the people abandoning the line behind you and scrambling to get positions in other lines. There is some time while the woman in the white blazer attempts to smoothly blend people into other lines during which *no* passengers get ticketed. A few raised voices quickly quiet down as the queues are rearranged. You are left alone in the closed line, watching as the agent walks away to begin a coffee break.

Leaning forward over the counter you shout "Oh Miss!" your voice hoarse with emotion. "I've been on line for 15 minutes and my flight leaves in less than 10!"

"You should have gotten here an hour before your flight," she says and turns away.

You can now see your candlelit dinner going up in smoke. Just about now the microwave is heating up the TV dinner...

"I hate my company. I hate this airline. I...," you say to yourself, half out loud. You notice the ticket agent processing the frequent flyer to Albania motioning for you to move in behind him, and the other passengers in that line move back slightly to make a space for you. The air conditioning system seems to have gone off in the terminal.

By now there are two customs agents and an inspector from the DEA questioning the guy in front of you, so you wave your bag at the agent but she refuses to take note of you and points for you to stand behind the Albanian general.

Beaten, you turn to see Miss Revlon and the Cab Man joking and walking with each other to the gate, boarding passes in hand. The man, noting your predicament, shrugs his shoulders and gives you a sympathetic grin.

Two more minutes on line and the Distributed monitor deletes your flight number from the screen.... "@#$%*!" You were really looking forward to that TV dinner!

Shoulders drooping, you shuffle back in line as the passenger in front of you is taken off for questioning. The next flight is in three hours! You realize you've experienced something about queuing theory that maybe Distributed people should study: it is generally better for tasks to be processed from one long queue by a fast server (the many agents at Central taken together appear to be one fast server) than by several slower servers handling proportionally more queues. With several slower queues the chances of *you* getting stuck behind a big job are greater than the chance that many big jobs will appear simultaneously at a single fast server (consider the likelihood of all of Central's agents being tied up at the same time by complicated ticketing problems.)

The reason for this is that in most physical phenomena (computer job scheduling included) there is a sort of a Gaussian or bell shaped distribution of tasks — some are long, some are short.

If all tasks were exactly the same length, it wouldn't matter whether there was one fast queue or several slower ones — both would take the same time. But in the Gaussian case it is more likely that the average queue time (response time) will be more varied in the several queue case than in the single queue case. You could just as well have ended up in a queue behind only short tasks — which would have been great. It's just that you could also end up behind long tasks leading to catastrophic failure (like missing your flight while your spouse equivalent is headed to the airport).

That is why most smart airlines have one long queue and know from experience almost *exactly* how long the wait is from any one point. That is also why when the line length exceeds a certain point, the airline will send out expediters to query the people at the end of the line in hopes of "moving up" those that would otherwise miss their flights. Overall, there is an economy of scale to managing one long queue and statistically the wait from any point is calculated with much more certainty (less standard deviation). The shorter queue approach is more of a crap shoot — sometimes you win big but other times you don't and the cost for losing (missing your plane) is much greater than the bonus for winning (having the extra time you need to buy a chili Superdog for $3).

In designing computer systems, the same economies of scale often apply. Amdahl Corporation, for example, makes fast mainframe UNIX machines. An Amdahl 5860 processor (with a single queue) is the computing equivalent of 15 to 20 VAX 11/780s (numerous queues). This allows the machine to run as many as 500 users on a single machine — with very even and quick response time. The impact of a big job running concurrent with other tasks is hardly felt, whereas it certainly would affect the 30 to 40 users forced to compete with it if it ran on one of a group of minis. The large machine offers a single file system — everyone is on the same tree and can look at other people's programs without having to **uucp** them around a network. In addition, a user with one HUGE UNIX job (something on the order of the computer animation done for *Star Trek II* — a job that took six weeks to generate on a VAX) can have the combined power of 15 to 20 VAXs running a single instruction stream. This would not be possible in the distributed approach since you can't parse up a single thread.

Indeed, such work as seismic study, cartography, weather forecasting and ray tracing of smooth-shaded CAD images requires the fastest engine possible. Even mundane UNIX maintenance benefits can be realized: the entire kernel can be compiled in eight minutes on a 5860. One big engine takes up less floor space, power, air conditioning, system programming time (only one copy of the operating system) and network time than do several slower ones.

So if you want your UNIX applications to fly, take a page from the book of Central Airlines. A single fast server can more equitably handle either a large number of tasks or a few compute intensive ones with less overhead and more even throughput than can the multi-queue distributed approach. The operating efficiencies of a large, fast UNIX server invariably lead to commensurate gains in consistency of response time and the ability to smoothly respond to load changes.

Neal Macklin joined Amdahl Corporation in 1980 as a Senior Systems Engineer. He has also worked for American Airlines and IBM, and is a graduate of MIT. He currently has business and development responsibility for Amdahl's UNIX-based products. ■

While everyone else has been squawking about data communications, one company has quietly become the industry leader.

Meet BBN Communications. For years we've been quietly solving the communications problems of government agencies and major corporations around the world. In fact, we started when we built the world's first packet-switched computer network back in 1969.

Now BBN Communications designs, manufactures, installs and operates wide-area data communications networks through a vertically integrated offering of products that run on UNIX* operating systems and networking software written in "C." We are currently designing state-of-the-art networks for MCI, WANG, MASTERCARD and other commercial customers, as well as developing a major presence in the Federal Government marketplace with exciting projects like the **Defense Data Network,** a common user data communications network for the Department of Defense.

Due to our continuing rapid growth, we presently have need for exceptionally talented individuals with UNIX* and "C" backgrounds to participate in advanced R&D and exciting commercial applications in our **Cambridge, MA** and **Washington, D.C.** area offices.

OUR CURRENT OPPORTUNITIES INCLUDE:

- **Sr. Systems Engineers**
- **Development Programmers, UNIX*/C**
- **Software Test Engineers**
- **Expert Systems Tools Developers**
- **Network Designers/Analysts**
- **Software Release Engineers**
- **Network Consultants**

*UNIX is a trademark of Bell Laboratories.

BBN Communications complements fully competitive salaries with such benefits as a medical insurance/HMO option, dental coverage, employee stock purchase plan, 3 weeks' paid vacation, an educational assistance program and an exceptional retirement trust program.

Please call or send your resume to Marilyn Tromer,
Manager of Employment, BBN Communications Corporation,
70 Fawcett Street, Cambridge, MA 02238, (617) 497-3164.

An Equal Opportunity Employer

BBN Communications Corporation

A subsidiary of Bolt Beranek and Newman Inc.

Circle No. 15 on Inquiry Card

BIG UNIX FOR

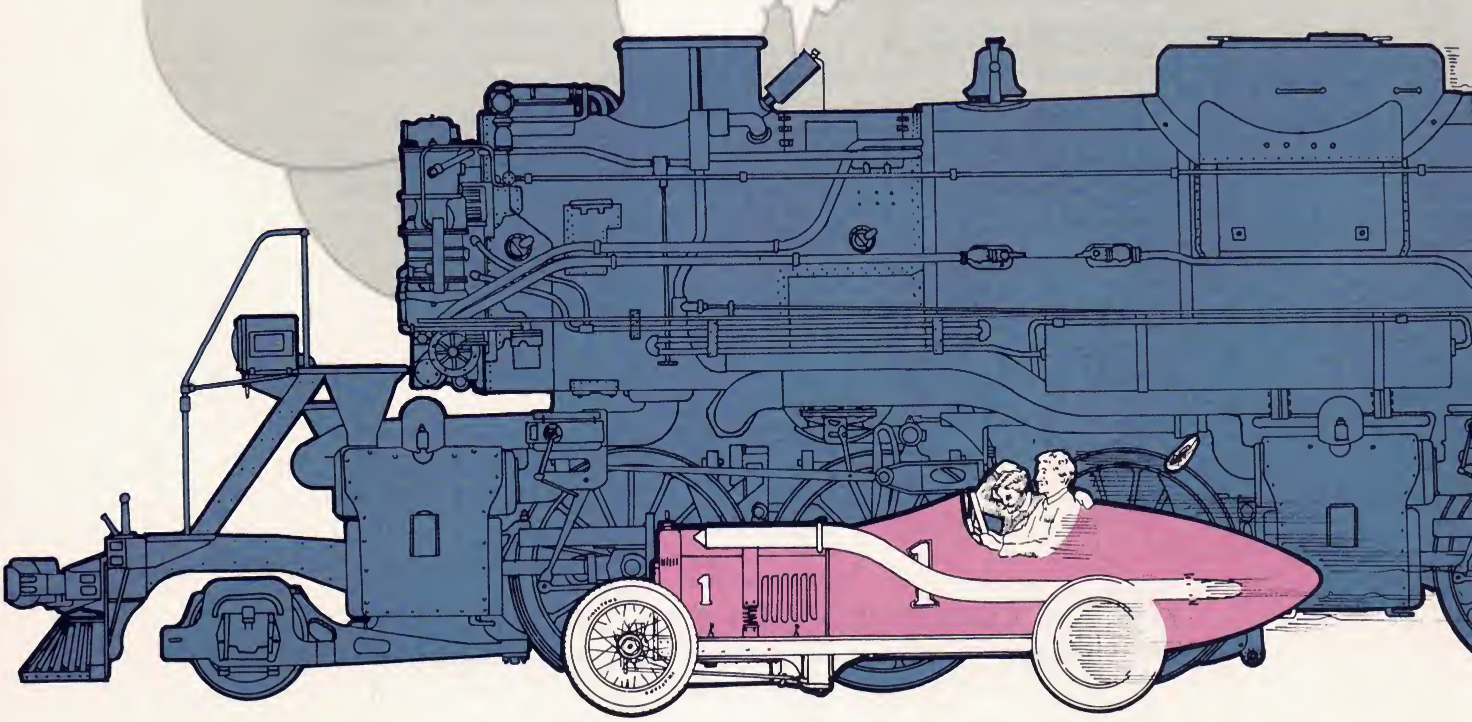
Keeping mainframes a

The migration of UNIX to mainframes indicates that, more than any other operating system, UNIX seems destined to become a true machine-independent work environment. If the system you are logged onto looks and acts like UNIX, chances are software portability concerns will soon be behind you.

Mainframe UNIX typically uses peripheral processors for terminal communication. Front end processors (FEPs) concentrate user sessions on mainframes so that host system resources can be efficiently utilized. The number of FEPs required varies according to the capacity of each individual node.

In the case of UNIX on the IBM 370, the range of FEPs includes IBM Series/I minicomputers, DEC VAXen and AT&T Technologies' 3B series of minicomputers. Upwards of 150 users can, in theory, run simultaneously on an IBM 3033 UNIX 370 system. A single VAX 11/780 or 3B20; two VAX 11/750s or 3B5s; or three or more IBM Series/I minicomputers would suffice as front ends for such a host system. Smaller systems can also be used, but some of the advantages of Big UNIX are lost as FEPs proliferate and networking overhead becomes a larger fraction of the host CPU load.

UNIX ports to new hardware can impose certain constraints. If a system is not easily conversant with asyncronous terminals and 8-bit bytes, it may resemble UNIX but will have apparent differences. This may in fact preclude use of your favorite terminal.



THE END USER

ont ends on the same track

Ned Peirce

A front end processor, though, can eliminate this difficulty by providing the standard UNIX communications interface.

FRONT END PLUSES

There are significant advantages to using a proven UNIX machine as a front end. First, a UNIX-based minicomputer FEP can run a closely related variant of whatever the current release of UNIX is. This provides for painless upgrades as new UNIX releases become available, and guarantees that your Big UNIX system won't get left behind as UNIX itself evolves. Because users interact with the FEP, additional development work has to be done to insure compatibility with existing communications software. But once this is done, the system should appear to users as generic UNIX — because that is in fact precisely what they'll experience.

Second, front-ending mainframes is analogous to adding such "speed up" hardware as attached processors (APs) to minicomputer systems. FEP software can be implemented as user level processes running on a system doing other work and providing service to other users not on the host mainframe. Hybrid front ending can make it unnecessary to dedicate expensive processors as FEPs. In this way, users can also use FEP machines as they would use any regular mini UNIX system. Administratively, the percentage of FEP and regular users can be dynamically changed without either group necessarily being aware of it. Mixing the user load on the FEP can be used to smooth the installation of Big UNIX

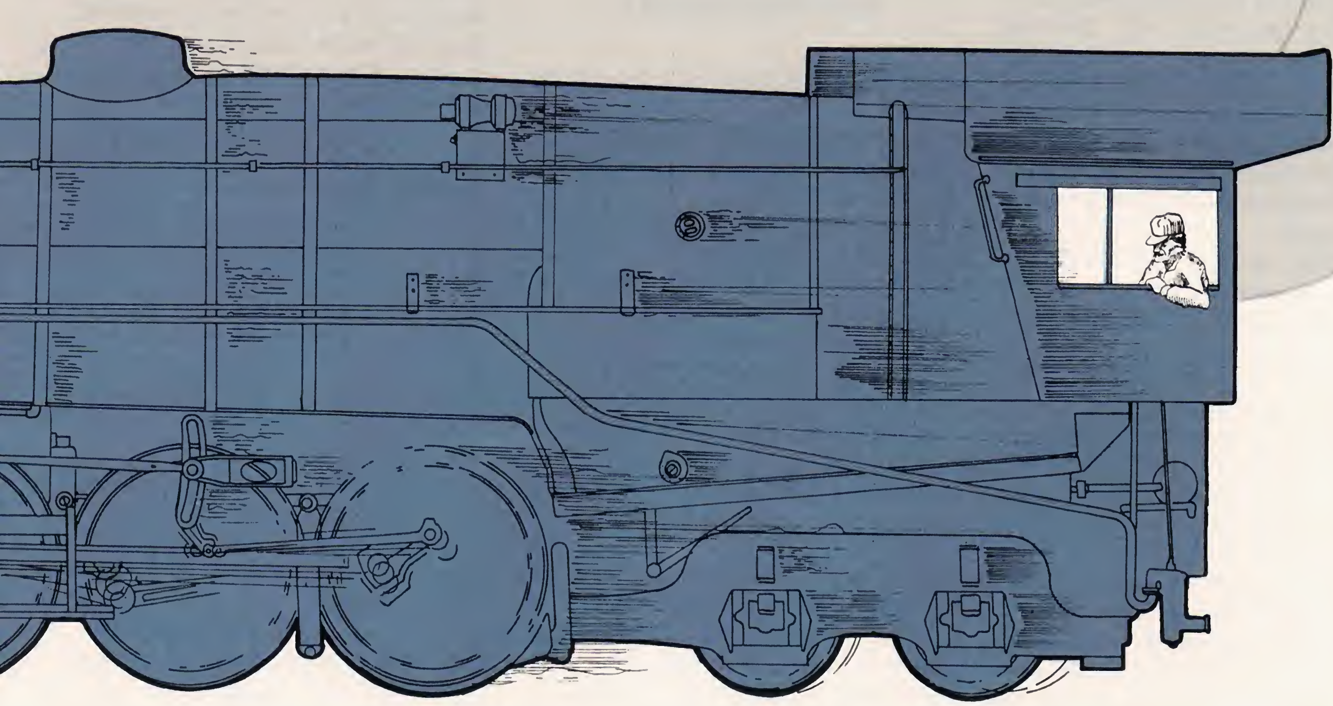


Illustration by Erik Jorgensen

as gradually more and more users are transferred to the host system until, happily, only administrative logins remain. UNIX expertise that is already available in-house can support FEP software directly with no delay for training. The high-speed link to the host machine can be part of a network already implemented on UNIX-based minis, such as the NSC Hyperchannel, and still be able to carry other traffic.

OBSTACLES

One problem that FEP designers are still grappling with is the fact that most UNIX users use screen editors and even visual shells. These programs run "raw," which is to say that they require the CPU to respond to each character as it is received. For a mainframe communicating with users on several FEPs over multiplexed high speed links, this is expensive. At some point, the advantage of big capacity hardware can get lost in simple communications with terminals. One strategy provides for placing screen editors and other raw I/O programs in the FEP itself. The migration of programs in both directions will probably continue as sophisticated applications such as Videotex continue to be developed.

UNIX on a VAX 11/780 or 3B20 will happily support, say, 60 users running CPU-inexpensive processes, such as library index programs. A single programmer, though, can bring the same machine "to its knees" while running a couple of "trivial" builds, such as, for instance, five versions of the operating system. No matter how big a machine is used, there are users who can overtax it. To mitigate the impact of such users, it is possible to create multiple "virtual" UNIX systems that can each run independently and simultaneously on the same host mainframe computer. This strategy may provide a way to control the saturation of the host processor.

There are, however, a number of applications where this will not work since system resources must be shared and therefore scheduled. UNIX 370 runs on top of IBM's Time-Sharing System (TSS) so UNIX system calls are made to TSS and not to the hardware. This is far from the most direct method for fair share resource allocation, and performance is of necessity decreased when an additional operating system is running. Thus the trend is toward "native" mode mainframe UNIX such as that employed in Amdahl's UTS system.

UNIX seems destined to become a true machine-independent work environment.

The main adjustment users will have to make to Big UNIX will be getting used to UNIX being fast. It is hard to scale saturated VAX loads, but one rule of thumb for the UNIX/370 system is that performance is two to five times that obtained with proportional loads on a VAX. In cases where users run many background processes, the performance improvement is even better. This does not refer to how many users can be supported, but rather to what kind of system response can be expected with the system lightly loaded. If raw I/O does not predominate, it is expected that 60 to 150 users can be accommodated depending on their resource demands.

BIG IRON BENEFITS

There are numerous reasons for users to be excited about mainframe UNIX systems. Perhaps foremost is that moving upscale in computer power is always fun. Users running large compute-bound programs will see dramatic improvement in their throughput. Other users, too, will benefit from faster response times. Software containing numerous processes that communicate with each other via signals can be made into larger, more efficient processes by using the extended process address space of the larger machines. Applications requiring large interactive databases are also made easier by the greater online storage capacity and faster disk speed available on mainframes.

Another advantage of Big UNIX is that teamwork is made easier since there are more people immediately available. In large distributed environments, just finding out what system a person is on can take several minutes, by which time the urge to communicate may have evaporated.

Users will also find that their personal system administration is easier on large systems. It will not be as necessary to have logins on several systems in order to have access to a variety of different resources. This necessarily means that users will not have to work as hard to keep their personal utilities and data updated on all the systems they use.

The standardization of UNIX on large computers is really a compatibility issue. UNIX is popular because of the friendliness of the environment ("friendly" once the user is acquainted with the system's quaint syntax). Tools, utilities and commands must work in similar fashion on all UNIX ports, both big and small, or there is little point in calling it a UNIX port. It doesn't matter whether the software comes from AT&T, Berkeley or Cray Research — just so long as it looks and acts like UNIX. And we all know what that is: a UNIX system is one that runs all our favorite programs just the way we expect it to. It's not UNIX if our programs don't run right.

Whether Big or small, UNIX should look the same to the user. After all, why port UNIX to a big

Why port UNIX to a big machine if not to avoid retraining users?

machine if not to avoid retraining users? It is increasingly irrelevant what hardware one is running on. This obviously does not mean that one compiler can do it all, but source code and the production of it can be identical. From a user's perspective, the value of moving up to Big Iron can be determined by looking at a complex function of the number of users, the type and amount of work, and the degree of interdependence between users. The bottom line is that larger systems can do more with more people and lower networking requirements.

Ned Peirce is a systems analyst working as a consultant at AT&T Bell Laboratories in Summit, NJ. He supports AT&T's UNIX development organization as part of the Dedicated Facilities Management Group. ■

UNIX MOVING ON: WHENCE AND WHITHER?

A sense of déja vu pervades the market

by August Mohr

The mainframe UNIX market is currently in transition on a number of different levels. Much of what is happening relates to the nature and history of UNIX.

The development of UNIX has followed a path radically different from that traveled by most commercial products. It should come as no surprise that the development of UNIX on mainframes is also moving in non-traditional directions.

It's been part of popular wisdom for a few years that UNIX owes its success in large part to Bell's foresight in making it available cheaply to universities. What appears in retrospect to have been foresight, though, may in fact have been happy coincidence. UNIX was developed by some very individualistic programmers who wanted a system for their own use. Marketing it was never a consideration. But when word got out in academic circles of its existence, Bell started getting requests for the code. Eventually Bell acquiesced and, through Western Electric, began distributing UNIX in the way that was easiest for itself: as is, unsupported, without any warranties or guarantees whatsoever. Universities got it virtually free while potential resellers had to pay through the nose, cash up front. Bell may not have known what it had, but it had no intention of turning UNIX loose until it found out.

UNIX inroads into the mainframe area have followed a similar course. Many software vendors initially installed UNIX strictly for in-house use simply to facilitate development work. Only later did these companies discover that other people wanted UNIX too.

Following this pattern, the demand for UNIX on mainframes seems to have come initially from the user's domain, rather than from corporate management. At a very basic level, software divisions of large companies are finding themselves moving over to UNIX as they hire more new college graduates and others with UNIX backgrounds. These companies are discovering that they save money by giving their programmers the environment in which they feel most productive.

Donal O'Shea, director of UTS products at Amdahl, explained that, "If someone's bringing people onto an engineering project and has to teach them VM/CMS, for instance, or TSO and MVS, they are not going to become productive for *x* months, whatever *x* might be. But for them to start using *his* UNIX system rather than the one they used in college would probably take only a week or two. We've had

Illustration by Stephen G. Luker

people come in here from college and be productive extremely quickly because they already knew UNIX. Some of them are crazy about UNIX. They're into every detail of it. But that translates directly into money — on the bottom line of an engineering manager's budget — and that's very important."

Amdahl offers an illustrative example. When it began its 580 project in 1976-77 (the 580 is the successor to the 470, the company's original IBM 370 look-alike), it had to build up a second engineering team. Although all the in-house simulation projects were using TSO, LBS or other IBM-world systems, many of the recruits fresh from college were experienced in UNIX and didn't take easily to these complex systems. Rather than fight it, the company put in some UNIX systems for Engineering.

At the same time, Amdahl went looking for an operating system for the 580 console. It eventually settled on a stripped-down Version 6, so in a way, Amdahl has been shipping UNIX for years — but strictly through the back door.

Prompted by programmer demands, Amdahl also decided to try putting UNIX up on one of its bigger machines. It brought in Tom Lyon from Princeton, who had done a paper on making UNIX run on IBM 370 architecture, and put together a team that got it running fairly quickly. It has been used in Engineering very heavily ever since. Amdahl Engineering now has a 5860 computer (its newest and fastest) running UNIX for 200 to 400 users.

Around 1980, Amdahl began thinking of branching out from processors to systems. Since UNIX was already around, it was a natural candidate to become the company's first software product. With this in mind, Amdahl put together a team to clean up its implementation by removing internal comments and references, a necessary first step toward making it a legitimate "outside world" product. (I only wish AT&T had been so diligent. In the UNIX manuals published by Holt, Rinehart, and Winston, there is still a reference indicating that **"troff -g"** can be used to send output to the Murray Hill Computation Center.)

UNIX..."the great equalizer between vendors."

In May of 1981, Amdahl released the product as UTS — a case of a productivity problem and a Princeton paper producing a public product.

Academic interest in the product fell off, though, as soon as a price was set and the UNIX group was faced with a dilemma: if the project was to continue, products had to be sold.

Amdahl's marketing staff had no idea of what it was that it had, so the programmers themselves were the ones who had to sell UNIX. Following "traditional" UNIX distribution routes, they sold it to other programmers who in turn sold their management. The Amdahl programmers were successful; corporate executives were convinced. Amdahl at last had a real UNIX product.

PERFORMANCE GAINS

Amdahl was not the only company with a large software staff interested in UNIX on larger machines. One of Amdahl's major customers, in fact, has been AT&T itself, which was quietly developing its own mainframe UNIX.

Andy Hall, director of the UNIX System Development Laboratory at AT&T Bell Laboratories, explained that, "what really motivated Bell Laboratories was productivity issues. We were finding that large software development projects, like switching systems, where the number of lines of code is numbered in the millions, were very difficult so long as they were distributed over a network of minicomputers. The problems of coordinating databases, of administering systems and networks and of deciding what's wrong when one machine goes down and can't communicate with another, were substantial and were actually getting in the way of development."

AT&T's initial transition involved moving 180 to 200 users from 10 to 12 minicomputers onto a single mainframe. Said Hall, "(By going to mainframes) you can consolidate the database, you avoid the networking problem altogether and you get the advantage of substantially more CPU power."

UNIX on mainframes is also growing from a different direction. Programs initiated on UNIX workstations often show a tendency to outgrow the little machines. There are advantages to being able to easily port programs from the workstations to mainframes.

Hall sees mainframe UNIX as the wave of the future, particularly when combined with UNIX workstations.

"I think mainframe UNIX is going to grow," he said. "It's essentially non-existent at this point but over time the mainframe vendors are going to see that proprietary operating systems are not to their advantage. In an environment where micro-based UNIX workstations predominate, vendors need to get compatibility and UNIX is one way of giving them a competitive advantage. I think

Radio Shack's Model 16B is America's #1 Unix-Based Microcomputer

There Has to Be a Reason!

And it's because our Unix-based software is easier to use (even for a computer novice) than any other system. This system allows two, three or more people to share the power of a TRS-80 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 could perform 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.

Based on a Virtual Industry Standard

For $9995, start with a two-user system that includes our 32/16-bit Model 16B with built-in fifteen-megabyte hard disk and 512K memory, one of our deluxe DT-100 data terminals, the letter-quality DMP-2100 dot-matrix printer, plus multi-user word processing and electronic filing software. TRS-XENIX, which is derived from Western Electric's UNIX multi-user operating system, is standard on every Model 16B.

Commercial Leasing Available

The TRS-80 Model 16B sells for as little as $3999, and the complete two-user system described can be leased for only $345 per month*. Add another terminal for a 3-user system like the one shown below at any time. 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

*Plus applicable use/sales tax. Prices apply at Radio Shack Computer Centers and participating Radio Shack stores and dealers. System includes 26-6006, 26-6011, 26-6012, 26-1256, 26-4401, 26-6431, 26-6412 and one 26-6052. RS-232 cables not included. Xenix/TM Microsoft Corp. Unix/TM Bell Laboratories.

Send Me a New 1985 TRS-80 Computer Catalog!
Mail to: Radio Shack, Dept. 85-A-541, 300 One Tandy Center, Ft. Worth, TX 76102

NAME
ADDRESS
CITY STATE ZIP
PHONE

that UNIX is going to become the predominant operating system on workstations and that people are going to insist that those workstations be networked to mainframes so that they get that kind of computing capability. The natural way to do that is through UNIX, not through proprietary operating systems. We're moving toward a distributed environment where every box is placed together in a synergistic way. The way to make that a success is to have a common operating system."

