VERSION 7 • XENIX • SYSTEM III • SYSTEM V • 4•2BSD • WORK-ALIKES

UNIX/WORLD

THE MAGAZINE FOR MULTIUSER, MULTITASKING SYSTEMS

JULY 1985

A TECH VALLEY PUBLICATION

UK £2.20 $3 IN USA

BEYOND C

FOURTH-GENERATION LANGUAGES

The Altos 486/Xenix: An AT Buster?

AT&T Bell Labs' Help Facility

Prolog, Modula-2 As A Unix System Tool

File System Checkups, C Standards, Part 2

0 71572 04782 07

How we improved Structured Query Language.

Actually, we didn't change a thing.

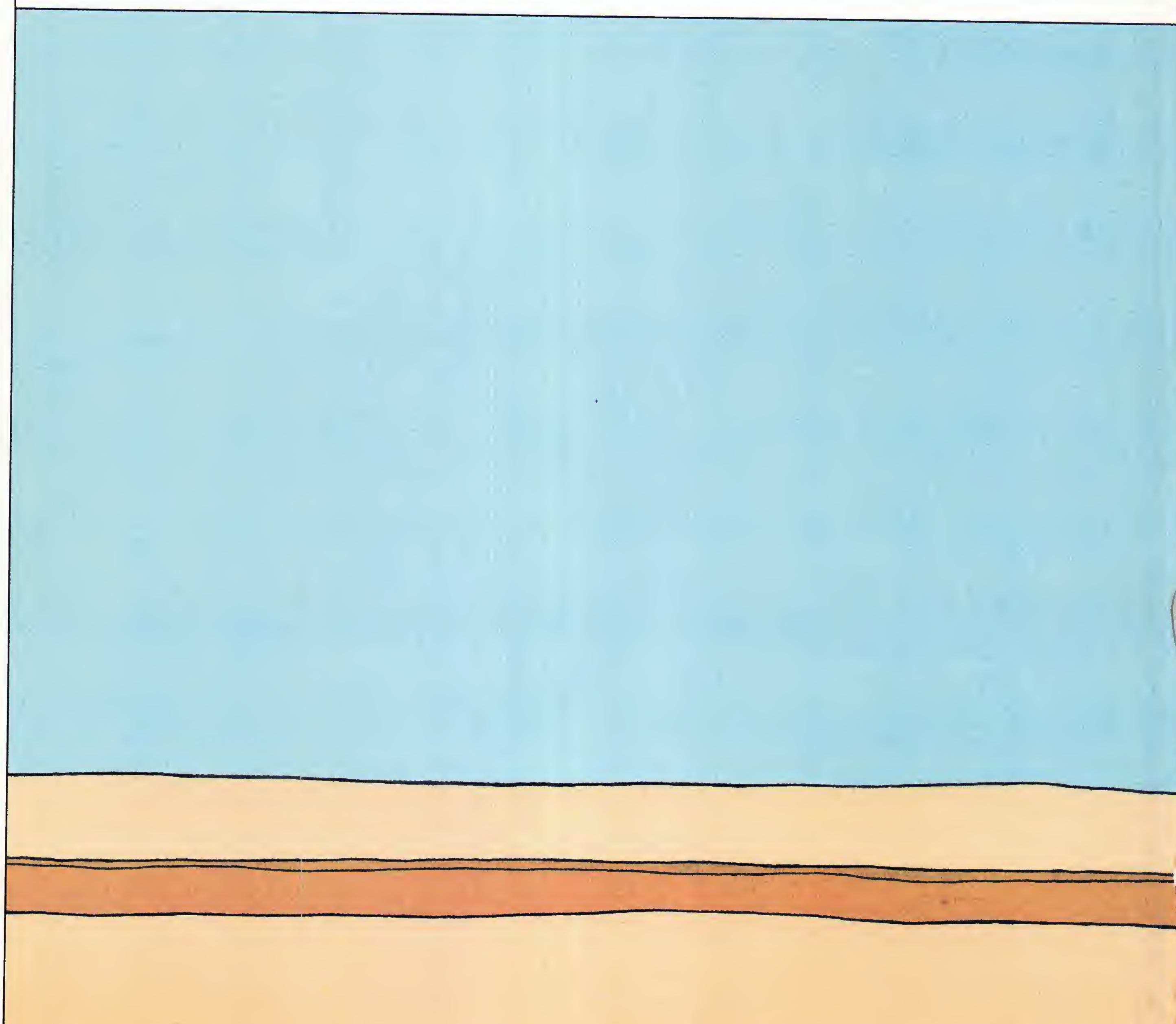
We just combined it with the best relational database management system.

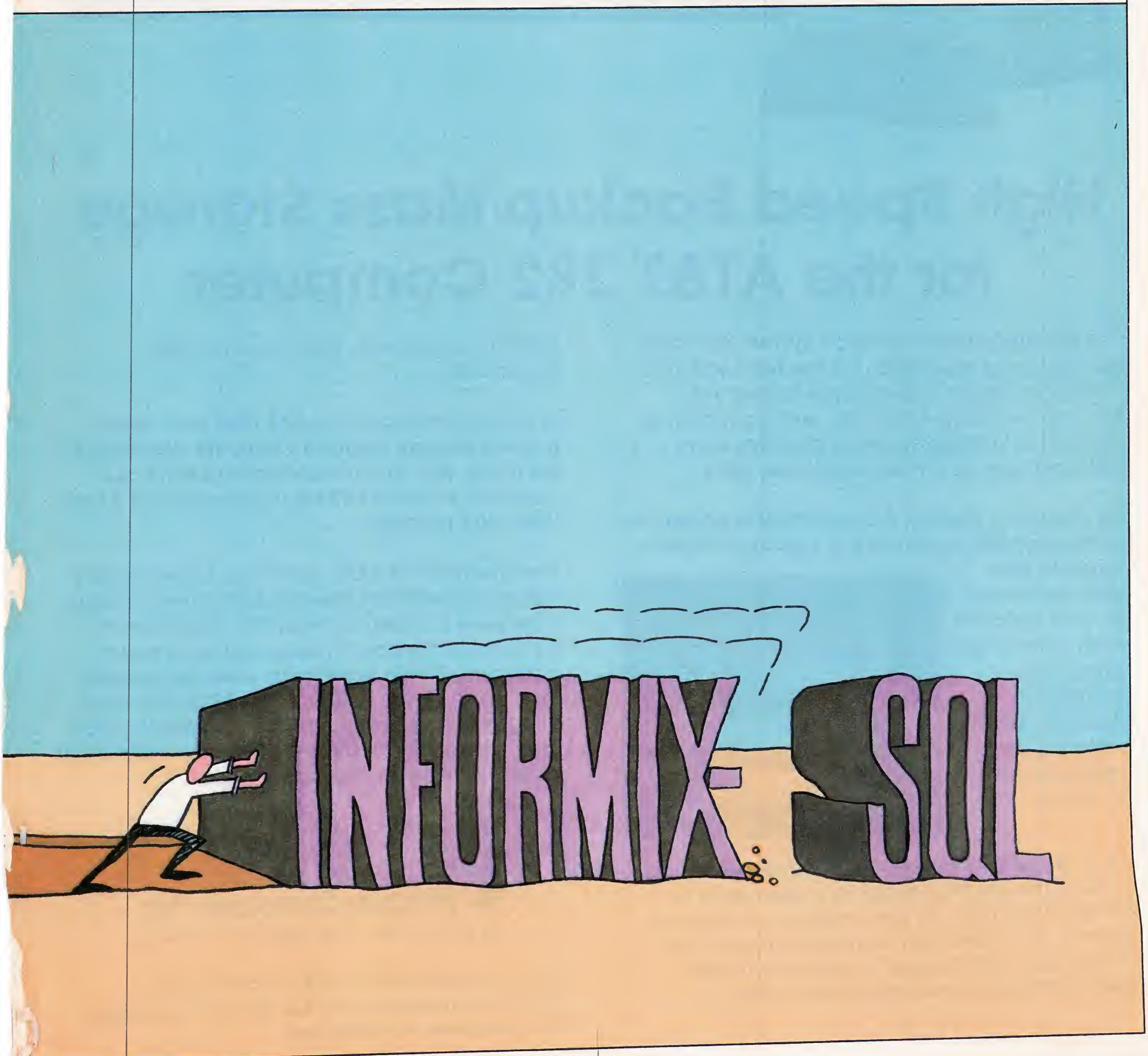
Introducing INFORMIX®-SQL.

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

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

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




INFORMIX-SQL

High Speed Backup/Mass Storage for the AT&T 3B2 Computer

ACI's 10 megabyte cartridge system provides dual purpose expansion for the AT&T 3B2. This cartridge system provides the fastest 3B2 backup available and also, unlike streaming tape drive storage systems, provides extra mass storage as a mountable disk drive.

As a backup device, ACI's cartridge system allows complete multi-volume backup. Single or multiple files may be saved to, and restored from, removable cartridges. Individual files are easily restored from multi-volume backups.

As a mountable disk under the Unix operating system, the cartridge system allows users to create directories on removable cartridges as desired. In a multi-user environment, administrators may find it useful to allow users their own cartridges, enabling them to have as much disk space as they need in 10MB increments.

In certain situations, the 3B2 user may desire greater storage capacity than the standard 32 MB of the 3B2. ACI's expansion system has a capacity of up to 240 MB of unformatted 24 ms fixed disk storage.

The operation of ACI's cartridge system is fully integrated with the standard AT&T Unix System V Release 2 system menus. This allows users to backup and restore files as well as to mount and unmount file systems by making menu selections. In-depth knowledge of the Unix operating system is not required, yet those familiar with Unix will find the cartridge system works as expected with all disk commands.

Absolute Computers, Inc.

1106 Clayton Ln. 108E • Austin, Texas 78723 • 512-458-5206

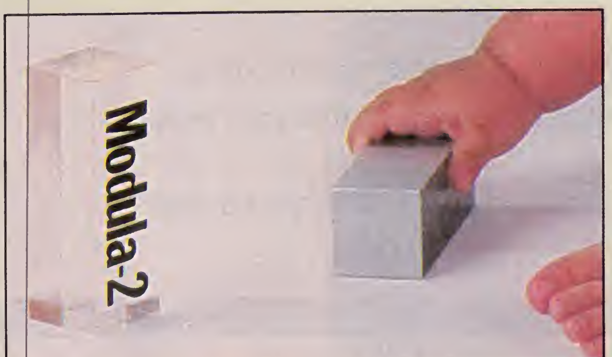
3B2 is a trademark of AT&T Technologies
UNIX is a trademark of AT&T Bell Laboratories

Please circle Ad No. 41 on inquiry card.

UNIX/WORLD

VOLUME II, NUMBER 6 A TECH VALLEY PUBLICATION

CONTENTS



THEMES

FEATURES

CONCENTRIC ASSOCIATES, INC.

WE DON'T PRODUCE TRAINING.

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

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

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

The first of these products is:

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

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

Call us for information about our on-line demonstration.

shacc
By Paul Ruel
Concentric Associates

SHACC UP WITH CONCENTRIC ASSOCIATES, INC.

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

Please circle Ad No. 5 on inquiry card.

NEXT MONTH

- MULTIPROCESSOR SYSTEMS
- NEW CHIP ARCHITECTURES

UNIX/WORLD (ISSN 0739-5922) is published monthly by Tech Valley Publishing, with offices at 444 Castro St., Mountain View, CA 94041, telephone 415/940-1500. RETURN POSTAGE GUARANTEED. Postmaster, send Form 3579 to UNIX/WORLD, P.O. Box 1929, Marion, OH 43305.

Subscriptions cost $18 for one year within the U.S.A. and its possessions. For Canadian subscriptions, add $6 for each year. For foreign subscriptions, add $20 each year for surface mail. Remit in U.S. funds drawn on a U.S. bank only. *Subscription questions and problems:* Contact UNIX/WORLD Subscription Department, P.O. Box 1929, Marion, OH 43305. Printed in United States of America. Requests for reprints or bulk orders should be addressed to the Merchandising Manager.

Address all editorial correspondence to the editor at UNIX/WORLD, 444 Castro St., Mountain View, CA 94041. Unsolicited manuscripts can be returned only if accompanied by a self-addressed, stamped envelope. UNIX/WORLD is not responsible for lost or damaged manuscripts, photos, or artwork.

The Power of Expression
The Expression of Power

Now, more than ever, expression of ideas through the printed word is a cultural imperative. Just getting through a day means dealing with massive amounts of information. And we each have a duality of roles — we consume information, we produce it.

As consumers of information, we must make choices. We choose material that appears more interesting, that is credible and clearly presented, that expresses a dimension of power through its appearance.

As producers of written information, we have the burden of creating that dimension. We have the responsibility to insure that our expressed thoughts become conveyed ideas.

Typesetting offers the solution. With the ability to choose typestyles and sizes, control spacing and position, you can give your message the added impact it needs to cut through the mass of material competing for your readers' attention. Contracts, correspondence, memos, reports, forms, documentation, training materials, everything you produce can carry the extra dimension of power that gets it read and remembered.

Laser xerography provides the means of typesetting in-house, effectively, immediately. VLS, through our *CompWare** product line, provides the means of using laser xerography.

Call now for more information on *DX-2700**, software support for the Xerox 2700* laser printer; our *DWB** release for AT&T's 3B* processors; VAX/UNIX* BSD Laser Printer support; and other *CompWare*products.

We'll show you how to express the power in your corporate publishing.

*DWB and 3B are trademarks of AT&T Technologies; DX-2700 and CompWare are trademarks of VLS, Inc.; Xerox 2700 is a trademark of the Xerox Corporation; UNIX is a trademark of AT&T Bell Labs; VAX is a trademark of Digital Equipment Corporation.

VLS INC. 44000 OLD WARM SPRINGS BLVD. FREMONT, CA 94538 (415) 490-3555

Please circle Ad No. 110 on inquiry card.

EDITOR'S CONSOLE

Here's another "Editor's Console" you can put in the debunking popular myths and industry trends category. This time, man/machine interfaces. Countless computer newspaper and magazine pages have detailed discussions of the new man/machine interfaces. In lay parlance, this means things like mice, windows, menus, soft-function keys, on-line help tutorials—in short, the stuff that glitzy press conferences, vacuous *Business Week* cover stories, innocuous product reviews, and the buzzword *user-friendly* are made of.

The only problem with all this is that magazine reviews and feature articles are as close as most business users will ever get to these products. Now some might say that's a bad thing, that most business users should use these aids in man/machine interaction.

I, for one, happen to disagree. Indeed, I would strenuously argue that most general business users *don't* want or need a mouse. I know I sound like some kind of conservative reactionary, the computer industry equivalent of a Jerry Falwell. But consider for a moment that history is on my side.

Every product of this nature yet to come to market has ultimately been a flop, failing to achieve its inventor's goals to revolutionize technology, capture the popular imagination (and market share), and shift the balance of power away from IBM. Not one, I remind you. Not Star, not VisiOn, not Lisa I, II, or III (the Macintosh XL), and, now, not the Mac.

Most of these "innovations" are but the PR and advertising hype of Silicon Valley's version of medicine men, garage mechanics, and soda pop salesmen. Indeed, most of these user interface innovations are little more than toys, of dubious value to the business user who wants to jot down a memo, redo a spreadsheet, or call up a file from the corporate database. But we shouldn't let a few bad apples spoil the lot.

Not all is lost, however. Real advances in man/machine interfaces are being achieved, mostly through fourth- and fifth-generation computer languages—in short, the stuff this month's issue is made of. In the meantime, however, users will continue to expand their computer usage through and only through the availability of application software that hit common chords among vast numbers of users. Otherwise, they'll just end up playing games.

Elsewhere, observant readers will notice some changes this month and last in the lineup of our regular editorial departments. Gone are "From the Publisher's Desk" (and the publisher) and "News from AT&T." In are a new publisher (Robert A. Billhimer) and "For the Record," a new monthly column of up-to-the-minute news, rumors, and noteworthy events. Out last month but back this issue is Bill Tuthill's "Starter Kit."

Philip J. Gill
Editor-in-Chief

UN

It's a big word these days. And no one has a bigger commitment to it than Apollo. In fact, we've raised the UNIX™ standard to new heights.

We're the first major computer company to implement both AT&T System V and Berkeley 4.2 on a professional workstation. So you can run either standard or both simultaneously.

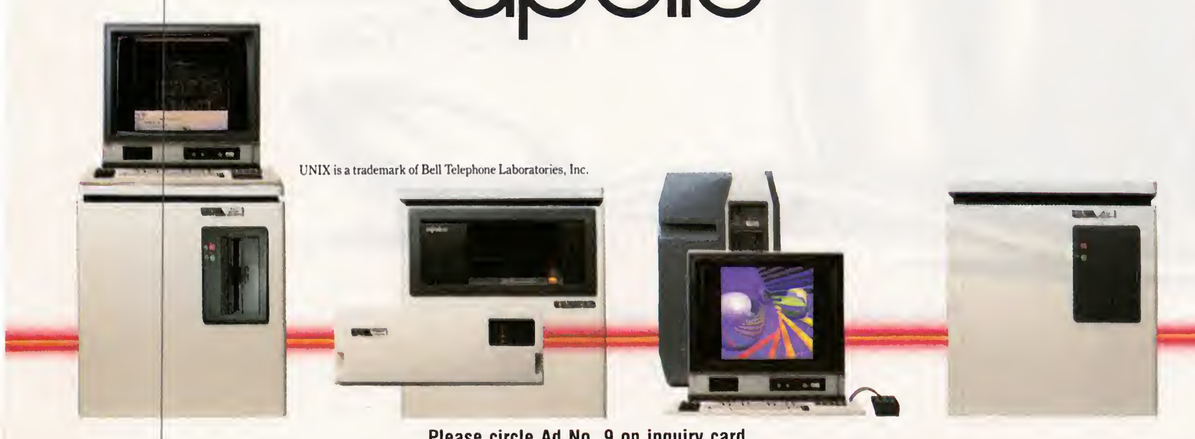
More important, we're the only company that gives you UNIX in a powerful, distributed processing environment. One that features 32-bit dedicated workstations, high resolution graphics, and a high-speed local area

IX

network that lets you transparently share information and resources.

Today, the right UNIX standard is a choice of standards. For a demonstration of a standard that's anything but ordinary, call (617) 256-6600 x4419. Or write Apollo, 330 Billerica Rd., Chelmsford, MA 01824, MS31.

apollo



UNIX™ & TODAY

The Magical Mixture

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

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

Features:

- Non-procedural specification with powerful recursive logic and Decision Tables
- Synonyms, Menus, Prompts, Helps and Defaults for streamlined definitions
- Tailoring control allowing multi versions of a base application
- Screen Painter
- A Report Generator which includes a Painter
- Push-button Self-documentation
- Audit Trails
- Source-code security through run-time only configurations
- Developed Applications instantly portable across **UNIX**-based machines

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

Call **TODAY —(408) 727-4464**

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

Please circle Ad No. 44 on inquiry card.

FOR THE RECORD

Top of the News . . . Some interesting products made their debuts at the Unix Systems Expo/Spring '85 at San Francisco's Moscone Center. Leading the list is **Relational Technologies Inc.'s** new RTI Ingres PC Link software, a PC-to-Unix system connection that permits the transparent information exchange between Ingres databases on Unix system hosts and IBM and compatible PCs running the most popular applications software, all through a Lotus 1-2-3-like interface (see Multiplex/Ingres PC Link preview in "Exposing the PC/Unix System Connection," this month) . . . Elsewhere on the DBMS front, **Unify Corp.** inked new OEM deals with **AT&T** for the Unix PC and with **NBI Inc.** for its Unix system-based gear. **Network Technologies Inc.**, also known as Neti, unveiled what is apparently the first in its class, an electronic conferencing software package called eForum . . .

Computer Consoles Inc., whose Officepower office automation system is reviewed in this issue, announced a new $16,500 floating point option for its 6/32 high-end supermini. Also, CCI's Irvine, Calif.-based Commercial Products Group said it had signed a new OEM agreement with England's only remaining indigenous mainframer, **International Computers Limited** (ICL). ICL has contracted to buy a minimum of 100 6/32s a year over five years and will also sell CCI's Officepower software on the 6/32 and other non-CCI-made ICL hardware. Company executives say they have shipped about 100 6/32s so far, with about 40 or so new 6/32s going out the door every month. They predict ship rates of 100 a month by December, and as many as 1000 next year . . .

Challenge Issued . . . Also during the Unix Systems Expo, Berkeley, Calif.-based **BASIS Inc.** (which stands for Bay Area Shared Interactive Systems) said it would donate a valuable Heuer Chronograph to the individual or organization who produces the best test of the Unix system's real-world performance—meaning a set of benchmarks that accurately portray Unix system performance in business applications such as word processing, accounting, database management, and the like. BASIS won the watch by being the first to successfully take the Plexus challenge, a test later taken by UNIX/WORLD Editor-at-Large Bruce Mackinlay. Interested parties should contact BASIS marketing director Gary Babcock for details . . .

Mis-Fortune . . . As predicted, a major shakeup did occur at beleaguered **Fortune Systems Corp.**, Redwood City, Calif., the "erstwhile darling" of the Unix system-based supermicro market, but since its IPO, a virtual financial dry hole. Officially, April's shakeup went like this: In a tersely worded press release, Fortune disclosed "a number of management changes which it believes will aid the company in achieving 1985 goals." In plain English, those management changes included promotions and demotions and the layoff of approximately 100 staffers "to properly size the company for an anticipated slower growth curve."

Prominent senior-level managers shuffled about were James Ferenz, formerly vice president, manufacturing, now senior vice president, operations; and Bob Ruebel, formerly senior vice president, marketing, now holding down the new position of executive vice president, corporate development. To top it all off, despite repeated public statements from the firm's new turnaround management team (headed by ex-Xeroids Jim Campbell and Bob Ruebel) that it would show a profit by the fourth quarter '84, the headquarters on Twin Dolphin Drive is still awash in a red sea of losses. And the firm projected that revenues for the first quarter '85 will be "somewhat disappointing," at 20 to 30 percent below the same quarter last year. Word is that the actual loss for this quarter will be between $1.5 million and $2 million . . .

Rumors of the Month . . . Those of you anxiously looking to AT&T to resolve your communications and networking dilemma will apparently have to wait at least until this December or next January. That is when, sources say, AT&T will unveil the next major release of System V, which will include the touted "Streams" communications feature and record and file locking. Till then, Bill Joy's NFS may well have free reign . . . Sluggish sales forced a major layoff in late April at **Sydis Inc.**, a San Jose, Calif.-based maker of integrated voice, data, and text office systems. Sources say that between 40 and 50 percent of the company's work force was let go, amounting to about 50 staffers. Gone are almost all marketing personnel. All this despite some big-figure OEM contracts, including one supposedly worth in excess of $150 million with GTE Corp. . . .

FOR THE RECORD

Microport Corp., the **Digital Research Inc.** spinof f charted with doing Unix System V ports for commercial OEMs, has been forced to retrench after losing a bid on a contract to do the System V port for the upcoming **Intel Corp.** 80386 chip, according to company insiders. However, the same confidants said the as yet unknown winner would be revealed in the next 60 to 90 days or so . . . Meanwhile, Microport chairman Rod Turner turned up as executive vice president of marketing at **Symantec Inc.,** Santa Clara, Calif., the software firm headed by ex-**Lotus Development Corp.** honcho Vern Rayburn. Turner, an **Ashton-Tate** alumnus who surfaced from semiobscurity last January to take the Microport helm, will fulfill his Microport duties on a part-time basis. Symantec, by the way, bought **C&E Software,** one of whose founders was ex-DRI VP Gordon Eubanks . . .

Look for a major new graphics software product this month from **Visual Engineering Inc.,** San Jose, Calif.. Called Display WorkBench during the development stages, the package is said to be highly modular and contain an integrated database along with other such facilities of importance to graphics software developers, OEMs, and VARS . . . Almost a year after its introduction at Comdex/Spring '84, **NEC Information Systems** has still to deliver its PC-UX Unix System III option on its highly touted APC-III, IBM-compatible PC. Word is now that the Unix system option will be available this month . . .

Contracts . . . **Sun Microsystems Inc.** and **Apple Computer** have announced an OEM agreement under which Sun will purchase and resell the Apple LaserWriter printer. The agreement calls for the purchase of $10 million worth of LaserWriters over the next two years . . . **Precision Visuals International,** the overseas subsidiary of Precision Visuals Inc., has signed its first major European license for GK-2000 with BritOil PLC . . . The new AT&T Unix PC is available for immediate off-the-shelf delivery from the four distribution organizations in the **Ducommun Electronics Group,** Cypress, Calif. . . .

Pyramid Technology Corp. has signed a major OEM agreement with VMark Computer Inc. of Natick, Mass., under which Pyramid will deliver several hundred Pyramid 90x and 90MX supermini computers valued in excess of $10 million over a three-year period. VMark makes UniVerse, its new proprietary database management system designed to execute Prime Information and Pick-like applications in the Unix system environment . . . **Stratus Computer Inc.,** Marlboro, Mass., has signed a multiyear OEM packet with System Development Corp. (SDC) of Camarillo, Calif., a subsidiary of Burroughs Corp. Under the terms of the agreement, SDC will market Stratus products for SDC's markets, primarily to the U.S. government and its agencies, for air space management, command and control, signal processing, computer security, weapons systems, communications networks, and office automation . . .

Computer Consoles Inc., Rochester, N.Y., has had its Officepower integrated turnkey office automation system selected to automate the offices of California Governor George Deukmejian. The system will automate both the governor's Sacramento and Los Angeles offices. Initial installation has been completed . . . **TouchStone Software Corp.** has signed a one-year agreement with First Software Corp. of Lawrence, Mass. The agreement provides for the distribution of TouchStone's Connectables Network software PCworks, UniHost, and MacLine. First Software will distribute the Connectables network software nationally as well as internationally on a country-by-country basis . . . **Systems Strategies Inc.,** New York, N.Y., a developer of IBM emulation software, has signed four OEM agreements worth over $500,000. All four companies have signed licensing agreements for Unix-based or Unix-compatible software. They include Siemens (West Germany), Tolerant Systems (San Jose, Calif.), Megadata Corp. (Bohemia, N.Y.), and Sequoia Systems (Marlborough, Mass.). The firms have selected Systems Strategies to supply IBM emulation packages for distribution with their hardware, thus allowing their non-IBM systems to communicate with IBM mainframes . . .

Noted . . . **Tech Valley Publishing Co.,** the parent of UNIX/WORLD Magazine, has announced the appointment of Robert A. Billhimer as new corporate president and publisher . . .

Encore Computer Corp. has filed for an initial offering of 5 million shares of common stock at an offering price of $5 per share. Hambrecht & Quist Inc. is managing the offering. ☐


"A UNIX™ TO BE PROUD OF!"
—Kaare Christian, PC Magazine
SCO
XENIX Operating System
Mankind searched the world over
for the multiuser operating system of the future.
Then IBM® chose XENIX® for the PC AT. And the future was *now*.
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
©MCMLXXXIV The Santa Cruz Operation, Inc.
UNIX is a trademark of AT&T Bell Laboratories • XENIX is a registered trademark of Microsoft Corporation • IBM is a registered trademark of International Business Machines Corporation • Apple and Lisa are registered trademarks of Apple Computer, Inc. • Multiscreen is a trademark of The Santa Cruz Operation, Inc.
Please circle Ad No. 11 on inquiry card.

THIS AD IS FULL OF LIES.

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

LIE 1.

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

NO!

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

Prevail components include:

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

LIE 2.

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

WRONG!

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

LIE 3.

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

NONSENSE!

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

LIE 4.

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

TOTALLY INACCURATE!

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

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

DISCOVER THE WHOLE TRUTH.

Discover PREVAIL

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

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

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

NAME ____________________ COMPANY ____________________
(PLEASE TYPE OR PRINT.)

ADDRESS ____________________

CITY ____________________ STATE ________ ZIP ________

TELEPHONE ____________________ HARDWARE TYPE ____________________

☐ END-USER ☐ VAR/DISTRIBUTOR ☐ OEM

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

Please circle Ad No. 19 on inquiry card.

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

mail

This space is yours. Tell us what we're doing right—or wrong. Raise issues and ideas that other readers may respond to. Propose projects. Reject market trends. Take a stand.

ALTERNATIVES TO SNA

Dear Editor:

There may be "No Way Around Big Blue's SNA" (UNIX/WORLD, April 1985), but there certainly are alternatives to implementing SNA on your Unix machine.

A large-scale IBM (or compatible) communications will use an IBM 37x5 (or NCR Comten 3690 or Amdahl 47x5) front-end processor running NCP. IBM saw the need to include a gateway to X.25 in its architecture, and any large SNA operation has the ability to talk X.25. X.25 is an international standard that is already available on many Unix system machines (Fortune, Intel, Altos, Apollo, etc.). X.25 can communicate to SNA networks via PDN's, on private networks, or on a single line with up to 4096 virtual circuits.

It would be better to implement a communication scheme based on an international standard than on a proprietary architecture controlled by a company that is a competitor of yours.

Do we want the company that made MS-DOS a "standard" to be in control of our communications? The world is not all Blue.

Sincerely,

Donald Berryman
Communications Analyst
TITN Inc.
Edina, Minn.
UUCP: `ihnp4!dicomed!titn!don`

WRONG IMPRESSIONS

Dear Editor:

In your article "Beyond PCs," by Gilbert and Branaman (UNIX/WORLD, March 1985), I can't tell if there is a type "O" or a big misunderstanding. The section titled "The Local-Area Network" leads one to believe that AT&T's alternative to the LAN is Ethernet while IBM's is Token Ring. Then it says PBX-based LAN is a strong alternative to LANs from IBM and AT&T.

First of all, Ethernet belongs to Xerox.

Second, AT&T has several offerings to the LAN market, two of which *are* PBX-based. Examples of this are System 85 and System 75. Both offer simultaneous voice/data over the same pairs of wire. This does not require rewiring as was implied; *that is our whole point*—to protect the wiring investments already made.

The second alternative, Information Systems Network (ISN) from AT&T, uses existing twisted pair already in place. Fiber optics is used between concentrator or host computers when needed. Twisted pair (the wire used for telephones) goes out to connect terminals and other devices.

There was no misinformation given, but because of the way it was stated, many people probably got the wrong impressions.

Sincerely,

Cindy Hostetler
Technical Consultant
AT&T Information Systems
Los Angeles, Calif.

Editor's Reply: Thanks for the clarification. You are indeed right, and our authors explained the situation pretty much the same as you describe in a section of that same article on LANs. Unfortunately, that detailed discussion of LAN requirements had to be left out for space considerations. We plan to fully address the topic of LANs in our December issue on communications and networking. Until then, my apologies.
—Philip J. Gill

NEEDS A HAND

Dear Editor:

I have been a subscriber to UNIX/WORLD for several months and wish to compliment you and your staff on the fine job you are doing in promoting the Unix system and its variants.

We are users of Xenix, which we run through a Radio Shack Model 16, using one terminal and four 8-Mbyte hard disks. The system runs well for us, but we do have one difficulty with which I hope you will be able to help us.

What we need is a simple and quick backup system. We have been backing up to floppies, but the entire procedure ties up the computer for about three hours. Hence, we only do the backup once every two or three weeks although we would like to have the ability to back up daily or even more often.

Radio Shack does not have a storage device; I have checked with several firms, including Hewlett-Packard and Altos, but I have been told that they have nothing either. I eagerly go through UNIX/WORLD each time it is published in the hope that I will find some information on backups, but nothing has yet come to my attention. Can you help?

We will be grateful for any assistance you may be able to provide.

Yours truly,

John T. Zubal
2969 W. 25th St.
Cleveland, Ohio 44113

Editor's Reply: Can anybody help? Please write Mr. Zubal if you've got the answer to his needs.

—Philip J. Gill
Editor-in-Chief

STILL THINGS TO DISCUSS

Dear Editor:

I would like to address the question "Is There Anything Left to Argue About?" which was raised in the March 1985 issue of UNIX/WORLD. The answer, of course, is yes.

In his article Stanley Shein asserts that all the major computer vendors regard the Unix system as being suitable for the office because they all offer a Unix system of some kind. As he points out, they are not babes in the wood; they know that you cater to the customer's wants or you lose business. Whether the customer's wants are rational or well informed is largely immaterial. As long as the customer perceives the Unix system as desirable, the vendors will supply it.

I have a copy of a sales memorandum distributed to DEC's sales force that makes it clear that Ultrix-32 is to be offered only to those customers already convinced that the Unix system offers them an advantage. If the Unix system is so great in the office, why isn't it being promoted by these vendors as the operating system of choice?

The answer is simply that the Unix system *per se* is not suited to the office, and it takes a lot of enhancements to make it even accessible to nonprogrammers. Production users need production operating systems—so why take a good development system and bastardize it to make a bad production system? It's not worth the effort, especially when we already have CP/M and MS-DOS.

The Unix system, like Pascal and relational DBMSs before it, is being sought by ill-informed users who mistake popularity for wide applicability. The Unix system has a place, but it isn't in the office.

Sincerely,

Roy Hann
Mountain View, Calif.

Editor's Reply: The Unix system does indeed have a place in the office, and a prominent one at that. Answer this one question honestly: For what environments are the hottest office automation and electronics publishing products now being developed? Only if you answered "For the Unix system and the Macintosh" would you be right. As for the DEC letter you refer to—of course, DEC is fighting the Unix system, but only to keep its installed base locked up under proprietary operating systems. Besides, we all know that nobody really wants VMS! —Philip J. Gill

BEWARE: READING THIS MAGAZINE . . .

Dear Editor:

Please be considerate to those of us ("non-wizards") on the other side of your magazine cover.

My husband spends many constructive hours inside UNIX/WORLD, and although it helps him become more of an expert in the state of the art, I hope he doesn't evolve into one of those futuristic types on your March 1985 cover.

Sincerely,

Ellen Faden
Fremont, Calif.

AT&T UNVEILS UNIX PC, STARLAN

BY OMRI SERLIN

A funny thing happened on the way to the 7300 inaugural party: Not only did the machine get a new name—the Unix PC—but, more importantly, AT&T seems not to be as bullish about its prospects as one would expect.

This mood evidenced itself in a series of telltale signs at the San Francisco unveiling in March. For example, Jack Scanlon, who heads the Computer Systems Division in AT&T Information Systems (AT&T-IS), suggested that the StarLAN network (not the Unix PC) was "clearly the star of this show."

Partly, this attitude is a result of history. The Unix PC was commissioned to Convergent Technologies by AT&T-IS in the days when Archie McGill was still around and when it was still called American Bell. Jack Scanlon and his Computer System Division troops, all veteran Bell Labs and Western Electric hands, at the time were part of a different AT&T group and had little enthusiasm for the project. They felt (and, apparently, still feel) that AT&T needs no outside help in designing computers, that they could have handled the job, and that going outside for such equipment adversely affects AT&T's prestige as a leader in electronic technology.

Then again, the emphasis on StarLAN might well be a welcome indication of acquired savvy. A year ago, when AT&T launched its commercial computer entry, the company's expectations were naively (arrogantly?) driven by a technological culture that paid very little attention to the realities of the marketplace.

A new frame of mind may be emerging, forged through AT&T's very unsatisfactory first year as a commercial computer supplier. The company, one hopes, may have learned some important lessons, the most significant of which is that IBM, not the Unix system, has laid the ground rules for the computer marketplace; and that a necessary condition for any serious progress here must be through coexistence with, not departure from, the installed IBM base.

The key hope AT&T pins on StarLAN is that it will be the "door opener" that will allow AT&T computers to gain entry into the important corporate marketplace, not by replacing IBM systems, but by allowing the two brands to communicate effectively. That is clearly the correct strategy.

Unfortunately, AT&T has priced the network interfaces and software far too conservatively. Unless an IBM shop can use existing wiring, there isn't much of a cost incentive for it to be more attracted to StarLAN than to the Ethernet, 802.3, or IBM PC Net alternatives (see Figure 1).

STARLAN

The most important feature of StarLAN (which is still being drafted as a proposed standard in an IEEE 802.3 subgroup) is that it can utilize unshielded twisted-pair telephone wiring that already exists in most plants and buildings worldwide. StarLAN can use the unused in-wall pairs, eliminating the substantial labor and cabling costs typically associated with LAN installations.

Even when new cabling must be installed, this is more readily achieved with the lightweight and unobtrusive twisted pair, as compared with the unwieldy 0.4-inch coaxial cable specified in the Ethernet and 802.3 standards. Even the 0.25-inch coaxial cable specified for "Cheapernet," another 802.3 variant now under consideration, is more difficult to handle.

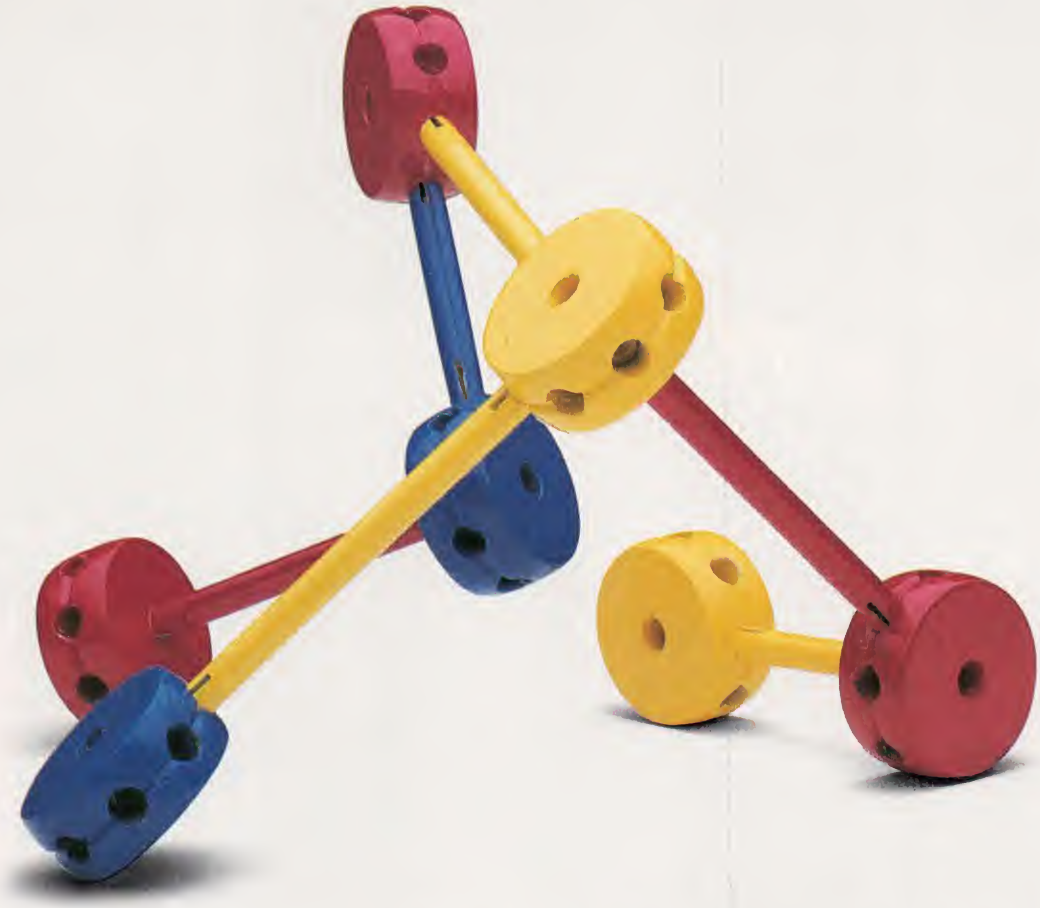
Low interface cost is another advantage of StarLAN. The StarLAN interfaces for AT&T's 6300, IBM PC, and the Unix PC are a full $100 cheaper than the equivalent IBM PC Network interface and are $50 cheaper than the least-expensive Ethernet/802.3 interface (see Figure 1). A further cost reduction will occur later this year when Intel begins delivering its 82588 low-cost network controller chip. (Current AT&T StarLAN interface boards carry the more expensive 82586 chip.)

StarLAN has another important cost advantage. For its interface to the transmission medium, it uses a simple, inexpensive RS-422 driver/receiver chip, whereas IBM has to use a far more expensive RF modem.

Of course, there is no free lunch. In return for the cost advantages, StarLAN gives up some speed (1 Mbps compared to 10 Mbps in 802.3 and Ethernet, 2 Mbps in the IBM PC Net). This is due partly to the use of unshielded twisted pair, and partly to the more rudimentary "collision detect" logic in the 82588 chip.

Note that the StarLAN interface board for the AT&T 6300 will, of course, fit in the IBM PC and other IBM clones as well. This allows the IBM PCs to coexist on the same StarLAN network that also supports

Continued on page 23



Networking Software So Simple To Use, It's Child's Play.

What do you think of when someone says "network"? Hardware? Complexity? Money? At TouchStone, we think networks should be simple.

The Connectables Software from TouchStone

The Connectables software — PCworks,™ MacLine,™ and UniHost™ — lets you build a network out of a collection of computers. Without special hardware, unnecessary complexity, or a lot of money.

UNIX + PC + Macintosh

Three great computers with different processors, different operating systems, and different data formats. The Connectables software lets you connect these systems together in whatever way you'd like to use them. Share electronic mail, exchange all types of files, run a program on another system, or use a remote computer's disk or printer. The only hardware you'll need is a standard cable or telephone line and a serial data channel.

It's So Easy.

The Connectables software will take care of all the details — dial the right numbers, ask you what you want to do, and figure out the rest. Move a spreadsheet from your PC to a UNIX™ system, or transfer a text file from the UNIX system to your Macintosh™ for editing. Print the file later on either system, or use another computer to backup your data. With The Connectables software, things that should be easy to do, are easy to do.

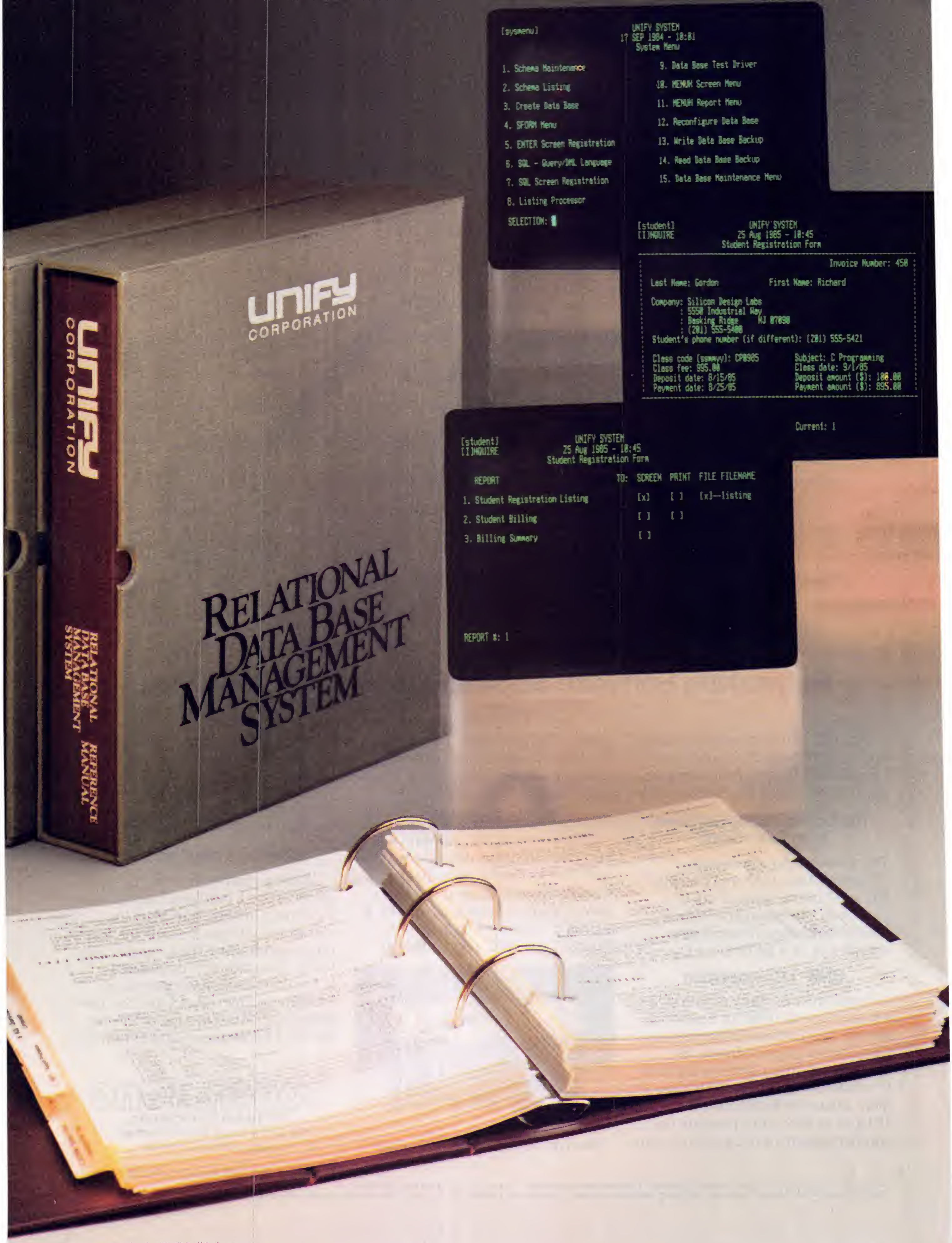
THE CONNECTABLES™:
FAMILY OF COMMUNICATION SOFTWARE PACKAGES.

Yesterday, Today, Tomorrow?

Making computers work together shouldn't be a problem. TouchStone's PCworks, UniHost, and MacLine software packages are only the first three members of The Connectables family. TouchStone is committed to the continued development of software products that solve compatibility problems. Even the problems we won't know about until tomorrow.

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

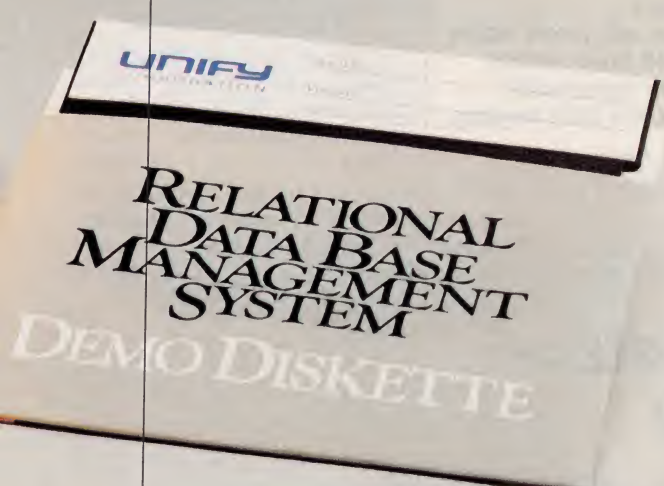
Please circle Ad No. 82 on inquiry card.


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

UNIFY. THE MIGHTY EASY DBMS.

UNIFY isn't only the fastest and most powerful of all UNIX*-based data base management systems—it's also one of the simplest to use. Which is why some 75 percent of those who see our manuals and buy a DBMS, buy UNIFY.

UNIFY guides the nonprogrammer through data base development with minimal steps, comprehensive menus, on-line HELP, elementary Query By Forms capability, and clear documentation.



UNIFY expedites applications development for users of all skill levels with features like PAINT, for effortless forms design; SQL, the powerful, English-like query language; and a menu-handler that lets you quickly compile screens, queries and reports into menus customized to each user.

And for the skilled programmer, UNIFY offers the most extensive host language interface, for limitless flexibility when you need it.

All of which makes it mighty easy to see why UNIFY has been selected by more computer manufacturers than any other UNIX-based DBMS.

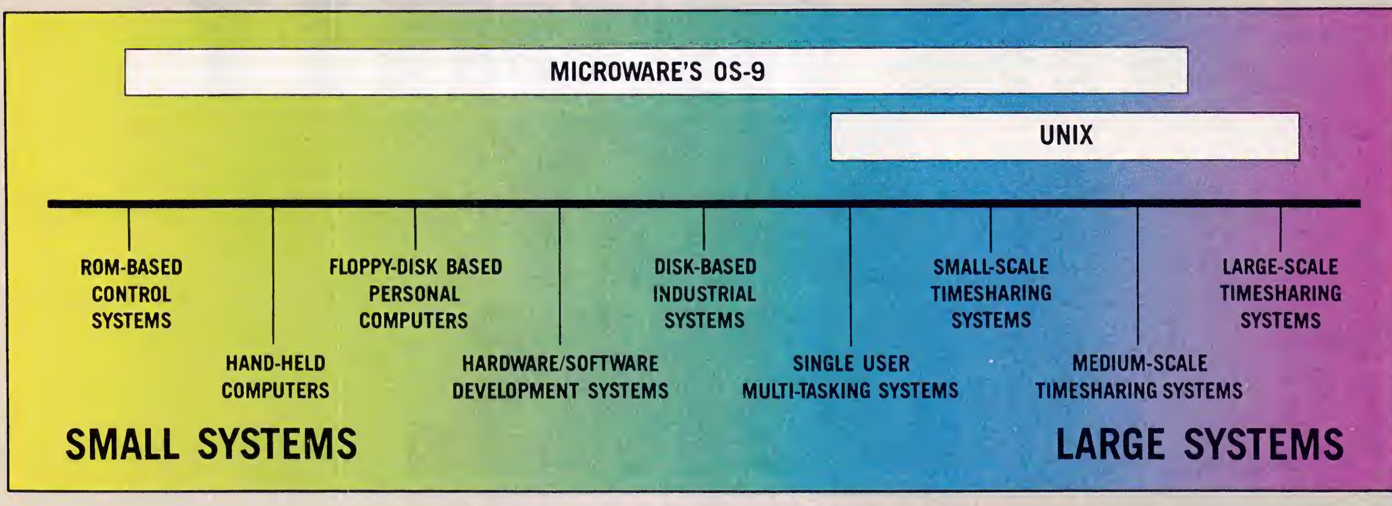
Judge for yourself. Send for our demo kit—disks, tutorial and reference manuals, all for only $150—that shows you how to build virtually any application. Contact UNIFY, 4000 Kruse Way Place, Lake Oswego, OR 97034, 503/635-6265.

Please circle Ad No. 63 on inquiry card.

UNIFY®

THE PREFERRED DBMS.

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
41-19 Honcho 4 Chome Funabashi City
Chiba 273, Japan
Phone 0474-22-1747
Telex 298-3472

SEE US AT BOOTH #1444 AT ★NCC '85

Please circle Ad No. [illegible] on inquiry card

Unix PCs, AT&T 6300s, and AT&T 3B2 machines. Furthermore, by using Microsoft's MS Net protocols in both the Unix system and MS-DOS, AT&T is able to promise (eventually) meaningful communications between these diverse machines, not just compatible physical contact.