Donal O'Shea agrees. "UNIX is the glue that sticks it all together," he said. "You have UNIX running on small workstations and in the office on PCs. You have UNIX running on the graphics workstations. We may soon see UNIX on the Cray. And you have all the systems in between. So you can take an application that starts small, and you can — without making major modifications to it or worrying about the vehicle it sits on — move it from processor to processor — to the point where in the mid-1990s, it's running at 1000 MIPS, or whatever is around at the time."

MAINFRAME MANUFACTURER PLANS

Enhancements to UNIX to make it more powerful on mainframes are coming from both inside and outside Bell Labs. AT&T's Andy Hall stated, "Many of the algorithms in System V Release 2 that were aimed at getting the maximum performance out of a minicomputer turned out to be entirely appropriate for getting performance out of a large mainframe."

The manager of Cray's UNIX project, Dave Sadler, is also very interested in the performance gains possible with UNIX systems. Cray's primary commitment is to performance, so UNIX has not been fully accepted yet, but Sadler said that the company didn't have any other projects as backup if UNIX turned out to be less than expected.

"We're putting most of our energy right now into the support of FORTRAN under UNIX," he said. "We're putting some facilities in there that allow you to do asynchronous I/O and large segment I/O, track at a crack I/O, that sort of thing — supporting very large files and data sets.

"Our business is to provide a tool to the marketplace that allows people to go beyond the laboratory, in generic terms. If you want to build an aircraft that runs at Mach 6, and to build a wind tunnel to test, experiment and design it, (you're talking about) a multi-billion dollar effort taking several years. When you can actually do that work through computer-based modeling on a supercomputer, there's a great deal of advantage there."

> **"UNIX is the glue that sticks it all together."**

So there *is* a market for faster and faster versions of UNIX, and UNIX *can* play a role on faster kinds of hardware. There are whole classes of problems that are simply too time-consuming for smaller systems — even dedicated ones.

Dave Lowry, marketing director of Denelcor, a new supercomputer manufacturer located in Denver, cited some research from Los Alamos National Laboratory predicting component speeds between now and the year 2000. According to Los Alamos, speeds may grow up to 100 times the performance of today's components, but there is a very real demand for computers that can run 10,000 times faster than those of today. The overall performance is a product of what the components can do and what the architecture can provide. Lowry's point is that we need architectures that are at least 100 times faster than today.

He claims it would be irrelevant what the computer cost if it could go 10,000 times faster than today's because the problems such a machine could solve would make an enormous impact on meteorological modeling, econometric modeling, nuclear weapons modeling and even air frame simulation, to name just a few areas. "If you could accurately model and replace the wind tunnel with a 'digital wind tunnel,'" Lowry noted, "the return would be such that you could charge $500,000,000 for the computer and nobody would whimper because the results would be so great."

According to Lowry, in universities worldwide, there are 150 projects to build parallel devices. There are a lot of different approaches to parallel architecture, but it's clear that to get that 10,000 fold increase in performance, you would have to have lots of widgits working on the same problem. Lowry believes that this performance is compatible with UNIX.

In the AT&T *UNIX Programmers Manual*, there are places where it says, "UNIX is eminently not a multiprocessor operating system; it's eminently not a real-time operating system." Lowry believes that these are not limitations of the intrinsic UNIX architecture but rather limitations of Bell's implementations. "Therefore you can have a relatively pure architectural version of UNIX that can be both a real-time system and a multiprocessor system," he said. "There have been lots and lots of complaints about the overhead involved in the UNIX operating system. We don't believe these relate to the basic architecture of UNIX; they are more related to the systems implementation on various computational engines."

Denelcor is not running on "traditional" hardware in any normal sense of the term. While its hardware will support the emulation of a conventional single-stream time-shared environment, that does not make full use of its unique capabilities. The Denelcor machine offers a massively parallel architecture — so parallel that processing units can be added modularly to achieve near-linear performance gains (as opposed to the diminishing returns experienced in many other parallel processing systems). The parallelism runs so deep that a normal UNIX program swap context switch can happen entirely in hardware, without calls to the kernel.

A NEW WAY OF THINKING

Highest performance with Denelcor may mean learning a whole new approach to programming, analogous to "object-oriented" programming. The mind set required to understand how many processes are happening and available to the programmer simultaneously is a mental leap some more traditional programmers may not be willing or able to make.

Even at this newest, most radical level, UNIX follows the patterns of its origins. The basic UNIX concepts involve a mental model foreign to older operating system concepts that are yet simple and powerfully extensible. The same is true of the object-oriented, parallel concepts at the forefront today.

Because of UNIX's "bottom-up" history, there are relatively few companies that take advantage, at the level of corporate strategy, of the unique nature of UNIX. In terms of corporate strategies, the beginning of UNIX's emergence as a "standard" operating system makes it possible for smaller companies to compete with larger ones, by giving customers more choices.

Gould, in particular, sees UNIX as a primary vehicle for its competition with mainframe manufacturers. Greg Hopwood, Gould's manager of special UNIX programs, called UNIX "the great equalizer between vendors." In a game where the winners are those that can gain market share at the expense of others, interchange standards and *de facto* standards make the market more fluid, which creates an advantage for the smaller manufacturers. For Gould, there is a great advantage to be gained if it can establish that mainframe competition is in the UNIX arena.

Thus UNIX again follows a path contrary to that of the big exclusive operating systems. By giving customers more hardware choices, UNIX is bringing competition to markets that have been dominated by single vendors.

The history of UNIX includes an inseparable link to academic communities. UNIX has been favored historically in university research environments. The recent wave of UNIX publicity, particularly word-of-mouth, programmer-to-programmer publicity, has made the system much more saleable — even in fields without strong academic ties.

Just as Bell experienced a steady demand for UNIX, manufacturers such as Digital Equipment are discovering that their installed customers are demanding UNIX. Corporate strategies are being forced to

realized.

to answer questions. Working with both end-users and OEMs, we'll find the best solution for any particular application.

Sound interesting? Call your local HP sales office right now about the HP-UX operating system. Or write to Hewlett-Packard, Attn. Pat Welch, Dept. 100201, 19447 Pruneridge Ave., Cupertino, CA 95014. In Europe, contact Henk van Lammeren, Hewlett-Packard, Nederland B.V., Dept. 100201, P.O. Box 529, 1180 AM Amstelveen, The Netherlands.

Productivity. Not promises.

HEWLETT PACKARD

Circle No. 19 on Inquiry Card

change in response to blossoming "grassroots" interest in UNIX.

We can expect that the future of UNIX will reflect the patterns of its past. AT&T is attempting to create a form of "legitimacy" for UNIX by trying to promote it through traditional marketing techniques. This will create anomalies in the overall pattern UNIX has followed. Instead of programmers demanding UNIX from management, some executives are beginning to respond to AT&T's campaign by asking their programmers, "What is this?" These are the executives most likely to become users themselves.

Because of the nature of UNIX and the strength of its historical patterns, acceptance of UNIX at upper levels of institutional and corporate management will likely continue to be slow and be primarily motivated by programmer and user demand. Word-of-mouth, as always, will be the best sales tool. New UNIX products will continue to arise out of developments initially created for internal use. Public demand for the products will likely be high until they are released.

UNIX has a historical and traditional antagonism to the "captive market" form of proprietary operating systems. Companies that have depended on that marketing strategy will be reluctant to give any support to UNIX, except to offer it on their hardware. That way they will attempt to avoid promoting UNIX and "competing with themselves." Yet they will still need to have UNIX products available to keep their share of the market.

We will see the concepts of UNIX extended in ways that maintain the integrity of the original, but allow development of bigger, faster, more intricately networked, actively parallel systems. Because it allows such hardware flexibility while maintaining a consistent user environment, we will likely see UNIX on virtually every new machine built in the foreseeable future.

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. ■

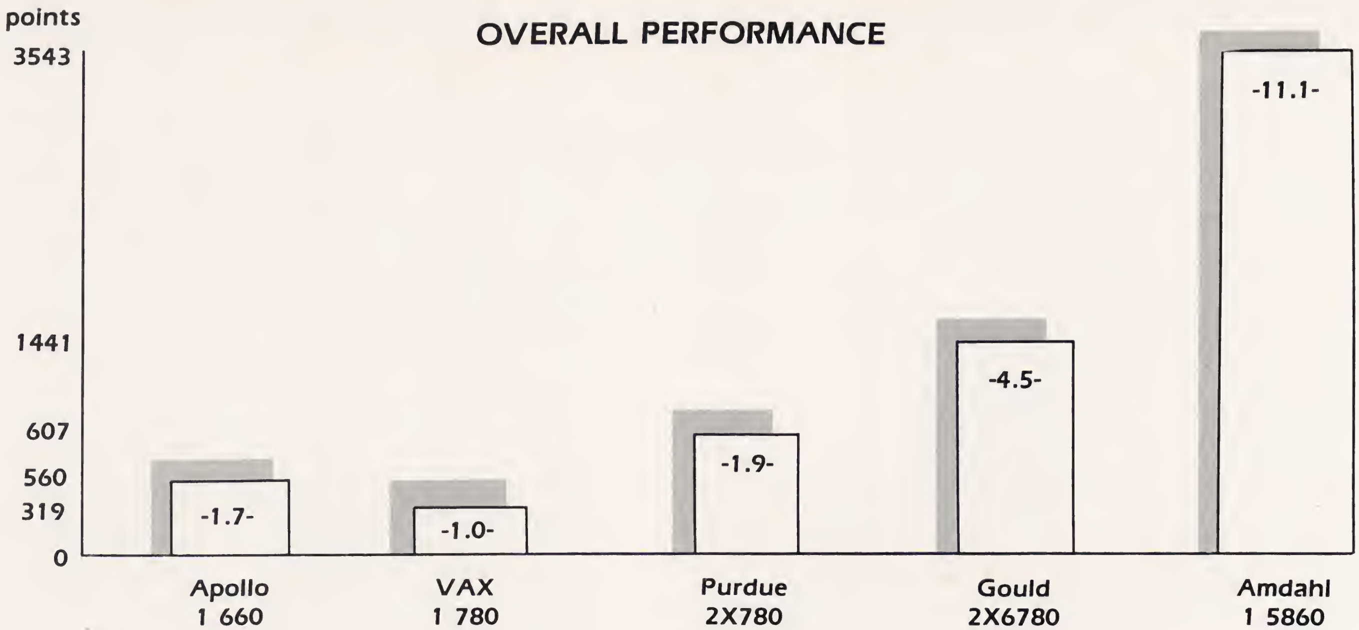
BIG IRON PERFORMANCE

Test results benchmarked against the VAX 780

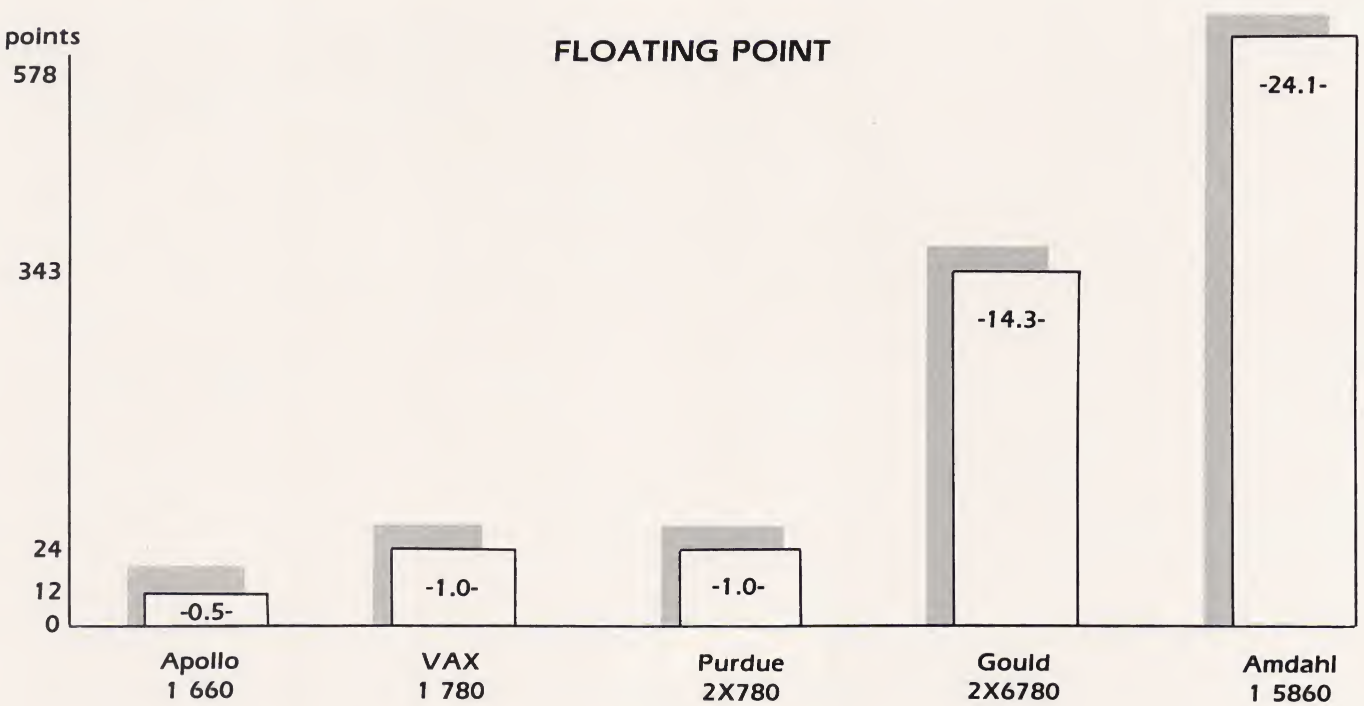
by Gene Dronek

There can be no doubt that both the Gould 6780 and Amdahl 5860 deliver BIG IRON performance — five to 10 times a VAX. The Gould machine actually ran dual processors to deliver the total compute power shown in the charts, but Amdahl ran only a single processor. It is particularly dizzying to consider the power of the Amdahl machine. If there were 11.1 users concurrently on it, they would EACH experience experience the compute power of an unloaded VAX. If you were a VAX user familiar with 10 or 40 users per machine, the ONE Amdahl could hold 100 to 400 users!! Forget all perceptions of time. Most commands execute during the RETURN key "bounce time" on the Amdahl.

Regrettably, these charts include data for only the Gould and Amdahl systems. Data for the other systems mentioned in the feature article did not reach *UNIX REVIEW* in time for publication, and so will be represented in a subsequent issue.



The figures shown here reflect overall performance scores for several high-performance systems. Each bar is labelled in VAX EQUIVALENTS for that system's overall performance score, based on the Aim Benchmark tests. The Apollo scored 1.7 VE, the Purdue Double-Vax scored 1.9 VE, the Gould 6780 dual processor scored 4.5 VE, and the Amdahl scored an astounding 11.1 VE. The points scale is based on a maximum of 3600.



This chart compares only FLOATING POINT instruction speeds for the same systems. Floating-point performance, shown in VAX EQUIVALENTS above each bar, was calculated based on the six Aim floating point test values of each system. The VAX 780 did not have an FP accelerator, while the other systems did. For floating-point power, the Apollo scored .5 VE, the Gould scored 14.3 VE, and Amdahl scored 24.1 VE. The points scale is based on a 600-point maximum.

Gene Dronek is the author of the standard benchmarking package for UNIX systems, the Aim Benchmark suite. He formerly served as the lead UNIX consultant at UC Berkeley. ■

SECURITY ROUND TABLE

Round two of UNIX REVIEW's security talks

UNIX REVIEW opened its security talks last month as part of its focus on system administration. The talks continue this month as an extension of our focus on UNIX on Big Iron. The extension is a natural one since many of the security measures devised for UNIX systems were first used on large machines serving major research and academic institutions.

It should also be said that neither of the experts featured in this month's installment — George Goble of Purdue University and Bob Chancer of AT&T Bell Laboratories — are strangers to large systems. For them, a personal computer is something on the order of a VAX.

As with last month's talks with Ed Gould of Mt. Xinu and Bob Morris of AT&T Bell Laboratories, the questions this month were provided by Dick Karpinski, manager of UNIX services at UC San Francisco. Though the interviews were conducted independently, the queries were kept essentially unchanged from one interview to the next to allow for easy comparisons.

George Goble

As the Computer Systems Engineer of Purdue University's Electrical Engineering Department, George Goble administers 25 systems, including 17 VAXen and a number of PDP-11s. His experience with UNIX reaches back to the Version 6 days. Goble made his mark on the UNIX community two years ago when he worked with Mike Marsh to develop a dual processor 780 VAX.

REVIEW: *What are some of the most common security problems you've encountered?*

GOBLE: One would be **mail**. If you can fake the sender (the "From" line) on **mail**, you can probably do bad things to organizations where most mail is electronic. You can give orders to people saying you're really somebody else (like their boss). We try to guarantee secure **mail** "From" lines internally on ECN (the Engineering Computer Network), but any mail from somebody else (not on ECN) is not secure. This includes **ARPANET**, **UUCP** and other campus machines.

REVIEW: *So, if you are going to make it secure, you'd have to do something like passing an encrypted signature.*

GOBLE: Yeah, or something like that. It's pretty rough, but internally that's been one of our biggest problems. It hasn't been misused yet, but there's a good possibility of it. We've been able to keep on top of it, but trying to make **mail** airtight is almost impossible.

If something in writing is forged, it can be traced. The police are pretty good at that stuff. But computer forgers can just walk away scot-free. You can just walk into a pay phone, hook up your TRS-80, put something in there and leave —without a trace.

Using **mail**, you can just tell somebody to go do something for you. Eventually it might get found out, and then you're up the creek, but there are a lot of people who will do anything they're asked to do. This is by no means the extent of all the security problems or even close to it. We are operating in a university environment here, and our users are somewhat hostile — certainly more so than in a research-only environment. You hear a lot of people talk about things like having callback on dial-up lines and encryption and all that stuff so as to keep people from getting access to an account to start with. But we have something like 10,000 accounts here since every student has an account based on being enrolled in engineering — and we don't charge for it, it's free. So most of our security is not directed at keeping people from getting on. We assume they're already on. But can they get into somebody else's file or do damage? That's the focus of most of our security concerns here.

We're dealing with a whole generation that grew up at universities and understands UNIX. They're very sharp. They can cut through it like butter.

There are cases here where people have put their friends, or what they think are their friends, in their search path, only to find one day when this guy doesn't like them that he can put a fake **ls** program in there or something.

REVIEW: *And wipe out all their files?*

GOBLE: Yeah, and he can also make a **suid** file to you, so that you can change your password all you want, but as long as he's got that **suid** shell, he can take shots at you until complaints to the system staff finally track it down. That's a dangerous hole.

Groups can have the same problem. They're fine and nice and all that, but a lot of people don't bother to look at **groupids** and probably reset all the group permissions. They just let something default to a creation mode like 775 (giving all permissions to anyone in the group), and then forget to choose their group. Then a lot of times, sources will get group "other," which everybody can get.

There are probably seven of us here who could walk into almost any other university UNIX system and have root within five minutes.

REVIEW: *How much security is practical or possible, and at what cost?*

GOBLE: It depends on your environment. If you're with the CIA, it's obviously one thing. In environments like this, it's another. But an extreme amount of effort is necessary to make anything totally secure or almost totally secure. With a reasonable amount of effort, though, you can make it harder to break into somebody's files on a computer than it is to break into his office.

To achieve 100 percent security is like approaching the speed of light. You can't. It's impossible. But you can get pretty close without too much effort — that is, within five to 10 percent. Our systems here at Purdue are by no means totally secure, but our ECN systems are generally eight to 10 times more secure than the average UNIX system.

REVIEW: *Would you measure that essentially in terms of the cost to break in?*

GOBLE: In terms of *how long* it takes to break in. There are probably seven of us here who could walk into almost any other university UNIX system and have **root** within five minutes. In fact, we used to give demos for the police or the FBI when they would be visiting for security talks.

There's always some loophole or some file not set up right. Here the same people take maybe 30 or 40 minutes to break **root**. These sorts of people exist at Berkeley and MIT and a lot of other places. It's not like the 414 hackers, who just dial up random numbers and stuff. I mean, these are folks that are on dedicated systems.

The average undergrad hacker probably doesn't even get into this league until he's been here for six or eight years. He might find his first **root** bug after going at it every night for, say, two or three semesters — if that's all he's doing. But by then we know who they are because...

REVIEW: *You've been watching.*

GOBLE: Yep. This is 1984. Okay, another part in this question is: you might take all the steps you can for security, but if the users can't or don't take the effort to protect their sensitive data — or maybe just don't know how — it's all for nothing. For example, I found that for 10 years now all salary information for the engineering staff and faculty here has been available for write access to any undergrad using the main campus CDC computer system. I gave a talk on computer security and used this as an example, using transparencies of an actual terminal session that covered up the read and write keys to the files. I warned the administration four or five times about this security problem, but they just said it was totally secure and couldn't be broken. After three warnings, nothing much had changed. So when I showed this live example at that security talk, a lot of people turned very red and a lot of jaws hit the ground.

REVIEW: *So what's the cost of security?*

GOBLE: We haven't really spent that much on special hardware for security here, because obviously anybody can have an account who wants it. Most of the cost has been the time and effort the system staff — me mostly — has had to spend trying to enforce security. The thing is, it's not only lost time — it's wasted productivity. Another thing, and this probably applies everywhere, is that when you get a serious enough case to call in law enforcement people, the time spent for what you accomplish is almost an order of magnitude worse. You can multiply it by about 10 because police don't really understand computers that well, so you have to train them.

We had an incident where somebody stole Dennis Ritchie's account. It was actually a Bell Labs contract employee on the East Coast who had broken into 20 or 30 systems, cleaned them out and gotten **root**. Purdue was the first system that stopped him. After catching him, we monitored him for a whole week. As soon as we realized the call was crossing state lines, we got the FBI in here and had as many as 10 or 12 people in my office just watching this guy for four nights in a row. He was trying to use **UUCP** to break into systems on the West Coast. So we set him up with fake data. He had retrieved data at three in the morning, like somebody else's password file, so we put fake passwords in it — encrypted ones — and that's what they nailed him with. It was a live display. I had a **root** terminal. We had the monitoring going on my other one, and we made sure he wouldn't find any hole. He would go down a hallway and see an open door, and we would slam it just before he could get to it.

REVIEW: *That's called active monitoring.*

GOBLE: He was a real sharp guy. We knew from the commands he was typing that he wasn't from Purdue, but he was very experienced at hacking. He tried 20 or 30 **root** bugs, most of which had been fixed. Maybe one of them hadn't, but everything he tried had either been fixed in previous years or we fixed them right on the spot before he got to try them. For two weeks he was dialing in for maybe five hours a night.

With a reasonable amount of effort, though, you can make it harder to break into somebody's files on a computer than it is to break into his office.

REVIEW: *What are some of the other security problems you've encountered?*

GOBLE: Our biggest security problem is management policies and people problems. Sometimes, the administrative people don't allow us to run security as we see fit. These people don't deal with the system every day and so they aren't really knowledgeable. We have pretty good mechanisms here that can usually tell when stuff's going wrong or people are doing illegal things. But we're not allowed to use them unless problems have already been demonstrated. It's a Catch-22. You can't find the problems unless you're allowed to look or unless somebody squeals on somebody else or unless a bad guy leaves his output behind after taking a joyride with **root**. We actually have a console switcher that's connected in parallel with all the consoles so we can get along that way, but we're not allowed to monitor people or even go looking for things — especially when it comes to grad students or faculty members.

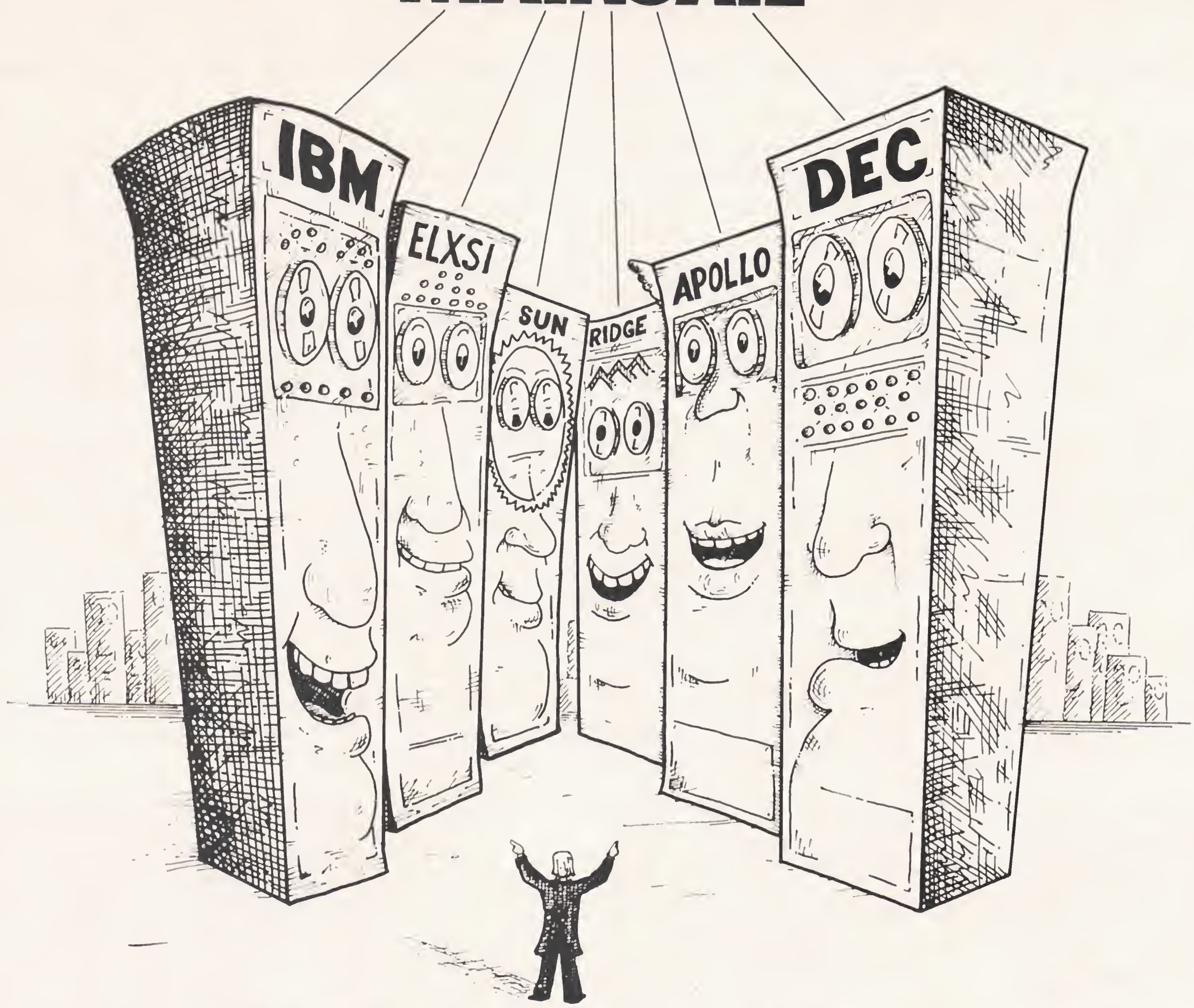
REVIEW: *In a sense, that sounds like a problem with educating your own management.*

GOBLE: Yes. But that's probably our biggest single problem. Otherwise, there has been much progress made recently. We pretty much nailed a whole frat running on stolen accounts and turned the case over to the Dean of Students not too long ago.

But I've got a couple more big security problems for you if you want them. One is the **set userid** feature of UNIX that a lot of people moan and groan about and say shouldn't exist. It actually is needed in a lot of cases, but it is very easy for inexperienced — or even experienced system programmers who don't keep an eye on security — to write these things and mistakenly think they're secure. But Kirk Smith, a graduate student here, and I have shown that's not the case.

As an example, one of the management people in our group

MAINSAIL™



SPEAK MAINSAIL. BECAUSE NOT EVERYONE SPEAKS UNIX.

Most computer programmers like the UNIX environment. It gives them a convenient set of software development tools. There's one problem, however. Applications must often run on machines that don't have UNIX.

That's where MAINSAIL comes in.

MAINSAIL is a powerful programming language that can help cut your development time and eliminate software conversion costs. You'll be able to take advantage of the development power of UNIX, while retaining the ability to move to other systems. And you'll be amazed at how easy it is to learn this proven, versatile language.

If you like UNIX, but need to keep your portability options open, just let us know. We'll show you how MAINSAIL can accelerate your development of portable, sophisticated application programs—not only under UNIX, but under VAX/VMS®, VM/CMS®, and a variety of other operating systems. For details, contact us at: XIDAK, Inc., 530 Oak Grove Ave., Suite 101, Menlo Park, CA 94025, (415) 324-8745.

UNIX is a trademark of Bell Laboratories.
VAX/VMS is a trademark of Digital Equip. Corp.
VM/CMS is a trademark of IBM Corp.

xidak

Circle No. 22 on Inquiry Card

had eight programs of his used to break **root** in the space of a week or a week and a half. One of the ones that was broken, by the way, is called VSS, for "Very Secure Shell." It was designed to let operators do dumps without having to be **root** themselves. But the log file it maintained could be moved because it was in a directory writable to a non-superuser and it could be forced to write out a password file, allowing anybody to put in a **root** account for themselves, using a program designed to keep **root** away from people.

You can look at almost anything, though — especially with symbolic links now in 4.2 — and shoot it full of holes if you spend maybe half an hour on it. It is very very difficult to write a good **set userid** root application of any sort without making it a security problem. You have to be at the level of the kernel before you even attempt to write these things. Distribution tapes come just full of this kind of stuff, and **mail** is by far the worst one. They're so big you can't prove them correct or even understand them.

Our other big security problem is uneducated or uninterested users who leave themselves unprotected by not using available protection mechanisms. That goes all the way from department heads to undergrads.

You can give a baby a gun and try to make it so that it won't shoot when he points it at himself, but that's not the problem. If you set people up right, they'll still manage to muck it up and open their directories again to the world. Of course, they won't realize it, and then they'll blame you for their problems. There are lots of hand-holding things written for these people, basically just to tell them if they've screwed up or not, by looking at obvious things, like 666 mode files, write bits on their directory and that sort of thing. There are just some people who are so naive that they have no concept of this stuff at all. They should be put on a personal computer stuck away in a closet somewhere.

REVIEW: *What sort of measures do you recommend for these concerns?*

Circle No. 25 on Inquiry Card

GOBLE: The first is to try and raise the awareness of management to the potential problems. I've written memos and given security talks that managed to alarm a bunch of people, but it's all worn off and nobody's done anything. The second measure would be to hire sharp and honest system people, when you can, to stay on top of things. Besides writing **suid root** programs, these people should try to play devil's advocate by working as hard as they can to break programs before they're installed for use. It's also good to have friends and hackers try to break new programs. That's how Kirk Smith came into things here. I'd rather have him break **root** than have an enemy do it.

The other step I'd recommend would be to set up standards in computer ethics. They are nonexistent now. People say it's unethical to go through and look at somebody's directory if his files are just laying there, readable — or even writable — to the world. What if somebody goes in and cleans them up or deletes them all? Where do you draw the line? A lot of times what the students consider to be ethical and what the faculty considers to be ethical are two different things.

REVIEW: *What's the distinction between the steps you recommend and what you actually use?*

GOBLE: I've got code in the terminal driver that we can use to monitor any traffic, beep my pager and transcribe everything to files. People trying to penetrate the system can't even tell what's happening unless somebody blows it on the management end. This gets used only very rarely, maybe two or three times a semester. Another thing we do is look for large amounts of login time at night by certain users. We also have a reward system for users that report somebody causing problems. We give those people maybe 500 blocks of extra disk space. This has proven to be fairly effective and it's almost free for us.

REVIEW: *Do you ever find that security flaws have left the system too secure, impeding people in their work?*

GOBLE: Not really. Right now we're just making default accounts mode 700, meaning they're locked to the world, but all the earlier ones are readable to the world. We're aware that there are places on campus where that can be a real pain, but as far as Engineering goes, I don't think we've impeded anybody.

Security is just like insurance. You buy insurance on your house, but if someone drops a nuke on you, it's null and void.

Software in the system is generally available to everybody. It's not like you have one group that can access this program, and somebody else that can run that program, because in general, things in the system can be run by everybody. That's not really a security issue here. Getting on the system is not an issue either, because anybody who is here has an account.

REVIEW: *So you're not employing any particularly sophisticated technical devices?*

GOBLE: No. Anybody can just walk into the terminal room, which is open 24 hours a day and do whatever he wants. So callback stuff and encrypting the phone line and that sort of thing doesn't even apply to us, where it would for a business.

REVIEW: *How much security is too much?*

GOBLE: The precedents haven't been set on that. Obviously the management people who make the rules here already think we have reached that too-much point since they essentially don't let us do anything unless something has already been proved. It's kind of a Catch-22.

I've got a console terminal here monitoring 25 machines. I just built a small 68000 with 35 ports on it that I want to use to reboot machines here on campus from home. It's not directly on dialup yet, so I go in through another machine. But when it is on dialup, it is going to have callback on it and extra tones to get into it and all that because we're talking about instant **root** access.

REVIEW: *Have you experienced any breaches?*

GOBLE: Of course. I don't think anybody can say they haven't.

REVIEW: *Well, in fact, when I talked to some folks from Bell, they said they'd be surprised if I got anybody to answer that question.*

GOBLE: A lot of people won't admit to it. It depends on what you mean by a breach, whether somebody got a file or whether somebody took **root** over. Bad guys get **root** here only a couple of times a year. But there's a lot of people that do it just for the fun of it. They break **root**, then they come and turn it in. They don't do anything with it. Sure, in a sense, that's a breach, but as far as people breaking in and then cleaning

The world's most widely-used computer language:

"@!#★?*"

All computers fail sometimes. And when they do, you'll be the first to hear about it. In no uncertain terms.

That's why we developed the Parallel 300. It's the computer you can swear by—not at. Because even when something fails, the system keeps on running. With no down time. No data loss. And no crises.

You can even fix it yourself while the computer's still running. The Parallel 300 diagnoses the problem, isolates it and sends you a message. You just remove the bad part and replace it with one we'll send you overnight. No tools. No midnight calls. No trained engineers.

With this kind of reliability, you can use the Parallel 300 anywhere. Where power's uncertain, in remote locations, for production lines or deadlines.

The Parallel 300's fault-tolerance is transparent to users and developers. And it's UNIX based so all your programs are portable.

So see how easy it can be to improve your language with a Parallel 300. Call us today. Parallel Computers, Inc., 3004 Mission St., Santa Cruz, CA 95060 (408) 429-1338.

Parallel

Computers you can swear by, not at.

See us at Booth 343

Circle No. 26 on Inquiry Card

out a machine or something, that is very rare.

REVIEW: *Didn't you have one particularly malicious occurrence?*

GOBLE: That didn't actually happen on our system. That happened in PUCC, where they can track anybody but can't do anything about it. Apparently an instructor either screwed up and set a whole entry level CS class up with mode 777 directories, or this kid — a freshman, I think — tricked the instructor into setting him up that way. There were 20 or 30 people in the class and this kid went through and **rm** *ed everybody but one or two friends. It was sort of like the Chicago fire with two or three buildings left standing. It was pretty obvious. They nailed him, basically, and he confessed. They took him to the Dean of Students, where they spent several days getting the Dean all primed on what a file was and what it meant to remove one maliciously. They finally agreed to throw the kid out for a year or something. But he went back to the computer center and asked if he could have his account back. Of course, they said no, so he said, "I'll just go use EE."

I was immediately called, and we went on red alert. It took about a day to figure out which account he was using. He didn't do anything bad, but just the fact that he was on using a stolen account led us to call the police. Those of us at the computer center worked together to get a sworn statement, so the police charged him with a bunch of stuff and arrested him. It made the headlines and all that, but there was nothing they could really do. They sent him back to the Dean of Students, where he got some more time slapped on his suspension.

He was a potential destructive case, but that's been the only really bad case. There have been two or three others, but nothing severe. We've suffered a lot more damage from management people believing they ought to hire people right off the street who know nothing about computers, but who are entrusted with **root** the first day. Typically, management just turns them loose and says, "Here, do these backups," so they go and remove whole file systems. They have no idea of what they're doing.

REVIEW: *Ed Gould made the point that because of convenience, administrators themselves often overuse* ***root*** *permissions. He decries that activity, suggesting that it would be better to get back out of superuser as soon as possible.*

GOBLE: Yeah. I overuse it a lot, too, but I've never done anything bad with it. If you get somebody that doesn't know what's going on, though, it can be a real problem.

REVIEW: *What about* ***UUCP****?*

GOBLE: It's a can of worms. And that's where I'll leave it. We've got it bottled up on one system, and nobody knows which one that is. If anybody wants to start digging for holes, I'm sure that's the place to start.

I think only 10 to 15 percent of **UUCP** is actually used. It was a big experiment in networking, and it's buggier than all get out. Even with the latest versions, you can still fake your ID. And then there's net news. Let's say someone dumps something on net news like, "This bug just showed up in 4.2 UNIX, and you can get **root** with it. It's on all systems, including System V and System V.2, and here's the five commands so you can use to invoke this thing." Suppose that goes out on a Friday night to hackers on 10,000 systems. By Monday morning, system administrators everywhere would come in to find everything in smoking ashes. That's why we had this secret security meeting at (the June 1984 Usenix Conference in) Salt Lake City. We need to defuse a lot of this stuff.

REVIEW: *Has that happened?*

GOBLE: I found a bug in January that I can't talk about that lets you get **root** on basically anything after System III, including all the Berkeley and Bell stuff. It would cause a nationwide disaster if that got dumped onto net news. It was fixed here the next day, before anyone could actually invoke it. Somebody had done something similar to it and caused a crash by mistake, but he didn't even know what he had done. He just knew the system went down. I went through the dump to try to find out what happened and when I looked at that code, I wondered, "What would happen if I tried this?" I did, and sure enough it worked. No one has actually used that bug anywhere to my knowledge. I tried it on 4.2, and it gave me **root**. I tried it on a 4.1 system, and it worked. I tried it on 4.0, and it worked, and I had somebody try it on System V, and it worked. So I called Dennis up then...

REVIEW: *Do you see any different impact for local area networks on system security?*

GOBLE: That's a mess too. We run our own local network here, but not really work stations. It's all Ethernets or DMC-11s VAX-to-VAX, so everything is basically under the same administrative control. It's not like some guy can take one of these machines and load his own stuff in it, get on and become **root** — which he certainly could do if there were user loadable workstations. So we use Ethernets in that sense, where everything's under our control, but if you get a string of work stations and some guy can download

In microcomputers today, UniSoft sets the standard.

AT&T has recently been advertising that their UNIX™ operating system will be the standard OS for microcomputers. That's true. But if you want AT&T's UNIX software on micros today, talk to UniSoft Systems.

UniSoft has been delivering AT&T's UNIX adapted for 68000-based microcomputers for two years. More than 75 different computer systems run the UniSoft software, UniPlus+.™ At each Bell release level, all these systems are object code compatible. This means that applications software developed on any UniPlus+ system will work on any other. This is where software portability pays off.

UniSoft enhances Bell's vanilla UNIX with the best features from the Berkeley BSD research version of the UNIX operating system. IP/TCP networking, record and file locking, and virtual memory from UniSoft turn UNIX into a commercial product. All this added value is still Bell-compatible.

Don't wait six months to get System V running on your hardware. UniSoft's customers can ship it now.

If you're building or selling a 68000-based UNIX system, your operating system should come from UniSoft Systems, the UNIX experts.

UniSoft
SYSTEMS

THE BERKELEY PORT AUTHORITY

Circle No. 16 on Inquiry Card

*UNIX is a trademark of Bell Laboratories

739 Allston Way, Berkeley, CA 94710 • (415) 644-1230
TWX II 910 336-2145 • UUCP ucbvax!unisoft!unisoft

See us at UNIX SYSTEMS EXPO/84, Los Angeles, September 11-14, Booth #117 his own stuff into one of them, with kernel permission, then that machine is certainly a mess. That guy can run in promiscuous mode or come on the net and fake himself to be somebody else, send resets to their host and blow them away. You have to have an Ethernet where hardware encrypts data or you have to basically put every guy in his own private channel somehow. Right now, we give workstations their own dedicated Ethernets so that they cannot be monitored by other workstations. That's kind of expensive. But once you can encrypt the data portion really well, that should solve the workstation on Ethernet problem. I don't know if anybody is really working on that. I talked to Interlan and some other people about it a couple of years ago.

REVIEW: *People are thinking about it, but I'm not sure anybody is putting any products out that are ready to be used.*

GOBLE: There's not that much to it.

REVIEW: *What do you see as the future of security management?*

GOBLE: There'll be a lot of people out there claiming they can do everything. There are a lot of managers out there who will buy anything advertised as security with big bucks. In big corporations, once you get three or four levels of managers up, you find people that are so far removed from the nitty-gritty work that all they can do is throw money at problems and not really get into the middle of them. So people are going to sell a lot of things, claiming to protect this and that.

Security is just like insurance. You buy insurance on your house, but if someone drops a nuke on you, it's null and void. And there are probably going to be lots of liability problems and lawsuits, once you start getting software vendors to sign contracts making people liable for damages if operating system bugs surface and cause damage. I mean, that's scary.

Bob Chancer

Bob Chancer is a member of the Computer Planning and Security Group of AT&T Bell Laboratories in Whippany, NJ. His responsibilities include monitoring corporate security and developing software security tools.

REVIEW: *How much security is practical or possible, and at what ultimate cost?*

CHANCER: That is very dependent on the environment. If you're talking about a military environment, you can lock all the user terminals hooked into a computer in a large metal box, from which no radiation can escape which is basically what they do in certain situations.

But if you were in a university, you'd possibly want to run a wide open system. It's like insuring your home. The amount of money you spend depends on what you have.

You probably want to lock up your jewels in a safety deposit box. But do you want to lock up a loaf of bread in a box downstairs? You've got to look at what you're securing.

REVIEW: *So what's the biggest security concern that you face? This is very much an extension of the question of how much security is practical.*

CHANCER: There is not just one big problem. There are things like management support, without which you can't have security. You also have to consider private machines. Many people own their own machines nowadays. How do you control private machines? It is very easy to control a computer center where you have a single management structure, but when anybody is able to own a machine, it becomes more unwieldy.

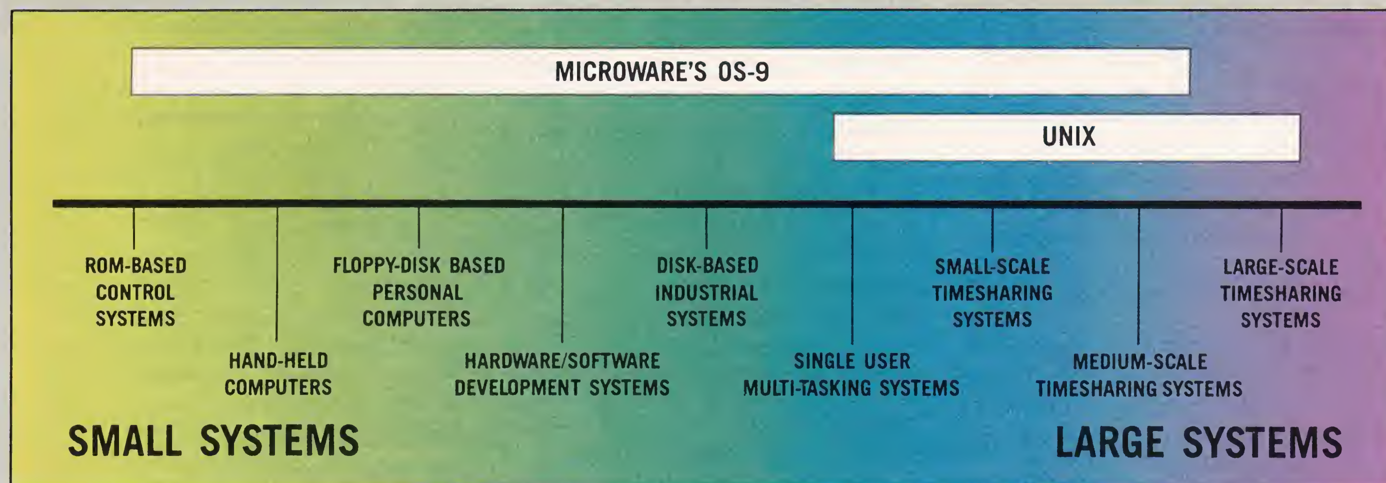
Another issue is how to provide both at a reasonable service level and reasonable security. Another area is user naivet$_6$e and carelessness. One of your biggest security problems is your body of users and what they're doing with their passwords and permissions.

And each machine has different data on it, so the level of security you might want on one machine where you have payroll information has absolutely no relationship to the level of security you might want on a machine that only secretaries use for generating low security letters.

REVIEW: *What are the measures you recommend to take for these concerns?*

CHANCER: Well, one of the biggest ones is education — educating management about

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. 28 on Inquiry Card

what the problems are and what can be done to solve them. You also have to have knowledgeable users.

The UNIX system provides you with a lot of capabilities for securing information, for example by placing permissions on files and creating login passwords. But unless your users know what those capabilities are and know that there may be a problem, you've lost the ballgame.

REVIEW: *Sometimes people feel differently about how important it is to secure particular pieces of data. How do you resolve these disagreements?*

CHANCER: Well, in a corporation, there are rules dealing with how secure certain types of data have to be. It's traditional that certain things, like patent information, be kept highly secret. There are also typically rules in corporations about how secure the computer has to be.

REVIEW: *Do you have a way of measuring the costs of security measures? One of the costs often not considered is the cost of interfering with people's work when they run into security-imposed barriers.*

CHANCER: There are a lot of security measures that do not affect production, like requiring people to pick good passwords.

Bell Labs makes a strong effort not to affect people's productivity while attending to computer security. There's a lot of data anybody in Bell Labs can access. But if you're outside of Bell Labs, the computers won't allow you to have this.

There is also some data that has to be restricted to a certain few users, like salary information. In most cases, security measures guarding this information will not affect anybody's productivity for the simple reason that those who need to know something will already know it or will have access to it.

REVIEW: *Have you experienced any breaches of security?*

CHANCER: Well, I can tell you about what I read in the papers and have heard from colleagues outside of Bell Labs, but I'm not at liberty to tell you of any problems within Bell Labs.

REVIEW: *Do you know of any breaches of security that have proved to be malicious or destructive?*

CHANCER: I've read about them, like the kid down in Atlanta who was stealing computer equipment. That's extremely destructive. But I've also read about people who break into systems but are not malicious. Maybe they try commands and accidentally wipe out a database.

> **Our machines, if we did not secure them, could not be worked on.**

REVIEW: *Do you have anything to say about the impact of UUCP on system security?*

CHANCER: **UUCP** can be a problem if it's set up incorrectly. If you allow other machines to take files off your machine, you're opening yourself up to problems. **UUCP** can be set up so that you can send files but nobody can take files. Of course there are other features that **UUCP** will allow, depending on the data on the machine. You could, for instance, require a callback. This means you don't trust the other user to be who he says he is, so you actually call him back before supplying any information. That way you know who you're talking to.

REVIEW: *Even then you're going to want to pay attention to how much freedom you supply to the UUCP login.*

CHANCER: The **UUCP** login doesn't invoke the shell. It invokes a program called **uucico**. I defy anybody to use a terminal and try to talk to that. You have to have another computer system to talk to **UUCP**. And there is a new, more secure release of **UUCP** being developed. It is delivered in a secure mode, because it takes a little less thinking that way.

Because the problem has been that programmers tend to get overloaded, they sometimes install things without looking at them from a security viewpoint. It goes back to the problem of having too much work to do, which tends to be a normal programmer problem.

REVIEW: *Do you see a different impact for local area networking on system security?*

CHANCER: There are many ways to set up local networks. You can set them up in such a fashion that when you login to one system, it's very easy to login to another, but you have to supply a password for the other.

REVIEW: *But when you login to another system and supply a password, couldn't a promiscuous listener on the network thereby uncover passwords?*

CHANCER: If your network software is secure, that doesn't tend to be the problem. The problem in a local area net, as far as I can see, is that your network is only as secure as the least secure machine. As you add machines to your network, you have to be careful you are not bringing down your security level.

REVIEW: *A point well made, leaving one big question: What do you see as the future of security management?*

CHANCER: It's a growing field.

REVIEW: *Growing both in numbers and academic attention?*

CHANCER: It's getting a lot more attention because of the current break-ins. Break-ins are making people more aware of the problem, so you are now getting more management commitment toward solving it. We're thus able to hire more people and expand security departments and their authority.

REVIEW: *Do you think that's good?*

> **It goes back to the problem of having too much work to do, which tends to be a normal programmer problem.**

CHANCER: Well, in the long run, yes. A lot of companies are sitting on a powder keg. Take, for example, the news accounts of a small company down in Atlanta, I think, that kept its backups in the same place as its original files. Somebody stole the disks, and there went the company down the drain.

REVIEW: *Recognizing these bet-your-company issues is a matter of importance and perhaps headlines will help us do that.*

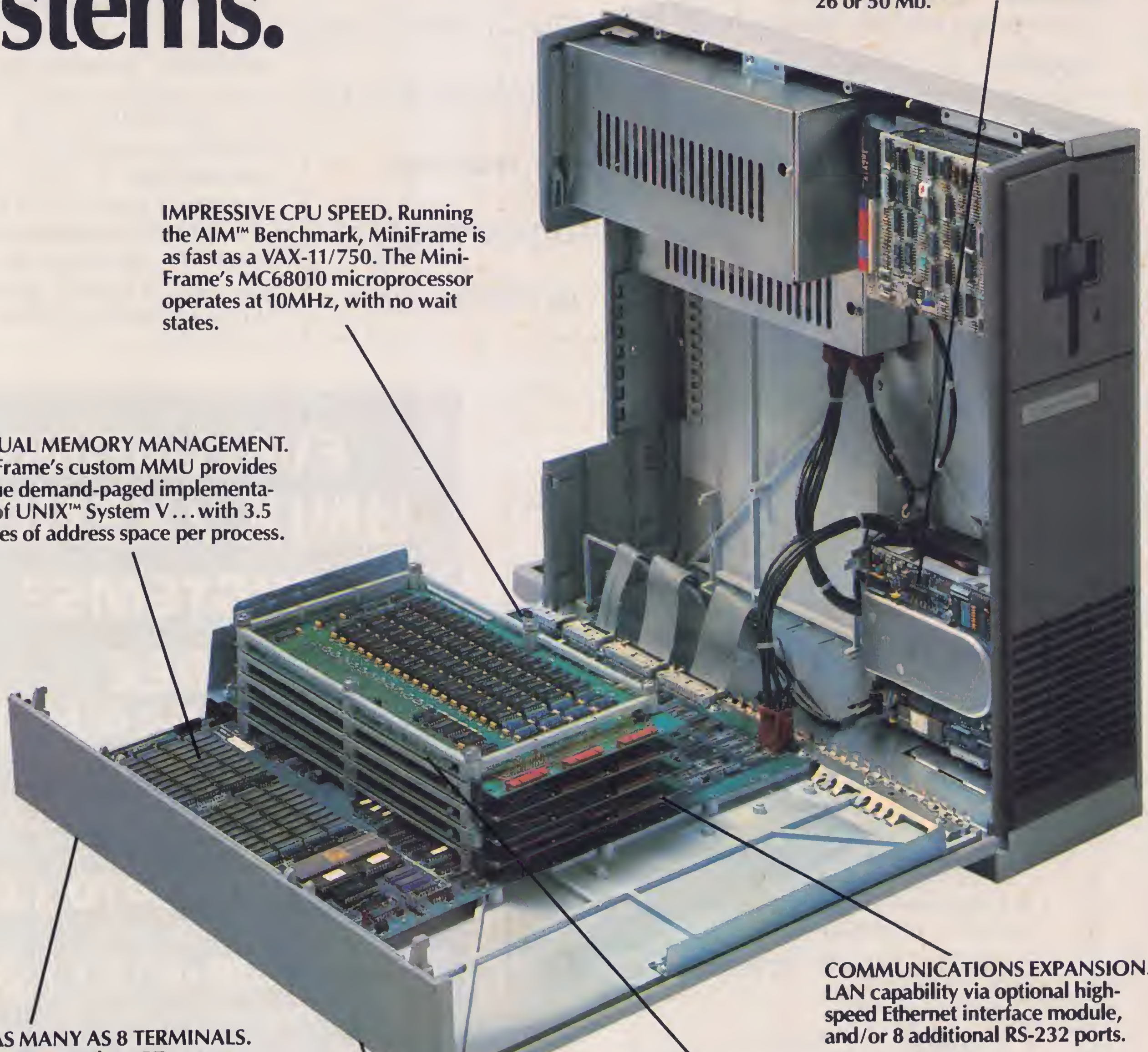
CHANCER: It will help us recognize that there is a problem. The biggest problem may not be what the headlines are pointing out, but at least it gets management thinking about security problems. ■

An open-and-shut case for making MiniFrame™ your first choice in low-cost OEM systems.