The StarLAN schedule calls for fourth quarter of 1985 availability of Unix/MS-DOS coexistence, meaning that Unix PCs, 6300s, and IBM PCs can attach to the same network but cannot use each other as file or print servers. This capability is planned for first quarter of 1986, when the 3B2 with the Unix system will provide server features for all connected PCs—from IBM and AT&T.

Unfortunately, when the cost of networking software is added to the interface costs, AT&T's StarLAN does not seem to have any dramatic edge over Ethernet, 802.3, or the IBM PC Net, except where existing unused telephone wiring is available to eliminate cabling costs. The lack of a clear-cut cost advantage detracts substantially from StarLAN's strategic role as a "door opener."

THE UNIX PC

A year ago, the Unix PC and its jazzy System V software would have been dazzling, dramatic, and compelling. Today, sadly, the system is a tad late.

Today, buyers looking for software ergonomics (windows, icons, mouse) have a much more cost-effective choice in the Apple Macintosh, not to mention the less expensive and fancier Atari models due to be released sometime this year.

Buyers concerned with standards and availability of software already have a good choice in the IBM PC/AT, which is at least equal to the Unix PC in basic power and which is similarly priced. The PC/AT features not only the MS-DOS standard but also the optional multitasking, multiwindow TopView environment. And, for those interested in multiuser operation, Xenix on the IBM PC/AT fits the bill.

	802.3	CheaperNET	StarLAN	IBM PC NET	AppleTalk
Signal rate (Mbps)	10	10	1	2	0.230
Technology	←—— Baseband		——→	Broadband	Baseband
Arbitration	←—— CSMA/CD			——→	CSMA
Topology	Bus	Bus	Star/ Daisy	Bus	
Medium	0.4" 50-ohm co-ax	.25" 50-ohm co-ax	Unshielded twisted pair	.25" 75-ohm co-ax	Shielded twisted pair
No. of nodes	1024	1024	200*	72	32
Max. segment length (meters)	500	185	240	240	300
Network interface	←— 802.3 MAU —→		RS-422	RF Modem	RS-422
Typical interface cost**	$650	n/a	$595	$695	$50
Available for:	IBM PCs, clones, other PCs, minis and mainframes	not yet	Unix PC, AT&T 6300, IBM PC, clones, AT&T 3B	IBM PC only	Apple MAC only

* Nominal, for the AT&T version
** Cost source: 802.3—3Com/IBM PC; StarLAN—AT&T 6300/IBM PC, UNIX PC.

FIGURE 1: STARLAN COMPARED TO OTHER LAN ALTERNATIVES

Buyers who prefer the AT&T logo over the IBM logo can get many of these advantages by buying the AT&T (Olivetti) PC 6300, which costs less and has just been enhanced with a color monitor and Xenix availability.

Who, then, would choose the Unix PC? The answer appears to be those already hooked on the Unix system. For them, the Unix PC is clearly the most interesting, most "civilized," and most cost-effective Unix system engine ever. Unfortunately, Unix system hackers are *not* the audience AT&T must have if the Unix PC is to be a hit.

AT&T will probably win a good deal of business away from Fortune Systems, Altos, and the Tandy 16/16B/6000 family. But it is doubtful that the Unix PC will draw new constituencies. On announcement day, AT&T was unable to disclose any major orders from any credible source. The one contract that was announced, nominally valued at $1 million, was with an obscure firm called PC One.

Given these and other uncertainties, it would not be surprising if the 40,000 units AT&T is reputed to have ordered from Convergent prove to be about twice the quantity actually needed.

A LOOK UNDER THE HOOD

The Unix PC is powered by a 10-MHz, 68010 microprocessor unit that supports demand paging. It comes with 512K bytes of memory and has three expansion slots for additional 512K-byte memory boards (for a total of 2 Mbytes) or for peripheral control boards. The expansion bus, incidentally, is proprietary. The computer has a 12-inch, high-resolution, bit-map monochrome display of 720 by 348 pixels. It also includes a 103-key keyboard, a three-button mouse, and a 300/1200-baud modem as standard gear. Built-in hard disks of either 10 Mbytes or 20 Mbytes are offered and are said to be immediately available.

System	Points*	$/Point
AT&T Unix PC**	2164	3
CT MiniFrame	2102	4
IBM PC/AT***	1747	5
NCR Tower XP	2146	8
Zilog 8000-12	2170	9
AT&T 3B2	1387	11
Plexus P35	1746	13
Altos 586	857	13

* Higher performance rates more points
** With 1-Mbyte RAM, 20-Mbyte disk
*** With floating point coprocessor

FIGURE 2: YATES VENTURES UNIX PC BENCHMARKS

Memory	Hard Disk*	Price
512K bytes	10 Mbytes	$5095
1 Mbyte	10 Mbytes	$5495
512K bytes	20 Mbytes	$6190
1 Mbyte	20 Mbytes	$6590
512K-byte	expansion board	$1195

Software	Price
Unix System V "core"	$495
Development tools	$395
Utilities	$495

* 1 320K-byte floppy included.
Source: Yates Ventures Inc.

FIGURE 3: PRICING FOR THE UNIX PC

Unix PC w/Unix V	$6190
PC/AT w/MS-DOS*	$6205
PC/AT w/Xenix 3.0	$6535
6300 w/Xenix 3.0	$5620

* Add $149 for TopView.

FIGURE 4: COMPARATIVE CONFIGURATION PRICING FOR AT&T UNIX PC, IBM PC/AT, AND AT&T PC 6300

Yates Ventures of Palo Alto, Calif., has tested the machine, along with several others, using Aim Technology's Suite II benchmarks, and concluded that the Unix PC provides the best "price-per-point." The more significant results are reported in Figure 2.

Note, incidentally, that both the Unix PC and the IBM PC/AT appear to be handily outperforming the AT&T 3B2, which costs about three times as much and which uses the vaunted AT&T (née BellMac) 32-bit MPU. Announced pricing for the Unix PC follows in Figure 3.

The most nearly comparable configurations (512K byte/20 Mbytes) for the Unix PC, PC/AT, and 6300 are shown in Figure 4.

As usual, point-by-point comparison is complicated. The Unix PC for example, has a built-in modem, and its screen has substantially higher resolution than those of the other two. On the other hand, the PC/AT has more expansion slots (eight versus the Unix PC's three) and, with Xenix, can accommodate more memory (3 Mbytes versus the Unix PC's 2 Mbytes). Both the 6300 and the PC/AT have color monitor options, but the Unix PC has monochrome only.

'CIVILIZED' UNIX

By far the most interesting aspect of the Unix PC is its "core" System V software. This software (which is entirely unbundled) complies with the recently released System V Interface Definition (SVID) book and is based on Release 2. But for the first time AT&T has addressed several of the most troublesome deficiencies in the Unix system, especially the system's notoriously user-hostile interface. As Jack Scanlon put it, "We have civilized Unix."

This has been accomplished with a software component called the "User Agent," which is responsible for managing a multiple-window, mouse-controlled user interface. Details on the implementation of this component are sketchy so far; in all likelihood, it is a "super-shell" that allows multiple windows belonging to different tasks (all spawned by the same user) to appear on the same screen. The User Agent supports a clip-board facility for interwindow data transfer, although, again, details on the types of transfers allowed are missing. On the whole, it appears that the User Agent has done a reasonable job of creating a Mac-like, software-ergonomic interface on top of the Unix system.

It is not entirely clear yet to what extent this interface is visible or usable by the applications packages. I suspect that few of the 28 applications currently released for the Unix PC take good advantage of the new user interface.

Any future application that does so will immediately become incompatible with all other Unix system variants, including System V on the 3B line. AT&T staffers indicate that a compatible User Agent is being developed for the 3B line and that it will eventually be available as a standard System V option. No timetable for these developments has been released.

Interestingly, the Unix PC version of the Unix system supports demand paging, the first non-Berkeley version to incorporate such a feature. Record locking is also supported, although it isn't yet clear how efficiently this is accomplished. In any case, it isn't terribly relevant until real multiuser configurations emerge.

Another interesting capability in the "core" software is the telephone management package, which manages two telephone lines (one for voice, one for data), maintains a directory, performs automatic dialing and the rest of the typical features found in such packages. A dumb-terminal emulator completes the core package.

OTHER ANNOUNCEMENTS

Along with the Unix PC, AT&T unveiled a series of enhancements to the PC 6300 and a new telephone management "Personal Terminal."

Among the 6300 enhancements are the following: a 20-Mbyte disk, a color monitor, a Communications Manager card (providing a modem and some telephony management functions, including a 200-entry directory), and, most significantly, Xenix 3.0. The port was done by The Santa Cruz Operation, which handles Xenix on non-286 implementations.

Also unveiled for the 8086-based 6300 were a mouse, an 8MHz 8087-2 math coprocessor, and an external box with a 30-Mbyte disk and a tape backup system from Mountain Computer.

The soft screen Personal Terminal (PT) is a $1795 desktop telephone management system. The soft screen (actually a gel layer over the CRT face) actually "gives" to the touch, which is a more pleasant sensation than the hard touch screen on the HP150, for one. All functions can be controlled by touching the screen, so a keyboard is $100 extra.

The key problem is this: Because we still live in a largely analog world, the PT cannot be connected to any plain telephone line. It can only be attached to a digital port on the AT&T S/75 or S/85 PBX. Curiously, it can't even attach to another PT without an intervening AT&T PBX digital port.

What about the second product (supposedly also a phone terminal) that Convergent Technologies is known to be working on for AT&T? This is apparently still coming.

By the way, with the Unix PC, AT&T is violating its own silly rule that the term *Unix* must always be followed by the word *system.* □

News Summary:* Sobered by a hard first year as a commercial computer supplier, AT&T now seems to pin more of its hopes on the StarLAN network, rather than the Unix PC (née 7300). AT&T also unveiled enhancements and software (including Xenix) for the 6300, hoping to revive its sagging fortunes and capitalize on PC/AT delays. Also unveiled: an attractive AT&T-made telephone-terminal; unfortunately, it works only with digital ports in AT&T's S/75 and S/85 PBXs.

Omri Serlin heads ITOM International, a Los Altos, Calif.-based research and consulting firm. He is the editor and publisher of FT Systems *and of* Supermicro, *newsletters that cover developments in the computer industry.*

BEYOND C

PROGRAMMING LANGUAGES

Past, Present, Future

Current third-generation languages such as C and FORTRAN will have to move aside at some point for a new family of fourth-generation languages.

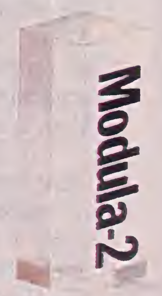
BY DAVID SPENCER

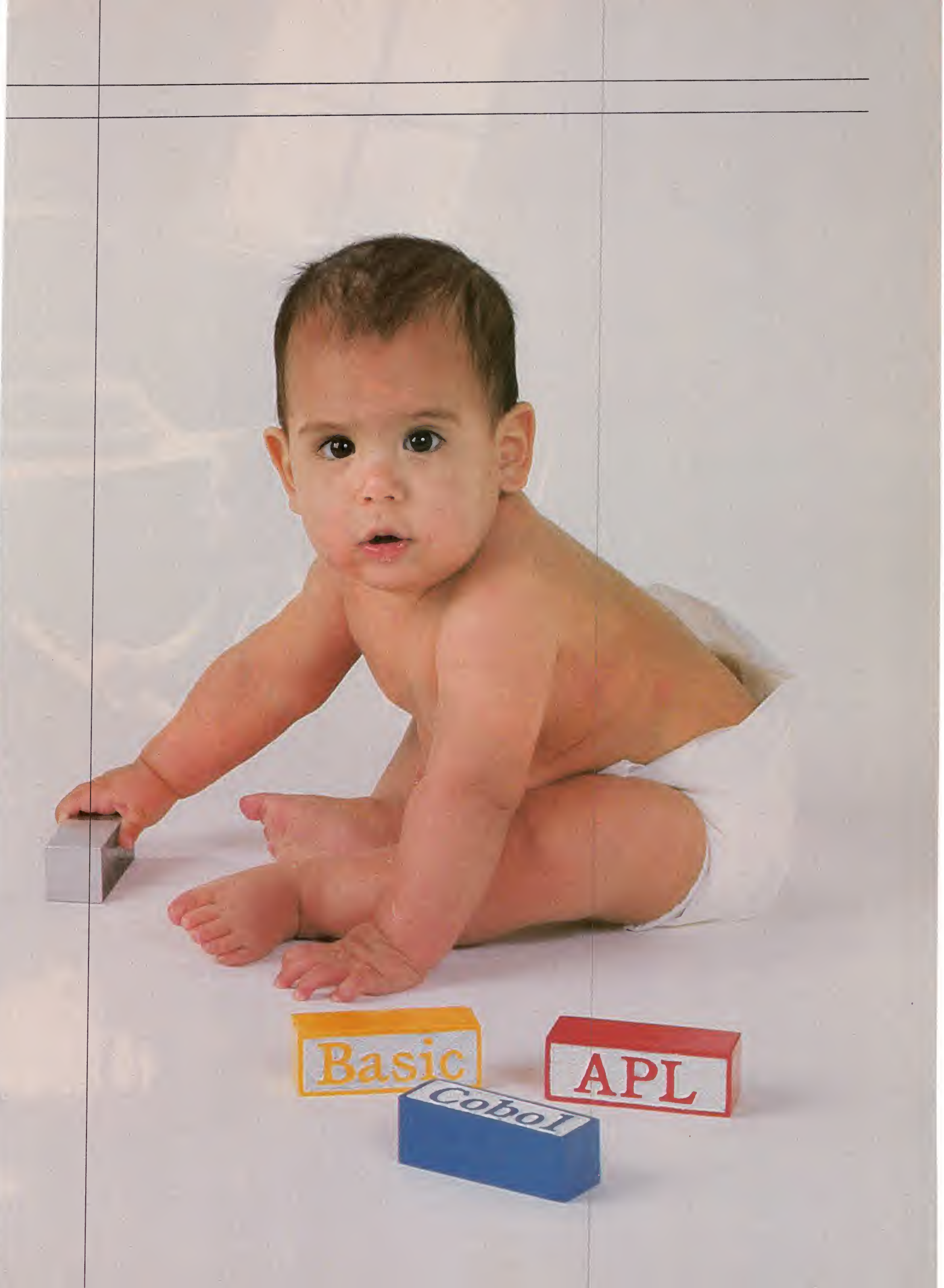
At 30 years old, FORTRAN is graying at the temples; third-generation programming languages are in their heyday. So you are probably wondering how we will speak to computers during the next decade. If current projections hold true, computers will seem (and talk) more like us fairly soon. In order for that to happen, however, current third-generation computer languages (such as C and FORTRAN) will have to move aside for a new family of fourth-generation languages.

Have no fear, though. As computer architectures and programming methodologies have come to simulate human thinking more closely, programming languages have increasingly abandoned the procedural approach (how machines do something) in favor of a non-procedural, functional one (what is to be done). As the address-related recall capability of the microprocessor is transformed into the associative recall of the brain, high-level languages (HLLs) move closer to, and will eventually be replaced by, the very high-level languages (VHLLs), also known as fourth-generation languages.

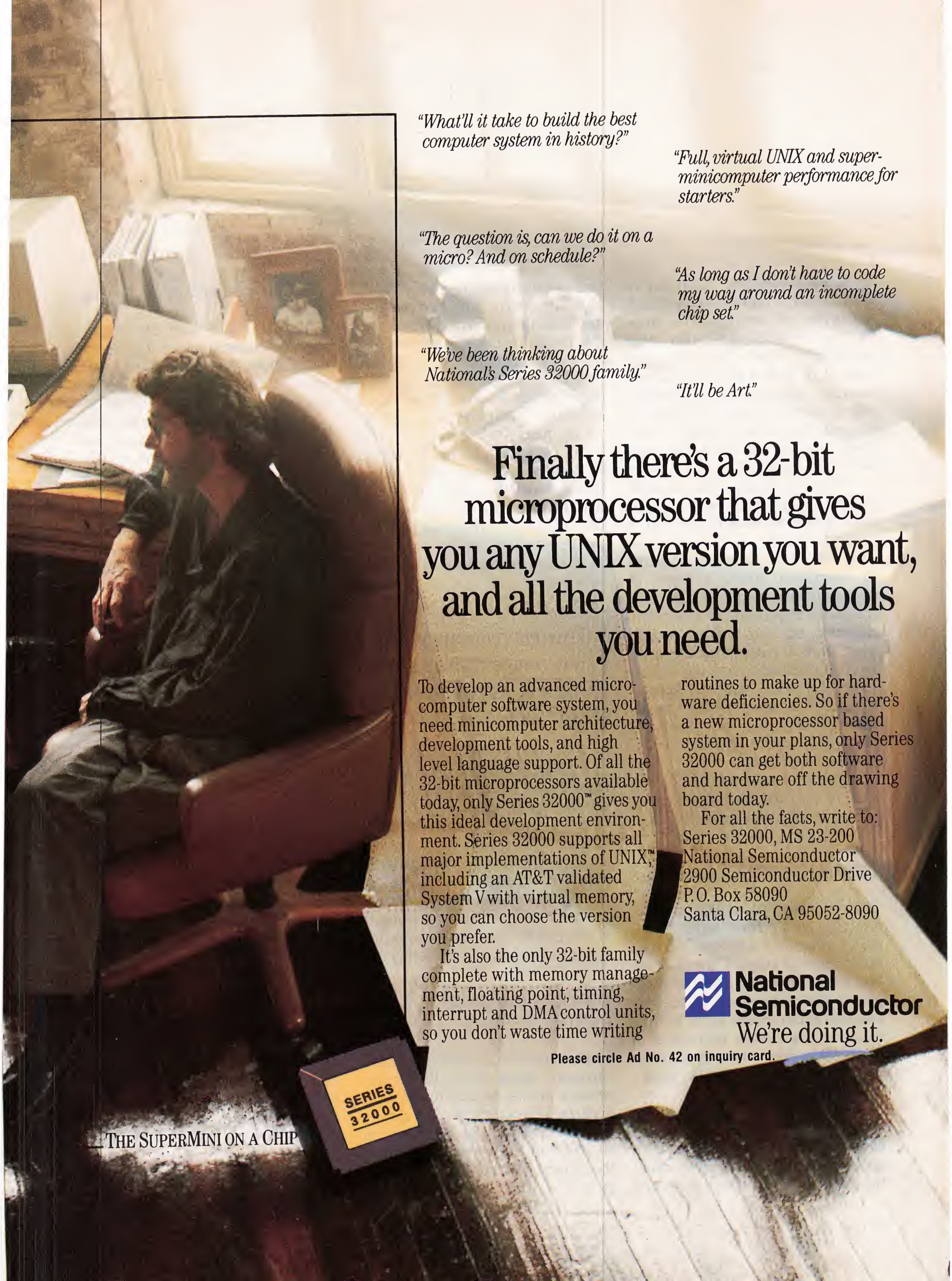
To belong to the fourth generation, a language must have crossed the threshold of a world where the programmer specifies the task to be done to the point where the knowledge of how to do the task is contained in the language itself. Historically, languages have moved further from the machine level, becoming more abstract with each generation. To understand fourth-generation languages, you must first have some idea of what came before.

First-generation languages were little more than machine code, which was closest to the computer's




Basic
APL
Cobol


UNIX IS A TRADEMARK OF AT&T BELL LABORATORIES. SERIES 32000 IS A TRADEMARK OF NATIONAL SEMICONDUCTOR CORPORATION.


"What'll it take to build the best computer system in history?"
"Full, virtual UNIX and super-minicomputer performance for starters."
"The question is, can we do it on a micro? And on schedule?"
"As long as I don't have to code my way around an incomplete chip set."
"We've been thinking about National's Series 32000 family."
"It'll be Art."
Finally there's a 32-bit microprocessor that gives you any UNIX version you want, and all the development tools you need.
To develop an advanced microcomputer software system, you need minicomputer architecture, development tools, and high level language support. Of all the 32-bit microprocessors available today, only Series 32000™ gives you this ideal development environment. Series 32000 supports all major implementations of UNIX™, including an AT&T validated System V with virtual memory, so you can choose the version you prefer.
It's also the only 32-bit family complete with memory management, floating point, timing, interrupt and DMA control units, so you don't waste time writing routines to make up for hardware deficiencies. So if there's a new microprocessor based system in your plans, only Series 32000 can get both software and hardware off the drawing board today.
For all the facts, write to: Series 32000, MS 23-200 National Semiconductor 2900 Semiconductor Drive P.O. Box 58090 Santa Clara, CA 95052-8090
National Semiconductor
We're doing it.
Please circle Ad No. 42 on inquiry card.
SERIES 32000
THE SUPERMINI ON A CHIP

flip-flopping switches. Each set of ones and zeros represented current-on/current-off settings. The steps for even a simple operation were necessarily very discrete and seemed artificial in their total lack of assumed information. Arithmetic was all done in binary, and how a program was coded directly mirrored how an operation was performed by the machine.

The second generation brought assembly languages, which separated operations (instructions) from operands (data) and substituted names for binary numbers. Assembly languages added a layer of address-related recall to the programmer's repertoire as assembler programs did the tedious, error-prone work of putting machine code together. Assembly languages were purely procedural because addition/subtraction involved the load-move-store of binary data. A simple operation still took many discrete steps, however, and decisions were made according to condition tests—the state of the machine's registers—instead of the task's inherent logic.

THIRD-GENERATION HLLs

The third-generation HLLs gave abstraction an algebraic form. Statements subsumed a greater number of incremental steps. Choices of action based on logical conditions replaced the comparisons of register contents and condition tests, moving closer to the human perception of the job to be done and away from the machine's demands for a binary representation.

In an article entitled "Programming Languages," James W. Hunt proposes task-centered criteria for a good language. Any HLL should allow the programmer to design programs easily, document programs, debug programs, move programs between machines (portability), verify a program's correctness, and compile parts of a program separately.

These criteria are met to some degree by most widely used HLLs: FORTRAN, COBOL, Pascal, and C. Some functions associated with fourth-generation VHLLS also appear in existing languages. Ada, for instance, employs a class concept to abstract data structures and also allows a basic type definition to incorporate new items in the class or to be changed in response to specified behaviors.

If current projections hold true, computers will seem (and talk) more like us fairly soon.

A similar extension to C, called Objective-C, seeks to make object-oriented programming part of the Unix system repertoire. Ada, some Pascals, and Modula-2 provide concurrency, which allows separate processes to share resources, including the CPU. But these languages are not true members of the fourth generation because their syntactical structures haven't really changed.

DIFFERS FROM ITS PREDECESSORS

As I said before, a fourth-generation language differs from its predecessors in the degree to which it minimizes the user's need to specify machine behavior. Heather Bryce, in an article in *Electronic Design,* lists the general characteristics of fourth-generation languages:

(one) designed for on-line operation;

(two) easily used by nonprofessional programmers (generally, users should be able to learn a subset of the language in two days and get satisfactory results);

(three) employs a database management system (DBMS) directly and requires one-tenth the number of instructions necessary for coding in COBOL or PL/1;

(four) uses nonprocedural code and makes intelligent default assumptions where possible, encourages structured code, produces code that is easily understood and maintained by others, and allows easy debugging of programs.

According to Greg Blanpied, Xerox vice president of technology development, *fourth-generation language* is an umbrella term that usually covers four areas of new software:

(one) presentation languages, such as formal query languages, natural query languages, reporting, and graphics;

(two) specialty/specialized functions, such as spreadsheets, modeling, analysis, and simulation;

(three) application generators, usually for COBOL;

(four) VHLLS, including such nonprocedural languages as LISP and Forth.

The decisions to be made in each area reflect the needs of the user. For example, a growing number of users are primarily ordinary business people whose needs may be served by different capabilities of each area. Such users generally want the presentation languages, coupled with some DBMS system, to give them easy access to a company's huge information system without becoming programmers or being totally dependent upon management information systems (MIS) staff.

They often need a special program for predicting the results of business decisions and modeling scenarios based on hypothetical situations or predictions. They don't want to wait for MIS to make changes in existing programs; instead, they want to use COBOL program generators that tailor general programs to their specific needs. And they need decision-support systems (sometimes called expert systems) to analyze and extract data based on knowledge stored in the system.

Fourth-generation languages are generally one of three types: declarative, functional, or object-oriented. Declarative, or rule-based, languages use a set of operators to define the relationship between data. Once all the rules are established, the program executes them (Malpas and O'Leary, 1984). Functional languages apply mathematical expressions to data to get a result. The relationships come from applied mathematics, and the program has no constructs that change the original data. Each statement is executed independently, and the state of the machine does not affect the program in any way.

In object-oriented programs, a construct called an *object* contains the data and commands to which the data responds. Objects can be organized into classes and analyzed for common features, "so that conclusions can be drawn or deductions made about the data" (Hindin, 1984).

WHAT COMES AFTER C?

Many people consider the Unix system and the C language to be the ultimate children of the third generation. But acceptance by systems programmers does not necessarily lead to general acceptance by the rest of the computing world. The Unix system and C form one kind of programming environment. However, several factors will determine whether that environment will support fourth-generation languages as well.