ONE TO EIGHT USERS. MiniFrame can be configured for up to 50 Megabytes of integral mass storage: 5¼" Winchester fixed disks of 13, 26 or 50 Mb.

IMPRESSIVE CPU SPEED. Running the AIM™ Benchmark, MiniFrame is as fast as a VAX-11/750. The MiniFrame's MC68010 microprocessor operates at 10MHz, with no wait states.

VIRTUAL MEMORY MANAGEMENT. MiniFrame's custom MMU provides unique demand-paged implementation of UNIX™ System V . . . with 3.5 Mbytes of address space per process.



COMMUNICATIONS EXPANSION. LAN capability via optional high-speed Ethernet interface module, and/or 8 additional RS-232 ports.

RUNS AS MANY AS 8 TERMINALS. Convergent PT and/or GT, or standard ASCII. Multidrop RS-422 communications line operates at 307 Kbit/sec.

ECONOMICAL MEMORY EXPANSION. MiniFrame provides ½ Megabyte of RAM standard; up to three boards can be added for system expansion to 2 Mb.

FOR HIGH-SPEED PRINTERS, a Centronics-compatible parallel printer port is standard in the MiniFrame processor.

MiniFrame: another first from Convergent Technologies. Super-minicomputer power at PC prices.

OEMs can now meet the needs of small to medium-sized organizations for low-cost, high-performance systems capable of handling large UNIX-based applications.

The MiniFrame system—which effectively opens a whole new market for you—is unique not just for its price/performance advantages. It also provides complete flexibility in meeting requirements ranging from single-user "personal UNIX systems" to eight-terminal data-processing installations.

BACKUP STORAGE with integral 5¼", 640 Kbyte floppy disk. Optional high-capacity backup available.

WE'VE CLOSED OUR CASE... now send for a complete information package. Write to Convergent Technologies, Data Systems Division, 3055 Patrick Henry Drive, Santa Clara, CA 95050. Phone: 408/980-0850. Telex: 176-825.

Convergent Technologies achieved this breakthrough by adapting the innovative hardware and software developed for its MegaFrame multiprocessor.

The result is a powerful, compact and expandable unit priced to give you the maximum opportunity of capitalizing on a rapidly growing UNIX market.

There are, of course, many other unique benefits of the MiniFrame system.

Convergent provides foundation software for office applications—including a powerful, Wang keystroke-compatible word processor, an advanced financial spreadsheet and complete electronic mail facility.

The Window Manager permits viewing and manipulating of up to four applications running simultaneously on Convergent's PT or GT terminal screens.

Major performance increases result from utilizing these terminals with the MiniFrame. At only slightly higher cost than standard "dumb tubes", our PT or GT terminals offer high-speed communications plus built-in processor with enough memory to execute key portions of the system code.

Programming languages include industry-standard High Level COBOL and BASIC, full FORTRAN-77, Pascal and C.

OEM prices for the MiniFrame start at less than $5,000; an eight-user MiniFrame can be configured for under $10,000. Prices like these make it an open-and-shut case for choosing the MiniFrame.

Graceful upgrade path to the MegaFrame.™

The MegaFrame, Convergent Technologies' revolutionary super-minicomputer system, utilizes multiple processors; has expansion potential to 128 users, 8 MIPS and gigabytes of disk storage. It enables OEMs to handle today's growing demand for computing services without discarding a single piece of hardware ... or being forced into expensive CPU upgrades.

Applications software can be transported—unchanged—from the MiniFrame to the MegaFrame whenever the workload requires it. The two systems are object-code compatible, allowing OEMs to provide a complete family of systems.

MegaFrame: proof that if anyone can build a UNIX system the way it should be built—it's Convergent Technologies.

Convergent Technologies

Where great ideas come together

Circle No. 31 on Inquiry Card

COPING WITH COMPUTER PIRACY

A 20th century legal dilemma

by Michael Donahue

The rain it raineth on the just,
And also on the unjust fella;
But chiefly on the just, because
The unjust steals the just's umbrella.
British legal maxim

All the world hates a data diddler. My friend, Raoul, is something of a fanatic about it. "Bullwhack 'em," he is fond of saying. "If that doesn't work, try dismemberment."

Raoul began collecting instruments of torture and hanging them in his office shortly after his first brush with computer crime in 1981. "These are thumb screws," he told me. He bought them the same day he found that the datafile for his mail order business had been zapped. In the place of thousands of names, addresses and purchase orders for the preceding 30 days, he discovered the message: "Ha ha." Raoul was not amused.

As crimes go, the unpermitted zapping of Raoul's file falls somewhere on the lower end of the heinousness scale. No blood was spilled; no money was taken.

Fortunately, Raoul is a careful man. He took his backup out of his fireproof cabinet and was open for business within 15 minutes. But Raoul broods. "What if I hadn't made a backup?" he asked. "I would have to explain to my ex-wife that my cash flow had been interrupted by some turkey with a TRS-80 and that I wouldn't be able to pay my alimony. Her lawyer would send me a pay-or-die notice and some black robed featherbedder would surely sing me a chorus of "Up a Lazy River" while the guys in the blue suits

dragged me out of the courtroom, and..." Raoul is nothing if not dramatic.

While my friend, the victim, collected more horsewhips and clubs than a well-prepared sadomasochist, the perpetrator struck again. "We got him this time," Raoul chuckled. The dynamic duo — the minions of the law and Ma Bell — captured, counseled and released a 14-year-old beardless hacker in Coos Bay who had found Raoul's 800 number on a users' group bulletin board and was just having a little fun.

The policeman was shocked at Raoul's suggestion. "My God, man, the boy is only 14 years old. Besides, what real harm has he done?"

Raoul thus discovered that flagellation had gone out of vogue, at least for computer abusers. "Well, at least you could put him in jail," he protested. The officer explained that although he and the deputy prosecuting attorney assigned to the case had a visceral feeling that some crime must have been committed, the prosecutor was having a little difficulty finding a section in the criminal code that fit the situation.

At that time, 1981, there were only 11 states enlightened enough to enact legislation dealing with computer crime. Technology had again outstripped law. In most jurisdictions prosecutors attempted to meet the demands of change by engrafting criminal acts committed on or by machines, unheard of before the second half of the 20th century, onto criminal statues rooted in the ancient common law of England.

"Somehow," the policeman told Raoul, "I don't see a jury convicting this boy of grand larceny or breaking and entering—or even mopery, hanging around with a long face, or whatever." Raoul could see the point, but he was still frustrated.

The problem is the complexity of means by which computer crime can be committed. The law had no trouble recognizing a crime committed on a computer with an axe. But it failed to discern that a more serious criminal act might take place with a few keystrokes thousands of miles away from the intended victim. Computer crime could be committed with malice aforethought or with a pure heart, an empty head and an overabundance of youthful curiosity.

Stumbling blocks along the tortuous path to justice were abundant. Investigating officers, hardened by a daily fare of hatchet murders and cat burglaries, couldn't whip up much enthusiasm for tracking down whiz kid file fiddlers. Prosecutors, wrestling with crowded criminal dockets, declined to file criminal complaints or seek indictments. When forced to, they invariably plea bargained down to the crime of attempted spitting on the street. Sentencing judges shied away from jailing computer abusers. They naively clung to the no-room-in-the-inn theory and refused to throw out hugger muggers to make room for computer age criminals.

No blood was spilled; no money was taken.

Raoul had a suggestion. "Chop the little finger of the right hand off at the first knuckle. That way every time they hit a return on the keyboard, they'd remember." We have all been a little worried about Raoul lately.

Federal legislation was inevitable. Everybody but Raoul hoped that our political leaders in Disneyland-on-the-Potomac would eventually come up with less draconian measures.

The media had created a new folk hero: a bespectacled, 14-year-old nebbish with terminal acne and a demoniacal 512K mind, who communicated at 1200 baud and spoke a language called Assembly. The newspapers gave him names like Capt'n Crunch, Asynchronous Attacker and Syscruncher and said he was something called a "system hacker." According to popular legend, he cut his teeth breaking into university systems and moved on to telephone companies, banks and other businesses. Like a modern day Dillinger, he leaped over security barriers with ease, robbing from the rich who made their living robbing from the poor.

A report found its way into the halls of Congress saying that the average computer theft nets the criminal something in excess of the national defense budget of Bolivia. Republicans and Democrats answered the call to arms and played "Can You Top This?" with offers of ever-escalating criminal sanctions.

Raoul wrote pages of sanguinary suggestions to his US Senator during this period. Fortunately, each of his letters was routed by an administrative assistant into the crank file before the senator spoiled his day reading it. Raoul received a form letter thanking him for his interest, assuring him that he was on the right track and asking him to keep those cards and letters coming.

The House Judiciary Committee received the testimony of a nine-year-old reformed hacker. He tearfully testifed that he had started innocently enough with two Campbell Soup cans and 50 feet of kite string connected between his room and his friend's house across the alley. Soon that wasn't enough to satisfy his itch for technology. He wheedled his parents into having his very own rotary dial telephone. Still he wasn't satisfied. He strong-armed younger children out of their lunch money so he could trade up to a touch tone model. He browbeat his father into buying him a used Commodore 64 so that he would have an even chance at matriculating into the fifth grade. He then shoplifted a $99 modem.

From that day forward, he was a prisoner of a technolust that knew no bounds. "The Source, Compuserve, Dialog," he whined. "I steal thousands of dollars in timeshare services. Long distance charges are eating me alive. I support my habit by stealing dimes out of telephone coin returns. You'd be amazed how many people walk away from them."

While his family slept, he spent hours haunting renegade bulletin boards seeking access numbers and codes to break into the telephone company's billing program before the next billing cycle. "My mom says if she gets one more $1800 phone bill, she'll kill me," he explained.

He was eventually brought to the bar of justice. While awaiting trial, he got a moment alone with the booking terminal, accessed JAILTRAK and single-handedly solved the entire jail overcrowding problem in that county. The newspapers said that he and 23 other inmates tunneled their way to freedom using a computer instead of a shovel. "Another

Technology has again outstripped law.

example of better living through electronics,'' the reporter gibed.

By now the furor has pretty much died down. Most states have enacted legislation making certain forms of computer abuse illegal. In some states certain acts have been declared felonies, while in other more *laissez faire* jurisdictions, they're regarded as misdemeanors. If Raoul's nemesis wriggled into the mail order company computer again and was convicted, he would receive up to six months in jail; but if Raoul moved 300 miles north, the sentence would be five years at hard labor.

The federal government keeps trying to pass a comprehensive computer crime bill to resolve these disparities. In early August, the Attorney General of the United States sent a bill to the Speaker of the House which, if passed, would make it a federal misdemeanor to fraudulently or illegally use a computer owned by the US government or a national bank — or any computer, for that matter, when the offense involves interstate commerce. It's about time.

Raoul sold his computer last year and has opened a wax museum. His therapist says he has every hope that in two or three years — or sooner if Congress passes the comprehensive computer crime bill — Raoul will be as good as new.

The average computer theft nets the criminal something in excess of the national defense budget of Bolivia.

Michael Donohue is a black robed featherbedder and writer living in Spokane, WA. ■

Circle No. 59 on Inquiry Card ➤

PDP-11 +AIS/3210 UNIX POWER!

The AIS/3210 gives your PDP-11 the performance of a VAX-11/780 with UNIX. At a fraction of the cost.

The AIS/3210 computer system can be configured as a co-processor utilizing your PDP-11 as the host, or as a stand-alone system with one or more co-processors. And you don't have to change any of your original PDP-11 peripherals. With the AIS/3210, your PDP-11 can have 32-bit processing power, up to 128 MB of physical memory, 16 MB of virtual memory per user, 64-bit floating-point hardware, up to 14 MHz operating speed, co-processor and multiprocessor flexibility, an operating system based on UNIX System V Release 2, and more. Much more.

All for less than 1/20 of the price of a VAX 11/780. Starting at $6,900.* Contact AIS today for complete information.

American Information Systems
1801 Page Mill Road
Palo Alto, California 94304
(415) 494-3210
Telex 755750

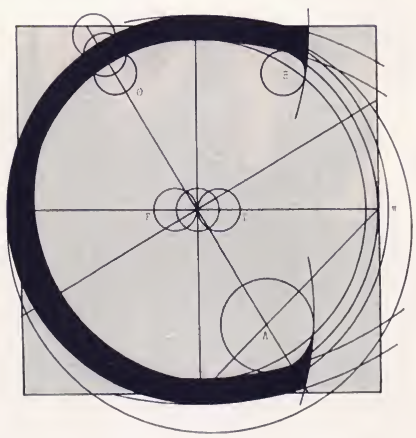
PDP-11 and VAX-11 are trademarks of Digital Equipment Corporation. UNIX is a trademark of AT&T Bell Laboratories. AIS/3210 is a trademark of American Information Systems Corporation. *Price subject to change without notice.

AMERICAN INFORMATION SYSTEMS

C ADVISOR

Program optimization

by Bill Tuthill



One reason for acquiring an expensive computer is that it runs faster than a cheap computer. In fact, an expensive computer may even turn out to be cheaper in the long run.

Suppose, for example, that two programmers each earn $30,000 a year. One works on an IBM PC running UNIX that cost $7500, while the other works on a 68000 UNIX system that cost $15,000. The 68000 machine is about twice as fast as the IBM PC, so the second programmer can be about twice as productive as the first. Editing will be easier, compiles won't take as long and program testing will be accomplished more quickly.

Amortizing the hardware over two years, a one-year project on the IBM PC would cost $33,750, while the same half-year project on the 68000 would cost only $22,500. These figures, of course, do not reflect the market advantage of bringing out a product half a year earlier.

The higher the labor costs, the greater the motivation to obtain large computers. The irony is that it takes extra labor to make software run fast. Since users expect software on large machines to run fast, programmers often work harder and longer when dealing with fast machines.

How can you make software run fast? Coding in C helps because C is one of the most efficient high-level languages available. This article discusses various strategies for C program optimization. Some of the techniques explored could be applied on small computers. Others, however, such as reliance on registers and main memory, would be effective only on large computers.

INVOKE THE OPTIMIZER

One of the easiest things you can do to make programs run faster is to compile them with an optimizer. Optimizers can do such things as take invariant computations out of loops, pre-evaluate common subexpressions, eliminate redundant register handling and convert strong machine instructions to weak ones (strong instructions are more expensive). Some optimizers are better at this than others, and some actually introduce errors by making ill-advised optimizations. If you can trust your optimizer, learn to rely on it:

```
% cc -O program.c -o program
```

The -O (upper case O) flag invokes the optimizer after compilation. The -o (lower case o) flag places the executable code in the file specified by the argument it precedes.

Working programs, even if slow, are always better than fast programs that don't work. So it's best to invoke the optimizer only after you've finished writing and debugging a program. The optimizer takes some time to do its work, and you don't want to wait for it each time you recompile after making a few changes. Also, since some optimizing C compilers generate incorrect code, it's best to complete your test suite both before and after optimization.

As Kernighan and Plauger demonstrate in *The Elements of Programming Style*, you should let the optimizer do the dirty work. By attempting to hand-optimize your code, you often make it harder to understand and maintain, and you may defeat the efforts of a good optimizing compiler. Kernighan and

```
C       before so-called optimization
        DO 4 J = 1, 1000
    4   X(J) = J
C       after so-called optimization
        Z = 0.0
        DO 4 J = 1, 1000
        Z = Z + 1.0
    4   X(J) = Z
```

Figure 1 — *A FORTRAN "optimization" that actually generates slower code.*

Plauger give the program listed in Figure 1 as an example of a recommended FORTRAN "optimization," one that actually generates slower code.

This "optimization" was made to avoid a thousand conversions to floating point. In fact, most FORTRAN compilers generate more expensive instructions for the second loop because of a thousand floating point additions, so the code actually runs slower as a result of the optimization. The moral: until you have evidence to the contrary, trust the compiler and let it optimize for you.

USE POINTERS INSTEAD OF ARRAYS

Arrays involve more computing than pointers because an address has to be computed for each index reference into the array. When you declare **str** as a character array and use **str[12]** in an expression, the compiler takes the address of the beginning of **str**, and adds 12 bytes to it. This becomes inefficient if you need to walk through an entire array, because the addition must be done for each array element. But if you declare **ptr** as a character pointer and initialize it to point at **str**, then you can walk through the array by incrementing **ptr** with a single machine instruction. Hopefully you have already learned how to make good use of pointers (this often takes a year of experience). If not, you can still speed up your programs by changing array indexes into pointers. The two functions listed in Figure 2 are identical in effect, but the second is about a third faster and doesn't require the declaration of the automatic variable **i**.

```
#include <ctype.h>
#ifdef USDL
# define tolower _tolower
#endif

map_slower(s)       /* map string s to lower case */
char s[];
{
        int i;

        for (i = 0; s[i]; i++)
                if (isupper(s[i]))
                        s[i] = tolower(s[i]);
}

map_lower(s)        /* map string s to lower case */
char *s;
{
        for ( ; *s; s++)
                if (isupper(*s))
                        *s = tolower(*s);
}
```

Figure 2 — *Two functions that demonstrate the value of pointers.*

For readers who are new to pointers in C, it would be good to read the second function aloud, saying "string pointer" in place of "s", and "what s points to" in place of "*s". The combination "char *s" is best read backwards — "s is a pointer to a character." The controlling **for** statement could also be written as follows:

```
for ( ; *s != NULL; s++)
```

The test for NULL is unnecessary however, because the expression "*s" is true until it is zero (or NULL). In fact both versions of the **for** statement generate exactly the same assembler instructions on many machines, including the PDP-11 and VAX.

DECLARE REGISTERS IN ORDER

When declaring an automatic variable, you can give a hint to the compiler to place it in a register. (External and static variables cannot be placed in registers.) Most C compilers will do this if possible, but if there are no registers left, they will ignore your hint. Thus, it behooves you to declare the most critical register variables first. Small machines have only a few registers available in a function, whereas large machines, particularly register-intensive ones (such as the Pyramid), have more available than you're likely to use.

The function listed in Figure 3 prints a bitmap, in **vfont(5)** format (developed at the University of Toronto and included in Berkeley UNIX), on the screen. Notice how the register variables are declared in order, from the one controlling the innermost loop to the one controlling the outermost loop. This way, if the compiler runs out of available registers, we are assured that registers will be used where they are most effective.

The line commented "bit on" may be somewhat opaque. What's happening is that one (1) is left-shifted bit positions. Since left-shifted quantities are always zero-filled, we end up with a 1 in the appropriate bit position, starting at 8 and going to 1 as the loop progresses. This quantity is then bitwise

```
#include <vfont.h>

bprint(n, d, bitmap)            /* print bitmap of glyph n - vfont(5) */
int n;
struct dispatch *d;
char *bitmap;
{
        register int bit;
        register char *cp;
        register int byte, row;

        if (d->nbytes == 0)     /* no glyph in this position */
                return;
        printf("glyph %d: %d bytes, l %d r %d u %d d %d:\n",
                n, d->nbytes, d->left, d->right, d->up, d->down);
        cp = bitmap;
        for (row = 0; row < d->up + d->down; row++) {
                for (byte = 0; byte < (d->left + d->right + 7) / 8; byte++) {
                        for (bit = 7; bit >= 0; bit--)
                                if (*cp & (1 << bit))   /* bit on */
                                        putchar('*');
                                else                    /* bit off */
                                        putchar(' ');
                        cp++;
                }
                putchar('\n');
        }
}
```

Figure 3 —*An example of declaring registers in order.*

AND-ed with ***cp**, which is a pointer into the bitmap. If the bitmap has that bit position turned on, the resulting expression will be nonzero, causing an asterisk to be printed. If, on the other hand, the bitmap has that position turned off, the expression will be zero, causing a space to be printed. Since we are left-shifting, rather than right-shifting, and since we are using only the least significant eight bits of an integer, sign extension will never cause portability problems. Some machines, such as the PDP-11 and the VAX, preserve the negative sign of negative characters converted to integers.

EMPLOY THE PROFILER

The strategies described above are easily applied while you are writing code. But if you get code from someone else, or if you're trying to improve code you wrote long ago, you'll certainly benefit from using a profiler. Never assume you know the relative efficiency of various constructions until you have compared them using a profiler. You may be surprised to learn that some are actually less efficient than you believed. For example, on most machines, this construction:

```
variable ? variable = 0 : variable++ ;
```

generates several more lines of assembly code, and runs a bit slower, than the following construction, which many programmers find easier to read:

```
if (variable)
        variable = 0;
else
        variable++;
```

Most programmers expect that when there are fewer lines of source code, the resulting machine code will be more compact. This is not always true. The question/colon operator is recommended only when you're trying to hide something unimportant from others:

```
printf("%d error%s\n", errors, \
(errors > 1) ? "s" : "");
```

This will print "1 error" if you have only one, but will pluralize, as in "20 errors", whenever you have more. Such detail may not be very important to someone reading your code for the first time, so it doesn't hurt to hide it.

When you're ready to measure your program, compile with the **-p** flag, run your program with some typical data, then use **prof** to generate a calling sequence profile of the sort shown in Figure 4.

The listing provided by the profiler can help you eliminate extraneous function calls. In particular, look out for functions that are called for each

FUSION™ is the only connection.

In an industry fragmented by diversity, FUSION makes connections.

FUSION is the LAN software that links different operating systems, diverse LAN hardware, and diverse protocols, to give you high speed communication across completely unrelated systems.

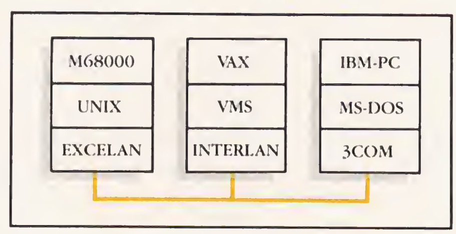
FUSION frees you to buy the best computer for the job—without worrying about compatibility. Your VAX mainframe can now communicate with your IBM-PC. You can even add M68000 work stations or a communications server. FUSION links them all—and more. Regardless of operating system, protocol, or network hardware.

FUSION™

- With FUSION, any user on any system can perform file transfer, remote login, and remote execution.
- FUSION offers a choice of two standard protocols—XNS or TCP/IP. Some customers have even developed private protocols.
- And you can create your own application programs, using FUSION utilities.
- FUSION accommodates all major LAN boards, too. So network interface is no obstacle.

When you add FUSION, you never lose your hardware investment. In fact, you get more out of it than ever.

Ask for FUSION. It's the connection you've been looking for.



A sample network using a few of the many configurations made possible with FUSION.

For further information please contact:
Network Research Corporation
1101 Colorado Ave.
Santa Monica, CA 90401
(213) 394-7200
(800) 541-9508

FUSION is a licensed trademark of Network Research Corporation.

network research corporation

Circle No. 35 on Inquiry Card

```
% cc -p prog.c -o prog
% prog < data
% prof prog
 %time  cumsecs  #call  ms/call  name
  67.7     1.85    256     7.23  _bprint
  21.3     2.43    114     5.12  __doprnt
   3.0     2.52      3    27.79  _read
   3.0     2.60     29     2.87  _write
   2.4     2.67      1    66.68  _main
   0.6     2.68     30     0.56  __flsbuf
   0.6     2.70      1    16.67  _lseek
```

Figure 4 — *Generating a calling sequence profile.*

character of input or output. There are none in Figure 4. Recently, I modified a program that used the **write( )** system call for each character of output, changing it to use the library macro **putc( )** instead. This speeded up the program by about a third. The standard I/O library defines **putc( )** as a macro that calls **_flsbuf( )** only when the output buffer is full. A profiler listing can also help you identify the functions where your program spends most of its time. The example in Figure 4 shows that **bprint( )** accounts for almost 68 percent of the program's processing time, so we should approach this function first when trying to speed up the program.

CHANGE FUNCTION CALLS INTO MACROS

Another thing to watch out for is code containing numerous trivial functions. This is often a good way to organize code, but it can greatly impede efficiency. Calling a subroutine always takes time, which will vary depending on the architecture of your computer. The example listed in Figure 5 shows two procedures that are too trivial to deserve status as functions.

It would be much better to turn these functions into macros. This would permit the code to be as legible as before, and would not require changing every line where these functions are called. Note how we don't need to declare the argument in a macro definition because the preprocessor makes simple string substitutions, without knowledge of the data type. Note, too, that we don't need a semicolon since this is supplied when the programmer calls one of the pseudo-routines. Finally, note how we parenthesize the substituted argument, in case it contains a complicated expression:

```
#define setsize(n) size = (n)
#define setfont(n) font = ftlist[(n)]
```

Functions should accomplish a single, nontrivial task that must be done repetitively. That way, they won't obscure your code with blocks of monotonous, repeated statements. As an added bonus, they will save program text space because a concentrated set of instructions can be called from more than one location. On the other hand, if functions accomplish only trivial tasks, they detract from your program's efficiency. It's best not to write code containing trivial functions. But if you're modifying someone else's code that contains trivial functions, turn them into macros.

```
extern int size, font;

setsize(n)
int n;
{
        size = n;
}

setfont(n)
int n;
{
        font = ftlist[n];
}
```

Figure 5 — *Functions that should be turned into macros.*

SUBSTITUTE MAIN MEMORY FOR TEMPORARY FILES

Main memory is much faster than disk, though it is more expensive per unit of storage. Just a few years ago, most UNIX programming was done on 16-bit machines, such as the PDP-11. This meant that programs could not use more than 64K bytes of memory, or 128K bytes with separate instructions and data. With the advent of inexpensive 32-bit chips, such as the MC68010, the NS32032 and the WE32000, most programmers are now able to work with large address spaces. Programs can use a megabyte of memory or more. This makes writing programs easier and also permits software to run faster.

Years ago, some people at Berkeley noticed that one reason **troff** ran slowly was because it was I/O-bound. Since it kept macros, registers and diversions in a temporary file, it was forever reading from or writing to that file. Because the VAX had lots of memory, particularly after Berkeley added paging and virtual memory, it was possible to hold all this data in main memory. This change alone cut **troff**'s execution time in half, without consuming more than a quarter megabyte of memory.

By convention, C programs use the name VMUNIX to indicate that software needs lots of

```
#define TEMPSIZE 512000
        char *tempspace;
#ifdef VMUNIX
        tempspace = malloc((unsigned)TEMPSIZE);
#else
        int tfd;
        tempspace = mktemp("/tmp/tfXXXXX");
        if ((tfd = creat(tempspace, 0777)) == -1)
                perror(tempspace), exit(1);
#endif
```

Figure 6 — *Using main memory instead of temporary files.*

memory. In the example listed in Figure 6, a temporary file is created only if memory is extremely limited.

In the case of VMUNIX, all the programmer has to do is write to or read from the appropriate location in memory. Otherwise, it is necessary to move to the appropriate location in the file using the **lseek( )** system call before writing and reading with the **write( )** or **read( )** system calls. These system calls cause needless overhead that can be eliminated on VMUNIX systems.

This and the other methods suggested here may help you write faster software, but nothing matters so much as a good algorithm. Binary searches are faster than linear searches, and trees are usually superior to flat lists. Quick sorts are faster than Shell sorts, which are faster than bubble sorts. Once you have identified the trouble spots within your profiler, you should work on improving your algorithm.

Bill Tuthill is a member of the technical staff at Sun Microsystems. He was formerly a leading UNIX and C consultant at UC Berkeley, where he contributed software to BSD 4.2. ■

INDUSTRY INSIDER

Computerrific elephants and soapy peanuts

by Mark G. Sobell

The clash of the titans is on. Or is it the dancing of the elephants? Mixed metaphors perhaps, but both are equally apt as descriptions of a raging market battle that is only going to get hotter. IBM and one of the master merchandisers in the mass consumer market, Proctor & Gamble, have joined forces to sell the PCjr, also known as the "peanut," side by side with Tide detergent. It is a bellwether of things to come.

As AT&T and IBM battle for shares in the consumer computer market, IBM has teamed up with Procter & Gamble for a "Computerrific Sweepstakes" mailing that will reportedly be sent to 60 million households. The mailing includes $3.30 worth of coupons for products such as Prell and Tide, an offer for a certificate worth $100 toward the purchase of a PCjr (in exchange for 25 Procter & Gamble proof-of-purchase seals), and a sweepstakes number that gives you a chance at one of 250 IBM PCs and 750 PCjrs. The theme of the mailing is: "Shouldn't your first computer be from IBM?"

In addition to the sweepstakes, IBM is going after the consumer market with IBM-sponsored credit cards and the most intensive PC campaign in the company's history — featuring the recently released PC/AT. August was the single biggest TV advertising month ever for IBM.

At their Dallas introduction of PC/AT, IBM projected that 16 million American households would have computers by the end of 1985 and that of those households, four million would upgrade to larger machines. IBM would like for many of those upgrades to be to IBM machines.

POPCORN

IBM announced its sweepstakes in August concurrent with its release of PC/AT (code named "popcorn"). The PC/AT derives its official name from "Personal Computer/Advanced Technology" and is based on the Intel 80286 microprocessor, making it several times faster than the original PC based on the 8086 chip. The PC/AT optionally comes with a 20 MB hard disk and, under its PC-DOS operating system, can run existing PC software in an 8086-compatible real address mode, making it a natural next-step-up for PC users.

IBM promises a PC/IX operating system release on the PC/AT later this year and a System III derived version of Microsoft's Xenix to be released in the first quarter of next year.

IBM advertises the Xenix machine as being capable of supporting a maximum of three terminals. But the PC/AT should be in the same performance ballpark as AT&T's 3B2, which is purported to be capable of supporting 16 users. Although the 3B2's performance might get sluggish with 16 people cranking away on it, even half that many users is still substantially more than the three people PC/AT is said to support. Based on this comparison, it would appear that IBM has deliberately limited the capacity of the PC/AT. Why?

Because of the market it's going after, there is some speculation that IBM didn't want to scare off low-end users with high-end specs. Another possible explanation of the three-user limit is that IBM is preparing to introduce a machine in the high-end multiuser microprocessor market (to compete with AT&T's 3B5). It's also possible that IBM may have imposed the limit to encourage

sales of the local area network (LAN) it will make available in October. The LAN allows up to 1000 desktop computers, located in one building or group of buildings, to share data and peripheral devices, such as printers and disk drives. The LAN will be accessed through a $695 PC network adapter board manufactured by Sytek, Inc.

Whatever the reason for the three-user limit, you can be sure the first non-IBM add-on will be a board that increases the number of terminals you can use on the system. You can also be certain that the PC/AT will prove to be an important entry in the UNIX arena. With a price tag between $4000 and $6000, IBM has taken a very aggressive pricing stance on what will certainly prove to be a competitive machine.

YOU, ME AND EVERYONE ELSE...

What about everyone else not ensconced in the AT&T or IBM camps? In a phrase: expect rough sledding. Just after the IBM announcement, Fortune Systems loaned North Star Computer almost $4 million as a sign of the importance it places on the merger talks going on between the two. Fortune, manufacturers of a mid-sized UNIX system, still has about $40 million left from its initial public offering last March, but has no machine to compete in the PC-DOS market. North Star, with three PC-DOS machines in its corner, needs money and a machine to sell to the UNIX market.

Just as GM joined the forces of Cadillac, Chevrolet, et al. to produce a complete line of cars, Fortune and North Star are attempting to protect themselves from the marketing campaigns of IBM and AT&T by joining forces to produce a more complete line of computers. One sales force offering a broader line should be able to sell more machines more efficiently than two sales forces selling narrower, separate product lines. This, at least, is the theory behind one strategy that may keep manufacturers alive in the face of the onslaught.

IBM has taken a very aggressive pricing stance.

Another strategy is to go up and out of harm's way — offering the consumer a high-performance machine with a complete selection of options. Cromemco's new high-end UNIX entry demonstrates this tactic: a fast machine that is expandable to 16 MB of RAM and 600 MB of hard disk with a complete high-resolution graphics package. The machine provides error-correcting RAM, cache memory on the disk controller and an efficient memory management board that supports scatter loading and fast context switching.

CONCLUSION

There's no doubt that the elephants are dancing. Attempts to convince consumers of the value of a variety of products will be in increasing evidence. AT&T is building its corporate image ("Watson, watch us now!") through advertising from all its divisions. IBM is going after the low-end market in a big way (watch for a 12-page insert in Reader's Digest designed to "prepare" the market with the theme: "Don't be scared, you already use computers in your daily life").

The little guys are scattering, taking up strongholds as each sees fit. The market is being reorganized from above. The shakeout is reality.

If you have an item appropriate for this column, you can contact Mr. Sobell by electronic mail at ***ucbvax!Shasta!olympus!its!mgs*** *or by US mail at UNIX REVIEW, 520 Waller Street, 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.

Everything You Need To Know About UNIX . . .

UNIX EXPO™

The Unix Operating System Exposition & Conference

October 16, 17, 18, 1984

Marina Expo Complex —

★ Exposition
★ Conference
★ Job Fair

Take advantage of all that UNIX EXPO has to offer, all in one location.

UNIX EXPO

The comprehensive, practical business/learning event designed solely and specifically to address the myriad business and technical aspects of the UNIX OPERATING SYSTEM. UNIX EXPO is *the* national trade show that will bring ISO's, sophisticated end-users, technical personnel, OEM's, software dealers, and other resellers face-to-face with the leading suppliers to the industry at the exposition, and the leading UNIX authorities at the conference program. By attending this three day forum, you can be prepared to position yourself at the vanguard of the UNIX revolution.

Inspect - Compare - Question - Select

all of the UNIX products and services on display at the 350 booth exposition. The nation's leading suppliers of UNIX and UNIX-like hardware, software, peripherals and services are anxious to talk business with you.

Learn To Earn at the Conference Program

A penetrating, multi-track slate of seminars focusing on *the* most vital technical and business areas of UNIX has been developed by noted UNIX advocate, James Joyce, President, International Technical Seminars. Attending the conference will help you achieve a full understanding of what is destined to be the major computer operating system for the coming decades.

Meet the Leaders in the Expanding UNIX Universe

For three days in October, New York City, the heart of the largest computer marketplace in the world, will become the core of the UNIX universe; creating an unparalleled opportunity for you to meet and exchange ideas, theories and information with your colleagues.

Expand Your Horizons At the Job Fair

PENCOM SYSTEMS, the national recognized leader in UNIX recruiting will host a special JOB FAIR at UNIX EXPO where exhibiting firms will disseminate information regarding employment opportunities. Your career objectives can be discussed, and meetings with company representatives scheduled.

UNIX EXPO

Simply complete this form and drop it in the mail with payment of $5.00 for each registration. (Registration at the door will be $10.00.)

Name ______________________

Title ______________________

Address ______________________

City ____________ State ____________ Zip ____________

☐ Please send complete seminar program information
☐ Please send discount hotel reservation form

Return to:
National Expositions Co., Inc.
14 W. 40 St.
N.Y., N.Y. 10018

Sponsored by Unigroup of New York.

SAVE TIME! SAVE MONEY! AVOID LINES!
REGISTER FOR UNIX EXPO TODAY

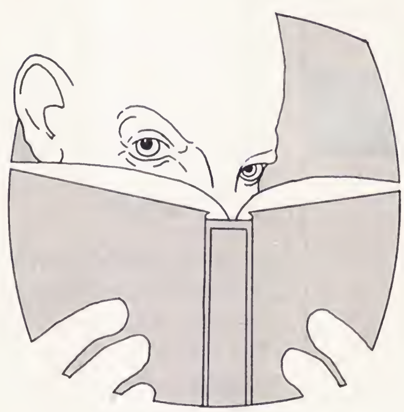
Circle No. 37 on Inquiry Card

/usr/lib

A mixed bag

by Jim Joyce

The C Programming Tutor



A mixed review of a book is one of the hardest to write, but that's exactly what's necessary for Leon A. Wortman and Thomas O. Sidebottom's *The C Programming Tutor* (Robert J. Brady Co., 1984, 274pp, $17.95). Their first example, given as Figure 1, may perhaps illustrate why the review must be mixed.

Starting off a tutorial on C with **#include "stdio.h"** seems excitingly ambitious, and is obviously aimed at instilling the practice of including standard files early on. Well and good, but nothing is *done* with **stdio.h** here, and so it clutters up the code unnecessarily. Perhaps the **#include** is required for the program to be universally runnable on the variety of systems they discuss in the book — CP/M-80, CP/M-86, MS-DOS and UNIX. But Wortman and Sidebottom do not say, and the reader is thus misled into thinking that the **#include** IS required, which — strictly speaking — it is not.

Perhaps more objectionable is the lack of a newline (**\n**) at the end of the control string, which on all UNIX systems I know about will result in the shell prompt being jammed up against the exclamation mark, as illustrated in the last line of Figure 1.

The comments on the braces for **main** and other functions are a clever touch, but I would have preferred blank lines between function definitions to break up the density of the code.

There is a distinct Pascal flavor to the examples, especially in variable names such as **Result**, **RawResponse** and so on. This is the way names are written in Pascal, but not, by convention, in C. They are valid names to be sure, but not the best choice of style. More alarming is the repeated statement that the underscore is a valid character for a variable name, without a single warning that variables in **stdio.h** and system function references in UNIX are named with a leading underscore character in order to avoid conflict with user-defined names.

Wortman and Sidebottom *do* offer sage advice regarding interactive data-handling:

> *Validate the response. You cannot depend on the user of the program to enter proper responses. Therefore, the program must check the correctness of the user's entry. (p. 13)*

The authors then make their point in a stepwise refinement of response-handling code.

Yet, misleading statements detract from the intelligence of their presentation. The illustration on page 3 of the steps of a C compilation lumps the preprocessor and the compiler together, which is quite an error in thinking. The authors then confuse standard **#include** files as part of the language. It is true that if the implementation of C is to be like that of C under UNIX, files such as **stdio.h** and **ctypes.h** need to have their usual contents, and it's also true that **printf**, **fprintf**, **scanf** and **fscanf** need to be defined, but none of these are part of the language.

Pointers are often the stumbling place for people learning C, and a book on C should do whatever it can to make the subject clear. This book tries but gets too involved with its "anecdotal" approach and, if anything, simply adds fuel to the fiery fear most programmers have about pointers.

The second part of the book is devoted to "Useful

```
$ cat fig.1.c
#include "stdio.h"

main()
{           /* main */
    printf("This is the start of something wonderful!") ;
}           /* main */
$ cc fig.1.c; a.out
This is the start of something wonderful!$
```

Figure 1 — *Some cluttered code from "The C Programming Tutor"*

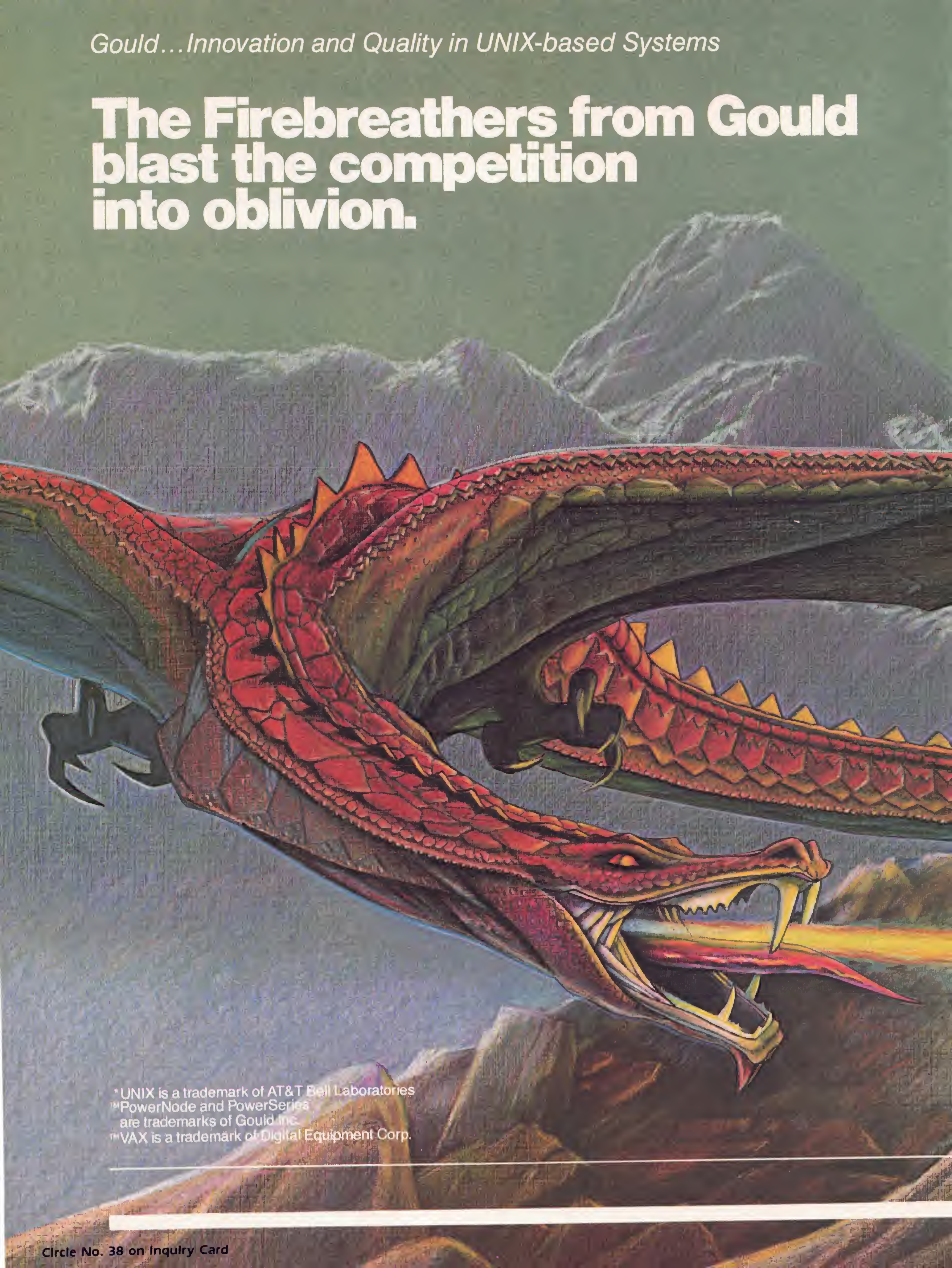
Programs," and on the whole most of the programs are. But by the time I had reached the second part of the book, I had come across enough misleading statements in the first part that there was little incentive to continue.

The Table of Contents is listed below:

Table of Contents for The C Programming Tutor

The C Programmer's Handbook

This gem is a Bell Laboratories production by M.I. Bolsky and P.G. Matthews at the Systems Training Center in Piscataway, NJ (Bell Telephone Laboratories, 1984, 72pp). It is a quick reference handbook for experienced programmers who simply


Gould…Innovation and Quality in UNIX-based Systems
The Firebreathers from Gould blast the competition into oblivion.
*UNIX is a trademark of AT&T Bell Laboratories
™PowerNode and PowerSeries are trademarks of Gould Inc.
™VAX is a trademark of Digital Equipment Corp.
Circle No. 38 on Inquiry Card

With blazing performance.

Great leaps in raw performance are rare in the computer world. Usually, changes occur in increments of half-a-MIP or so. Now real Firebreathers roar into the arena, quadrupling the best the competition has to offer.

With scorching speed.

These creatures don't run a little faster. Running real production code supplied by VAX™ users, our PowerNode™ 9000 benchmarks at 4.5 times faster than the VAX 11/780...at a comparable price! Even our second-in-line PowerNode 6000 runs 1.5 times faster. But with all this power and speed, the PN6000 has a dainty footprint...60% less than the VAX 11/780.

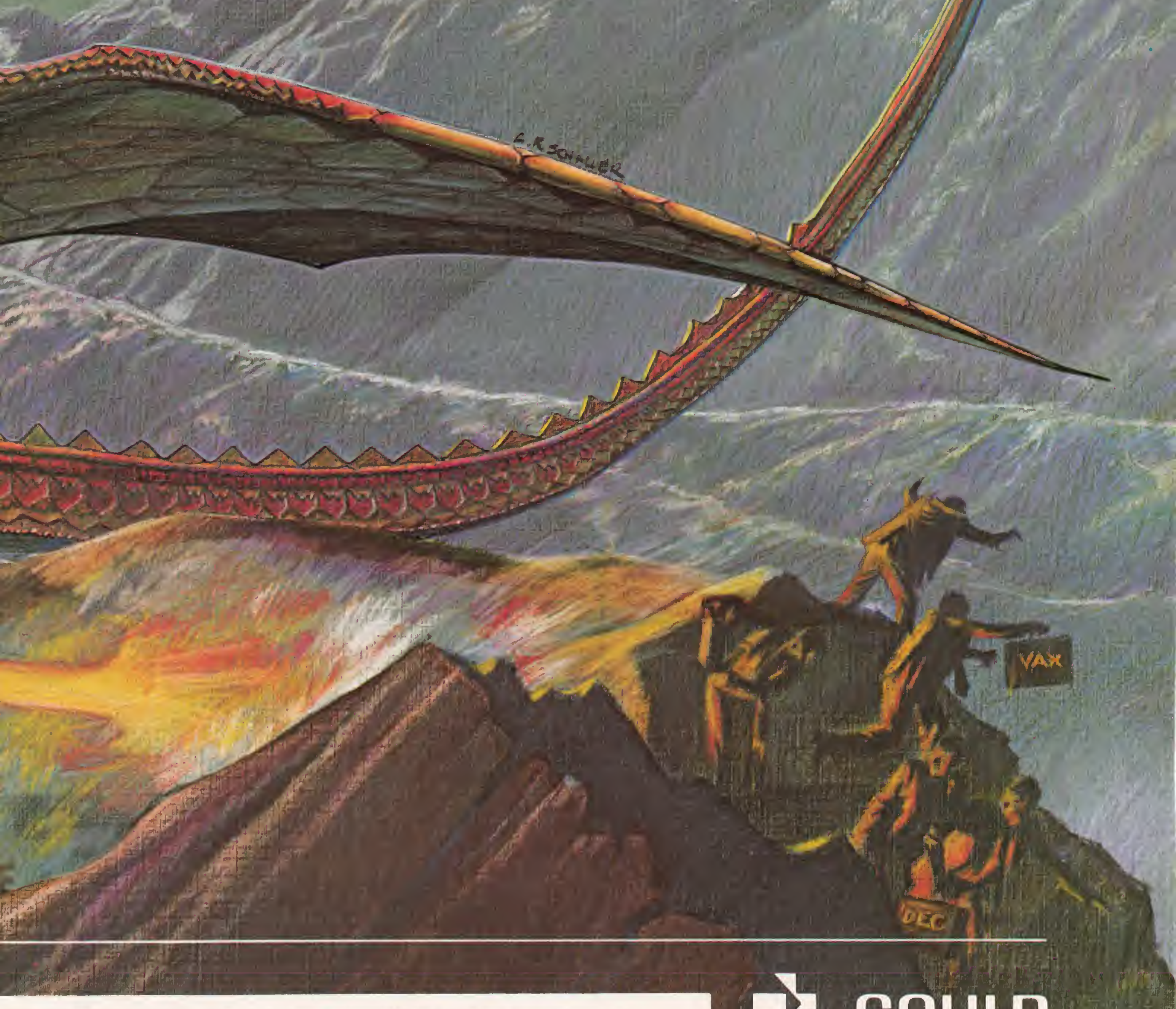
With sizzling simplicity.

Even the hottest firebreather won't fly if it's hard to run. Ours are UNIX*-based, so ease of use and compatibility become part of the power. Plus we offer the "Compatibility Suite", application software packages which are consistent across our entire Gould PowerSeries™ product line...the widest range of UNIX-based systems in the world.

More than a myth.

We know these claims sound outrageous. Make us prove them. Give us your benchmarks and production code. We'll show you numbers that surpass all expectations. Call 305-797-5459 for a test flight on the Firebreathers. Mainframe performance disguised as a supermini. The scorching performance you expect from Gould.

Gould Inc.,
Computer Systems Division
Distributed Systems Operation
6901 West Sunrise Boulevard
Ft. Lauderdale, FL 33313



GOULD
Electronics

need to check a point or two about the language. It is *not* a tutorial, despite some of the comments below.

The use of red for the language element to be discussed and black for the discussion helps users scanning for quick answers. The handbook's strength lies in its examples, although regrettably these are not always in the form of complete, working examples.

The lack of working examples, though, is not crucial here because the information is offered to jog the memory rather than to teach a concept. Each control structure, for example, has bullet-marked points summarizing its purpose followed by one or more examples and some notes about the control structure.

The discussion on page 11 about the increment (++) and decrement (--) operators is pleasantly terse and yet covers the topic well. The note at the top of the page is also charming:

> ***Note*** *— Side effects occur for the + + and - - operators. That is, the value of the variable which appears as the operand is changed.*

Regrettably, such a note is not printed on nearly every page though C is *full* of side effects, and much of its efficiency *relies* on those side effects.

The authors have even included an extended example of a program in C in the form of **bblsort.h** and **bblsort.c**, which together perform a bubble **sort** of

How can anyone outside of Bell Laboratories get a copy?

data. Although not as wonderful as examples included in Kernighan and Pike's *The UNIX Programming Environment* (*UNIX REVIEW*, Feb., 1984) the example does illustrate important programming style concepts.

This spiral-bound paperback is destined to become dog-eared and grimy for all C programmers who can get a copy. The section on identifier length on page 3 is worth the price of the handbook to anyone concerned about portability issues. The problem is: how can anyone outside of Bell Laboratories get a copy?

Dave Chandler of the Independent UNIX Bookstore placed a call to one of the authors to learn that the book is to be released outside Bell Laboratories "soon." Interested readers might wish to contact M.I. Bolsky, AT&T Bell Laboratories, 8 Corporate Place, Piscataway, NJ 08854, to relay comments that can be forwarded to management. The Table of Contents is listed below:

Table of Contents for The C Programmer's Handbook

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 questions about books, call 415/621-1593.

RULES OF THE GAME

What can you take when you clean out your desk?

by Glenn Groenewold

Jack Megabyte has been an employee of Goliath Corporation almost since its inception. During the past several years he's directed systems development for the company. But now, together with Helga Termcap, a consultant who's a whiz at system design, Jack is starting a new business. Jack and Helga believe they can come up with an operating system capable of outperforming Goliath's while requiring less computer capacity.

Shortly after giving Goliath notice of his resignation, Jack receives a letter from the company's attorneys notifying him that his new activities constitute a breach of his employment contract with Goliath. Goliath also tells him that it considers itself the owner of any systems he has devised while in its employ, and demands that he cease his activities in starting the new company. The letter concludes with the announcement that Goliath will file suit to obtain an injunction and damages against him if he does not comply.

Naturally, Jack becomes upset. He has a vague recollection of signing a document when he started with Goliath, but has not thought about it since. During his tenure with the company, at least a dozen other key employees have started their own companies without anybody raising a fuss.

Jack may be unaware of it, but he's run head-on into a change in attitude that has begun to surface in the computer industry. Major corporations have started filing suits against former employees — and, in some instances, even against the *lawyers* who represent them — in an effort to stem what they view as unfair competition from new enterprises.

In the face of this, what can Jack Megabyte do?

It would be a good idea for Jack and Helga to talk to a lawyer, of course, but before they do, they'll need to collect the information the attorney will want. One document the attorney is sure to ask for is any agreement Jack has signed at Goliath. Obviously Jack's task would be easier if he had had the foresight to keep a copy.

OBLIGATIONS OF EMPLOYMENT

Suppose that Jack essentially promised never to engage in any activity in competition with Goliath. At first glance, this might look grim for our fledgling entrepreneurs. But a restriction as broad as this is probably unenforceable. Some states, including California (with its concentration of computer-related industries), have flatly outlawed absolute prohibitions on the right to compete with a former employer. In other states, the courts are likely to view such limitations in light of what is reasonable under the circumstances, and enforce them only to that extent.

Nevertheless, from the employee's standpoint, it's important to be aware of the provisions in any agreement he or she has signed, since some or all may be valid.

There are also legal obligations that are independent of any written agreement. These arise from the very existence of an employment relationship. Employment contracts, like any others, always carry an implied covenant of good faith and fair dealing. And since an employment relationship is considered a *confidential* one, the employee has a *fiduciary* duty to the employer.

Put in simple terms, this means that an employee has a legal obligation to faithfully and loyally serve the *employer*'s interests. This is fine in theory but potentially complex in its application to specific situations. The best rule by which to guide oneself is to adhere to the dictates of common sense.