DBMS, the user interface, menus and windows—whole issues of various publications have been devoted to the ways in which the Unix system can be tamed and brought into the "user-friendly" world now inhabited by the Sun workstations and Apple Macintosh systems (see "References"). But the languages themselves, the VHLLS used to implement and control such systems, will still have to become less machine-oriented before the Unix system loses its "fit for true hackers only" reputation.

> **Fourth-generation languages are generally one of three types: declarative, functional, or object-oriented.**

Specific implementations of existing languages such as C and Pascal often provide a programming environment in which fourth-generation characteristics might be incorporated. Proponents of the Unix system and C, for example, may suggest extensions to bring C into the fourth generation (Cox, 1983, 1984) or the addition of a widely used functional language to the Unix and C environment (Saunders, 1984).

Building the next generation's language on top of the current one is a common and sensible approach, considering the time needed to start from scratch. It remains to be seen whether C, which most closely resembles assembly language, is the best third-generation base for such development. Adding an object-oriented component or fourth-generation interface to an effective Pascal implementation (such as Pascal-2) would be equally practical approaches.

Adding languages is a relatively easy way to make the Unix system accessible. Two special-purpose declarative languages, `make` and `yacc`, for example, are already part of the Unix system's utility set. With relatively little training, users of these utilities can greatly increase their productivity (Malpas and O'Leary, 1984). Query languages accompany many of the relational databases available for Unix-based systems. General-purpose declarative languages are the next step. Prolog, for example, offers increased efficiency for users and for machines. William Wong claims that "Prolog compilers on larger machines generate code that is as efficient as C or LISP so that programming logically does not necessarily imply inefficiency" (Wong, 1984).

FUTURE LANGUAGES FOR THE UNIX SYSTEM

For the Unix system, the new languages developed by artificial intelligence (AI) researchers are generally expected to succeed the third-generation languages now in use. LISP and Prolog are most often mentioned; APL and Forth are less prominent, perhaps with good reason. To give you a taste of each, without getting bogged down in academic generalities or technical specifics, sup-

```
country ← 4 6 ρ 'usa    india china brazil'
data ← 4 2 ρ 203 3 548 1 800 4 108 3
density ← data[;1] ÷ data[;2]
max ← ⌈/ density
mostdense ← (max = density) / [1]   country
```

FIGURE 1: A SAMPLE PROGRAM IN APL

pose we solve a simple problem with a demonstration program in each language.

This problem comes from the best-known book on Prolog (Clocksin and Mellish's *Programming in Prolog*, 1981). In a database, we have four countries. For each country, we know the population (in millions) and the size (in millions of square miles). From that data, we want to know each country's population density and relative information such as the country with the largest area, largest population, or greatest population density.

APL

APL (A Programming Language) was an early attempt to program according to the logic of the problem rather than to the architecture of the machine. Originally a notation for applied math algorithms, APL has been adopted by IBM and DEC and is now available on supermicros.

Proponents tout APL's terse form, simple rules, and concise representation of concepts through graphic symbols. You enter calculations as if you were using a calculator. Figure 1 illustrates our sample problem in APL. The first two lines establish our database. The "country" matrix has four items; each item has six characters. The "data" matrix is a corresponding matrix of four items, each with two numbers. The last three lines define the relationships as mathematical calculations.

To compute the density, we divide the first element of each item pair in the data matrix by the second element. In the fourth line, the function symbol for "maximum" is compressed onto density with the function symbol ("/"). Compression allows one element to be picked from a group, in this case the largest one. Using this same technique, the final line gives us the name of the country with the highest population.

Many people consider the Unix system and the C language to be the ultimate children of the third generation.

I can hardly discuss APL without mentioning the "funny symbols"; they are the focal point, it seems, for the real debate about APL's usefulness as a language. APL's features may appeal more to mathematicians than to business people. Its creator, Kenneth Iverson, offers in his book *A Programming Language* this incentive: "The descriptive and analytic power of an adequate programming language repays the effort required for its mastery." The degree to which APL rewards casual users is debatable: Other languages allow users to concentrate on the problem, free from "concern with computer-oriented details." And these languages use more familiar symbols.

LISP

LISP stands for List Processing Language (not Lots of Inconsequential Silly Parentheses). It was one of the first AI languages to become well known outside AI research centers. Popular architectures for Unix systems, such as the Motorola 68000, have allowed LISP to reveal its true potential.

Several LISP features are a definite move toward associative memory recall. One such feature is modularity: Data structures can be linked to form larger ones. By changing a set of pointers, you change the relative location of the structure. As a result, the essentially dynamic allocation of storage areas replaces the lengthy definitions of parameters for each program. LISP also provides "automatic garbage collection" as part of its efficient management of memory.

Figure 2 shows how to solve our demonstration problem with a LISP program. As you can see, the program in Figure 2 requires little more than a knowledge of the relationships between the data. Using English words or recognizable abbreviations, we define those relationships in the function definitions. In the applications definition, the relationships may be ordered to produce the desired results. Most computer manufacturers are offering LISP or Common LISP (a more recent and more consistent implementation) as part of their system software packages.

```
Def pop lambda X
    (cond (eq X 'usa') 203)
     (eq X 'india') 548)

:  ———————— Function Definitions for  China, and Brazil, same as above
                a

)  ——————————————— ending parenthesis matches the one before cond

(def area lambda X
 (cond (eq X 'usa') 3)

:  ———————— Function Definitions for India, China, and Brazil, same as above

)  ——————————————— ending parenthesis matches the one before cond

(Define DENSITY lambda X
 (div (pop X) (area X))

   ——————————————————————————— Application Definition begins
(DENSITY india) ————————————— the X is replaced by the condition india
 (div (pop india) (area india))
```

FIGURE 2: THE DEMONSTRATION PROBLEM IN LISP

```
    pop(usa, 203).
    pop(india, 548).
    pop(china, 800).
    pop(brazil, 108).
                                              Facts
    area(usa, 3).
    area(india, 1).
    area(china, 4).
    area(brazil, 3).
                                              Rule
    density(X,Y) :-
          pop(X,Pop),area(X,Area),        Y is pop/area.
```

FIGURE 3: THE SAMPLE PROGRAM IN PROLOG

PROLOG

Prolog (the name stands for Programming in Logic) allows you to do many of the same things you can do in LISP. Both languages employ object-oriented relationships. Prolog's basic relationship is the Horn clause, a predicate calculus formula that contains only one conclusion.

Prolog uses a rule-based search to manipulate the data in our sample problem, a slightly different approach from LISP's function/application definitions. Facts are added to the database and are then interpreted according to the rule.

The statements in Figure 3 headed by `pop` and `area` are clauses stating the facts from which conclusions may be found if we know the population and area of *X* and divide the population by the area.

Prolog is heavily endorsed in Japan and England, and many American firms are using it as an interface to Unix system databases.

FORTH

Forth is more compact than LISP, so it's more popular on current microcomputers. It's also very transportable and can do real-time applications.

A Forth routine consists primarily of addresses that point to commands written in machine code, "primitives" in Forth terminology. The user's instructions, called "secondaries," are written in primitives, and all secondaries for a program go into a dictionary. The syntax of the language is a string consisting of Forth words separated by spaces. Subroutines are called implicitly by "words" that start actions.

The big difference between Forth and other functional languages is mostly a matter of applicability. Like APL, Forth's syntax is terse and specialized, but unlike APL, it would be very difficult to do even our simple demonstration program. We would have to extend the language by entering new words into Forth's dictionary, or we'd have to do our arithmetic using registers and a stack, as we would in assembly language.

Forth is a machine-level programmer's language. Even more specialized than APL, Forth is an object-oriented language for those who want to manipulate the computer itself.

TOO SOON TO CHOOSE?

Fourth-generation languages may be common about 10 years from now. However, government funding might accelerate the process and make micro-based artificial intelligence software available even sooner. But success of any one language or product will probably depend as much on the marketing of the product as on the technical development.

Harvey J. Hindin, special features editor for *Computer Design,* expressed skepticism about the readiness of AI languages in an article (Hindin, 1984) on the new software: "Prolog may end up being fundamentally flawed just because it is a logic-based language. It turns out that logic-based languages (some with even more features than Prolog) have been proposed before and found lacking. They have turned out to be duds because logic-based languages are not flexible enough for the real world. . . . Other issues in the great debate between LISP and Prolog await the test of time."

The fifth-generation languages will be AI-inspired natural-language systems that characteristically handle a variety of grammatical/nongrammatical constructions, infer from user inquiry where data will be found in a database, and execute necessary manipulations, procedures, and formatting.

These systems are yet to be made generally available, but you can see the beginnings in such AI languages as Prolog and LISP.

In 1969 J. E. Sammet concluded her seminal book on the history of programming languages (*Programming Languages: History and Fundamentals*) with the prediction that future developments would either be theory-oriented or user-oriented. The goal of a theory-oriented approach is to give the system a complete characterization of all objectives to be achieved and let it design the program accordingly. Many fourth-generation languages take this approach.

The goal of a user-oriented approach is to implement a natural language to allow the system to respond to the specific idiosyncrasies of each user's needs. Sixteen years later, the debate still continues, and both approaches are still in progress. □

David Spencer is technical publications manager at Oregon Software Inc., Portland, Ore. After teaching English for 10 years, he has written software documentation over the last five years for Sperry Univac and Oregon Software. His interest in programming languages stems from his graduate days at USC, where he studied rhetoric, linguistics, and literature.

References

Baker, Linda, and Mitch Derick. *Pocket Guide to Forth,* Reading, Mass.: Addison-Wesley, 1983.

Berg, Eric H. "Bell Labs Unveil New 1-Megabit Chip." *The (Portland) Oregonian,* December 21, 1984, page E9.

Blanpied, Greg. "Using Fourth-Generation Languages Well." *Software News,* July 1984.

Bryce, Heather. "Software Engineers Seek to Make Computers Understand Natural Languages." *Electronic Design,* May 3, 1984.

Cantral, David. "Speaking the Users' Language." *Software News,* July 1984, page 24.

Clocksin, W. F., and C. S. Mellish. *Programming in Prolog.* New York: Springer-Verlag, 1981.

Cox, Brad J. "Object-Oriented Programming in C." *Unix Review,* October/November 1983, page 67, and February/March 1984, page 56.

Epstein, Arnold, Jeffrey D. Morris, and Barry Unger. "Forth Efficiency Blends with C and Pascal Syntax." *Computer Design,* November 1984, page 183.

Fletcher, George W. "Talking to Your Computer in English." *Software News,* July 1984, page 30.

Grosch, Herbert. *Computer Design,* November 1984, page 150.

GLOSSARY

Declarative languages. A program consists of a set of rules governing the relationships between various types of data.

Procedural languages. Program consists of flow-control constructs and data structures; users must bind data to types.

Rule-based language. Same as declarative language.

Functional programming. Program consists of a set of computational instructions.

Object-oriented programming. Data is coupled with a set of operations; the combination (called an *object*) is activated by commands to *do* things.

Query language. A query language such as SQL is used to access a database.

Expert system. Stores knowledge that a human expert might give in response to questions.

Knowledge system. Same as an expert system.

Hindin, Harvey J. "Fifth-Generation Computing: Dedicated Software Is The Key." *Computer Design,* September 1984, page 150.

Hunt, James W. "Programming Languages." *Computer,* April 1982, page 70.

Li, Deyi. *A Prolog Database System.* New York: John Wiley & Sons, 1984.

Lipkin, Efrem, and Theodore Goldstein. "Software Wars and the Development of Post-Structural Programming." Unpublished paper, 1985.

Malpas, John, and Kathy O'Leary. "Declarative Languages Under Unix." *Microsystems,* August 1984, page 94.

Roland, Jon. "Unix Database Management Systems." In three parts: *Unix Review,* December/January 1983, page 43; February/March 1984, page 26; and April/May 1984, page 24.

Roland, Jon. "AI, Unix, and C." UNIX/WORLD, Vol. 1, No. 3, 1984, page 96.

Sammet, J. E. *Programming Languages: History and Fundamentals.* Englewood Cliffs, N.J.: Prentice-Hall, 1969.

Santiarelli, Mary-Beth. "What Kind of Aid? It Depends on Skill." *Software News,* July 1984, page 22.

Saunders, David. "Unix, C, and APL." UNIX/WORLD, Vol. 1, No. 6, page 59.

Shindler, Max. "Engineering Software." *Electronic Design,* January 12, 1984, page 150.

Spivey, Mike. *University of York Portable Prolog System User's Guide,* University of York, 1984.

Wong, William D. "Prolog: A Tutorial/Review." *Microsystems,* January 1984, page 104.

UNDERSTANDING FOURTH-GENERATION LANGUAGES

At last. With fourth-generation languages finally coming to market, truly intuitive software may be on the way.

BY DOUG MAHUGH

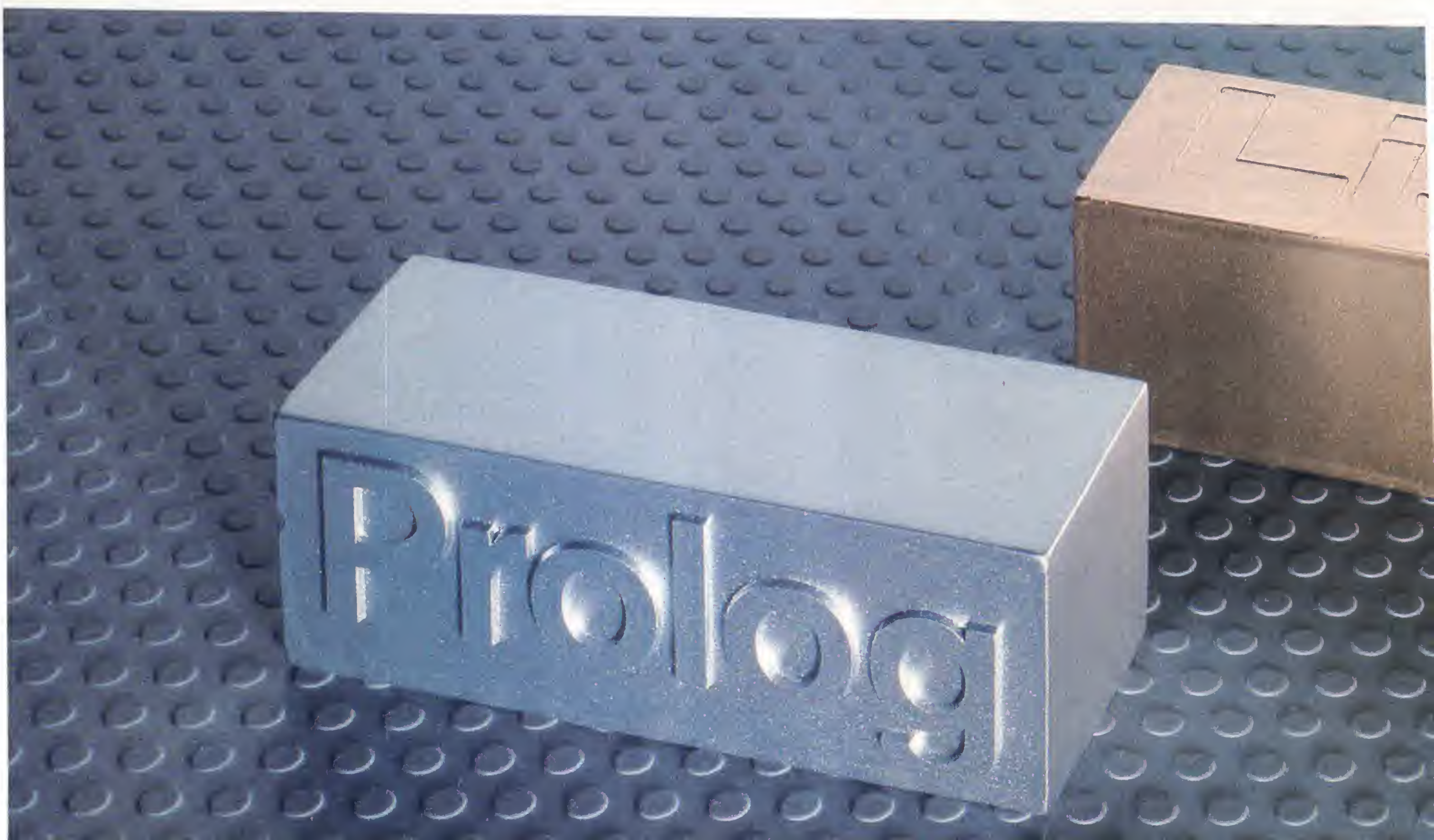
From the first assemblers and compilers to today's icon-based operating systems and windows, software has become progressively easier to use. Now, with fourth-generation languages (those that can understand English-like commands) finally coming to market, truly intuitive software may at last be on the way. After all, what could be easier than programming and working in plain English?

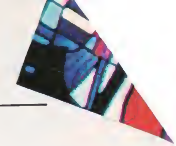
Unfortunately, that ultimate capability isn't available yet, although some new products come close. Many popular database management system (DBMS) packages have begun to incorporate fourth-generation languages, and new types of software such as expert systems use similar capabilities.

In this article, I'll be looking at the history of this field and at some of the commercial products that have come out of it. Not every application can benefit from a fourth-generation language, so I'll also consider how you can decide whether it's appropriate for you. Finally, I'll take a look at the future of this interesting class of software products.

Fourth-generation languages are a commercial application of the field of research known as the natural language (NL) problem. Simply stated, the NL problem is the task of developing a computer program that can understand written or spoken English commands. The fields of computer science, linguistics, and cognitive psychology have all played a part in attempts to solve this problem.

Fourth-generation languages vary widely in their capabilities and applications. They range from simple





query languages that can recognize a few synonyms to powerful applications generators that can understand complicated sentence structures, overlook spelling errors, resolve ambiguous requests, and learn new words and procedures.

The most common use of fourth-generation languages to date has been in DBMSs; some products offer a fourth-generation language for interactive queries, some for developing applications, and many offer both. Another popular application of fourth-generation languages is in *expert systems,* which help users solve problems in specific domains, much like an expert would, by responding to requests in English and by asking relevant questions when more information is needed.

Almost every major software developer is working on one of these products. Many vendors believe they will find a big market with managers and executives who have always had to rely on data-processing personnel to get them the information they need. With a fourth-generation language query system, managers themselves could find the information they need quickly and easily.

A more certain market is in small companies that cannot afford to hire programming staff to develop the custom applications. Traditionally these firms have had to get by with existing software or hire expensive contract programmers. Using a fourth-generation programming language, however, personnel from marketing, accounting, or production could generate the applications they need without professional help. Some analysts have even predicted that steady improvement of fourth-generation languages will eliminate the need for application programmers altogether.

A HISTORY OF NATURAL LANGUAGE SOFTWARE

A look at the history of natural language (NL) research, however, shows that these predictions should be taken with a grain of salt. Progress on the natural language problem has been steady but slow. In the late 1950s, many researchers believed that the development of computer programs that could understand English was only a few years away. Now, 25 years later, the most optimistic researchers believe that we are still a few years away from any general solution.

Most of the early work on natural language software was done by

the artificial intelligence (AI) research community, largely in response to adoption of the Turing Test as a measure of machine intelligence. The Turing Test was proposed by Alan Turing in 1950 as a way of determining whether machines can think. In it, a person presents questions to two unseen entities, one of which is another human being and the other a machine. If the person asking the questions cannot determine which of the two is the machine, then the machine has passed the Turing Test. Obviously, natural language capability is a prerequisite to passing the Turing Test.

The first program to give a convincing appearance of understanding natural language was developed in 1959. Known as the Conversation Machine, it could provide seemingly relevant responses to comments about the weather and could answer simple questions. However, because the meaning of the dialogue was not important, the Conversation Machine's natural language capability was merely an illusion.

The first program that could be said to understand natural language requests was Sad Sam, developed in 1963 by Robert Lindsay. You could tell this program the relationships between persons in your family, and the program would then answer questions about those relationships. You didn't have to base the questions on particular facts that you had already given the program because Sad Sam figured out answers by deduction, based on what it had been told. This was also the first natural language program to incorporate *extensibility*, meaning that it could acquire knowledge and grow in capability.

Another program developed in 1963, Baseball, was the first to hint at the bright future of fourth-generation languages for database queries. Baseball was developed at Lincoln Laboratories under the direction of B.F. Green, and it answered English-like questions about the previous year's American League games. A small database stored all the information, and the program translated questions into appropriate searches.

After these early efforts, natural language programs became popular. Although most were simple illusions, some were significant breakthroughs. Joseph Weizenbaum's Eliza program, developed in 1966, was the first programmable natural language product because you could give it a script that would make it parody any type of behavior. This program, using a script for a Rogerian psychotherapist, is available for microcomputers and is still popular today.

Another significant development was Terry Winograd's SHRDLU program. Completed in 1970 at MIT's Artificial Intelligence Laboratory, SHRDLU was a fully functional fourth-generation language that responded to requests about a mythical world of colored blocks and pyramids. The program could manipulate its model of the world by stacking objects on top of one another and moving them around at the user's request. It could also answer questions about the relative positions of objects .

SHRDLU's two most impressive features, however, were the ability to understand ambiguous requests by referring to earlier requests and the ability to answer questions about its reasoning (such as "Why did you put the green cube there?"). These capabilities are now included in many commercially available fourth-generation languages.

Specialized natural language products for mainframes started to appear on the market in the late

DEFINING FOURTH-GENERATION JARGON

When you consider that simpler communication is the goal driving fourth-generation software development, it is a little ironic that the field has developed a confusing language of its own. The most common terms used to describe fourth-generation software are defined below.

Context: Natural language (NL) programs are usually classified as either *context-sensitive* or *context-free*. (Technically, these terms refer to the grammar used.) Context-sensitive means that the interpretation of words is affected by their context, while context-free means that a word has only one possible interpretation, no matter how it is used. English is obviously context-sensitive; the word *run,* for example, has more than 40 possible interpretations.

Expert System: A program that has access to the facts and reasoning processes needed to solve problems in a particular field. Users consult expert systems in the same way they would consult a human expert; the expert system analyzes the problem and either gives an analysis or asks for more information.

Extensibility: The ability of a fourth-generation language to acquire new knowledge about the data it works with. Syntactic extensibility is the ability to learn new definitions; semantic extensibility is the ability to learn new relationships, or semantic links.

Fourth-Generation Computer: Any computer based on VLSI (very large scale integration) technology. This term has no direct connection to fourth-generation software (see below), so its use can be misleading. The first five generations of computer hardware are vacuum-tube technology, transistor technology, integrated-circuit technology, VLSI technology, and KIPS (knowledge information processing systems). There is currently a great deal of debate about the differences between fourth- and fifth-generation computers.

Fourth-Generation Language: A programming language that takes English-like sentences as input and performs the tasks that these sentences describe. Fourth-generation languages are not as much like written English as many users expect them to be, but they are powerful and easy to use. The most important criterion of a fourth-generation language is that it provides more automation of features than third-generation languages such as FORTRAN, COBOL, and C. The first three generations of programming languages are machine languages, low-level compiled or assembled languages, and high-level languages such as FORTRAN and COBOL.

Natural Language Software: Programs that can understand inputs given in standard written English. Although no general-purpose natural language program has been developed yet, many programs can understand English sentences of a certain type or those that are limited to a specific topic.

Nonprocedural Language: A programming language that allows the programmer to state goals to be achieved, rather than the specific procedures necessary to accomplish them. In contrast, procedural languages such as C and Pascal allow the programmer to specify *how* the program will run, as well as *what* it will do.—*D.M.*

1970s, 20 years after the AI community began to tackle the problem. The first commercially successful NL product was Intellect, from Artificial Intelligence Corp. It ran on IBM and compatible mainframes and interfaced with many popular DBMSs.

Savvy, from Excalibur Technologies, was the first NL product for micros. It used a limited pattern-matching approach but managed to fit into a mere 4K bytes of memory. Savvy gave users the capability to modify data as well as display it, so it was both a query language and a programming language.

In late 1984, Microrim released Clout, a natural language interface that works with several popular microcomputer DBMS products. Clout's capabilities are limited, but it is a significant product because many developers had felt that true NL capability required too much memory and processing power to ever be successfully implemented on micros.

MODERN FOURTH-GENERATION LANGUAGES

The Unix system has been a late beneficiary of natural language research. However, over the past year many Unix system software vendors have added fourth-generation languages to their DBMS products. (The acronym that seems to be catching on for this type of product is DBMS + 4GL). Data General, Prime, Micro Data Base Systems, Hewlett-Packard, SIR, Relational Database Systems, and Data Language Corp. have all added some type of fourth-generation language capability to their DBMSs, and many other Unix system software developers are doing the same.

As I mentioned earlier, the two most common uses for fourth-gener-

Stick this in your RAM.

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

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

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

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

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

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

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

R Desk™

BY R SYSTEMS, INC.

Only $295!

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

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

NAME ______________________

COMPANY ______________________

ADDRESS ______________________

CITY ______________________

STATE ______________ ZIP __________

TELEPHONE ______________________

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

___ VISA or ___ MasterCard

CARD # ______________________

EXP. DATE ______________________

SIGNATURE ______________________

Rolodex is a registered trademark of the Rolodex Corporation. UW785

Please circle Ad No. 67 on inquiry card.

ation languages are query languages and application generators. These applications are ideally suited to fourth-generation languages because the limited vocabulary (terms relevant to a specific database) simplifies the interpretation process.

Although the vocabulary of fourth-generation languages may be limited, advertising and public relations copywriters have come up with an unlimited number of adjectives to describe them. I'll avoid most of the jargon here, but one distinction deserves special attention: procedural versus nonprocedural fourth-generation languages.

In the past, fourth-generation languages were considered nonprocedural by definition. Supposedly, the whole purpose of a fourth-generation language was to let users describe *what* they wanted to do and not *how* it should be done. After some real-world experience with nonprocedural languages, however, this approach was clearly imposing limitations on programmers; many types of tasks can be specified only in a procedural language.

Consequently, developers began to use the term *procedural* fourth-generation language—a seeming contradiction in terms—for languages that offer fourth-generation language ease of use while allowing users to specify procedures if necessary. This type of language is so popular that the working definition of fourth-generation language has changed to allow both procedural and nonprocedural languages.

SIR/DBMS is one product that offers a solution to the procedural-versus-nonprocedural problem. SIR's nonprocedural fourth-generation language generates a short program (in a procedural language) for each command. You can then modify this program if you need more control than the nonprocedural fourth-generation language provides.

TWO TYPES OF EXTENSIBILITY

Another important concept is *extensibility:* the ability of the system to acquire new knowledge. Modern fourth-generation languages have two types of extensibility: syntactic and semantic.

Syntactic extensibility allows a user to add new definitions to the data dictionary. Ideally, the fourth-generation language should be able to do this whenever a user enters an unrecognized phrase; this way, users can stick with the terminology they're used to, and the system will learn new terms as they are needed.

With Microrim's Clout2, for example, the system displays an unrecognized phrase and then asks the user to "Please enter a synonym or change spelling." If the user enters a synonym made up of recognized phrases—that is, a new definition—the system stores the definition for future reference.

The other type of extensibility, *semantic extensibility,* allows users to define new relationships and new processes. For example, you might specify that "customers have credit limits." This is known as a semantic link, and it lets the fourth-generation language know that when you say "Who is over their credit limits?" you're asking for a list of customers. This semantic information, combined with the data dictionary, is the "real-world knowledge" that the fourth-generation language calls upon.

A fourth-generation language can also use semantic links to spot meaningless queries and errors. Jim McGuire, vice president of marketing at Data Language Corp. (DLC), points out that "a fourth-generation language can recognize errors that a third-generation language like COBOL or BASIC can't, such as inappropriate use of a certain record type." DLC's Progress DBMS provides this type of intelligent error checking in its fourth-generation language.

Another way to resolve ambiguities is to refer to the user's previous commands. This is called *accumulating context,* and it's similar to the way people figure out what is being referred to in everyday conversation. If you say "I saw Joe today" and then say "He wants you to call him," most people will realize that "He" refers to Joe. Likewise, if you ask a fourth-generation language to "list the products with sales less than 1000" and then type "Which of those have sales less than last month?" accumulated context will make the fourth-generation language assume you mean "Which products with sales less than 1000 have sales less than last month?"

Ironically, one problem with fourth-generation languages is the large amount of power they provide. Through implied calculations and searches, a user can make a simple request that might require a great amount of machine time. For example, a user may ask for the top three revenue-producing products in this year's product line. If revenue to date is a stored parameter, this is a simple request; if it isn't, revenues may have to be calculated for every product in the database in order to find the top three, and each of those revenue calculations may in turn require searching through all the orders for that product so far this year.

The more sophisticated fourth-generation languages check for such cases and let you know that the request will take a long time. The program Intellect, for example, tells you that "You have just made a very expensive request in terms of CPU time. Do you wish to make this a batch request?"

THE NATURAL LANGUAGE PROBLEM

Although we take it for granted, the process of understanding a sentence is not a simple one. Furthermore, the information needed to understand a sentence is not always contained in the sentence itself: You need to know its context as well as general information about what makes sense in the real world. For example, most people would interpret the sentence "Who makes the most money?" as a question about earnings because the other meaning (counterfeiting) is not common.

It would be impossible to store all of the general information necessary to understand any possible sentence, so fourth-generation languages all work in a limited domain. This keeps the amount of necessary "background information" manageable. In a database query language, for example, the fourth-generation language typically has access to the terms in the data dictionary and the logical relationships between them, along with some general information about what types of operations are allowed on each type of record or variable.

Along with this information, a fourth-generation language has some general rules for determining meaning. Although fourth- generation languages use many different approaches, most of them go through a process that can be broken down into three steps: syntactic analysis, semantic analysis, and deduction.

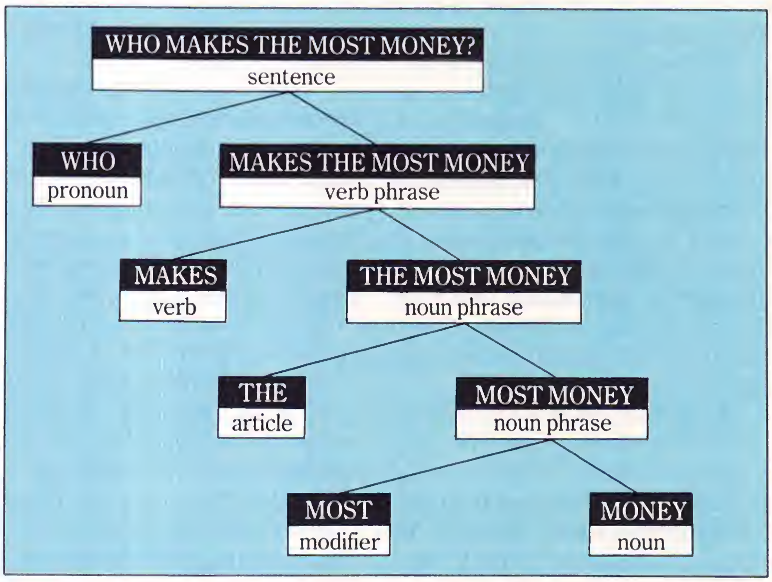


FIGURE 1: SYNTACTIC ANALYSIS OF A SENTENCE BY PARSING

Another potential problem with fourth-generation languages is the complete lack of direction they provide. Menu-driven systems show new users what the system is capable of, but a fourth-generation language provides no hints about where to start. Texas Instruments, a leader in natural language research, has come up with a combination menu-driven/natural language approach that addresses this problem.

CHOOSING A FOURTH-GENERATION LANGUAGE

Can your application benefit from a fourth-generation language? This is a question only you can answer, but here are a few general guidelines to follow.

If you're looking for a DBMS + 4GL, consider how many users the system will have and how often they will be using it. If your application will have only a few users and if all will be using the system daily, a fourth-generation language may not be worthwhile. A fourth-generation language will be more useful if you need to give occasional users and first-time users easy access to the system.

If you're a developer, the choice of a fourth-generation language will be determined by the system you're working with. Because almost all fourth-generation languages have been designed for specific DBMSs, your choice will be limited; in fact, you may not have any choice at all. Although several fourth-generation query languages (such as Savvy, Clout2, and Natural Link) have the ability to work with more than one DBMS, fourth-generation programming languages do not yet offer this capability.

If you have decided that a fourth-generation language can help your application, there are many features to consider.

First, how large is the data dictionary? Can users add new definitions? Is there a limit to the number of user additions?

Second, does it provide semantic extensibility? Can the end-user do this?

Third, does it use accumulated context? Can users see the current context?

Fourth, does it provide a procedural language? Can you combine procedural and nonprocedural languages for a specific problem?

And finally, what are the hardware and software requirements? Does the fourth-generation language interface directly with your existing data or provide a utility that will translate your data to the format it requires?

In addition to these features, you'll want to check how the language handles errors in syntax and logic. Does it recognize misspelled words? Does the fourth-generation language check for inappropriate use of a data type?

Another feature to look at is the installation process. Some fourth-generation languages require that you specify plural definitions explicitly in the data dictionary; in other words, you must define *companies* to be the same as *company*. More advanced fourth-generation languages have this capability built in and will recognize the plural form of a defined data type for what it is. In general, the installation process should be as automated as the fourth-generation language itself.

Because you cannot always get a demonstration of software before you buy it, you may have to make a decision based on printed materials or conversations with salespeople. In

this case, be wary of the jargon used in ads and marketing literature for natural language products. Pay more attention to specific examples than vague descriptions. And, perhaps most importantly, if an example uses an unusual sentence construction, assume that this construction is required. If a simpler or more obvious example were possible, the ad probably would have used it.

THE FUTURE

It remains to be seen whether fourth-generation languages will take the software industry by storm. For one thing, enough products aren't yet available to see how users will react. In addition, heavy advertising of natural language "vaporware" and misleading ads have jaded many potential buyers.

Despite these public relations problems, however, the merits of natural language software for DBMS query are too great to be ignored. Considering that nearly every major DBMS has a fourth-generation query language or has plans for one, it seems likely that natural language capability will soon replace IBM's SQL as the de facto standard query language. This will give corporate information centers the ability to take advantage of new DBMS technology without retraining end-users.

The future of fourth-generation languages for programming work is not as clear. It is already obvious that English-language capability in a fourth-generation language will not be of great help to programmers. Granted, such a language would give nonprogrammers the ability to write simple programs, but designing powerful and innovative applications with that language would be no easier than describing such applications in English. And—as any documentation writer can attest—it sometimes takes many pages of English to describe the operation of a single page of code.

Of course, computer languages exist (Algol-68, for example) that are better suited than English to the task of describing programs, and future fourth-generation languages for programming will probably mimic those languages. This will increase programmer productivity, but even greater gains may be made through the use of icon-based systems, as in the Xerox Star, Apple Macintosh, and the AT&T Unix PC.

And what of the future of fourth-generation languages under the Unix system? One possibility is the development of a fourth-generation language front-end for the Unix system that would generate shell scripts to carry out specified actions. Users could build their own data dictionaries and then ask for results by using English-like queries. Such a shell could easily be ported to any Unix system and would provide the Unix system's full power to inexperienced users.

If this sounds like an overwhelmed Unix novice's secret fantasy, bear in mind that two products already on the market—S1 and the Unix Consultant—come close to this capability, and others are likely to follow. Continued improvement of products such as these will make working with the Unix system as enjoyable for beginners as it is for experts. □

Doug Mahugh is an independent programmer and writer based in Chicago. The former editor of Joystik *Magazine, Mr. Mahugh has also worked as a programmer for the Boeing Company and as a writer and editor of books on microcomputers.*

PROLOG AS A UNIX SYSTEM TOOL

As Prolog interpreters and compilers become faster and as libraries of code become available, Prolog will become a worthy competitor to C.

BY JOHN MALPAS

The logic programming movement (along with the most widespread logic programming language, Prolog) is beginning to attract attention from people on all levels of the computer industry. Born and nurtured in European universities, Prolog is now being used by several government and business institutions in Europe and the United States.

Some of the most enthusiastic support for the new technology can be found in Japan, both on the part of researchers in the Fifth-Generation project, in particular, and in the Japanese software industry, in general. For example, officials at ASR International, a Tokyo software-development company, expect Prolog to become a standard language (along with BASIC, FORTRAN, and C) that all programmers will learn.

To listen to its advocates, logic programming has great theoretical promise. The technology is based on using symbolic logic (as expressed in so-called *Horn clause form*) as a programming language. Symbolic logic was originally developed as a universal language for expressing situations and problems. Logic programming is a subset of logic, but it is equally broad in scope. It can serve as a universal language for expressing other paradigms of computer science, including relational database theory, software engineering, parsing theory, and artificial intelligence (AI) knowledge representation.

What makes logic programming unique is its declarative semantics. A logic program is a set of rules that declare the conditions under which some relationship is thought to exist. Unlike procedural languages, logic programming is not based on the destructive assignment statement. This means that once a variable has assumed a value, the programmer cannot arbitrarily change it.

Because of this, a logic program exists on a higher level than the process it is modeling and is said to *describe* the process. By way of contrast, most procedural language programs exist on the same level as the process they are modeling, and thus they can only *simulate* the process.

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

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

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

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

FORTRIX™ has already been installed on 26 different brands of hardware, so whichever FORTRIX™ version meets your needs, you can be sure it will exceed your expectations in terms of speed and cost savings realized. Why not act now to save **your** bundle? Get full technical details, plus references from among over 100 satisfied licensees, from Jim Flynn at (212) 687-6255, Extension 22, or write to him at Rapitech Systems Inc., Dept. A1, 565 Fifth Avenue, New York, NY 10017.

NEW RELEASE! NOW, FORTRIX FOR VAX/VMS AND MS-DOS!

FORTRIX™ Fortran-to-C Conversionware℠ from

RAPITECH™ Rapitech Systems Inc.

Telephone (212) 687-6255/Telex 509210

*UNIX is a trademark of AT&T Bell Laboratories

Please circle Ad No. 22 on inquiry card.

Finally, logic programming allows the co-existence of a *metalanguage* with the *object language* (Prolog itself). Such ability gives logic programming great extensibility because it allows a programmer to change the behavior of the interpreter without rewriting it.

If this is logic programming's theoretical promise, what of its reality? As a Unix system and C contract programmer, I had occasion to find out last summer, when I wrote a software testing application using Prolog for a major financial institution. The experience confirmed many of the strengths of Prolog but also revealed some features that currently make it an immature application language. Fortunately, the version of Prolog I used for this project is well integrated with the Unix system. The experience leads me to believe that Prolog may have a bright future as a Unix system tool.

THE PRACTICE OF PROLOG

What is immediately attractive about Prolog as a practical language is its inherent discipline: The number of ways to express an idea is limited. As a result, programmers tend to solve the same problem in the same way. Prolog has few lexicographical symbols, so the code is easy to read. Because of Prolog's high level, you can express in a single page a routine that might require 10 or 20 pages of C. This bodes well for maintenance.

Prolog is similar to several high-level declarative languages found in the Unix system, such as `yacc`, `make`, and `tbl`. For example, both `yacc` and Prolog are rule-based languages. Whereas you can usually use `yacc` rules to describe a grammar, you can use Prolog rules to describe anything. A rule in either language consists of a conclusion followed by multiple conditions.

One difference between the two languages is that `yacc` uses a forward-chaining resolution strategy suitable for implementing bottom-up parsers, while Prolog uses a backward-chaining strategy similar to that of a top-down parser.

Another major difference is that `yacc` allows C language procedures to be attached to rules. As a result, each `yacc` rule has side effects that are unexplainable according to the semantics of `yacc` itself. In Prolog, on the other hand, the only way to express anything is with a rule. There are no attached procedures, so the language has precise and completely uniform semantics.

I was able to rip out large sections of code and fix or replace them in a day or two.

When I first started using Prolog, it seemed like a good idea to be able to mix C code with the rule base in a manner similar to `yacc`. I had the impression that C would perform certain jobs such as screen handling more appropriately. But Ken Bowen, a professor of logic programming at Syracuse University, indicated that Prolog is a complete programming language that can stand on its own.

My own experience proves this point of view to be correct because Prolog turned out to be adept even at low-level tasks such as screen handling. Furthermore, a Prolog program is easier to read than a `yacc` program precisely because there are no attached procedures.

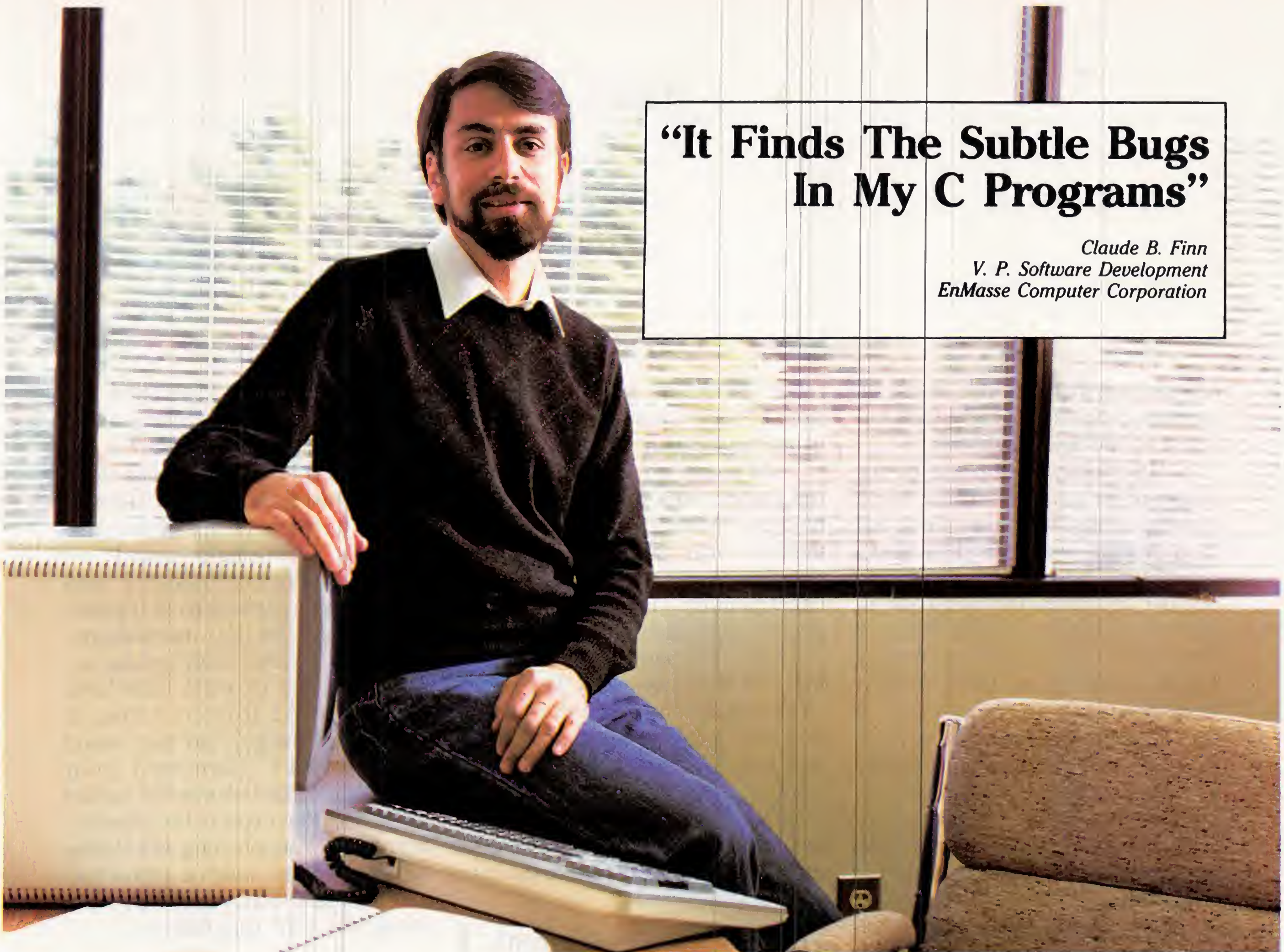
PATTERN MATCHING

Prolog has pattern-matching abilities that in some ways exceed those of Unix system utilities such as `grep` or `sed`. One of my first projects with the language was to write a `sed`-like text-filtering program that performs a set of textual transformations on a file. But where `sed` can match patterns only on the lexical level (that is, patterns involving characters and repetitions of characters), my Prolog program can handle both lexical and word-level patterns.

An example word-level pattern specification is the following: "Find any number of words, beginning with the word *Charles* and ending with a word containing the string *decid.*" This will locate a span of words such as "Charles, who just returned from India, decided . . ." Because this Prolog program operates on a sentence at a time instead of a line at time (as do `grep` and `sed`), it also finds word patterns that cross over line boundaries.

At the beginning of last summer, I was charged with doing human factors testing of PC software for a department in a large financial institution. The typical test situation involved sitting subjects down in front of a PC with the relevant software running and asking them to perform a series of tasks. The subjects were videotaped to determine if certain screens made them nervous, and a file was generated containing every keystroke they made, along with time, error conditions, and so on.

Additional complications arose because we frequently added new tasks and were always changing them. Furthermore, subjects never seemed to go through the tasks in the prescribed order. Nontechnical personnel needed to be able to add new tasks to the system's repertoire at


"It Finds The Subtle Bugs In My C Programs"
Claude B. Finn
V. P. Software Development
EnMasse Computer Corporation

describe anything. A rule in either language consists of a conclusion followed by multiple conditions.

The problem was to analyze the output of the tests, determine which task the subject was attempting, and then produce statistics about tasks completed, number of backspaces, number of erroneous menu paths, and so on. Analysts could then tell from the statistics which instructional wording was bad or which screens were incomprehensible.

Fortunately for me, the director of this project had a predilection for doing the test file analysis on a Unix system. We decided to build a prototype in `awk`, a language he favored because it is of a higher level than C and therefore easier to maintain.

The `awk` prototype worked well for a file containing a single known task. We implemented an `awk` code-writing scheme that enabled an analyst to add a new task to the system simply by going through the task once on the PC, and then using a "build" program on the resulting file.

The aspect of the problem that seemed beyond `awk`'s reach was discovering which task in a file that contained multiple tasks the user was attempting. Because this is essentially a parsing problem, I suggested either `yacc` or Prolog. After long deliberations, we decided to go with Prolog. Like `awk`, Prolog is usually an interpreted language. This gives it an advantage over `yacc` for strategies involving automatic code generation.

THE SYSTEM

The final system was made up of four components: a two-stage lexical analyzer, a parser that also counted errors, a program that automatically added new tasks to the parser, and a report generator. In the initial version, all four components were part of one large Prolog program.

Although we were developing the system on a Convergent Technologies MiniFrame with virtual memory, input data files above a certain size caused the program to run out of memory. (This is where Prolog gets its reputation for being a memory hog.) The bottleneck turned out to be the lexical analyzer. We needed an alternative strategy if the system was ever going to be able to handle files that had more than about 700 lines.

The answer we came up with was to separate the components into three Prolog programs, with the output of the lexical analyzer piped into the input of the parser, and the output of the parser piped into the input of the report generator. Each component thus acted as a code generator for the next one.

Prolog's biggest current disadvantage is that it requires large amounts of memory and CPU resources.

For instance, the lexical analyzer asserted a set of facts (containing the input data) into the database of the parser and then caused the parser to begin executing by sending it a query. The parser looked up the data asserted by the lexical analyzer, parsed it, and then asserted a set of facts containing its output into the database of the report generator. Using this strategy, the lexical analyzer could read the input file in discrete, manageable segments, and therefore there was no danger of running out of memory.

A shell script controlled the whole process. It sent a query to the first Prolog program to start it up, and then `cat`ed the input data file to it. This shell script could also turn on tracing for parts of the Prolog program that were being debugged.

The version of Prolog we used did not have a particularly rich library of predicates for data manipulation. To produce respectable-looking reports, we had to write such basic items as `atoi` (convert a string of digits to an integer and vice versa) and `printf` (formatted print statement). This lack must be caused by the fact that many of the other researchers experimenting with Prolog are not concentrating on report formats. In fact, both predicates were straightforward to implement.

CONCLUSION

So far, the people for whom I built the system seem to be happy with it. The aspect of Prolog that most impressed me in this project was ease of maintenance. During the development process, I frequently brought a new version of the system to the customer, demonstrated it, and then listened to feedback about what was wrong with it or what new features I needed to add. I was able to rip out large sections of code and fix or replace them in a day or two. To perform such major surgery on an equivalent C program would have taken at least a week.

Prolog's biggest current disadvantage is that it requires large amounts of memory and CPU resources. When evaluating the cost-effectiveness of doing an application in Prolog as opposed to a leaner technology, consider the trade-off: programmer productivity versus efficient machine utilization. Personally, I will always choose programmer productivity at the expense of machine efficiency, having learned the hard way that no specification is immutable.

It should be clear from this article that Prolog thrives in the Unix system environment. What may not be as clear is that Prolog has a great deal to offer the Unix system. For example, you can use it when the limitations of `awk` or `sed` become unworkable.

> **Input data files above a certain size caused the program to run out of memory; this is where Prolog gets its reputation for being a memory hog.**

Prolog is also a highly productive application language in its own right. As Prolog interpreters and compilers become faster and as libraries of Prolog code become available, we can look forward to a time when Prolog will be a viable and worthy competitor to C and other high-level languages for many types of applications. □

John Malpas, a consultant in New York, teaches a Prolog programming workshop at Structured Methods Inc. He is the author of a book entitled Introduction to Prolog and its Applications, *to be published this fall by Prentice-Hall.*

MODULA-2 KNOCKS ON THE DOOR

Because of its special features—in particular, its modular structure—Modula-2 could become a leading programming language by the end of the eighties.

BY GIACOMO MARINI

The C language has been integrated with the Unix system for many years, but this historical partnership is an insufficient reason to preserve C as *the* primary language for the Unix system. Although the basic capabilities of the Unix system, as well as the utilities and applications available with it, have evolved substantially since the Unix system's inception, C has changed minimally during this time. The dual popularity of the Unix system and C stems from attributes of the Unix system, not from special features of C.

Given that this situation is an accident of history, we should be prepared to consider some alternatives. The purpose of this article is to propose Modula-2 as one such alternative.

Modula-2 is a general-purpose programming language designed by Nicklaus Wirth, the father of Pascal. During the last 20 years Wirth has defined several structured programming languages. Modula-2, the latest of Wirth's linguistic developments, benefits from its roots in Pascal, Modula, and Mesa, a language developed at the Xerox Palo Alto Research Center (PARC). Modula-2 incorporates the positive features of its ancestors as well as remedies for their difficulties.

Wirth believes that in the field of software design "nothing can replace a creative person's way of thinking," and after developing Pascal he began directing his own creativity toward multiprogramming. In his experimentation with building a language that reaches beyond the limitations of Pascal, Wirth defined a language, which he dubbed *Modula,* that was strongly oriented toward concurrent processing.