Let's apply these guidelines to Jack Megabyte's situation. Poor Jack could be in trouble if he's made any significant use of Goliath's time and property in pursuing his personal project. Under these circumstances, there could be a serious question as to the ownership of the new operating system Jack and Helga dreamed up while he was working for Goliath.

The basic rule of law is this: while an employee is acting within the scope of employment, the employee's work product is the property of the employer. But the more intellectually creative an employee's activities become, the more difficult it is to apply this rule. After all, an employer does not own an employee's mind.

Such considerations as where and when Jack did the thinking that led to the concept of the new system can be important in determining whether it was done within the scope of his employment. So can the extent to which Jack may have used Goliath's proprietary information and trade secrets as points of departure.

OTHER CONSIDERATIONS

The free-wheeling environment which traditionally has characterized the computer industry leads to unique problems in applying these precepts. Employee use of the employer's computer facilities during periods when they otherwise would be idle has been commonplace in this field — as have elasticity in working hours and worksites. Irregular hours and work done at home or at some other convenient site make it more difficult to determine when employees are acting within the scope of their employment. Moreover, when an employee is engaged in "pure" research rather than "applied" work, it can be even harder to draw the line between employment work product and outside-the-job activity.

The use of special skills acquired during employment can't be prevented. However, employees do not have the right to disclose former employers' trade secrets and customer lists, and as a general rule are barred from *soliciting* the former employer's customers for business. As a practical matter, though, strict adherence to this rule may have the effect of preventing employees from engaging in business. Since the courts are reluctant to go along with this, there's a huge "gray area" with respect to what is allowable and what is not. By and large, the courts tend to protect employee mobility when possible.

But, there are other restrictions on a former employee's actions that will be upheld by the courts. For instance, it would be improper for Jack Megabyte to solicit other Goliath employees to join him in his new enterprise. And if Jack by chance has been a corporate officer of the Goliath company, he has to be particularly cautious in involving himself in competitive activities — lest he find himself on the losing end of a lawsuit.

How does Helga, Jack's crack systems designer partner, fit into this picture? Can she safely assume this is simply a matter between Goliath and Jack?

Not at all. The range of possible legal actions available to Goliath isn't limited to suits against poor Jack. Helga, too, could find herself a defendant in court under a claim of having induced Jack to breach his contractual obligations with the company, or having interfered with his employment relationship with Goliath. Helga also could be in jeopardy if she had been a consultant for Goliath, and thus had access to the company's trade secrets and proprietary information.

What this all means is that

An employer does not own an employee's mind.

Jack and Helga should probably have obtained legal advice at a point well before Jack pulled the plug on his employment relationship with Goliath. Be that as it may, now that Goliath has tipped its hand, it's imperative that they consult with counsel before making another move. This doesn't mean, though, that Jack and Helga should panic. In this showdown, they may have some effective legal weapons of their own. We'll discuss some of these in a future column.

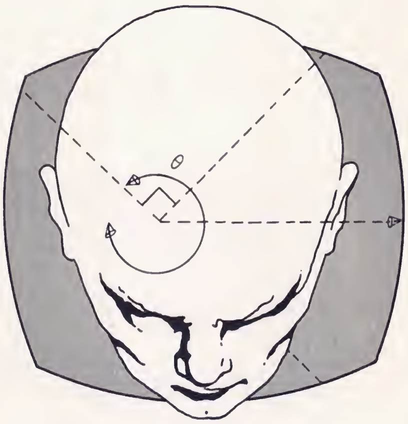
Glenn Groenewold is a California attorney who devotes his time to computer law. He has served as an administrative law judge, has been active in trial and appellate work and has argued cases before the state Supreme Court. ■

PROBLEM SOLVER

Dealing with disaster

by Bob Toxen

Sooner or later every UNIX system will sustain file system damage. It may be caused by a power failure, a hardware or software failure or an operator error. The chances of these occurrences may be reduced by using proper techniques. Preparing for the inevitable reduces the impact of the damage. When damage does occur, moreover, the proper response may minimize data loss and prevent further damage. File system damage is like a cancer: unless it is stopped, it will grow and destroy more and more files.



PREVENTION

A kilobyte of prevention is worth a gigabyte of cure. If power lines are unreliable or noisy, or if your equipment or data is particularly sensitive, then investing in an *uninterruptible power supply* is well worth the doller-per-watt cost. It also pays to keep equipment properly maintained.

Be very careful with the files needed for booting. Other system files, too, should be handled with care. Removing **/dev/console** or accidentally entering:

```
# chmod 666 / usr/file
```

instead of:

```
# chmod 666 /usr/file
```

can be disastrous. The former will instantly render the root file system unusable and unbootable, since it takes execute (directory search) permission away from the entire file system —except for references relative to the current directory that do not go through the root directory.

Make sure knowledgeable people know how to reset the system, know how to turn the system off, and understand the tradeoffs of connecting other equipment to the same electrical circuit (causing electrical noise or an overload). Ignorance is not bliss but an accident waiting to happen.

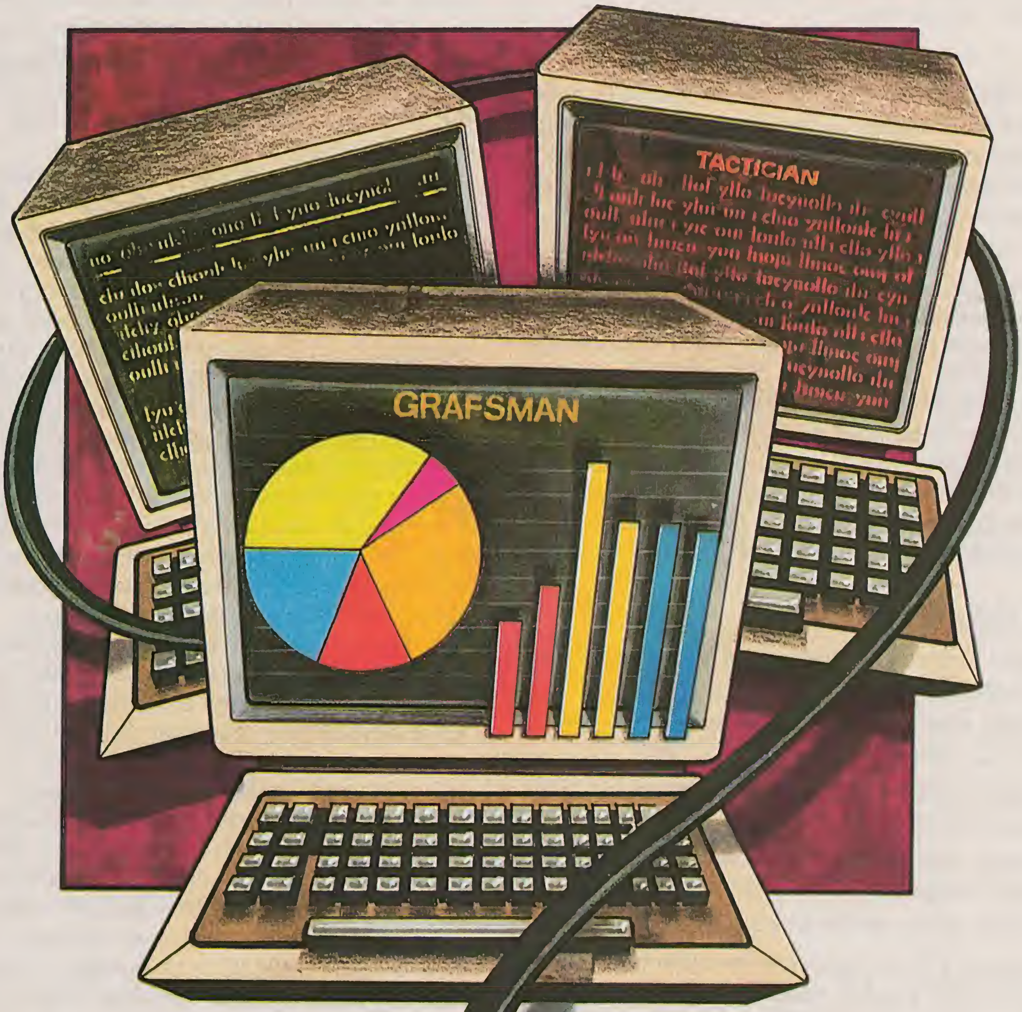
MINIMIZING THE IMPACT

The best and simplest way to minimize the impact of a crash is to perform frequent file system backups. Backups should usually be done every day or two and certainly at least weekly (unless data is static). Users should be encouraged to record their valuable data on either a tape (or floppy), another system, a different disk or, if necessary, on the same disk. Backup tapes (or floppies or disks) should be read periodically to make sure they are readable. Some people have learned this lesson the hard way. Alternate between at least two tapes (or sets of floppies) in case the system crashes in the middle of a backup, destroying both the disk and tape. Store some backups off-site to guard against fire, earthquake and sabotage.

Make provisions for easy recovery in the event the system will not boot. One method is to have some way of booting off a different disk. Another common method is to provide a way to backup the disks with standalone utilities that can be booted from instead of the default UNIX kernel. You could also provide a way to overwrite the disk with a bootable UNIX system and essential standalone utilities that should be bootable from tape (or floppy). There are 10 files needed for UNIX to boot, including:

```
/unix (name may vary)
/dev/console
/dev/md0a (name may vary)
/dev/swap
/etc/init
```

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 NCR TOWER 1632 and PLEXUS P/35 installations. Additional IPT ports will be available soon.

Call for further information about the fast, versatile and powerful Integrated Productivity Tools family.

SOUTHWIND SOFTWARE, INC.
4520 E. 47th St. So.
Wichita, Kansas 67210
316-788-5537
1-800-346-3025 EXT 234

SOUTHWIND SOFTWARE

Circle No. 42 on Inquiry Card

```
/etc/inittab (System III & System V)
/etc/rc (System III)
/bin/sh
/bin/csh (Some configurations)
/bin/su (System V)
```

Also, all directories leading to these files must be readable and executable by all. Some versions and implementations will need different files. One way to find these files is to reboot the system (after properly shutting down) and issuing the command:

```
# ls -lut / /bin /dev /etc
```

as soon as you get a single user prompt. This will list the files in the specified directories with the time they were last accessed (read, written or executed as a program) — sorted by access time. Those files with an access time after the time the system was shut down are probably those needed for booting. For systems with several disks, these critical files should be duplicated on a second disk, and the capability of booting from that disk should be provided. In most implementations, one can boot off any disk or tape.

SHUTTING DOWN THE SYSTEM

1. Make sure everyone is logged off including those on dialups and nets.
2. Make sure that printers, tape drives and other peripherals are inactive.
3. Make sure **UUCP** and similar networking programs are inactive.
4. Make sure various daemons such as mailers, news and networks are inactive.
5. Take the system down to single user mode.
6. Do a **ps** and kill any process besides process 1, your shell and **ps**. Do another **ps** to verify that they all went away. A **kill -9** may be needed. Don't worry about **getty**s that do not go away. This is a harmless problem caused by a defective **tty** driver. International Technical Seminars offers an excellent class on how to write drivers correctly.
7. Issue a **sync** command.
8. Turn off the system or press the reset button.

People who have the **reboot** (or facsimile) program may use it in place of steps 5 through 8.

WHAT FILE SYSTEM DAMAGE MEANS

In addition to the data in everyone's files, UNIX must keep track of the names of files, their permissions, ownership, time of last modification, links, directories, unused disk portions (free blocks and free inode numbers), counts of files and counts of blocks of data that will fit in each file system — as well as an assortment of other concerns. When changes are made to such things, they are not immediately written out to disk but instead are kept in memory. If anyone wants to read any of this changed data, UNIX knows to use the copy in memory rather than the old copy on disk. Likewise, any new changes will affect the copy in memory.

Some portions such as the superblock, which keeps track of free blocks, free inodes and such, and the **/tmp** directory, which contains the temporary files used by editors and compilers, change often. Other areas, such as the **/bin** directory, do not change often but are read often (every time you execute a program). By keeping this rapidly changing or frequently read data in memory rather than having to read it continually from disk and write it back, UNIX runs much faster than it would otherwise.

This buffer area in memory is limited by a fixed size. If there isn't room to fit in some new data that someone wants to read from or write to a disk, then a portion of the buffer will be written to disk — if a change has been made. That portion of the buffer then will be available for new data.

There is usually some changed data sitting in the memory buffer waiting to be written to disk. UNIX is in no hurry to write this data to disk. Why should it? If anyone wants it, they can get the memory copy of it. The only problem is that if the system crashes, the disk will contain some old data.

If some of this old data is information on whether a particular block of data is free; is contained in a file; is a list of where the data for a particular file is kept on disk; or is a list of files in a directory, then UNIX will be confused when it is rebooted. Suppose you just created a file with **vi**. Imagine that the block on the disk that records the place where this file's data is kept is written to disk and that the actual data blocks are also written on the disk. If the system then crashes, the block that records the file's data blocks will have been allocated to a file (rather than being unused), and the data block of the directory that this file was created in will not have been written to disk. If you then reboot, you will not be able to access that file because its name is not in the directory. Also, if you create another file, it may use the same blocks that were used by your first file, destroying the first file's data. This is why, when rebooting after a crash, **fsck** must be invoked *immediately* — before the file system is changed further.

RECOVERING FROM A CRASH

First, log the crash in the system logbook. Include any error messages and any other significant items that will help determine the cause of the crash — thus minimizing the impact of future crashes of a similar nature. For example, if the system crashed with the error message:

```
panic: IO err in swap
```

displayed several times, one would suspect that either the disk used for swapping or its related controller, device driver or the like was having problems. Similarly, the message:

```
panic: parity
```

appearing more often than, say, once a month probably indicates memory hardware problems. In most implementations, the message will also tell which section of memory the parity error has occurred in. After several such panics, a field service engineer may be able to see a pattern and determine which section of memory should be replaced.

After logging a crash, reboot the system by pressing the reset button or performing whatever routine you normally use to start up your system. It should come up in single user mode. The very first thing to do at this point is to run **fsck** — as in file system consistency checker. It will check each of your file systems to make sure they are not corrupt. This is usually done with the command:

```
# /etc/fsck
```

Some systems are configured to start up **fsck** automatically. The **fsck** command will read the file

A kilobyte of prevention is worth a gigabyte of cure.

/etc/checklist to get a list of file systems to check.

The **/etc/checklist** file is a textfile that may be edited with **vi**. It contains the name of each disk that contains each file system, one name per line. The first line should have the name of the root file system. This name (on every line except the first) should be the same name used in a **mount** command — except that there should be a small letter "**r**" after **/dev/**. For example, if your root file system is on **/dev/md0a** and you issue the **mount** commands:

```
# /etc/mount /dev/md0c /usr
# /etc/mount /dev/md1a /mnt
# /etc/mount /dev/md1c /image
# /etc/mount /dev/md1b /tmp
```

then your **/etc/checklist** file should look like:

```
/dev/md0a
/dev/rmd0c
/dev/rmd1a
/dev/rmd1c
/dev/rmd1b
```

If **/etc/checklist** is not configured, list the file systems to be checked on the command line, like so:

```
/etc/fsck  /dev/md0a /dev/rmd0c /dev/rmd1[acb]
```

The **fsck** command will then read through each file system (from disk) and check for inconsistencies, such as blocks that are both on the free list and in a file or files that don't appear in any directory. Each time **fsck** finds something wrong, it will indicate what the problem is and ask whether it should be fixed. Almost always you will want to type the letter "**y**" (for yes) followed by RETURN. One case where you might want to type "**n**" (for no) is when **fsck** asks you whether it should delete a file and gives only its inode number rather than its name. You will want to find out the file's name before it is removed so you can recover it from backup tape. To find out the name of inode number 387 on **md0c**, give the commands:

```
# /etc/mount /dev/md0c /usr
# find /usr -inum 387 -print
# /etc/umount /dev/md0c
```

Another time to say no is when **fsck** asks for permission to remove **/unix** or another equally important system file. Recovering from this or other more complex problems is beyond the scope of this article.

Bob Toxen is a member of the technical staff at Silicon Graphics, Inc. He has gained a reputation as a leading ***uucp*** *expert and is responsible for ports of System V for the Zilog 8000 and System III for the Motorola 68000.* ■

HIEROGLYPH™
UNIVERSAL REPORT PRODUCTION SYSTEM™

INTEGRATED TEXT/DATA/ADVANCED GRAPHICS SOFTWARE

A "report" is information presented in organized form—typically a printed document. If you are a professional whose work involves preparing reports HIEROGLYPH™ is for you. HIEROGLYPH is designed to meet the needs of technical and office report preparation and production where the combination of text, data and advanced graphics are essential elements. Whether you are an engineer, scientist architect, manager of business reports, a writer or graphic artist, HIEROGLYPH integrates all information in your UNIX Universe. HIEROGLYPH combines all the elements necessary to generate the final composition. You have the option of camera-ready copy or multiple copies in Color or Black and White. HIEROGLYPH uses simple, single-stroke commands. Both text and commands can be in English or several foreign languages. The user manual is written for three levels: New, Experienced and Expert.

Regardless, you can be doing productive work the first day. HIEROGLYPH software incorporates Text processing, Document Aids, Document Filing, Graphics, Data Handling and Production Tools. Together they comprise the most productive system for creating and producing superior reports—documents that make lasting impressions.

PRESCIENCE

HIEROGLYPH is the product of Prescience, Inc. (pronounced pres-ce-ence) 820 Bay Avenue, Suite 300, Capitola, California 95010 (408) 462-6567.

HIEROGLYPH is referenced in the following publications: AT&T UNIX System V Software Catalog, CONVERGENT TECHNOLOGIES Software Catalog, ENVISION Independent Sofware Supplies Catalog, PYRAMID Prism Program, SUN Microsystems Catalyst Program.

Circle No. 44 on Inquiry Card

THE UNIX GLOSSARY

A guide to shell terminology

by Steve Rosenthal

Note: where terms have several meanings, only those directly pertaining to the shell are listed.

— in the standard UNIX shell (the Bourne shell), the # is used as the default character meaning "delete the previous character on the command line." Successive #'s delete successive characters going back towards the start of the line. Many systems have changed the default from # to the backspace (CTRL-H) or the DELETE (also called RUBOUT). You can set your own choice with **stty.**

— the default prompt for superuser.

— when this symbol is used as the first character of a shell script (either Bourne or C shell), the remainder of the line is treated as a comment, and the line's contents are ignored.

$** — when used as a first character in a name included in a command line to the shell, the **$ symbol serves as a metacharacter that tells the shell to take the name as a variable and substitute in the current value before executing the line.

$ — also serves as the default system prompt used by the Bourne shell (the standard UNIX system command interpreter) when it is waiting for user input. This character can be changed, and other shells (such as the C shell) use different characters.

$# — a shell variable that is set by the system to the number of arguments supplied by the user when invoking the shell. It is most often used for testing shell scripts to see that the right number of arguments was supplied.

$$ — a shell variable set by the system to the current process id. It is frequently combined with other variables or strings to create names for temporary files or provide traceback information for debugging.

$* — a metastring expanded by the shell to the full set of positional parameters (the variables **$1**, **$2**, **$3**, et al.). It is used in **for** expressions to make a shell script repeat a block of commands for each argument supplied when the shell is invoked.

% — as a system prompt, the % sign is the character used by most installations running the C shell to indicate that the system is waiting for input. In many implementations,the % symbol is preceded by a number indicating the sequence number of that input line for the current login session.

& — when used at the end of a command line, the ampersand sends a message to the shell requesting background execution of the specified command. It is frequently used to start long, low-urgency tasks.

) — in a case statement, this character is used by the shell to select one of several possible blocks of commands based on a string match. The) symbol ends the matching string at the beginning of each of the possible command blocks.

***** — a metacharacter translated by the shell as matching any string. It's normally used to match filenames, often combined with an initial letter or suffix to limit the match to certain files. Thus **name* matches *name*, *thatname* or *anyname*. Read aloud, the * symbol is usually pronounced as "star" rather than "asterisk."

- — as a prefix to a word following a command name in a command line, the - character means that the word is a flag rather than an ordinary argument. Flags (also

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
COMDEX™/Fall '84
November 14-18, 1984
MGM Grand Hotel
Las Vegas, Nevada
Booth M410

Circle No. 45 on Inquiry Card

called options) control the way certain commands are executed.

$0 — a shell variable containing the name of the current shell procedure. It is set by the parent shell when the shell script is invoked.

: — the character used to separate pathnames in a path command that tells the shell where to search for commands and in what order. An initial : represents the current directory.

: — also used as a null command for the shell that can be used at the start of a line to designate what follows as a comment. Shell variables and expressions are evaluated inside this comment, however.

; — the symbol used as a logical end of command in shell command lines. It allows more than one logical command to be given on one physical line.

;; — in a **case** statement, this is the symbol used to end one of the case's command lists. See *case*.

< — the "less than" sign or left angle bracket. When placed after a command name in a command line, it indicates that, rather than receiving standard input, input should be read from the file named next.

<< — in a shell procedure this identifies the following token as a beginning marker for a *here document*, in which all lines that appear before a matching string end marker are treated as input to the command preceding the <<. This tool is commonly used for in-line **awk** programs.

= — in the shell, the **=** sign acts as the assignment and variable creation command. The variable on the left of the **=** is set to the value of the expression on the right. If the variable did not previously exist, it is created. No space is permitted around the **=** sign.

= — the logical equality operator in expressions. At least one space must be on either side of **=**.

> — the "greater than" sign or right angle bracket. When placed after a command name in a command line, it says to redirect the output that normally would have gone to standard output to the file named on the right.

>& — in the C shell, this serves as the command to redirect diagnostic output as well as standard output to the file named to the right of the **>&** sign.

> > — used in a command line to the shell, the > > says to append the output from the command on its left to the file whose name appears on the right.

? — as part of a file specification for the shell, the **?** matches any single character. Thus *?file* would match *ofile*, *ifile* or *afile*.

-v — a flag selecting the verbose option for the shell. In this mode, the shell prints each command before executing it. It is most often used for debugging.

-x — a flag selecting the execute option for running a shell procedure. In this mode, the shell prints out all commands, variables, substitutions and expansions for each command line. It is used for debugging.

[ ] — in a command line, this string matches any one of the characters enclosed within the brackets. If two characters in the list are separated with a - (minus sign or dash), then all characters in the character set range between those two characters are also considered to be included.

\ — the backslash character, used as a quoting symbol to tell the shell to interpret the character following the backslash as a normal text symbol instead of a metacharacter for processing by the shell. For example, a string starting with \$ will be interpreted as a string actually starting with $ rather than as a variable standing for a value.

{ } — the "curly braces" are used to set apart an embedded shell variable within a text string argument. For example, *${which}one*, when *$which* has the value "any," will be expanded to mean *anyone*.

| — the character used to tell the shell to set up a pipe, taking the standard output from the command names on the left and routing it to the standard input named on the right. The | character itself is often called "pipe" in UNIX, whereas in other contexts it is more commonly called the "or bar," "alternation bar" or "vertical bar."

What do you like
better about the VISUAL 2000 . . .
its power or its versatility?

Personally,
I like its price!

Never has a UNIX-based multi-user system given so much to so many for so little.

Introducing the VISUAL 2000

The VISUAL 2000 is the full-featured system with the power and flexibility to support multiple users in real business applications at a surprisingly low cost per station. It can be used with inexpensive video terminals. Or as a database manager or file server for a cluster of intelligent workstations or PCs, including both the IBM® PC and VISUAL's own lightweight, portable, totally IBM PC compatible COMMUTER. In all applications it offers greater performance, more flexibility, and lower cost than any other system in its class.

Powerful Intel 286 processor

The Intel 286 is today's chip of choice for UNIX™-based systems. Only the Intel 286 has on-chip memory management, an instruction set optimized for multi-tasking, and the optional 287 numeric co-processor to speed up floating point by a factor of 10.

What do these features mean to the VISUAL 2000 end user? Faster response time, more users supported, and lower system cost!

Cost-effective one-board design

A basic advantage of the VISUAL 2000 is its one-board base-level design. A single high-density board includes the 286 CPU, 512KB-2MB of RAM, controllers for Winchester, floppy, and streaming tape, an intelligent communications processor, six RS-232 ports, and a parallel printer port. Even a real-time clock with battery backup. One-board design means higher performance, lower cost, and greater reliability than comparable multi-board implementations.

UNIX trademark Bell Labs/XENIX trademark Microsoft/IBM® trademark International Business Machines/COMMUTER trademark Visual Computer Incorporated/AT&T trademark American Telephone and Telegraph/RM/COBOL trademark Ryan-McFarland/TOM trademark The Office Manager/SOFTBOL trademark Omtool Corp./MicroFocus Level II COBOL trademark Microfocus/INFORMIX trademark Relational Database Systems/RealWorld trademark RealWorld Corp./20/20 trademark Access Technology/XED trademark Computer Concepts Ltd.

Configurability and Expandability: VISUAL gives you more

The VISUAL 2000 spans a much wider range of configurability and expandability than other systems in its price class. Up to 16 independent users. 6 megabytes of RAM. 4 Winchesters. Floppy. And streaming tape for simple, reliable disk backup. All in a small stand-up enclosure which looks right at home next to a desk.

And if a fully expanded VISUAL 2000 isn't enough, you can connect up to 254 VISUAL 2000s, PCs, and workstations in a local area network.

Extensive system software simplifies system integration

The VISUAL 2000 runs XENIX, Microsoft's popular, enhanced version of UNIX, derived from UNIX under license from AT&T, and designed to be faster, more secure, and easier to use in business applications.

And VISUAL has worked hard to simplify the system integrator's job, by providing all the tools needed to deliver end-user applications with a minimum of effort.

Languages such as C, SMC BASIC, RM/COBOL, TOM BASIC, SOFTBOL, and MicroFocus Level II COBOL, to provide instant compatibility with hundreds of proven business application programs.

Other system-building tools, like the INFORMIX database management system and RealWorld modular accounting system.

And productivity software, such as the 20/20 integrated spreadsheet and XED office-grade word processor.

The Bottom Line

High performance. Superior flexibility. Extensive software. And low cost. . . VISUAL 2000 systems start at under $10,000, suggested list. No one gives you more in a UNIX-based multi-user system.

Whether you're an OEM, system house, distributor, or end-user, call today for further information on the VISUAL 2000 and see for yourself!