Filled with ideas sparked by his experimentation with Modula, Wirth left for a year's sabbatical at Xerox PARC. There, he worked with the architects of Xerox's Alto computer, the prototype for its Star 8010 Professional Workstation, and studied the Mesa language.

When Wirth returned to the Swiss Institute of Technology, he drew upon his experiences with Pascal, Modula, and Mesa to produce Modula-2. He then designed the Lilith machine, a customized, micro-coded computer with a high-resolution bit-mapped display similar to that of the Xerox Alto workstation. The entire software environment of the Lilith was implemented in Modula-2.

DIFFERENCES FROM ITS FOREFATHER

Although similar to Pascal, Modula-2 differs from its forefather, both syntactically and conceptually, in four distinguishing features. First, the basis for program development and separate compilation is the module. Second, the language supports concurrent processing using coroutines. Third, the syntax of Modula-2 is more systematic. And fourth, Modula-2 maintains the structure of a high-level language while providing for low-level programming and machine-level access.

The most important feature that distinguishes Modula-2 from other languages is its use of modules, which allow you to extend Modula-2 for special-purpose applications. You can divide large programs into collections of separately compiled modules, and you can then place them in a library that different programs can use. For example, Modula-2, like C, relegates input/output (I/O) functions, system parameters, and commonly used routines to modules that are included in a standard module library.

Another important attribute of the language's modules is that you can compile them separately, but not independently. When you compile a program that refers to (imports from) a library module, the compiler performs complete type checking to verify the data compatibility of the interface between modules. A correct implementation of the language provides as much checking between two separately compiled modules as within one module.

However, the module concept goes beyond simple program partitioning. Each module itself is composed of a definition section (definition module) and an implementation section (implementation module). This sectioning separates the interface aspects of a module from the implementation details so that revisions of the implementation module do not affect the interfaces.

Modula-2's co-routines allow programmers to write powerful interrupt drivers, process schedulers, and quasi-concurrent operations easily. Co-routines execute independently but not simultaneously.

Modula-2 also has a systematic syntax (which provides a better bracketing of control and data structures), an improved case statement, and several other less significant improvements that make it easy to learn and use.

SOFTWARE DEVELOPMENT

Although Modula-2 is a high-level language, it was designed for all types of software development, including the lowest levels of I/O drivers, operating systems, and read-only memory (ROM) code. It provides low-level facilities for manipulating machine-level data, determining the memory address of variables, and accessing peripheral device registers residing at fixed memory locations. Modula-2 also retains the high-level features of its predecessor, Pascal, including compulsory declaration of variables, clear description of data structures and algorithms, and type checking of data.

Modula-2 is inherently portable; system-dependent facilities may be easily restricted to specific modules.

Because of its special features —in particular, its modular structure—Modula-2 has the potential to become a leading programming language by the end of this decade.

The trend of software development, for both in-house and market distribution, lies with large programs designed by teams of programmers. Modules are an essential feature of this type of large development project because they allow for clear and explicit division of the program into modular parts. The library, to which you relegate modules and system parameters, facilitates interface between programmers.

Consistent with the philosophy of structured programming, the separate compilation capacity of Modula-2 helps you to produce bug-free, reliable, and maintainable computer programs because you can build large programs from collections of distinct, smaller ones. Furthermore, Modula-2 is inherently portable because system-dependent facilities may be easily restricted to specific modules.

A COMPARISON

Modula-2 and C have many similarities. Both provide facilities for relaxed type checking and direct access to memory words and addresses. Both also provide primitive operations close to the level of the machine.

However, the two languages also have significant differences. In C, memory words and addresses are freely accessible, whereas in Modula-2 this access is tied to specific language constructs. Furthermore, unlike C, Modula-2 can enforce type checking of parameters to library routines. Also, Modula-2 has more extensive low-level capabilities (specifically, support of co-routines), and it is a strongly typed language.

In Modula-2, you can override type checking systematically, and the text of the program indicates exactly what you have overridden. This feature is important when several programmers are involved in the development and maintenance of software and when new programmers will inherit code.

The concept of modules is the strongest attribute of Modula-2. C uses independently compiled "compilation units"—source files into which you can break a program down—which are similar to the modules in Modula-2. C's "external" variables correspond to Modula-2's "imported" objects, or externally declared objects that are visible within a module.

The root of the differences between the compilers, though, lies in what a C compiler knows about an external object and what a Modula-2 compiler knows about an imported object. A Modula-2 compiler knows everything, but a C compiler knows little or nothing.

A Modula-2 compiler performs data type checking across compilation units—it performs separate compilation. A C compiler does not check data types across compilation units—it performs independent compilation. Separate but not independent compilation is another valuable feature when more than one programmer will be working on a program.

The advantages of more stringent type checking and control over module interfaces do not prevent the production of efficient code. A recent Modula-2 implementation (by Mike Powell, then with the DEC Western Research Center in Santa Clara and now at UC Berkeley) demonstrated that Modula-2 code can be as efficient as C code.

Modula-2's relevance to the Unix system becomes clear when you consider the future of computers. Because of the Unix system's extensive capabilities and progressive adaptation during the past 10 years, many people predict that it is the operating system of the future.

Modula-2 is a strong candidate for the programming language of the future primarily because of its modu-

> **A Modula-2 implementation for the Unix system would also make the development of new applications easier, safer, and faster.**

lar structure and because of features such as its provisions for concurrency, portability, and type checking. Because Modula-2 can perform type- and consistency checking, its libraries offer advantages over available Unix system C libraries. A Modula-2 implementation for the Unix system would also make the development of new applications easier, safer, and faster. In short, its modular structure is paving the way toward a connection between Modula-2 and the Unix system world. □

Giacomo Marini, a co-founder of Logitech Inc., Redwood City, Calif., is vice president of that firm's software division. He is responsible for the family of Logitech Modula-2 software development tools as well as contract software development project. Prior to the founding of Logitech, he was with IBM and Olivetti.

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

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

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

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

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

Available now. For additional information, call or write,

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

Mark
Williams
Company

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

Please circle Ad No. 21 on inquiry card.

EXPOSING the PC to Unix System Connection

Wait, don't throw away those dedicated terminals. Consider hooking your PC to a Unix system.

BY ALAN WINSTON

Face it. There's a lot to be gained by hooking your personal computer to a Unix system, ranging from a little extra desk space to a powerful new multitasking environment for nontechnical users. The range of possible connections is immense—from dumb-terminal connections to full-network configurations—and one or more of them might be right for your application.

Let's look a little closer at the reasons not to buy a terminal but instead to connect the PC you already have to a Unix system.

First, buying a terminal costs money. The cheapest terminal I've run across lists at almost $600. Terminal-emulator software, which will make your PC appear as a familiar terminal to the Unix system host, is usually much cheaper.

Second, you have to learn to use yet another keyboard. The Macintosh and the IBM PC have idiosyncratic keyboard layouts that take a while to learn. Switching from one to another all day can be slow and irritating.

Third, multiplying hardware eats up desk space. A few months ago I visited the office of a friend who was working on some IBM PC software. On his desk were a special-purpose text-processing terminal connected to the Unix system host, a Compaq PC for testing software, and a 3270 PC serving as a terminal to an IBM system. Although all this created a high-tech-looking environment, it didn't leave any desktop space for such important programming aids as a cup of coffee or a telephone.

Fourth, if you want anything fancy, such as a very large display or a Dvorak keyboard layout, it's fairly easy to get it on a PC. Although most terminals don't have any provision for alternate video output or keyboard reconfiguration, most PCs do.

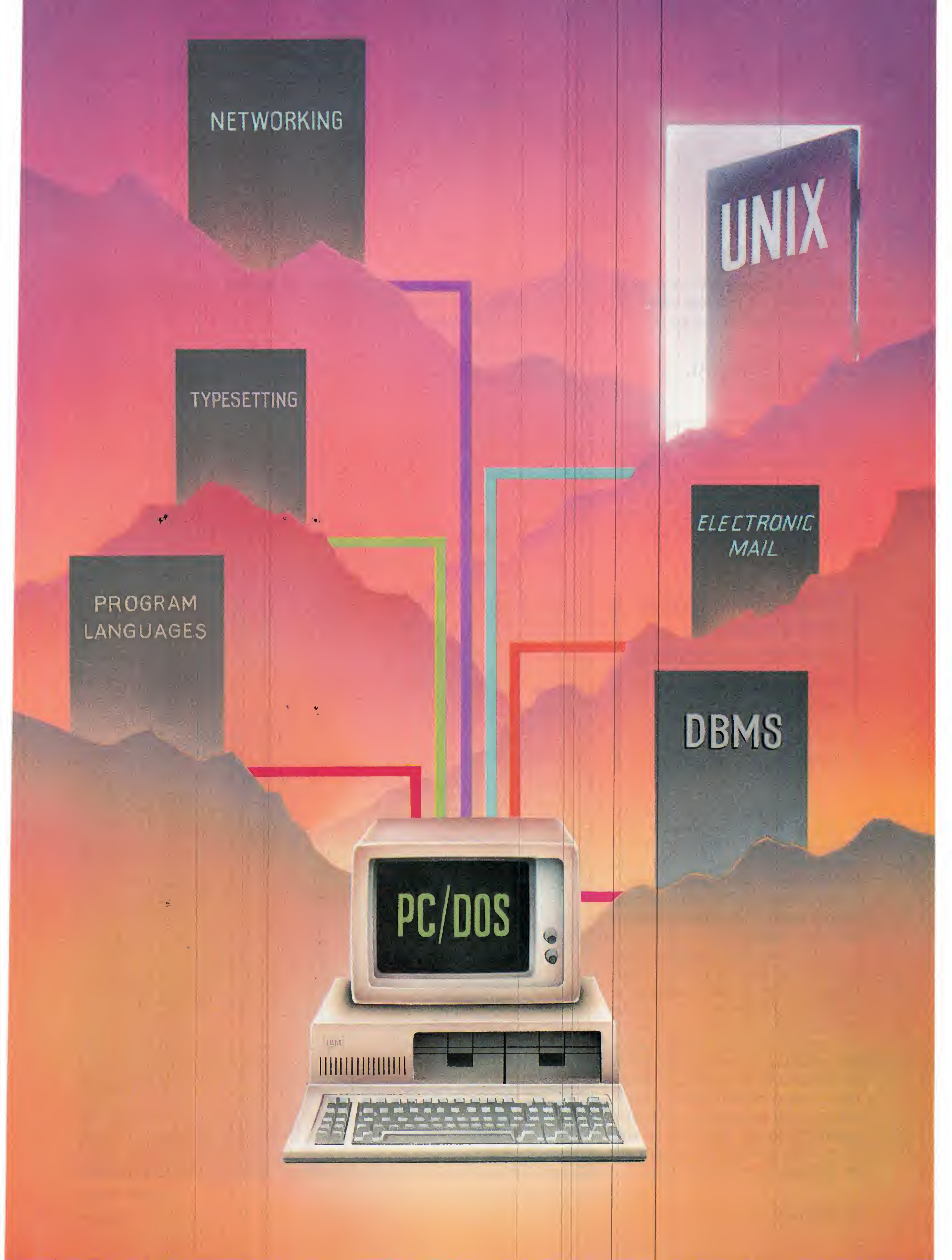
If you already have a PC on your desk and want to use it as a terminal, you have several choices.

TTY EMULATION

To start at the least sophisticated end of the spectrum, you can easily use almost any microcomputer that has an RS-232 port as a dumb terminal in TTY emulation mode—a mode in which the micro acts like a teletypewriter terminal. The host system communicates line by line, with new messages appearing on the bottom line of the screen and old messages scrolling off the top. This is the kind of terminal most on-line databases expect, and it is as generic a communications protocol as you can find.

TTY emulation is easy to code on the micro. All the PC has to do is read one character at a time from the communications port, echo it to the screen, read the keyboard, and echo that key press to the host. This software doesn't take many lines of BASIC or even a great deal of technical knowledge to produce. Moreover, you don't even have to code it yourself for most micros; vendors such as IBM, Kaypro, Apple, and many modem manufacturers offer TTY emulation packages fairly cheaply. Such software is often available free in the public domain, too.

Running your PC as a TTY terminal has its disadvantages, though. Any application that requires cursor addressability (the capacity to put a character anywhere you want on the screen) won't run. This means that full-screen editors, word-processing


NETWORKING
UNIX
TYPESETTING
ELECTRONIC MAIL
PROGRAM LANGUAGES
DBMS
PC/DOS
IBM

packages, spreadsheets, and so on won't work. `vi` is a lost cause, although you can still use `ed`.

SMART-TERMINAL EMULATION

If you want to run your PC as a smart terminal, with cursor addressability and page-mode transfer (the ability to read everything on the screen at once, which is very helpful if you want to use the local editing capability of your terminal or micro), you have two choices.

> **Berkeley Unix's `termcap` offers a means of describing new terminals to the system so that editors such as `vi` can use them without being laboriously changed.**

One way is to teach your PC to talk to the Unix system in the same manner that it talks to a terminal it already knows. Apple sells MacTerminal and LisaTerminal, which emulate DEC VT-100 terminals (among others), for the Macintosh and Lisa, respectively.

More terminal emulators are available for the IBM PC than I can count, and they are also available for various CP/M systems. (The Kaypro uses the same control sequences as the adm-3a terminal, which makes that emulation a breeze.) Emulators are available for many popular terminals on many popular micros, and if the particular terminal you want isn't supported on the particular PC you have, it's reasonably straightforward for a programmer to write one.

FIGURE 1: TERMINAL CHARACTERISTICS

Terminal Characteristics

`co` — number of columns (numeric)
`li` — number of lines
`pt` — physical tabs (Boolean)
`os` — overstrike capability (Boolean)
`hc` — hardcopy terminal (Boolean)

Screen Movement

`bs` — backspace capability (Boolean)
`am` — automargins (Boolean)
`up` — cursor up
`nd` — nondestructive forward
`cm` — cursor motion
`ho` — home
`sr` — scroll reverse
`do` — down
`bt` — back tab
`hd` — halfline down
`hu` — halfline up

Screen Updating

`cl` — clear entire screen
`ce` — clear to end of line
`cd` — clear to end of display
`dl` — delete line
`al` — add line
`dc` — delete character
`im` — insert mode
`ei` — end insert mode
`ic` — insert character

Arrow and Function Keys

`ku` — keyboard up
`kd` — keyboard down
`kl` — keyboard left
`kr` — keyboard right
`kb` — keyboard backspace
`kh` — keyboard home
`ks` — keyboard start (for special keypad work)
`ke` — keyboard end (for special keypad work)
`kn` — number of function keys (numeric)
`kN` — function keys (N equals 1 through 10)

Highlighting

`so` — standout
`se` — standout end
`ue` — underscore end
`us` — underscore start
`vb` — visual bells
`sg` — standout glitch
`ug` — underline glitch

Software Control

`tc` — use another `termcap` definition
`is` — initialization string (for setting options)
`if` — initialization file (for setting tabs, etc.)
`ma` — map (for old versions of `vi`)
`te` — terminal end (for cursor motion)
`ti` — terminal initialize (for cursor motion)
`ve` — visual end (for `vi`)
`vs` — visual start (for `vi`)
`ae` — alternate character set end
`as` — alternate character set start

Special Characteristics

`ip` — insert pad after each character inserted
`mi` — safe to move in insert mode
`xn` — newline glitch
`ll` — last line, first column
`nl` — newline character if not ^J
`ta` — tab character if not ^I
`bc` — backspace character if not ^H
`pc` — pad character if not NULL
`cr` — carriage return if not ^M
`ms` — safe to move in standout
`cs` — change scrolling region (vt100)
`bw` — backspace wraps from column 0 to last column

Continued on page 63

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

```
Mt|trs80|trs-80|radio shack trs-80 model I:\
        :am:bs:co#64:li#16:
Ma|apple2|apple II (unextended):\
        :am.bs:os:co#69·li#24:cl=^L:
```

FIGURE 2: TERMCAP ENTRY FOR TRS-80 AND APPLE II

```
kb|h19|heath|h19b|heathkit|heath-19|z19| zenith|heathkit h19:\
        :am:bs:mi:ms:pt:co#80:li#24: cl=\EE:cd=\EJ:ce=\EK:cm=\EY%+
          %+ :\
        :al=1*\EL: dl=1*\EM:ei=\EO: im=\E@:dc=\EN:up=\EA:do=\EB: nd=\EC:\
        :ho=\EH:as=\EF: ae=\EG:sr=\EI:so=\Ep: se=\Eq:vs=\Ex4:ve=\Ey4:\
        :kb=^H:ku=\EA: kd=\EB:kl=\ED:kr=\EC:kh=\EH:kn#8:\
        :k1=\ES:k2=\ET·k3=\EU: k4=\EV:k5
=\EW:k6=\EP:k7=\EQ:k8=\ER:
```

FIGURE 3: TERMCAP ENTRY FOR H89 AND Z89

```
z8|trs80ii|trs-80-II|radio shack trs-80 model II (P&T CP/M):\
        :am:bs:pt:ms:co#80:li#24: cl=^L:ce=^A:cd=^B:cm=\EY%+
%+ :\

:al=^D:dl=^K:ho=^F: up=^^:do=^-:nl=^-: nd=^]:se=^O:so=^N:\
        :kb=^H:kd=^-:kl=^\: kr=^]:ku=^^:
z2|apple2e|appleII|apple II plus:\

:am:bs:pt:co#80:li#24:cl=^L:ce=^]: cd=^K:cm=^^%r%+
%+ :\

:is=\024T1\016:so=^O:se=^N: ho=\E^Y:nd=^\:up=^-:do=^J: ma=^ul:\
        :vs=\024\103\066:ve=\ 024\103\062:vb=\024G1\024T1:
zi|apple-80|apple II with smarterm 80 col:\

:am:bs:co#80:li#24:cl=50^L:cd=50^K· ce=3^]:cm=5^^%%+
          %+ :\

:nd=2^\:up=2^:do=^J:bt=2^R:is=TR^V^A:
```

FIGURE 4: TERMCAP ENTRY FOR TRS-80 II AND APPLE IIE

A colleague of mine just spent three weeks writing a program to emulate a Televideo 950 on an IBM PC. (It took him that long only because he was at first unfamiliar with the Intel 8086 assembler and had to learn it to write the code.)

The second choice is to teach your Unix system to talk to your PC. The Berkeley Unix system offers a means of describing new terminals to the system so that editors such as `vi` can use them without being laboriously changed. Called `termcap`, this text file is a database of terminal descriptions, each one containing appropriate information—such as cursor-positioning sequences, backspace characters, number of lines on the screen, how to highlight parts of the screen, what values the function keys return, and other vital information—for each major terminal and many popular PCs.

A new entry in the `termcap` file may suffice to enable you to use your PC as a smart terminal. Reprinted in Figures 1 through 6 are some sample personal computer `termcap` definition files from Bill Tuthill's `termcap` tutorial articles in UNIX/WORLD, Vol. 1, Nos. 1 through 3.

However, nothing I've described so far will handle file transfers, present a "user friendly" interface, or do anything at all for you that a VT-100 or a TeleVideo 950 won't do. And for many novices, this situation presents a problem.

BRIDGING THE GAP

Several products have recently reached the market that bridge the gap between the Unix system and popular PCs (mostly IBM and those running MS-DOS). Others have been announced and are currently in beta-site testing. These products use the PC and the host to better advantage—some of them may even make it worth buying a PC and getting rid of the terminal.

> **Several products have reached the market that bridge the gap between the Unix system and popular PCs.**

One "bridge" product connecting MS-DOS and Unix systems is the Locus PC Bridge, from Locus Computing Corp. (Santa Monica, Calif.). This product, which was reviewed fully in Vol. 1, No. 6 of UNIX/WORLD, makes the Unix system a file server for the PC, which is under the simple impression that it has a very big disk drive.

All PC applications run locally and can take data from the Unix system host across a local-area network (LAN) link, such as Ethernet or AT&T's IS Net. This lets the executive who wants to run Ashton-Tate's dBase III against massive files on the host do it without worrying about where the data live and what the fully qualified name of the file is. Users never have to know that the

Unix system is on the other end of the wire.

Another part of the product is a terminal emulator, which does input/output (I/O) to the host in screen-size blocks, requiring only one interrupt for the entire transfer. This considerably reduces the load on the host compared to one interrupt per character, and makes a substantial difference when several users are running vi, for example.

Locus' strategy is to sell only to original equipment manufacturers (OEMs), so the price is dependent on where you buy it. As far as I know, AT&T is still the only OEM of this product, which means that the only hosts you can buy it for are the 3B series computers.

Another software product that distributes the load of editing and text production away from the Unix system host and down to a PC —in this case, even an IBM PCjr— is Sprouts, a family of software products from Slater Towar (Estes Park, Colo.).

The first of these products, the Writer, is in beta test as I write and was due for release by February. In essence, it is a "what you see is what you get" text and graphics editing system that displays user-definable characters—including graphics characters and charts—formatted exactly as they will be printed, even down as far as proportional spacing. This could be a great aid if you want to produce publications that integrate text and graphics in a single document, and it costs substantially less than, say, a Xerox publication system.

According to the company, the Sprouts family will be able to use data from another machine so that the editing terminal—which requires only 128K bytes and a graphics card—can edit files much too big

```
Mo|osb|osbornel|osborne 1:\
        :bs:ms:xt:co#80:li#24:cl=^Z: ce=\ET:cm=\E=%+%+ :\

:up=^K:do=^J:nd=^L: ic=\EQ:dc=\EW:al=\EE:dl=\ER:im=:ei=:\

:ku=^K:kd=^J:kl=^H:kr=^L: so=\E):se=\E(:ul:us=\El:ue=\Em:
```

FIGURE 5: TERMCAP FOR OSBORNE I

```
Mf|f100|freedom100|freedom|freedom 100 by Liberty:\
        :am:bs:bw:mi:ms:pt:co#80:li#24:kn#20: ct=\E3:st=\El:is=\Eg\Ef:\

:kr=^L:cl=^Z:up=^K:do=^J: ho=^^:kb=^H:kl=^H:\: kd=^V:ku=^
           K:nd=^L:\

:ko=dc,al,dl,cl,bt,ce,cd:ma= ^Hh^Vj^Kk^Ll^^H^Z^L:\
        :ch=\E]%+ :cm=\E=%>+ %+: cv=\E[%+:sr=\Ej:bt=\EI:\
        :al=6.5*\EE:cd=\EY: ce=\ET:dc=\EW:dl=11.5*\ER:ei=\Er: im=\Eq:\
        :se=\EG0:so=\EG4:ue=\EG0: us=\EG8:as=\E$:ae=\E%:\

:k1=^A@\r:k2=^AA\r: k3=^AB\r:k4=^AC\r:k5=^AD\r:\

:k6=^AE\r:k7=^AF\r:k8=^AG\r: k9=^AH\r:k0=^AI\r:\
        :hs:ts=\Eg\Ef:fs=\r:ds=\Eg\Ef\r:
```

FIGURE 6: TERMCAP FOR FREEDOM AND MICRO DECISION I

for any of its local storage. Slater Towar will be offering a cartridge version for the PCjr so that it can run as a diskless workstation and edit large files on a Unix system host.

The Sprouts package also allows the PCjr to run as a standard terminal. The only difference is that text sent to the screen is formatted as it comes in, and you can then capture and manipulate it as desired. File-server software is required on the host and is included free with the purchase if that host machine is already supported. If it isn't, Slater Towar requires a minimum order of 400 copies (at $50 each) to make the port.

DOING THE TANGO HELPS

Commercial Office Systems Inc. (COSI) of Ann Arbor, Mich., offers a product now called Tango but originally known as Communiqué. Tango, I suppose, is easier to pronounce.

To a Unix system host, Tango emulates a VT-52, VT-100, IBM 3101, or Tektronix 4010 graphics terminal, either through a modem or directly. Even when logged on to the Unix system, users can direct the PC to execute an MS-DOS program using the pcexec command. File transfer is supported with handshaking to ensure data integrity, using commands such as pucp to copy from the PC to the Unix system and upcp to copy from host to PC. Users can direct program output from host to PC or PC to host. Furthermore, they can direct the host to run a program on a PC other than the one from which the command originated.

On the PC side, Tango has capabilities such as automatic log-in, auto-dial of numbers, and execution from a "script" stored in an MS-DOS file. Tango runs on IBM PCs and compatibles that have 128K bytes of RAM and MS-DOS Version 2.0 or higher. It runs in conjunction with any host operating system that supports ASCII file transfer, but the company claims it does significantly more on a Unix system. AT&T will market it for the

3B series and AT&T 6300 PCs, and COSI itself sells Tango for many different Unix system machines and PCs.

The Connectables series from Touchstone Software (Seal Beach, Calif.) networks IBM PC-compatibles and Macintoshes as workstations with a Unix system host. The Connectables family currently includes PCworks for MS-DOS machines, MacLine for the Macintosh, and UniHost for one of the more than 50 host machines that Touchstone supports. Each package provides file and document transfer from all workstations to the host or to each other, electronic mail, and terminal emulation (of VT-52 and VT-100 terminals).

The Unix system can either be used as a "command server," performing I/O for the PC user without requiring special knowledge of Unix, or it can be logged in to directly or through a menu. Various levels of technical knowledge are supported, making this system appropriate for everyone from a secretary to a programmer.

Don't throw away your dedicated terminals; I hear they make great paperweights.

In sharp contrast to this approach is Ultra-Mac, from Lutzky-Baird Associates of Culver City, Calif. This Macintosh software, now in beta test, will not provide terminal emulation capability at all. In fact, the Unix system interface is off-limits to the Ultra-Mac user, who is expected to be a nontechnical type—such as an advertising copywriter—working with others on a team project.

Ultra-Mac displays a hierarchy chart, something roughly akin to a genealogical chart on the Macintosh, and this chart contains only the portion of the Unix file system that a given user is allowed to access. By using the mouse, the user selects a file to check out—to copy to his or her local diskette. The user can then edit the file with MacWrite or Microsoft Word and check it back in—all without ever knowing the Unix system is on the other end. Also part of the product is a Unix system-compatible electronic-mail system that facilitates communication between people working on related projects.

Subsequent releases will in-

SAMPLING OF PC/UNIX SYSTEM LINKS

PRODUCT/VENDOR	REQUIREMENTS	PRICE HOST/PC
Tango COSI (Ann Arbor, Mich.) (Soon to be available through AT&T)	IBM PCs and compatibles with 128K-byte RAM and one disk drive; supports "a variety" of Unix system hosts.	$295/$195
Locus PC Bridge Locus Computing Corp. (Santa Monica, Calif.)	IBM PCs and compatibles with 64K-byte RAM, one disk drive, and LAN connection.	OEM only
Ultra-Mac Lutzky-Baird Assoc. (Culver City, Calif.)	Apple Macintosh with 128K-byte RAM and Zilog or Charles River host LAN—Applebus or other.	$1595/$200
Sprouts Slater Towar (Estes Park, Colo.)	Any model of IBM PC with 128K-byte RAM and a graphics monitor; various host systems.	free/varying from 1 @ $80 to 500,000 @ $10
PCWorks Touchstone Software (Seal Beach, Calif.)	IBM PC or compatible with 128K-byte RAM, 280K-byte disk storage, MS-DOS 2.0 or later, and RS-232 port and cable.	$195
MacLine Touchstone Software	Apple Macintosh with 128K-byte RAM, and RS-232C port and cable.	$145
UniHost Touchstone Software	More than 50 different Unix systems with 30-60K-byte disks and serial log-in ports.	$295

PREVIEW: MULTIPLEX AND RTI's INGRES PC LINK

Interest in PC-to-Unix system gateway products is rising, and products that address this need are among today's hottest sellers. But anxious users may quickly find that many of these gateways don't go far enough.

And that's where Network Innovations' new Multiplex PC-to-Unix-system connection software comes in. It goes one (and maybe two) steps further than many competitive products, providing not only for file storage and resource sharing, but also for data interchange between the most popular PC applications programs—Lotus 1-2-3, dBase II, WordStar, MultiMate, Multiplan sylk, and VisiCalc DIF files—and Unix system-based relational DBMSs.

Typically, most PC-to-Unix-system gateway products allow the PC to run off-line as an MS-DOS engine and on-line as a standard Unix system terminal, and to share the Unix system's typically larger, more powerful, and usually more expensive peripherals.

However, that's as far as most of those products go.

Multiplex allows an end-user to take a Multiplan file using Microsoft's sylk data format, for example, to store a spreadsheet in a Unix system-based relational DBMS and to later call it up from an ASCII terminal as a standard Unix system file. Multiplex converts Unix system files to PC files for Lotus, Multiplan, MultiMate, VisiCalc, dBase II, and WordStar, as well as for any other PC applications software that supports those packages' file formats. And vice versa.

Introduced rather quietly at this year's UniForum '85 conference in Dallas, Multiplex is the first product of startup software house Network Innovations of San Jose, Calif., and is the brainchild of co-founders Jim Groff (president) and Paul Weinberg (vice president), both of whom are former Plexus staffers.

Multiplex's capabilities are all housed behind a Lotus-like front-end to boot, so the 800,000 or more Lotus users in corporate America won't have to learn a new set of commands. And at the back-end, Multiplex supports SQL or SQL-like interfaces to the DBMS. Currently, it works with Unify, Relational Database Systems' Informix, and Relational Technology Inc.'s Ingres under System III, System V, 4.2BSD, and Xenix.

Multiplex's license fees are quite reasonable, about the price of a popular PC software package—$595 for a Plexus-class Unix system host and $125 for each PC.

The first commercial implementation of Multiplex to reach end-users will likely result from Network Innovations' recently announced OEM/joint development agreement with Relational Technology Inc. (RTI) of Alameda, Calif. RTI, perhaps best known as the purveyors of commercial Ingres, will market the product as Ingres PC Link.

—Philip. J. Gill

clude print spooling, Unix system terminal capability, and use of the Unix system as a "disk" for the Macintosh. This last capability will eliminate the need for documents to be downloaded to a local floppy. Later, similar capabilities will be implemented for the IBM PC, according to Lutzky-Baird representatives.

Lutzky-Baird said that the non-technical users in the firm's beta-test site love the product and don't want to give it up when the testing is over. As unusual as that might sound, that's how you might feel about any one of these products or any one of the other PC-Unix system connection products on the horizon. Don't throw your dedicated terminals away, though. If nothing else, I hear they make great paperweights. □

Alan Winston is a freelance writer and full-time programmer/analyst who has been interested in the Unix system for several years. His last work for UNIX/WORLD Magazine appeared in the June 1985 issue.

IN SEARCH OF SOFTWARE EXCELLENCE . . . AT&T LEADS THE WAY

AT&T COMPUTER SOFTWARE GUIDE

AT&T Information Systems, Inc.

Announcing Certified and Reviewed software for your AT&T computer

Here is the authorized reference of software products for the AT&T 3B2, 3B5 and 3B20 computers utilizing UNIX™ System V and the AT&T PC 6300 running MS™-DOS. This guide is the source when making intelligent decisions regarding your purchase of hardware and software. The AT&T COMPUTER SOFTWARE GUIDE contains over 500 pages of definitive listings including:

- Company and Product Name
- Application Category
- Product Description
- Programming Language
- Hardware and Software Requirements
- Support
- Computer
- UNIX™ System V Release No.
- Availability/Retail Price Distributor

This striking guide includes a full color advertising section and multiple indexes.

Please Rush

____ copies of The UNIX™ System V Software Catalog R8068-0 ($19.95) each.

____ copies of the AT&T Computer Software Catalog R9276-8 ($19.95) each.

☐ Bill me ☐ Check Enclosed

Name ________________

Company ________________

Address ________________

City/State/Zip ________________

Phone Number ________________

Please include your state's sales tax.
Enclose payment with your order and publisher pays shipping, up to 6 books.

Reston Publishing Company, Inc.
11480 Sunset Hills Road
Reston, Virginia 22090
(703) 437-8900 within Virginia
1-800-336-0338 outside Virginia
Hours are 9 am to 5 pm EST.

V-0955-8H(9)

THE UNIX™ SYSTEM V SOFTWARE CATALOG

AT&T Technologies, Inc.

UNIX™ System V sets the standard—here is your authorized reference manual.

This is the new AT&T catalog of software products designed to run under the popular UNIX™ System V. THE UNIX™ SYSTEM V SOFTWARE CATALOG demonstrates how the UNIX™ operating system works as a common denominator for different computer hardware and software. This comprehensive manual provides over 200 objective, non-evaluative listings of software products including the following information:

- Company Name
- Product Name
- Application Category
- Product Description
- Programming Language
- Code Format
- Hardware and Software Requirements
- Computer Systems Supporting the Product

The attractive UNIX™ SYSTEM V SOFTWARE CATALOG contains multiple indexes by Company, Product and Category for easy reference.

Please circle Ad No. 68 on inquiry card.

MCBA introduces shrink-to-fit software.

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

Down to micro size.

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

Alter the fit? Absolutely.

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

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

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

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

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

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

So whether you're a dealer or a user, find out about it. Call us now at (818) 957-2900.

Shrink-to-fit software. For growing businesses.

MCBA®

Minicomputer Software for Micros.

2441 Honolulu Avenue, Montrose, California 91020

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

Please circle Ad No. 94 on inquiry card.

THE ALTOS 486: AN 'AT' BUSTER?

BY BRUCE MACKINLAY

This month our gonzo reviewer discusses the crazy notion of competing directly with IBM.

Altos 486, a Xenix-based multi-user supermicro.

"Compete directly with giant IBM, are you crazy?" But that is just what little Altos is doing with its 486, a Xenix-based multiuser supermicro for small businesses. So the question is, how does the Altos 486 compare to the IBM PC/AT? To find the answer to that question, read on.

The Altos 486 has a number of things going for it in its battle against the PC/AT. First, it is an Altos, and Altos has a good reputation for making quality machines. Second, it is cheaper than the PC/AT ("AT" for short). A comparably equipped PC/AT costs $7315, while the 486 costs $6490, excluding title, destination, and license. Finally, and most important of all, the Altos 486 is one of a family of machines, from the 186 to the 986, that are tried-and-true small-business machines. So the long and short of it is that the Altos 486 has a lot going for it. (The newly introduced 3068, though, is not a member of this family.) The real question is, will this be enough? Many bigger companies with better

Last month the employees of Fortune Systems did something extraordinary.
They took up a voluntary collection to place this message in UNIX World.

What motivates employees to spend their hard earned cash to make such a statement?
It's simple. They're very proud of what they do.

And with reason. It *is* a damned good computer system. Just ask the people who build it.
Or ask our thousands of customers. They'll tell you the same thing.

The employees of Fortune Systems believe in their product.
As President of Fortune Systems, I believe in them.

FORTUNE SYSTEMS IS A DAMNED GOOD COMPANY. AND I BOUGHT THIS AD TO TELL YOU SO.

James S. Campbell
President and CEO
Fortune Systems Corporation

Please circle Ad No. 33 on inquiry card.

products have succumbed to IBM's iron fist.

The Altos 486, which fills the gap between the 186 and the 586, is a one-to-four-user Xenix-based business system. All the Altos *x*86 family machines are cross-compatible, meaning that software on one will work on another without recompiling; the IBM PC, PCjr, and PC/AT, on the other hand, don't have this capability.

Furthermore, the Altos *x*86 machines all support WorkNet, Altos' proprietary network. Currently, IBM does not support any form of networking on Xenix. If you are running a small business in which you need a small time-sharing system, then you should consider an Altos; if you have an Altos *x*86 machine and need four more terminals, then you should consider the 486.

At $6490, the Altos 486 includes 512K bytes of random-access memory (RAM), 21 Mbytes (formatted) hard disk, five RS-232 terminal/printer ports, one WorkNet local-area network (LAN) port, an Altos III terminal, and Xenix. One thing that sets the 486 apart from the PC/AT is that the 486 is a complete working Xenix (I have a habit of pronouncing this Unix) computer. With the PC/AT, you need to add significant hardware and software to get a working multiuser system. This makes the real (working) price of the 486 lower than that of the PC/AT.

HARDWARE

The most attractive feature of the Altos 486 is that it is an all-in-one computer system. As noted above, a basic 486 Xenix system includes one Altos III terminal. Furthermore, as you grow to the maximum of four users, all you need to add is a terminal for the second, third, and fourth users; no other additional hardware is required. Compare this to the IBM PC/AT, where you have to be an engineer to put together a Xenix system or have to pay top dollar to have someone else put it together. If you can recall the old Osborne I (it only seems long ago), its most attractive feature was the fact that it was an all-in-one computer, including the application software (hint, hint).

The Altos 486's most attractive feature is that it is an all-in-one computer system.

In addition to local-area networks, the 486 also supports 3270 and 3780 bisynchronous. You can do this with an IBM PC/AT (PC Path), but again you have to purchase extra (expensive) hardware. Also, there is WorkNet hardware for the IBM PC, allowing you to share the Altos 486 Xenix disk drives between multiple IBM PCs. This will allow you to get some of the advantages of time-sharing with your current set of IBM PCs.

I can never resist the temptation to open the machine and look at the hardware. Altos made this simple by including an overview of the hardware (this is standard with the machine). The 486 design is very clean; it should have a long mean time between failures and will be easy to fix when it does fail. However, if you are not an engineer, you will have to go to Altos or to an Altos dealer for repairs.

COMPANY OVERVIEW

Company name	Altos Computer Systems	
Public/private	Public Traded over the counter	
Stock symbol	ALTO	
In business since	1977	
Headquarters	Altos Computer Systems 2641 Orchard Parkway San Jose, CA 95134	
CEO	David Jackson	
VP marketing	Philip White	
General sales contact	Ron Conway, Vice President, Sales	
	½ FY '85	FY '84
Net sales	$59,991,000	$102,739,000
Net income	$5,489,000	$9,748,000
Employees	600	550
% of total expense spent on R&D	6%	6%
Units shipped	N/A	30,000
Other major products	Altos 186, 586, 986, 986T, 3086, and Altos III terminal	
Major funding	No long-term debt No venture capital $15 million untapped credit line	

BENCHMARK MEASUREMENTS

Aim Technology Suite II

Altos 486

Arithmetic Instruction Times (microseconds per op)

	short	*long*	*float*	*double*
+ Add	1	4	105	137
* Multiply	6	55	111	184
/ Divide	7	50	122	228

Memory Loop Access Times (microseconds per byte)

	read	*write*	*copy*
Char type	9	9	14
Short type	5	5	7
Long type	3	3	6

Input/Output Rates (bytes/sec)

	read	*write*	*copy*
Disk	24K	18K	9K
Pipe	105K		
TTY 1		749	
TTY 1+2		979	
RAM 1-byte			70K
RAM 4-byte			172K

Array Subscript References (microseconds)

short[ ]	long[ ]
6	8

Function References (microseconds/ref)

0-parameters	1-parameter	2-parameters
funct()	funct(i)	funct(i,i)
29	33	36

Process Forks

(−16K bytes)
20 per second

System Kernel Calls
(calls-per-second and microseconds per call)

`getpid()` calls:	2K calls/sec or	443 microseconds/call
`sbrk(0)` calls:	2K calls/sec or	637 microseconds/call
`create/close` calls:	63 pairs/sec or	15873 microseconds/pair
`umask(0)` calls:	2K calls/sec or	491 microseconds/call

On the other hand, you should be able to get a PC/AT fixed at almost any retail computer store.

While the Altos 486 is the embodiment of completeness, the PC/AT is the picture of expandability. With the PC/AT you have five slots free, whereas the 486 has only one slot. Actually, once you add four serial ports and networking, you are down to three slots. Here's a suggestion: Don't buy the IBM serial cards, they waste slots. This makes the PC/AT much more flexible *and* more expensive. The question is, do you need this flexibility. I would, but a small business would not.

Also, you can expand the PC/AT to have up to 3 Mbytes of internal memory, but the 486 is limited to 1 Mbyte (expansion from 512K bytes was scheduled to become available in June). Although this might seem like a big difference, it really isn't. Experiments with the PC/AT running Xenix show no real improvement in speed above 1 Mbyte of memory. On a four-user business system, you just don't need more than 1 Mbyte (except with large spreadsheets). With all this said, I still think the PC/AT beats the 486 on expandability and flexibility.

FLOATING POINT RESULTS

Comparing the 486's floating point results to those of the PC/AT is very interesting. The PC/AT without floating point hardware is *very* slow, while the 486 is significantly faster. The PC/AT is so slow that I strongly suggest that you put an 80287 chip into your PC/AT. The 486, on the other hand, is fast enough for a business system. The only reasonable explanation for this difference is that nonhardware floating point operations on the PC/AT result in substantial

ALTOS 486 VS. IBM PC/AT

Note: The ratios in the following section reflect the number of times faster that the IBM PC /AT is over the Altos 486 running Xenix 3.0. The numbers in parentheses show where the 486 out performs the PC /AT.

* Numbers not available for IBM PC/AT

Arithmetic Instruction Times

	short	*long*	*float*	*double*
+ Add	1.015	1.131	(10.062)	(9.714)
* Multiply	1.422	1.231	(9.729)	(7.390)
/ Divide	1.501	1.194	(8.708)	(5.707)

Arithmetic Instruction Times (with 80287 floating point chip in the IBM PC/AT)

	short	*long*	*float*	*double*	
+ Add	**	***	2.865	3.596	
* Multiply	**	**	2.778	4.188	Avg.
/ Divide	**	**	2.065	3.686	3.196

Memory Loop Access Times

	read	*write*	*copy*	
Char type	1.394	1.665	2.796	
Short type	1.381	1.598	2.598	Avg.
Long type	1.295	1.631	3.080	1.938

Input/Output Rates

	read	*write*	*copy*
Disk	(1.121)	1.700	2.354
Pipe			1.702
TTY 1		*	
TTY 1+2		*	
RAM 1-byte		2.796	
RAM 4-byte		3.080	

Array Subscript References

short[]	long[]
1.811	(1.623)

Function References

0-parameters funct()	1-parameter funct(i)	2-parameters funct(i,i)
1.348	1.368	1.458

Continued

kernel (context switching) overhead with every floating point operation.

Because the Altos 486 is a business system, Altos did not put in a socket for a floating point hardware

The PC/AT without floating point hardware is very slow, while the 486 is much faster.

chip. In the benchmarks that accompany this article you'll see a comparison of the PC/AT using the 80287 floating point chip and the Altos without it. As you can see, the 80287 chip makes a big difference in floating point speed! If you are doing a large amount of floating point operations, then consider a different machine altogether (both the PC/AT and the 486 are weak here).

I was surprised to see how slow the Altos 486 is when compared to the PC/AT (again, see the benchmarks). The PC/AT uses the 80286 processor, running at 6 MHz, while the Altos 80186 runs at 8 MHz. Based upon the processor speeds, you would think the 486 would be faster. But the 80286 chip is such an improvement over the 8086 and 80186 chips that the clock speed is misleading.

Very telling are the memory access times, which are a basic measurement of the performance in a business system. The PC/AT is almost twice as fast as the Altos 486, comparing only memory speeds. Disk input/output (I/O) is also another important measurement. The PC/AT is 1.2 times faster than the 486 (when comparing only disk speed). This number is even more surprising when you consider that both machines use the same half-height, hard-disk technology.

ALTOS 486 VS. IBM PC/AT Continued

Process Forks

(49K bytes)
1.350

System Kernel Calls

`getpid()` calls:	1.646
`sbrk(0)` calls:	(1.165)
`create/close` calls:	1.079
`umask(0)` calls:	1.216

ALTOS 486 VS. IBM PC/AT

	IBM PC/AT	Altos 486
CPU	Intel 80286	Intel 80186
Clock speed	6MHz	8MHz
Floating point hardware	Not available	Intel 80287
Serial I/O controller	—	Z80
Internal memory—standard	512K bytes	512K bytes
—maximum	3 Mbytes	1 Mbyte
Hard disk—standard (formatted)	21 Mbytes	21 Mbytes
—maximum	42 Mbytes	42 Mbytes
Floppy disk	1.2 Mbytes	720K bytes
Size	21″ × 17″ × 6″	17″ × 15″ × .5″
Weight	37 pounds	20 pounds
Performance	excellent	good
Projected reliability	excellent	excellent
Floating without hardware	very poor	good
Floating with hardware	excellent	N/A
Expansion	good	very poor
Network port included	no	yes
Network/Xenix compatibility	no	yes
Expansion bus	published standard	proprietary
Expansion slots available	5	1
COST		
Base Price	$5795	$6490
First terminal and interface	$525	Included
Expand to 5 serial ports	$600	Included
Xenix	$395	Included
Total	$7315	$6490

Networking hardware is included as standard with the 486, and that's a *big* advantage. On the PC/AT, the networking hardware costs from $300 to $1000 per machine. Also, not only is the networking software not integrated into the system on the PC/AT, but IBM's Xenix 3.0 does not directly support networking.

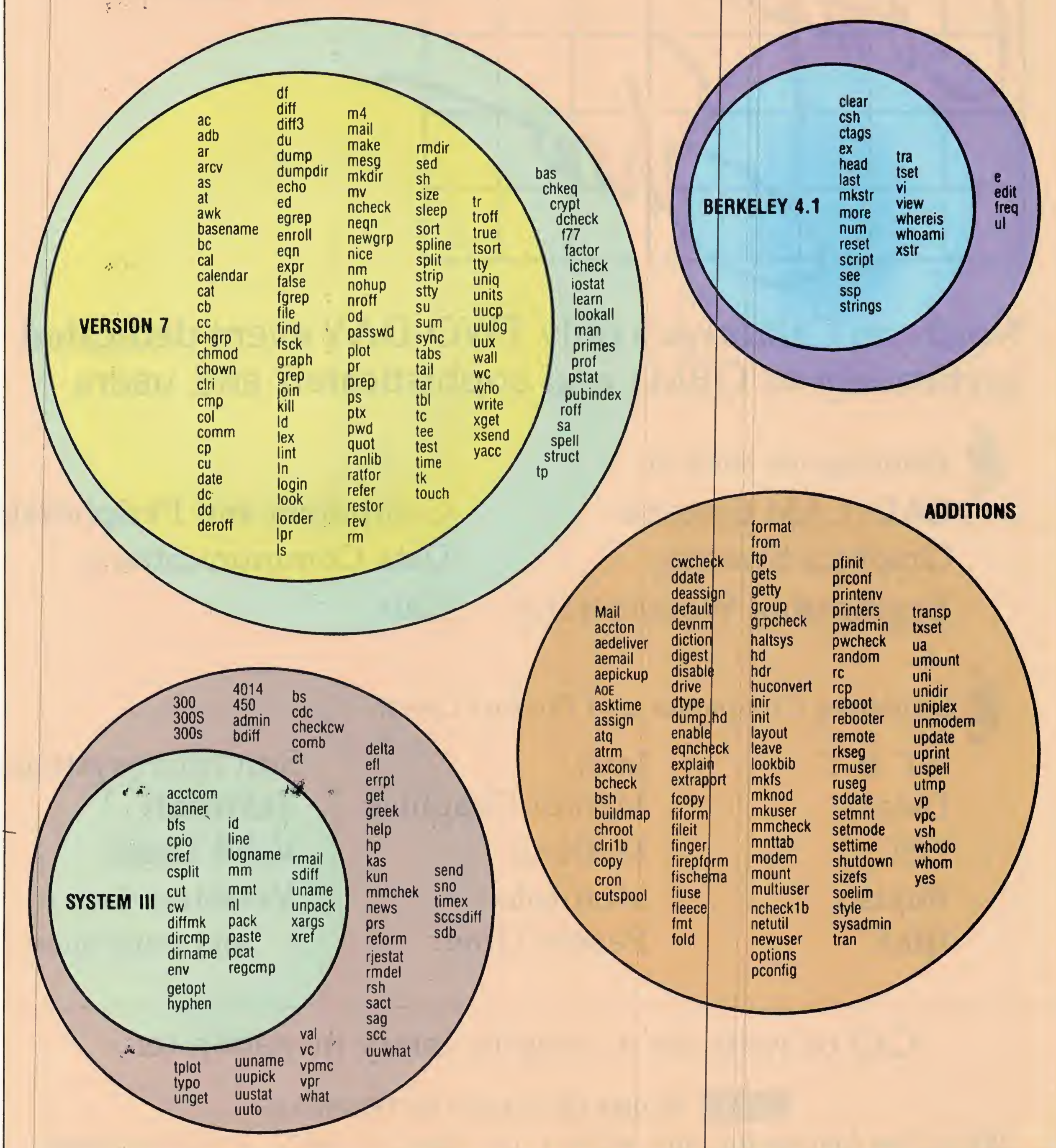
RUN-TIME XENIX 3.0

Nondevelopment "run-time" Xenix 3.0 comes standard on the Altos 486. The run-time Xenix includes the Xenix kernel and the basic utilities. For an additional $1000, you can purchase the extended utilities package, which includes `csh`, `vi`, C compiler and optimizer, Source Code Control System, `cu`, and `uucp`. As you can see from the command completeness test, the extended utilities include a very large set of commands. I couldn't run the command completeness test on the "run-time" Xenix because there were not enough Unix utilities for them to run on. For the `uucp` buffs out there, I can report that the Xenix 3.0 `uucp` and `cu` utilities work. Other than networking and the limit on the number of users supported on the PC/AT, Xenix 3.0 is nearly identical on the PC/AT and the Altos 486.

An attractive feature of the 486's version of Xenix is the integration of WorkNet into the disk I/O routines.

Let me take a moment to let off a little steam on the subject of Unix versus Xenix. Xenix *is* a Unix system! The big difference is that the

ALTOS 486 COMMAND COMPLETENESS



HARDWARE/SOFTWARE OVERVIEW

Model	Altos 486
Price (of review machine)	$6490
Configuration (of review machine)	512K bytes RAM 1-Mbyte floppy 21-Mbyte (formatted) hard disk, 1 LAN and 5 asynchronous serial ports, external disk port
First delivered	November 1984
Related models	186, 586, and 986
PROCESSOR	
CPU	80186
Cycle time	8MHz
Bus	Proprietary
Cache size	None
Min. memory	512K byte
Max. memory	512K byte
Floating point	Software
STORAGE MEMORY	(range of available hardware)
Floppy	1 Mbyte, 1¼ inch, half-height
Winchester	21 Mbyte (20-Mbyte expansion was scheduled to become available June 1985
Backup	Floppy
I/O processors	Terminal I/O (Z80A)
OTHER HARDWARE	(expansion)
Serial ports	5 standard, no expansion
LAN medium	RS-422/WorkNet
Protocols	3270, 3780 Bi-sync
Standard software and development (denote development software)	
Unix version	Xenix 3.0
Shells	Bourne and AOE, `csh` available
Libraries	Standard Xenix 3.0
Utilities	Xenix extended utilities
Languages	C, COBOL, FORTRAN, Pascal, BASIC are available

Unix system is marketed by AT&T and Xenix is marketed by Microsoft. This is not to say that there are no differences; in fact, Xenix has many advantages over a straight System V Unix, but all this talk about Xenix being easier to use than Unix is a lot of hot air. I have used Microsoft's visual shell (`*vsh`), and I did not like it; I find most of the other visual shell programs easier to use. If you want to try a truly nice visual shell, look at HP's PAM. There is no essential difference between Xenix and Unix. I welcome the letters pointing out the differences because this will only prove my point. Now, back to the matter at hand.