VISUAL See for yourself®

Visual Technology Incorporated
540 Main Street, Tewksbury, MA 01876
Telephone (617) 851-5000. Telex 951-539

REGIONAL OFFICES:

Northwest:	(415) 490-1482
Southwest:	(213) 534-0200
North Central:	(513) 435-7044
South Central:	(214) 255-8538
Northeast:	(201) 528-8633
Southeast:	(301) 924-5330

Circle No. 49 on Inquiry Card

Bourne shell — the standard UNIX shell distributed by AT&T. It is named after its principal developer, Dr. Stephen R. Bourne, formerly of Bell Labs and now software engineering manager at Silicon Graphics, Inc. (and a member of the *UNIX REVIEW* Board). The Bourne shell is a powerful programming language based on ALGOL-68 syntax.

case — a shell program flow command that executes the first of a set of blocks marked with a string matching a stated case variable. Each block starts with a string terminated with a) and ends with a double semicolon, ;;. The case statement itself ends with an **esac**, which is "case" spelled backwards.

csh — the command used to invoke the C shell. See *sh* for more details.

C shell — a popular shell that is more useful for interactive work than the original Bourne shell, but takes up more memory and runs slower on non-paging hardware. Developed by William N. Joy, Vice President for Research at Sun Microsystems (and a member of the *UNIX REVIEW* Board), the C shell is part of the Berkeley Software Distribution and is available at most UNIX installations. It features C language-like structures, aliases, command histories and several other features.

.cshrc — one of the two login profiles the invoked C shell looks for. The other is **.login**. Login profiles are shell procedures that set initial parameters and invoke setup programs according to the preferences of users.

do — a reserved word that starts a list of commands after a **for**, **while** or **until** statement. The list ends with the matching **done** reserved word.

done — the reserved word that ends a list of commands started with **do**. Data passes to the statement following the **done** after the commands in the block marked by the two reserved words are executed for all argument values.

elif — a reserved word that ends a block of shell commands started with an **if** clause, and then starts a new **if** clause. It is used to combine multiple conditions without resorting to nested **if** levels.

esac — a reserved word that ends a **case** statement in a shell procedure. It is, simply, "case" spelled backwards.

exit — a shell command to end the current procedure, discontinuing execution of further commands. Because the shell procedure is invoked by an instance of the shell itself, exiting the procedure completes that execution of the shell, returning control to the calling program.

expr — a shell command to produce as output the value of a list evaluated as an arithmetic statement. The command accepts **+**, -, *, / and % (remainder) as operators (but * must be quoted, either with quote marks or a backslash, or it will be taken as a metacharacter matching any word).

fi — a reserved word that ends a block started with **then** or **else** in an **if** statement. The word "fi" is "if" spelled backward.

for — a shell-reserved word that says to repeat the procedure block between the immediately following **do** and **done** lines for all items in an argument list.

here document — a section of text embedded in a shell procedure used as an argument to a command. It is marked with a << symbol, extending from after the word following the << to the point where the next match to that starting word occurs.

HOME — the shell variable containing the name of the user's home directory, if set by default, a **.profile** specification or an entry at the keyboard.

if — the shell construction that starts a conditional execution block. If the condition following the **if** evaluates to a true value (a zero), the command list following the then reserved word is executed. If not, and there is a command list following an **else** reserved list, that second list is executed. The **if** construct ends with the **fi** reserved word.

.login — one of the two login profiles used by the C shell. See *.cshrc*.

PATH — the shell variable specifying which directories to search through and the order in which they should be searched in order to find executable files corresponding to command lines. The default path is **HOME:/bin:/usr/bin**.

pipe — a facility offered by the shell routing standard output of one program to the standard input of another. The connection is intended to be one-way and synchronization is handled by the kernel. The | symbol is used to indicate that the output from the process on the left should be routed to the process on the right.

pipeline — a succession of commands connected by pipes, such that the output of each one becomes the input of the next. This aggregate construction allows small building block commands to be combined to accomplish complex tasks.

positional parameter — a shell variable corresponding to arguments provided when the shell was invoked. These are referred to by number, with the first argument appearing as **$1**, the second as **$2**, and so on.

.profile — a Bourne shell procedure automatically invoked during the process of signing on to the system. It is used to set variables and run standard startup commands according to individual user preferences.

PS1 — the shell variable containing the current prompt string. It defaults to **$** for the Bourne shell, but you can set it to anything you want by **PS = "newprompt "**.

PS2 — the shell variable containing the current special prompt used for requesting additional input from the user. It defaults to >, but can be changed if you want some other prompt string.

reserved word — one of a set of words that the shell takes as having special meaning. Reserved words include **if**, **then**, **else**, **elif**, **fi**, **case**, **esac**, **for**, **while**, **until** and **done**.

sh — the command to invoke the standard (Bourne) shell. Because the shell is itself an executable program, it can be invoked by the user. Berkeley Software Distribution offers two shells, the Bourne shell and the more complex C shell. The Bourne shell is invoked with **sh** and the C shell is invoked with **csh**.

shell — a program that interprets user commands, invoking the required programs, expanding or evaluating expressions, and directing the input and output. Two shells, the Bourne shell and the C shell, are popular, but UNIX also allows you to write your own.

shell procedure — a file containing a set of lines to be executed by the shell. Normally, the lines contain shell commands and arguments. A shell procedure is executed by invoking the shell with the procedure as an argument or as a command. If created with one of the normal editors, it must be made executable (by changing its permissions with **chmod**) before it can be run as a command. Shell procedures are also known as "shell scripts."

shell scripts — see ***shell procedure***.

shell variable — a variable that can be set, tested or used in shell programming. The shell variable **var** is set by the assignment statement **var=value** and the value is retrieved by prefixing the variable name with a **$**.

tee — a command to preserve data, provided by a file named as an argument, and pass the data on via a pipe to a succeeding file in a pipeline. It can be used to save intermediate results in a pipeline, or to create multiple temporary files.

then — the reserved word in an **if** statement that starts a block of commands to execute if the condition is true. The block ends with an **else** or **fi** reserved word.

trap — a command to tell the shell what to do before exiting if a shell procedure is interrupted. UNIX defines more than a dozen signals for various causes of interruption, and the user can define what to do for most of them. For example, **signal 2** indicates that the user tried to interrupt a running procedure from the keyboard.

until — a shell construction which indicates that the immediately following block should be repeated so long as the expression in the until statement is false. The block is marked by a **do** reserved word at the beginning and a **done** reserved word at the end.

while — a shell construct indicating that the immediately following block should be repeated as long as the expression in the **until** statement is true. The block is marked by a **do** reserved word at the beginning and a **done** reserved word at the end.

Comments, arguments, corrections? Send them to UNIX Glossary, Box 9291, Berkeley, CA 94709.

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

BEAT OUR BENCHMARK.

GET THIS WATCH.

Take the Plexus Challenge.

Everybody in the UNIX-based supermicro business <u>talks</u> performance.

Only Plexus dares to prove it... with the Plexus Challenge:

Show us a UNIX-based supermicro that can beat Plexus—running any recognized benchmark (or even your own application code*) —and we'll present you with this elegant Heuer chronograph valued at $485.00.

How do we dare make this offer? Simple: our multiprocessor architecture and highly-tuned version of UNIX leave the single-CPU competition in the dust. Multibus architecture and intelligent I/O processors let you expand modularly from 5 to 40 users—without changing a single byte of object code.

Are you thinking "Fine, but I really don't need all that performance"? Well, some of our customers tried other hardware first. They <u>used</u> to think the same thing.

Challenge details: This offer is extended to bona fide OEM's, value-added resellers, and volume end-users in the multiuser commercial UNIX systems market. Limit one award per company. Offer expires Jan. 31, 1985 unless withdrawn earlier. For complete rules and participation information, call (800) 556-1234 Ext. 560 (in Calif. (800) 441-2345 Ext. 560)

PLEXUS

Supermicros built for speed.

Plexus Computers Inc., 3833 North First Street, San Jose, CA 95134

*Subject to prior approval

RECENT RELEASES

AT&T made it clear with a late June announcement that it means to move in on IBM's marketing territory. The ranking brass of AT&T Information Systems assembled in New York on June 26 to unveil an MS-DOS Personal Computer with two significant UNIX links. A dozen application software packages for the 3B series of UNIX computers were also announced.

The Personal Computer, a 16-bit 8086-based machine manufactured by Olivetti, resembles the IBM PC in many ways. The keyboard is virtually identical and the machine runs most IBM-compatible software. While prices were not announced, AT&T clearly means to be competitive.

In referring to IBM, AT&T Information Systems Chairman Charles Marshal said, "We welcome electronic high noon."

One point in favor of the new AT&T low-end machine is its strong UNIX link. Two software products being marketed with the Personal Computer, for $100 apiece, give businesses already running UNIX a number of enticing options.

The first of these products, PC Interface, was initially announced in January as one of the 3B suite of products. Through it, MS-DOS users can make a connection to either a 3B2 or 3B5 computer and thus make use of the larger computer's resources. This can be particularly useful for data transfer and storage. PC Interface also allows Personal Computer users to escape MS-DOS altogether and operate as a UNIX workstation.

A second software product, dubbed the "Context Switch" allows Personal Computer users to jump from MS-DOS to UNIX at the flick of a switch. MS-DOS applications can be frozen in process while the user shifts over to UNIX to do other work. When the user switches back to MS-DOS, the unit will return to the same process it was working on before. Context Switch also allows users to run various UNIX utilities (such as **spell**) on files created under MS-DOS.

Concurrent with announcements related to the Personal Computer, AT&T unveiled 12 application software packages for the 3B2 and 3B5.

Reflecting AT&T's Fortune 1000 orientation, the mix of packages has a strong business application flavor. Five of the 12 are database-related packages: AT&T INGRES, Informix, File-it!, dBASE II and C-ISAM. Two others, AT&T Business Accounting System and Multiplan, have an accounting orientation.

The Handle family of software — WRITER/SPELL, HANDLE/CALC, HANDLE LIST and HANDLE GRAPH — gives AT&T an office automation product.

The remaining software offerings are: Microsoft Word, Ryan-McFarland COBOL, AT&T Gift Registry and the Communications Management Control System.

Circle No. 70 on Inquiry Card

MASSCOMP ANNOUNCES MC-500DP SERIES

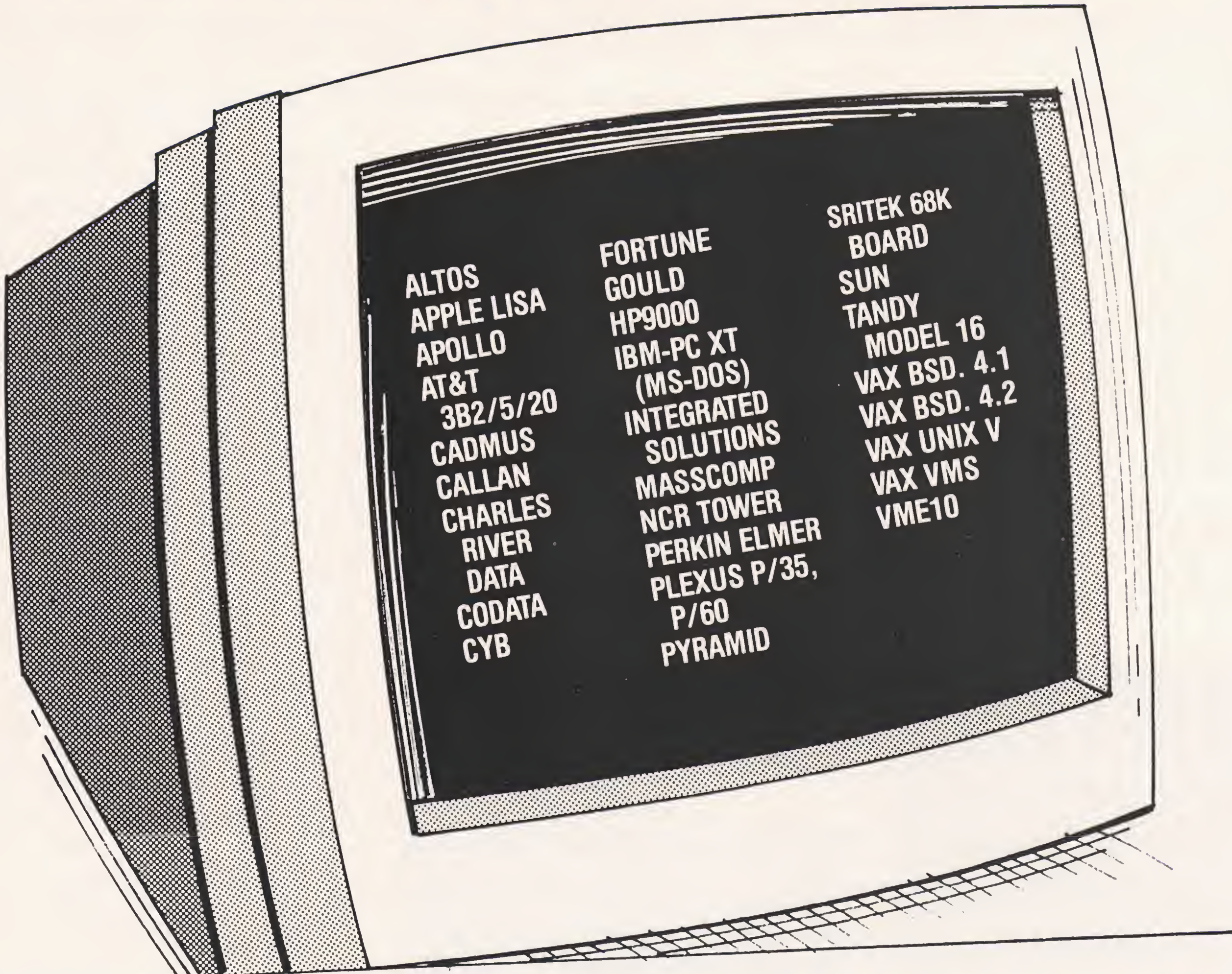
MASSCOMP has announced an addition to their family of supermicros: the MC-500DP Series, the first UNIX-based, real-time dual-processor supermicro system for scientific, engineering and general technical applications. The dual processor increases processing power by up to 99 percent over MASSCOMP'S single-processor systems.

The MC-500DP uses the single-processor CPU, packaging and peripherals so current customers can upgrade from a single-processor to a dual-processor system.

The MC-500DP series architecture consists of two high-speed, 8 MB-per-second buses, and MULTIBUS and an STD bus. Each high-speed bus supports a general-purpose, 32-bit virtual memory CPU and ECC memory. In addition, the system supports two floating-point processors.

The architecture incorporates all of MASSCOMP's single-processor features, including an integrated array processor, multiple independent graphic processors, 8 MIPS (million intructions per second) data acquisition and control processors and an Ethernet communications processor. Various disk sizes from 50 to 474 MB are supported, as well

Whatever System You Use . . .



. . . there's plenty of UniPress Software from our product library

PRODUCTIVITY AIDS

- Emacs Multi-window extensible editor
- Minimacs—Emacs without MLISP
- Phact Isam File Manager
- /RDB — Relational Database
- The Menu System

APPLICATIONS

- QCalc Spreadsheet
- Lex Word Processing
- Leverage List Processor

LANGUAGES

- Lattice C Native
- Lattice C Cross to MS-DOS
- Amsterdam C and Pascal Compiler Kit

SUPPORTED O.S.

UNIX 4.1, 4.2, System III, V, V7
VMS
MS-DOS
XENIX

In addition, we carry the UniPlus + Unix System V for the Apple Lisa and a full line of application software dedicated to the Lisa.

Call or write our trained staff for more product information

UniPress Software, Inc.
2025 Lincoln Highway, Edison, NJ 08817
201-985-8000 • Order Desk: 800-222-0550 (outside NJ) Telex: 709418
Mastercard and Visa

See us at UNIX EXPO, October 16-18, New York, Booth No. 246

Circle No. 55 on Inquiry Card

as floppy disks, 1/4-inch cartridges and 1/2-inch MAG tapes.

The MC-500DP system can be used in two modes. It can automatically distribute the computing load between CPUs, or the user can define which tasks are to be run by each CPU.

The MC-500DP Series consists of seven new systems: three workstations, one table-top and three cabinet configurations. The lowest-priced system, the MC-531DP table-top, is available at $36,900, including installation and a 90-day, on-site warranty. It includes two CPUs, 2 MB of ECC (error correcting memory) memory, a 50 MB disk, a 1 MB 5 1/4-inch floppy disk, a 15-slot backplan, an alphanumeric terminal, five serial lines, and the FORTRAN and C languages.

Circle No. 71 on Inquiry Card

NCI DESIGNS SYSTEM V COMPATIBLE KERNEL FOR INTELLIGENT CARDS

Network Consulting, Inc., has designed a UNIX System V compatible kernel that is transportable to Intel-based intelligent peripheral printed circuit baords. NCI's kernel brings UNIX to intelligent peripheral pcbs or processors handing any I/O activity. It reportedly allows sotware developers to write System V applications without being hindered by the design constraints of the hardware.

A developer writing a System V application can make it run faster by letting the intelligent card handle many of the operations that formerly were reserved for the host processor. The NCI kernel allows the host processor to run and interact with UNIX, COHERENT or any other UNIX-compatible operating system.

Circle No. 76 on Inquiry Card

PLEXUS APPLICATIONS SOFTWARE CATALOG

Plexus Computers, Inc., has released the Plexus Software Referral Catalog, listing over 150 UNIX applications software packages currently available for use on its high performance supermicro systems.

The catalog serves both as a reference for integrating the Plexus systems for vertical applications and for improving performance through networking, language development and other options.

All the software listed in the catalog is fully compatible with UNIX and is available immediately. Typical packages offered are for high-level language compilation, database management, word processing, spreadsheet, text editing, general accounting and financial analysis.

The catalog is available from Plexus Computers, Inc., 2230 Martin Avenue, Santa Clara, CA, 95050.

Circle No. 77 on Inquiry Card

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.

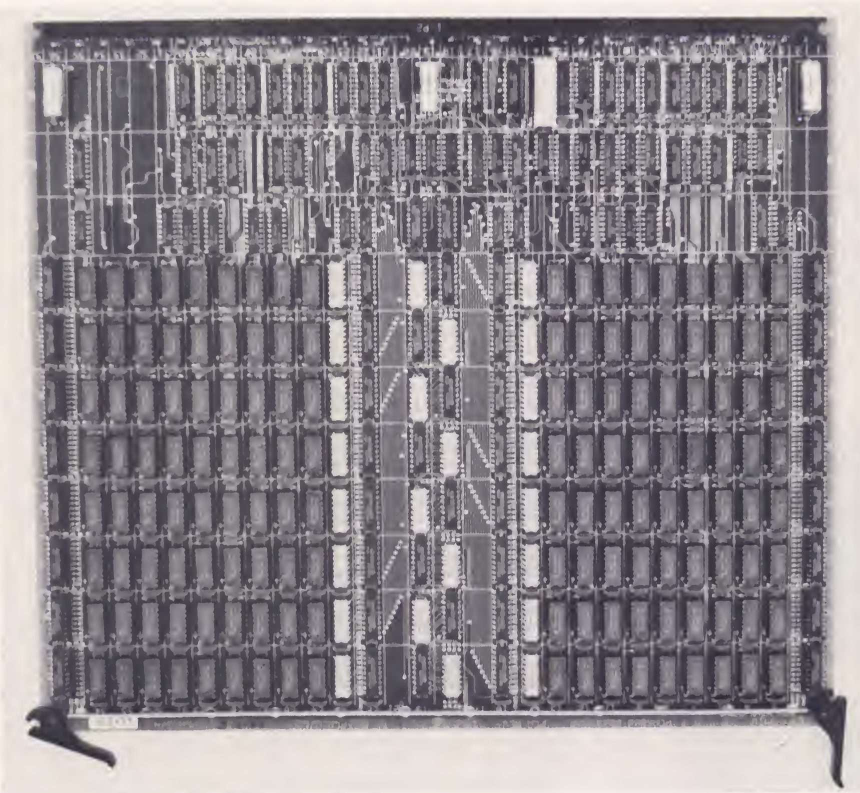
The 15-inch diagonal display boasts 100 dots per inch, which gives you high resolution graphics and font capabilities. Complete with a full 32-bit processor and up to one megabyte of memory, you can also offload the host by running programs in the 5620.

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. 56 on Inquiry Card



PYRAMID DOUBLES MEMORY CAPACITY OF 90X SUPERMINI COMPUTER

Pyramid Technology Corporation has introduced a four MB memory board that doubles the memory capacity of its 90x supermini computer, a 32-bit UNIX-based machine. The 4 MB board uses 256K DRAM chips.

The expanded memory capacity provides up to 16 MB of main memory. In addition to the 4 MB board, Pyramid offers a new 2 MB board that alos uses 256K DRAM chips. Both the 2 and 4 MB boards are installed in new or existing 90x systems and use the present memory control unit without modification.

The 4 MB memory board is priced at $22,000, while the 2 MB board runs for $11,500. Both are now available for shipment.

Circle No. 78 on Inquiry Card

MICRO-MAINFRAME COMMUNICATIONS FOR UNIX-BASED SYSTEMS

Pathway Design, Inc., announced UniPATH, a family of gateway products for micro-mainframe communications operating under UNIX System III and V. UniPATH will be resident on a 32-bit microcomputer system, permitting up to 32 devices to communicate with a variety of IBM hosts in SNA/SDLC and BCS networking environments.

UniPATH, occupying 256K bytes of memory, will be resident on the supermicro and serve up to 32 asynchronous terminals and printers. It will offer value-added concurrent emulation of 3270, 3770, 3780 and 2780 devices. The new products will be transparent to the user, running under UNIX as devices.

UniPATH includes modularized software components, facilitating porting to various UNIX and UNIX-compatible operating systems. The modular software will operate as a logical extension of UNIX, transparently managing address translation and logical unit identification for communications and multi-tasking.

Pathway Design currently manufactures a variety of micro-to-mainframe communications products permitting value-added 3270 interactive communications and 3770 and 3780 file transfer between MS-DOS-based personal computers, including the IBM PC, Sperry PC and Wang Professional, and IBM hosts in SNA/SDLC and BSC environments.

For more information, contact Pathway Design at 177 Worcester Street, Wellesley, MA, 02181, 617/237-7722.

Circle No. 81 on Inquiry Card

LINEAR CAD I ALLOWS DESIGN OF LINEAR LSI ON PC

Micro Linear Corporation has introduced LINEAR CAD I, a computer-aided design package that allows system engineers to design linear and linear/digital large-scale integration (LSI) above the resistor/transistor level.

LINEAR CAD I, which runs on the IBM PC and compatibles, combines graphics and simulation software for designing linear circuits with Micro Linear's macro cell library. System engineers can design linear circuits for Micro Linear's Series FB900 bipolar fixed arrays by performing schematic capture, generating a net list and executing SPICE simulation on an IBM PC.

The package includes the macro cells and modeling parameters, schematic capture, SPICE and an interface that allows designers to perform higher-level simulation, if required, on

Cybernet, a network of time-share computers from Control Data Corp. The Cybernet interface is offered for the simulation of circuits requiring more than 140 transistors.

The traditional semi-custom linear design approach required transistor/resistor level design; Micro Linear's macro cell library now enables designers to design with "blocks" such as op amps, voltage regulators and comparators.

LINEAR CAD I is priced at approximately $7900, with delivery in four weeks. Packaged macro cells are available off the shelf.

Micro Linear Corporation is located at 2092 Concourse Drive, San Jose, CA, 408/262-5200.

Circle No. 84 on Inquiry Card

NEW ETHERNET SYSTEM

SCS Semiconductor Corporation recently introduced an Ethernet local area network (LAN) System for its SAMSON computer system.

SAMSON's front-end Ethernet processing offers transport, network and data link protocols, which are off-loaded from the system CPU. This feature allows a variety of protocol architectures to be used and combined, and by reducing the workload of the CPU, it maintains performance even during periods of heavy network activity.

The SAMSON Ethernet system meets the Ethernet specification, version 1.0, of up to a 10-megabit-per-second transfer rate and a 16-bit dedicated CPU, with 128K bytes of on-board RAM for message buffering and high-level protocol software. SAMSON's hardware also features bipolar, finite-state Ethernet management; up to 32 variable-length chained receive buffers; and up to 60K bytes available for packet buffering. Full hardware address recognition is included with up to 256 contiguous multicast addresses filtered.

High-level protocol software provides a front-end environment for protocol execution and is down-line loadable. SAMSON's Ethernet software supports OEM-written protocols for special applications and OEM-ported pro-

A NEW DIMENSION IN DATABASE MANAGEMENT

Booth #1996-7

See us at

COMDEX™/Fall '84

November 14-18, 1984

Las Vegas Convention Center

Las Vegas, Nevada

See us at UNIX EXPO, October 16-18, New York, Booth No. 103

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

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

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

RHODNIUS Incorporated

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

Circle No. 57 on Inquiry Card

tocols to protect previous investments.

The application, presentation and session layers of the ISI Reference Model are included in the SAMSON Ethernet software package. In addition, several Berkeley enhancements are included. Berkeley's remote copy program (RCP) makes possible remote execution of UNIX commands, including remote mail and I/O redirection on UNIX systems. Berkeley's remote login program (RLOGIN) supports virtual terminal access to other systems running UNIX protocols.

SAMSON's Ethernet is available worldwide through SGS sales offices and representatives. The price is less than $200 per user, based on a fully implemented SAMSON system.

SGS USA is located at 1000 East Bell Road, Phoenix, AZ, 85022, 602/867-6241.

Circle No. 86 on Inquiry Card

MORROW ENTERS 'SUPER-MICRO' ARENA

A multiuser UNIX supermicro that brings the cost per user down to under $2500, has been introduced by Morrow, Inc.

Morrow's Tricep, which is now available, supports 4-8 users running UNIX System V on the 16/32-bit MC68000 microprocessor and is priced starting at less than $9000 retail, and at less than $5500 in OEM quantities.

Tricep hardware includes an MC68000 CPU running at up to 10 MHz, with an on-board MC68451 memory-management unit; 512K bytes of main memory (expandable to 2 MB); an I/O controller with four RS232C serial ports (expandable to eight ports); a Centronics-compatible parallel printer port; hard and floppy-disk DMA controllers; and 80188-based slave processors with 128 or 512 KB of on-board dual-port RAM. All boards plug into the 14-slot IEEE-696 (S-100) bus backplane.

Mass storage includes one to four 16 or 32 MB 5 1/4-inch Winchester disk drives (for maximum 128 MB storage); an optional 5 MB removable hard disk cartridge drive; one to four 400 KB 5 1/4-inch floppy-disk drives; and one to four optional 1.3 MB 8-inch floppies.

Tricep's UNIX System V port,

THE CODATA DIFFERENCE

A Major Improvement in OEM Microcomputers.

With so many microcomputers on the market, and with each manufacturer claiming to have the solutions to everyone's problems, why are more and more OEM users choosing CODATA?

Very simply because the CODATA 3300 works!

That's an important part of what we call THE CODATA DIFFERENCE - a refreshing, new concept in OEM marketing.

More performance...Lower price...Extended warranty.

We've increased the performance of the CODATA 3300 and, at the same time, we've lowered the price. That means our CODATA multi-user, multi-tasking 16-bit computer systems have price/performance ratios nobody else can match. We're so confident in our system's integrity that we've extended the warranty on every CODATA 3300 to 180 days. That's the longest warranty in the business and, let's face it, you've got to be pretty sure of your product to guarantee it for six full months. We're sure.

These are more elements of THE CODATA DIFFERENCE.

An uncomplicated pricing policy.

We have only two price sheets: one for suggested retail, the other for OEM prices. Simple, straightforward, easy-to-understand. Beyond that, there are no quantity commitments, no bill-backs, no hassles.

Full support and technical assistance if needed.

Our customers select the CODATA 3300 because it fulfills their needs, because it does what we say it will do, and because we have taken great pains to assure that it is functionally simple and capable of performing its assigned tasks with no problems. But, if technical support is required it's only as far away as your phone and available at a moments notice via our TOLL FREE number during normal business hours in every U.S. time zone.

A multi-user, 33-megabyte system for only $9,600.

The CODATA 3300 is a powerful 16-bit, 68000-based, MULTIBUS system that can effectively accommodate up to ten users. It's a complete UNIX system that runs full ANSI standard FORTRAN-77, RM/COBOL, BASIC+, SMC BASIC, APL, and PASCAL. The 3300 provides up to 33 megabytes of unformatted on-line storage via an integrated, high speed Winchester drive, and a removable, quad-density 5-1/4″ floppy disc system. The 3300 features 320 K bytes of parity protected RAM.

An expanded CODATA 3300, with 84 megabytes of disc storage is priced at $13,500. The 3300 is also available as a 12-megabyte system for as low as $7,800.

LET US SHOW YOU THE CODATA DIFFERENCE.

Write or call us for more information.

CONTEL CODATA

285 North Wolfe Road, Sunnyvale, CA 94086
1-800-521-6543. In California: 1-800-221-2265

Circle No. 58 on Inquiry Card

KEYBOARD AND MONITOR OPTIONAL

sub-licensed from UniSoft Systems of Berkeley, is fully compatible with Bell Labs' System V. In addition, UniSoft, which has done more than 30 System V ports optimized for the 68000 chip, has added such enhancements as record-locking and IEEE floating-point capability. An optimizing C compiler comes with the system; other available languages include BASIC, COBOL, FORTRAN 77, Ada and Pascal.

Morrow has implemented a three-point DMA architecture that allows all three intelligent controllers — I/O, hard and floppy disk — to address areas of main memory directly through the bus.

Tricep's serial I/O controller communicates with terminals and printers at rates of up to 19.2 Kbaud. The unit's hard disk controller features a seek time of 85 milliseconds with the standard 16 MB drive, or 35-45 mS with the optional 34 MB drive; data-transfer rates are up to 625K bytes per second.

Future enhancements to Tricep will include boards providing Ethernet local-area network support and PC graphics capability.

For further information, contact Morrow, Inc., at 600 McCormick Street, San Leandro, CA, 94577, 415/430-1970.

UNIX IN JAPAN

AT&T announced in mid-July the opening in Tokyo of a UNIX Systems office which will be responsible for the marketing, licensing and support of software products in the Far East. The company also announced it will produce a standard Japanese version of UNIX System V.

Both announcements were made in Tokyo by John A. Hinds, vice president of market development for AT&T International, the overseas marketing unit of AT&T, and Thomas H. Crowley, software systems vice president in AT&T's Computer Systems division.

Larry L. Crume, head of UNIX Systems Engineering at AT&T Bell Laboratories and member of the *UNIX REVIEW* Editorial Review Board, was named director of the UNIX Systems office.

"I wish to heck someone would come up with SNA and BISYNC gateways for my UNIX™-based multiuser system."

"Someone has. UniPATH* by Pathway Design. Now save your pencils and call 800-343-0515. Or send this coupon."

I'm running out of pencils. Send me all the information you have on UniPATH. Soon.

Name ______________________

Company ______________________

Address ______________________

City ____________ State ________ Zip ________

Phone ______________________

PATHWAY

Pathway Design Inc., 177 Worcester Street
Wellesley, MA 02181 • 800-343-0515
(In Massachusetts, 617-237-7722)

Circle No. 32 on Inquiry Card

*UniPATH Gateways for UNIX III, UNIX V and XENIX™ operating environments.

UNIX is a trademark of Bell Laboratories XENIX is a trademark of Microsoft Corporation

CALENDAR

OCTOBER

October 2-4 East Coast Computer Faire, Boston, MA. Contact: Computer Faire, Inc., 611 Veterans Boulevard, Redwood City, CA 94063. 415/364-4294, or CFI in Newton, MA, 617/965-8350.
October 16-18 UNIXEXPO, The UNIX Operating System Exposition, New York, NY. Contact: National Expositions Co., Inc., 14 West 40th Street, New York, NY 10018. 212/391-9111.
October 25-28 The Second PC Faire, San Francisco, CA. Contact: Computer Faire, Inc. (see October 2-4).
October 30 Uni-Ops Monthly Meeting, Palo Alto, CA: "The Mac as a Workstation." Presentation and discussion sponsored by the non-profit UNIX user group. Contact: John Bass, 408/996-0557 or Paul Fronberg, 408/988-1755.

NOVEMBER

November 14-18 Comdex, Las Vegas, NV. Contact: The Interface Group, Inc., 300 First Avenue, Needham, MA 02194. 617/449-6600.
November 20 Uni-Ops Monthly Meeting, Palo Alto, CA. Contact: Uni-Ops (see October 30).

JANUARY 1985

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.

TRAINING CALENDAR

OCTOBER

October 1 AT&T Technologies, Sunnyvale, CA: "Shell Command Language for Programmers." Contact: AT&T Technologies, Corporate Education & Training, PO Box 2000, Hopewell, NJ 08525. 800/221-1647.
October 1 AT&T Technologies, Hopewell, NJ: "Fundamentals of the UNIX Operating System for Users." Contact: AT&T (see October 1).
October 1 AT&T Technologies, Lisle, IL: "Fundamentals of the UNIX Operating System for Programmers." Contact: AT&T (see October 1).
October 1 NCR Education Seminars, New York, NY: "UNIX Operating System." Contact: NCR Customer and Support Education, 101 W. Schantz Ave., Dayton, OH 45479. 513/445-3905.
October 1-2 Computer Technology Group, New York, NY & Washington, DC: "Shell Programming." Contact: CTG, Telemedia, Inc., 310 S. Michigan Ave., Chicago, IL 60604. 800/323-UNIX, or in Illinois, 312/987-4082.
October 1-3 CAPE Seminar, Rochester, NY: "A User-Oriented Evaluation Three-Day Seminar: UNIX." Contact: Center for Advanced Professional Education, 1820 East Garry St., Suite 110, Santa Ana, CA 92705. 714/261-0240.
October 1-3 Computer Technology Group, Chicago, IL: "UNIX Fundamentals for Programmers." Contact: CTG (see October 1-2).
October 1-3 Structured Methods Seminar, New York, NY: "UNIX System Internals." Contact: Structured Methods, 7 West 18th St., New York, NY 10011. 800/221-9274, or in New York, 212/741-7720.
October 1-5 Computer Technology Group, San Francisco, CA: "UNIX Internals." Contact: CTG (see October 1-2).
October 1-5 Bunker Ramo Information Systems, Trumbull, CT: "Advanced UNIX." Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06609. 203/386-2223.
October 2-5 Integrated Computer Systems Seminars, Los Angeles, CA: "Hands-on UNIX Workshop." Contact: ICSS, 6305 Arizona Place, Los Angeles, CA 90045. 800/421-8166, or in CA, 800/352-8251.
October 3-5 Computer Technology Group, New York, NY & Washington, DC: "Using Advanced UNIX Commands." Contact: CTG (see October 1-2).
October 4 AT&T Technologies, Hopewell, NJ: "Shell Command Language for Users." Contact: AT&T (see October 1).
October 4-5 Computer Technology Group, Chicago, IL: "Shell as a Command Language." Contact: CTG (see October 1-2).
October 4-5 Structured Methods Seminar, New York, NY: "Using **lex** and **yacc**." Contact: Structured Methods (see October 1-3).
October 8 AT&T Technologies, Sunnyvale, CA: "Fundamentals of the UNIX Operating System for Users." Contact: AT&T (see October 1).
October 8 AT&T Technologies, Hopewell, NJ: "UNIX System Internals." Contact: AT&T (see October 1).
October 8 NCR Education Seminars, Chicago, IL & Los Angeles, CA: "UNIX Operating System." Contact: NCR (see October 1).
October 8-12 Computer Technology Group, New York, NY & Washington, DC: "UNIX Internals." Contact: CTG (see October 1-2).
October 8-12 Computer Technology Group, Chicago, IL: "C Language Programming." Contact: CTG (see October 1-2).
October 8-12 Structured Methods Seminar, New York, NY: "UNIX System Workshop." Contact: Structured Methods (see October 1-3).
October 9-11 Computer Technology Group, San Francisco, CA: "UNIX Administration." Contact: CTG (see October 1-2).
October 9-12 Integrated Computer Systems Seminars, Los Angeles, CA: "Programming in C: A Hands-on Workshop." Contact: ICSS (see October 2-5).
October 10 AT&T Technologies, Lisle, IL: "Fundamentals of the UNIX Operating System for Programmers." Contact: AT&T (see October 1).

October 10-12 CAPE Seminar, Uniondale, NY: "A User-Oriented Evaluation Three-Day Seminar: UNIX." Contact: CAPE (see October 1).
October 11 AT&T Technologies, Sunnyvale, CA: "Shell Command Language for Users." Contact: AT&T (see October 1).
October 15 AT&T Technologies, Lisle, IL: "Shell Command Language for Programmers." Contact: AT&T (see October 1).
October 15 AT&T Technologies, Sunnyvale, CA: "UNIX System Internals." Contact: AT&T (see October 1).
October 15 AT&T Technologies, Dublin, OH: "UNIX System Administration." Contact: AT&T (see October 1).
October 15 NCR Education Seminars, Chicago, IL & Los Angeles, CA: "UNIX System Administration." Contact: NCR (see October 1).
October 15 Pulsetrain Seminar, New York, NY: "PROLOG Programming Seminar." Contact: Pulsetrain (see September 10).
October 15-16 Computer Technology Group, San Francisco, CA: "Advanced C Programming Workshop." Contact: CTG (see October 1-2).
October 15-16 Computer Technology Group, Chicago, IL: "Shell Programming." Contact: CTG (see October 1-2).
October 15-19 Bunker Ramo Information Systems, Trumbull, CT: "Intro to UNIX." Contact: Bunker Ramo (see October 1-5).
October 15-19 Plum Hall Training, New York, NY: "UNIX Workshop." Contact: Plum Hall, 1 Spruce Ave., Cardiff, NJ 08232. 609/927-3770.
October 16-18 Computer Technology Group, New York, NY & Washington, DC: "UNIX Administration." Contact: CTG (see October 1-2).
October 17-19 CAPE Seminar, Raleigh, NC: "A User-Oriented Evaluation Three-Day Seminar: UNIX." Contact: CAPE (see October 1).
October 17-19 Computer Technology Group, San Francisco, CA: "Advanced C Programming Under UNIX." Contact: CTG (see October 1-2).
October 17-19 Computer Technology Group, Chicago, IL: "Using Advanced UNIX Commands." Contact: CTG (see October 1-2).
October 17-19 Digital Seminar Program, San Francisco, CA: "Comprehensive Overview of the UNIX Operating System." Contact: Digital Educational Services, 12 Crosby Drive, Bedford, MA 01730. 617/276-4949.

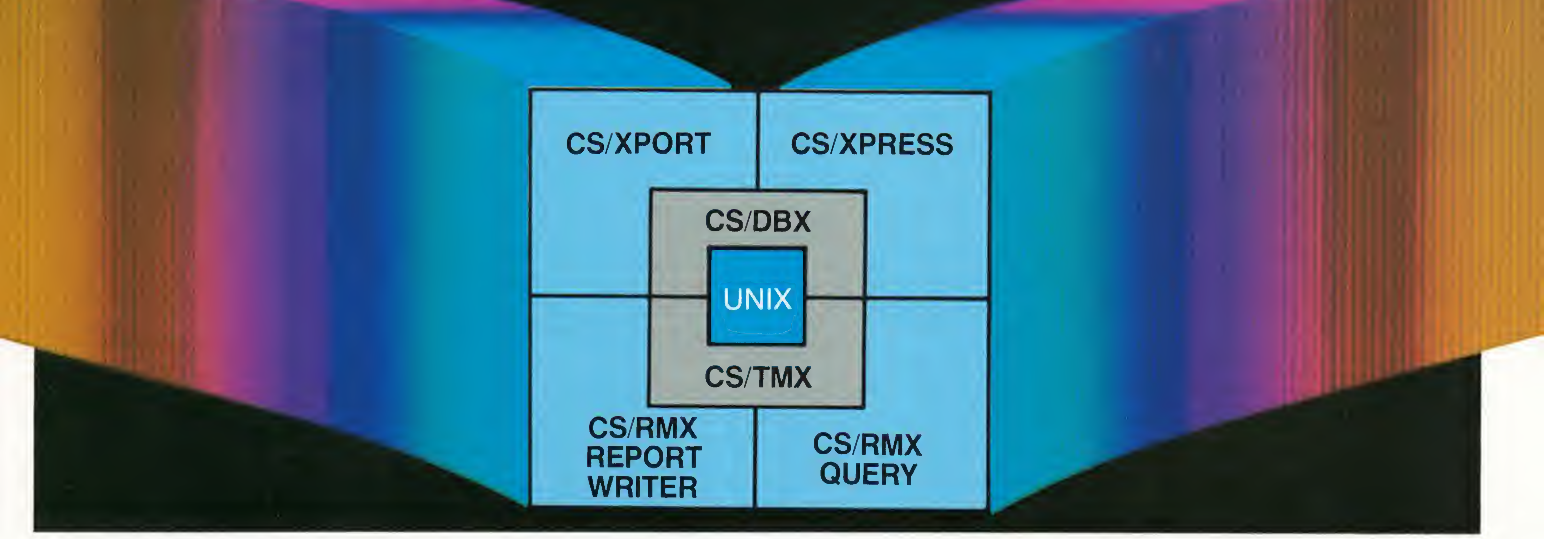
October 22 AT&T Technologies, Hopewell, NJ: "C Language for Experienced Programmers." Contact: AT&T (see October 1).
October 22 AT&T Technologies, Sunnyvale, CA: "Fundamentals of the UNIX Operating System for Programmers." Contact: AT&T (see October 1).
October 22 AT&T Technologies, Hopewell, NJ: "UNIX System Internals." Contact: AT&T (see October 1).
October 22 NCR Education Seminars, Houston, TX: "UNIX Operating System." Contact: NCR (see October 1).
October 22-23 Computer Technology Group, New York, NY & Washington, DC: "Advanced C Programming Workshop." Contact: CTG (see October 1-2).
October 22-26 Bunker Ramo Information Systems, Trumbull, CT: "Advanced C." Contact: Bunker Ramo (see October 1-5).
October 22-26 Computer Technology Group, San Francisco, CA: "Berkeley Fundamentals and '**csh**' Shell." Contact: CTG (see October 1-2).
October 22-26 Computer Technology Group, Chicago, IL: "UNIX Internals." Contact: CTG (see October 1-2).
October 22-26 Plum Hall Training, New York, NY: "C Programming Workshop." Contact: Plum Hall (see October 15-19).
October 23-26 Integrated Computer Systems Seminars, Boston, MA: "Hands-on UNIX Workshop." Contact: ICSS (see October 2-5).
October 24-26 CAPE Seminar, Baltimore, MD: "A User-Oriented Evaluation Three-Day Seminar: UNIX." Contact: CAPE (see October 1).
October 24-26 Computer Technology Group, New York, NY & Washington, DC: "Advanced C Programming Under UNIX." Contact: CTG (see October 1-2).
October 24-26 Digital Seminar Program, San Francisco, CA: "The C Programming Language." Contact: Digital Educational Services (see October 17-19).
October 29 AT&T Technologies, Lisle, IL: "C Language for Experienced Programmers." Contact: AT&T (see October 1).
October 29 AT&T Technologies, Hopewell, NJ: "UNIX System Device Drivers." Contact: AT&T (see October 1).
October 29 AT&T Technologies, Hopewell, NJ: "UNIX System/C Language Interface." Contact: AT&T (see October 1).
October 29 NCR Education Seminars, Dayton, OH & Houston, TX: "UNIX System Administration." Contact: NCR (see October 1).
October 29-November 2 Bunker Ramo Information Systems, Trumbull, CT: "Advanced UNIX." Contact: Bunker Ramo (see October 1-5).
October 29-November 2 Bunker Ramo Information Systems, Trumbull, CT: "C Programming." Contact: Bunker Ramo (see October 1-5).
October 29-November 2 Computer Technology Group, New York, NY & Washington, DC: "Berkeley Fundamentals and '**csh**' Shell." Contact: CTG (see October 1-2).
October 29-November 2 Plum Hall Training, New York, NY: "Advanced C Topics." Contact: Plum Hall (see October 15-19).
October 29-November 2 Structured Methods Seminar, New York, NY: "C Programming Workshop." Contact: Structured Methods (see October 1-3).
October 30 Computer Technology Group, San Francisco, CA: "UNIX Overview." Contact: CTG (see October 1-2).
October 30-November 1 Computer Technology Group, Chicago, IL: "UNIX Administration." Contact: CTG (see October 1-2).
October 30-November 2 Integrated Computer Systems Seminars, Boston, MA: "Programming in C: A Hands-on Workshop." Contact: ICSS (see October 2-5).
October 31-November 2 CAPE Seminar, Dallas, TX: "A User-Oriented Evaluation Three-Day Seminar: UNIX." Contact: CAPE (see October 1).
October 31-November 2 Computer Technology Group, San Francisco, CA: "UNIX Fundamentals for Non-Programmers." Contact: CTG (see October 1-2).

NOVEMBER

November 5 AT&T Technologies, Lisle, IL: "Fundamentals of the UNIX Operating System for Users." Contact: AT&T (see October 1).
November 5 AT&T Technologies, Lisle, IL: "UNIX System Device Drivers." Contact: AT&T (see October 1).
November 5 NCR Education Seminars, Chicago, IL: "C Programming." Contact: NCR (see October 1).
November 5 NCR Education Seminars, New York, NY: "C Programming." Contact: NCR (see October 1).
November 5-9 Bunker Ramo Information Systems, Trumbull, CT: "Intro to UNIX." Contact: Bunker Ramo (see October 1-5).
November 6 Computer Technology Group, Los Angeles, CA: "UNIX Overview." Contact: CTG (see October 1-2).
November 6-7 Computer Technology Group, Chicago, IL: "Advanced C Programming Workshop." Contact: CTG (see October 1-2).
November 6-9 Integrated Computer Systems Seminars, San Diego, CA: "Hands-On UNIX Workshop." Contact: ICSS (see October 2-5).
November 7 AT&T Technologies, Hopewell, NJ: "Fundamentals of the UNIX Operating System for Programmers." Contact: AT&T (see October 1).
November 7-9 Computer Technology Group, Los Angeles, CA: "UNIX Fundamentals for Non-Programmers." Contact: CTG (see October 1-2).
November 8 AT&T Technologies, Lisle, IL: "Shell Command Language for Users." Contact: AT&T (see October 1).
November 8-10 Computer Technology Group, Chicago, IL: "Advanced C Programming Under UNIX." Contact: CTG (see October 1-2).
November 12 AT&T Technologies, Hopewell, NJ: "Shell Command Language for Programmers." Contact: AT&T (see October 1).
November 12 AT&T Technologies, Lisle, IL & Sunnyvale, CA: "UNIX System/C Language Interface." Contact: AT&T (see October 1).
November 12 NCR Education Seminars, Dallas, TX: "C Programming." Contact: NCR (see October 1).
November 12-14 Computer Technology Group, Los Angeles, CA: "UNIX Fundamentals for Programmers." Contact: CTG (see October 1-2).
November 12-16 Bunker Ramo Information Systems, Trumbull, CT: "Advanced C." Contact: Bunker Ramo (see October 1-5).
November 12-16 Computer Technology Group, Chicago, IL: "Berkeley Fundamentals and '**csh**' Shell." Contact: CTG (see October 1-2).
November 13 Computer Technology Group, Boston, MA & New York, NY: "UNIX Overview." Contact: CTG (see October 1-2).
November 13-16 Integrated Computer Systems Seminars, San

Diego, CA: "Programming in C: A Hands-On Workshop." Contact: ICSS (see October 2-5).
November 14-16 Computer Technology Group, Boston, MA & New York, NY: "UNIX Fundamentals for Non-Programmers." Contact: CTG (see October 1-2).
November 14-16 Digital Seminar Program, Dallas, TX: "Comprehensive Overview of the UNIX Operating System." Contact: Digital Educational Services (see October 17-19).
November 15-16 Computer Technology Group, Los Angeles, CA: "Shell as a Command Language." Contact: CTG (see October 1-2).
November 26 NCR Education Seminars, New York, NY: "UNIX Operating System." Contact: NCR (see October 1).
November 26 AT&T Technologies, Sunnyvale, CA: "UNIX System Screen Editor '**vi**'." Contact: AT&T (see October 1).
November 26 AT&T Technologies, Dublin, OH: "UNIX System Administration." Contact: AT&T (see October 1).
November 26 AT&T Technologies, Lisle, IL: "UNIX System Internals." Contact: AT&T (see October 1).
November 26-28 Computer Technology Group, Boston, MA & New York, NY: "UNIX Fundamentals for Programmers." Contact: CTG (see October 1-2).
November 26-30 Bunker Ramo Information Systems, Trumbull, CT: "Intro to UNIX." Contact: Bunker Ramo (see October 1-5).
November 26-30 Bunker Ramo Information Systems, Trumbull, CT: "C Programming." Contact: Bunker Ramo (see October 1-5).
November 26-30 Computer Technology Group, Los Angeles, CA: "C Language Programming." Contact: CTG (see October 1-2).
November 26-30 Integrated Computer Systems Seminars, Washington, DC: "Hands-On UNIX Workshop." Contact: ICSS (see October 2-5).
November 28 AT&T Technologies, Hopewell, NJ: "Fundamentals of the UNIX Operating System for Programmers." Contact: AT&T (see October 1).
November 28-30 Digital Seminar Program, Dallas, TX: "The C Programming Language." Contact: Digital Educational Services (see October 17-19).
November 29-30 Computer Technology Group, Boston, MA & New York, NY: "Shell as a Command Language." Contact: CTG (see October 1-2).

Extend the power and performance of UNIX™ with CS/XTEND



Cincom Systems, the industry leader in data base/ data communication software systems, passes the power of its proven high performance software technologies for mainframe and minicomputers to the world of UNIX. CS/XTEND, our new fully integrated family of development and decision support components, makes it easy to create and implement even the most complex applications. And, our experience and expertise in developing quality software ensure that you get the high performance you need while using a minimum of machine resources.

CS/XTEND includes integrated solutions for professional system developers as well as productivity tools for non-technical information users:

CS/DBX—a high performance shared Data Base Executive with the power to handle multiple users concurrently accessing any number of files.

CS/XPORT—a Distributed System Interface that links computers so they can share data, providing the foundation for distributed data base networking.

UNIX is a trademark of AT&T Bell Laboratories

CS/TMX—a powerful Terminal Management Executive that simplifies interactive programming by making programs independent of the terminal, and provides overall network management.

CS/RMX—an easy-to-use Retrievals Management Executive that provides relational access to the data base for interactive inquiries and report writing by both end-users and computer professionals.

CS/XPRESS—a dynamic Application Builder that lets non-programmers automatically generate, maintain and access custom-built files of information.

CS/XTEND is fully compatible with Cincom's TOTAL® Data Base Management System. That means you can implement applications to operate on different types of computers using identical data base techniques.

Find out today how CS/XTEND extends the power and performance of your UNIX system. Simply call or write, Cincom Ventures Division, 2300 Montana Avenue, Cincinnati, Ohio 45211.

800-543-3010 (In Ohio: 513-661-6000)

Cincom Systems®

Excellence in Software Technology.

Circle No. 63 on Inquiry Card

ADVERTISERS INDEX

THE SANTA CRUZ OPERATION PRESENTS

XENIX NOW!

AN SCO PRODUCTION IN EXCLUSIVE ASSOCIATION WITH MICROSOFT CORPORATION
THE MULTIUSER, MULTITASKING PC BLOCKBUSTER "XENIX NOW!"
STARRING VISUAL SHELL • MULTISCREEN™ • MICNET • THE BERKELEY ENHANCEMENTS
AND INTRODUCING C-MERGE AS THE MS-DOS DEVELOPMENT ENVIRONMENT
FEATURING WORLD FAMOUS SCO TRAINING AND SUPPORT FOR DEALERS • END USERS • ISVs • OEMs
AND AN INTERNATIONAL CAST OF HUNDREDS OF XENIX APPLICATIONS
PRODUCED AND DIRECTED BY THE SANTA CRUZ OPERATION
SCREENPLAY ADAPTED BY THE SANTA CRUZ OPERATION FROM ORIGINAL STORIES BY MICROSOFT AND AT&T
IN BREATHTAKING SELECTABLE COLOR
NOMINATED FOR ★ BEST DOCUMENTATION! ★ BEST SUPPORT! ★ BEST TRAINING!
★ BEST ELECTRONIC MAIL AND NETWORKING! ★ MOST APPLICATIONS!
★ MOST COMPLETE UNIX SYSTEM!

NOW SHOWING AT SCO QUALIFIED DEALERS!

SCO
THE SANTA CRUZ OPERATION

RELEASED FOR IBM PC XT, AT&T PC, OTHER COMPATIBLES AND APPLE® LISA® EXCLUSIVELY THROUGH THE SANTA CRUZ OPERATION, 500 CHESTNUT STREET, P.O. BOX 1900, SANTA CRUZ, CA 95061. IMMEDIATE SEATING FOR OEMs.

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

M MULTIUSER OPERATION SUGGESTED
XENIX WILL TURN YOUR PC INTO A REAL COMPUTER

Circle No. 64 on Inquiry Card

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. 65 on Inquiry Card