A very attractive feature of the Altos 486's version of Xenix is the integration of WorkNet into the disk I/O routines. In a networked Altos x86 environment, you can read and write files on another computer directly. For example, to display a file stored on another machine you would type `more @mach2/usr/john/his/file`.

This would cause a file named `/usr/john/his/file` to be read from a remote machine and displayed on the local machine. Other

Another feature of WorkNet is PC Path, which allows the IBM PC or compatible to share files on the Altos WorkNet network.

than the fact that the file name is lengthened to indicate a remote machine, there is no difference between executing a command on the remote machine or on the local machine.

Xenix 3.0 also supports the Berkeley 4.2 feature of symbolic links. Symbolic links and WorkNet combine to give you much of the functionality of a network file system. For example, if your accounting system looks in a local directory called `/usr/accounting/data` for its data—but you want to use the same files on two different machines—you could use symbolic links to create a `/usr/accounting/data` directory on the local machine that points at a

ALTOS SOFTWARE AVAILABILITY PROGRAM

APPLICATIONS GENERATORS	
Software Express Inc.	APPGEN Development System
American Information Systems	RediWrite
GRAPHICS	
High Tech Marketing	High Tech Business Graphics
Altos	Altograf
VERTICAL APPLICATION	
Orion Microsystems	GLOWS (client accounting)
ProMation	The Contractor's Edge
Health America	Patient Management System (dental and medical)
Advanced Business System	EasyPost System (insurance)
Dickens Data Systems	ALOT (Automated Law Office Tool)
NMI Inc.	Magazine Circulation Fulfillment
MicroManufacturing	Manufacturing Control System-MCS-3
MCBA	Manufacturing And Distribution System
The Computer Center	Club Manager
Clinical Data Design	MDX System (medical)
The Computer Center	MUNIS (municipal goverment)
NMI Inc.	Personnel Searcher
PAS Inc.	Rx-II (pharmacy)
Spectra Data Systems	Property Management System
Romax Computers	Uni-Max Retailer
NMI Inc.	Customer Profile
Orion Microsystems	GLOWS Pratice Management
Software Solutions	FACTS (wholesale distribution)
Tracline	Wholesale and Retail Accounting
SRD Inc.	ADAM Auto Repair Management
TCF Software	Client Ledger
NFP Management Systems	Non-Profit Fund Accounting
Holm-Dietz	Farm Management System
WORD PROCESSING	
Altos (Uniplex)	Executive Word Processor
Horizon	Horizon Word Processor
SofTest Inc.	Lex-86 Word Processor System
OFFICE AUTOMATION	
Altos	ABS-86 Business Solution
Altos	AOE (Altos Office Executive)
SPREADSHEET	
Altos (MultiPlan)	Altos Financial Planner
DATA COMMUNICATIONS	
SofTest Inc.	SofGram

Continued

`/usr/accounting/data` directory and the shared machine.

In this way all file references would automatically reference the single shared files. This is not as clean or as flexible as a true networked file system, but it should work for many applications. A single shared printer could be implemented by making a symbolic link for all machines to a shared `/usr/spool/lpd` directory.

Another feature of WorkNet is PC Path, which allows the IBM PC or compatible to share files on the Altos WorkNet network. This is done by creating a special PC file under Xenix that contains the image of an IBM PC floppy or hard disk. This will allow people who have PCs to share the 486 hard disk while still working with PC-DOS. Altos also provides a free Altos III terminal emulation program for the IBM PC.

When you compare the version of Xenix 3.0 on the Altos 486 with that on the PC/AT, the Altos 486 is a winner, but not a big winner. The 486 has three advantages. First, you can put four users on the 486, versus only three on the PC/AT. Second, the networking is integrated into Xenix on the 486, but it is ignored on the PC/AT. These points will be very important to people who need them. And third, PC Path will sell a number of systems because it creates a bridge between PC-DOS and Unix.

OTHER SOFTWARE

We are still hearing that the Unix system does not have any applications software, but when you consider the software available for the Altos *x*86 machines, you should forget this nonsense It's all disinformation foisted on the world by PC propagandists. The list of software

AVAILABILITY CONTINUED

DATA MANAGEMENT	
RDS Inc.	Informix C-ISAM
SMC Software Systems	Thoroughbred IDOL
Unify Corporation	Unify
GENERAL BUSINESS ACCOUNTING	
Open Systems	Accountant
Americal Business Systems	BACS Accounting System
RealWorld Software	RealWorld Accounting
SMC Software Systems	Thoroughbred Accounting System
Software Express Inc.	Financial Series
NMI Inc.	Complete Accounting
OPERATING SYSTEM	
Altos/Microsoft	Xenix Run Time
Altos/Microsoft	Xenix Extended Utilities
LANGUAGES	
Ryan-McFarland	RM/COBOL
Micro Focus	LEVEL II CIS COBOL
Digital Research	CBASIC-16
SMC Software Systems	Business Basic III
Microsoft	MBASIC
Microsoft	MS Pascal-86
Microsoft	MS FORTRAN-86
Microsoft	MS COBOL-86
Omtool	SOFTBOL
COMMUNICATIONS	
Altos	WorkNet
Touchstone Software	PCworks
Touchstone Software	MIMIX
Altos	PC Path
Altos	2780/3780 IBM bisync
Altos	3270 IBM bisync
Altos	3270 SNA/SDLC emulator
Altos	X.25 packet switching

packages available for the Altos 486 includes sixteen vertical applications, nine compilers/interpreters, three database management packages, three word processors, and more. This should allow end-users to choose the software package that most closely fits their needs.

Included with the machine I reviewed was the integrated Altos Office Executive (AOE) office automation system. AOE combines a data manager, spreadsheet, word processor, and electronic mail program into one semi-integrated package. With AOE, you can create a mass mailing using data from the database, and you can include data from the database into a spreadsheet, etc.

Altos made AOE by taking three existing packages and writing interfaces between them. The spreadsheet is Multiplan from Microsoft; the data manager is File-it from Relational Database Systems; and the word processor is Uniplex from Redwood Software of the UK (known as Uniplex Integration System here in the U.S.). Unlike other similar integration efforts, Altos' interface between these different packages is fairly clean. One sign of just how clean is the fact that you can "mail-merge" a form letter with a File-it database.

I have worked with Multiplan and File-it before, but the word processor was new to me. I liked the combination of function keys and menus, which made Uniplex very easy to use and extremely powerful. I found myself wishing that it worked on my day-to-day computer.

There are some weaknesses in Uniplex, to be sure, but these are minor and repairable. One thing I found surprising is that you could not nest special printing features (you can't get boldface underline, for example). Also, the spelling checker has very few options. When the software finds a misspelling, it brings up only two choices: You can accept the spelling, or you can change the spelling. With more advanced spelling checkers, you have many choices, including having the program give you a menu of possible correct spellings.

RDS' File-it is not a database manager simply because you cannot create cross database joins, but it is one of the easiest to use and one of the simplest data management tools. What it lacks in power, it more than makes up for in simplicity and ease of use. I have given this tool to technophobes and have found them using it 30 days later to keep track of all sorts of things. More powerful database management systems are often intimidating.

I don't have too many good things to say about Multiplan, other

ALTOS ADDS HIGH-END UNIT TO x86 LINE

As we were going to press, Altos Computer Systems Inc. expanded its x86 line of Unix system supermicros with the Altos 3068, which accommodates up to 30 users and a wide range of system options. The 3068, designed for original-equipment manufacturers (OEMs) and major accounts, incorporates Unix System V and the 32-bit Motorola 68020 microprocessor.

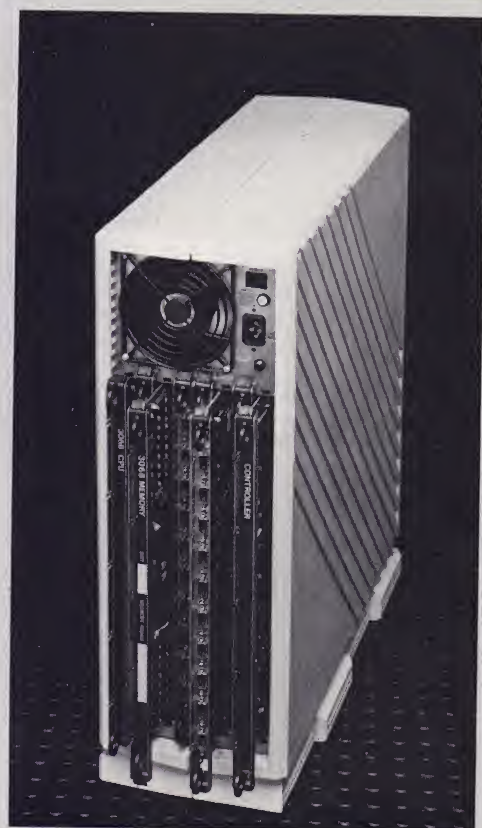
The 3068 is designed to run as a stand-alone, general-purpose computer or as a node in a distributed network. The 3068 can be networked with all other Altos multiuser systems via the Altos WorkNet local-area network, and with mainframe computers via 3270 Bysync, SNA, X.25, and 3780 communications options. With Altos PC Path attached to WorkNet, the 3068 can act as a file server and communications gateway for personal computer users.

The Altos 3068 supermicrocomputer's base configuration includes 1 Mbyte of random-access memory (RAM), a 20-Mbyte hard disk, and a 1.2-Mbyte floppy-disk drive supporting up to 10 users. Users can expand the main memory to 16 Mbytes in 1-, 2-, or 4-Mbyte increments. Hard disks can be upgraded to 240 Mbytes (unformatted) in 20-, 60-, or 80-Mbyte increments. A streaming magnetic tape unit, with up to 60 Mbytes of backup storage, is also available with the 3068.

The modular systems design includes eight board slots, with four available for custom configuration. Additional boards can be added or replaced, at user discretion, depending on the need for additional users, memory ports, or proprietary boards.

The Unix System V operating system available for the 3068 supports demand-paged virtual memory with 1K-byte page size, allowing programs that exceed physical memory to be run on the system.

The 3068 is scheduled to become available in July on a worldwide basis. Base configuration price is under $7000 in OEM quantities.



than that it runs on a wide range of computers and that it is a spreadsheet. What I want to know is, when will some of the better microcomputer spreadsheets move to Xenix/Unix.

CONCLUSION

I really don't know how the Altos 486 will do against IBM's PC/AT; it's too early to predict, for one thing. Currently, Altos is selling the 486 quite well against the PC/AT, but there is a maxim in the computer world: To compete with IBM, you have to be half the price, be twice as good, or sell something else. It is clear from my review that the Altos 486 is neither twice as good as the PC/AT nor half the price. But that does not mean that it will fail.

Perhaps Altos is selling an integrated, low-hassle, low-maintenance, Xenix business computer. I am not sure what IBM is selling with the PC/AT. But with all the options and add-on hardware and confusion about PC-DOS and Xenix, the PC/AT is not a low-hassle, low-maintenance, low-priced multiuser microcomputer.

I do have a simple suggestion for Altos: Why not include with the 486 all the basic business software—a spreadsheet, word processor, data manager, and accounting system. This would make the Altos 486 the Osborne-1 of the multiuser microcomputer world. □

Bruce Mackinlay, UNIX/WORLD's gonzo reviewer and editor-at-large for reviews, is vice president of research and development at WMZ/Novatech, Concord, Calif. His last work for this magazine, a review of the Pyramid 90x supermini, appeared in the May 1985 issue.

THE AT&T UNIX SYSTEM help FACILITY

BY THOMAS BUTLER AND LISA KENNEDY

The new Unix* system `help` facility provides quick, easy access to information about the Unix system. The facility consists of five Unix commands—`help`, `starter`, `glossary`, `usage`, and `locate`—each of which is the name of a separate module of the facility.

Each module contains a different kind of information:

(one) `help` is the top-level menu interface to the entire facility. Through it, all other modules can be entered.

(two) `starter` contains information about the local Unix system and general information for beginning system users.

(three) `glossary` contains definitions of terms and symbols important for the Unix system.

(four) `usage` contains detailed information about specific Unix system commands.

(five) `locate` is a command "thesaurus." It identifies Unix system commands related to keywords submitted by the user.

The facility is quite flexible, and most information can be accessed in several different ways, depending on the proficiency and preference of the user. All information in the facility can be retrieved by typing "help" at shell command level and then entering menu choices until the desired information is reached.

Users can also use individual modules of the facility by typing the name of the module at shell command level, thereby avoiding the use of some menus. The `glossary`, `locate`, and `usage` commands also accept command line arguments so that users can enter all required information on the command line and directly retrieve the information they desire.

The facility also contains a mechanism to collect and store data about its use as well as a set of software tools for command developers and system administrators who wish to modify the `help` database.

Much of the data contained in the facility was collected directly by questionnaires and surveys sent to Unix system users. The user interface of the facility was designed with special care to be terse enough for experienced users but simple enough for beginners.

The first release of the Unix system `help` facility is an initial step toward providing a comprehensive, easy-to-use interface between users and all system documentation. Further expansion and refinements of the facility are now under way.

The wide proliferation of the Unix operating system is a strong indicator of its power and ease of use for programmers and other computer professionals. It is not, however, always the easiest system for beginners to use. The shortcomings of the Unix system have been widely discussed and need no further elaboration here. One early, and obvious, step we can take toward making the Unix system easier to learn is to give its users a helpful on-line assistance facility. This article describes the initial `help` facility we have designed to fill this need.

The first section of the article briefly describes a prototype facility that we built and the reasons we built it. The next section describes the initial set of requirements adopted for the first version of the facility. This is followed by a discussion of data gathering to collect information for inclusion in the facility and, finally, by a discussion of the human interface we designed.

THE PROTOTYPE

Our first decision was to build a prototype of the facility and to throw it away at the end of the proposed seven-month development period. While this may seem an odd way to begin a development project with short deadlines, the idea did not originate with us (Brooks, 1975), and the prototype approach has several distinct advantages. The most important of these are the following: the prototype could be implemented quickly; the code need not be optimized until we are sure we have designed the system we really want; and the prototype could be tested with real users and revised before it is actually released as a Unix system product.

The prototype we designed and implemented was quite small, but it allowed us to do the following: evaluate the user interface to the facility; evaluate our data-collection procedures; evaluate the data collected to populate the facility; and evaluate the development method.

REQUIREMENTS AND DESIGN

The Unix system `help` facility is an interactive utility available at shell command level. It allows users to retrieve a variety of information about the Unix system while they are working on-line and to retrieve that information easily and conveniently. The facility consists of the top-level interface `help` and four major modules—`starter`, `glossary`, `usage`, and `locate`.

The four modules provide the following information, respectively: *(one)* general information about the local Unix system; *(two)* a glossary of technical terms often used in relation to the Unix system; *(three)* information about the usage of specific commands, including examples; and *(four)* a means of identifying unknown Unix system commands using keywords related to the desired command functions.

It is easiest to think of the `help` facility as a tree structure, with the top-level `help` menu interface at the root of the tree and the other four modules subordinate to it. Individually, the four modules are all quite different in their structure, and each of them will be described separately.

The `starter` module can be thought of as a sort of formatted bulletin board. It contains general information for users, much of which is specific to the local system where the facility is installed. In Release 1 of `help`, `starter` includes a list of the Unix system commands and terms we believe new users should learn first; a list of education centers offering Unix system training; a list of important Unix system documents for beginners; a list of on-line teaching aids and tutorials available for the system; a set of local environment information, including the system administrator's name and phone number, the name of the system, and the type of processor running the system.

Users retrieve information from these `starter` components by selecting the appropriate category on a menu. The user interface for the facility will be described in greater detail later in the article.

The `glossary` module provides definitions for many common Unix system terms and symbols. Many of the definitions contain examples to illustrate the concepts being defined and references to related Unix system commands. The first screen of this module includes a complete list of the terms and symbols supported. Like `starter`, the `glossary` module is mainly menu driven.

The `usage` module gives users specific information about individual Unix system commands. One part of `usage` provides a syntax summary and a short paragraph describing the action and uses of the command. Much like the `synopsis` section of the *Unix System V User Reference Manual,* the syntax summary includes a list of allowable options for the command and a command line showing the relative positions of the command name, its options, and its arguments.

The two other parts of `usage` include a list of all options for the command and an explanation of each, and examples of typical command lines using the command and an explanation of what each command line does. Users move through these different sections of `usage` by making menu selections.

The final module, `locate`, accepts from the user a set of keywords related to the work the user wants to do. It then identifies a set of Unix system commands likely to be helpful in doing this work. For example, if users are seeking a command to print the contents of a file, they might enter the keywords "file," "print," "contents," and "output," or some subset of this list.

The keywords themselves are entirely free form; users can submit any keyword they think is related to the work they want to do. The `locate` module keeps a quite large set of keywords on file for each of the Unix system commands it supports. After accepting the keywords, `locate` checks the keyword list for each command and returns the names of all commands having a keyword on file that matches any of those entered by the user.

After identifying a command, users can directly access the detailed command information in the `usage` module. If they find that the command they selected is not the one they need, they can return directly to the list produced by `locate` and make another selection. As with the other modules, choices are presented and input is solicited using menus.

In addition to these four modules, `help` contains a logging mechanism to collect and store data about its use. It also has a set of software tools for use by command developers and system administrators who want to add information about their local system, add data for local commands, or otherwise modify the `help` database.

These additional features are available through a separate command called `helpadm`, which is provided with the `help` facility. Using this command, system administrators can turn on or turn off the logging of usage data on the facility; and both developers and administrators can add, modify, or delete `starter` information, `glossary` definitions, the descriptions, option descriptions, and examples in `usage`, and the keyword information used by `locate` for each command. It should be possible, for example, to use `helpadm` to insert foreign language text and data into the `help` database.

DATA GATHERING

Some of the data contained in the facility was obtained through consulting standard reference documents for the system. For example, this is the way we obtained all the command syntax summaries in the `usage` module. Also obtained this way were information about Unix system training locations, Unix system beginner documentation, the names of currently available on-line teaching aids, and the definitions of terms and symbols.

All the remaining information in the system was obtained primarily through questionnaires and surveys administered to a controlled mixture of experienced and inexperienced Unix system users within AT&T Bell Laboratories.

After a substantial amount of information was collected with the questionnaires and surveys, we collated the results, checked the data for correctness, and edited all the information for consistency. The important difference between our procedure and the usual one in a design effort of this type is that, rather than relying only on the intuitions and judgments of those involved with designing and developing the facility, the intuitions and judgments of a large sample of both experienced and inexperienced users were collected and considered when we decided what to include and what to exclude from the system.

The importance of the data obtained through surveying real Unix system users cannot be overemphasized. The contents of all of the following were primarily the result of input from Unix system users: the command descriptions in the `usage` module; the examples of typical usage for each command in the `usage` module; the command keyword list placed on file for use by the `locate` module; and the specific set of Unix system commands included in the suggested "minimal" set given in the `starter` module.

In addition, all our decisions about precisely what sorts of information to include in the `starter` module were strongly influenced by user input from surveys. This approach is quite different from the usual strategy of relying on designers' intuitions for the first version of a system and then testing the users' reactions to it later. So far, we have found it to be a pleasant change.

THE HUMAN INTERFACE

Designing the user interface for something like a help facility is frustrating. On one hand, the facility must be simple enough for a beginning user; on the other hand, the system must be terse enough that expert users won't be irritated by verbose guidance and instruction. We have tried to consider both populations of users in the present design and have produced a flexible, vari-

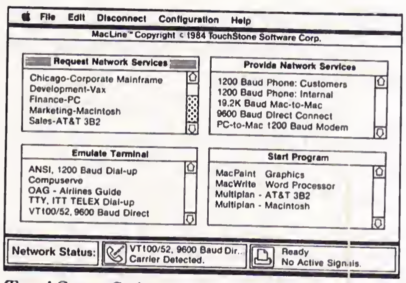
Continued on page 102

APPLICATIONS SOFTWARE

TOUCHSTONE CONNECTS UNIX SYSTEMS, MACs

TouchStone Software Corp. has released MacLine, a Macintosh networking software package. With the MacLine terminal emulator, users can connect Apple Macintosh computers to a variety of other computers and information services.

When MacLine is used with the other Connectables network products, PCworks and UniHost, the Macintosh can transfer all types of files to and from Unix systems or PCs, exchange mail with other system users, and access a remote printer or disk.



TouchStone Software Corp.'s MacLine networking software package

MacLine software is priced at $145. UniHost for Unix systems costs $295, and PCworks costs $195.

For more information, contact TouchStone Software Corp., 909 Electric Ave., Suite 207, Seal Beach, CA 90740; 213/598-7746.

Please circle Reader Service Number 160.

WINDOW SYSTEM INTRODUCED FOR XENIX

Viewnix windowing interface for Unix system-based microcomputers is now available from Five Paces Software Inc. The product is being shipped for the IBM PC/AT and XT running the Xenix operating system and will soon be offered on other Unix systems that employ memory-mapped displays, the company said.

Viewnix allows the user to configure up to 10 windows on the screen of the PC/AT, each of which can contain any standard Xenix application. The windows can be expanded, contracted, or moved about the screen to suit the user's specific needs. In addition, a cut-and-paste capability is provided to facilitate the integration of data between different applications.

Viewnix offers a programming interface that will allow applications to take control of the windows in which they are run. The system allows applications to write directly to the window memory buffers so that programs running on the memory-mapped console will not have to employ the slow, character-at-a-time video output associated with Unix systems.

Viewnix retails for $249, with discounts available to volume purchasers; UniMove and UniLink are each priced at $149.

For more information, contact Five Paces Software Inc., 9635 Wendell Rd., Dallas, TX 75243; 214/340-4933.

Please circle Reader Service Number 161.

NEW ACCOUNTING SOFTWARE FROM AMI

Accountants Microsystems Inc. (AMI) has released Datawrite Corporate General Ledger. The new software operates under Xenix or MS-DOS on most business microcomputers and can be tailored for virtually any industry or firm.

The software's flexibility derives from the Datawrite proprietary Matrix Report Writer, essentially two report writers in one. The first allows the user to specify any account in any order and to change the account description without affecting the basic chart of accounts. The second report writer lets the user set columns, headings, subtotals, and totals in any format for management and financial reports. Matrix Report Writer is menu driven rather than chart driven.

AMI accounting software

Corporate General Ledger retails for $2995 and is available from AMI's direct sales force and its dealers and agents.

For more information, contact Accountants Microsystems Inc., 3633 136th Place S.E., Bellevue, WA 98006; 206/643-2050.

Please circle Reader Service Number 162.

NEW VUE PORTS FROM NIS

National Information Systems has announced that VUE, a project management system available for Unix system processors, is now available on a variety of computers, including Convergent Technology's MiniFrame, Gould, and NBI, all running under the Unix system.

VUE allows either I-J or precedence notation and uses the critical path method of scheduling. Fourteen reports are standard, including printer graphics versions of bar charts and network diagrams. Functions include sorting on the desired field, specifying a time window, picking selected material for reporting, and summarizing for less detailed reporting.

VUE now runs on the AT&T 3B series, Convergent Technology MegaFrame and MiniFrame, DEC 10/20, VAX, and PDP-11, Fortune, Gould, Honeywell DPS-6 and -8, HP 3000, IBM mainframe VM/CMS, NBI, Perkin Elmer, and Zilog S8000. The user interface remains the same across the entire selection of computers.

Price ranges from $2995 for supermicro versions to $20,000 for large mainframe versions.

For more information, contact National Information Systems, 20370 Town Center Lane, Suite 130, Cupertino, CA 95014; 408/257-7700.

Please circle Reader Service Number 163.

BIG 8 ACCOUNTING RUNS UNDER UNIX

A multiuser business accounting software package designed to run under the Unix system, MS-DOS, and MS-DOS networks is now available from Modern Technologies International Inc. (ModTech).

The Big 8 Accounting Series from ModTech eliminates the need to purchase a single-user product for current needs and then replace it with a multiuser product as the company grows.

The series offers five integrated, menu-driven accounting packages designed to satisfy business accounting needs, including inventory control, general ledger, accounts receivable, accounts payable, and payroll.

The Big 8 offers this multiuser capability on Unix Systems III and 5, Version 7, Xenix, Venix, and PC-IX.

Retail price per module is $695 for DOS and $895 for the Unix system. ModTech is distributing the Big 8 Accounting Series through value-added dealers, microcomputer retail chains, and OEMs.

The Big 8 from Modern Technologies

For more information, contact ModTech Inc., 656 Bair Island Rd., Suite 302, Redwood City, CA 94063; 415/367-6855.

Please circle Reader Service Number 164.

CREATIVE APPLICATIONS' ACCOUNTING SOFTWARE

Creative APLications Inc. has expanded its oil and gas accounting software to run under the Unix operating system.

This means that smaller independents can start out with a low-cost but single-user computer and upgrade to a multiuser system later without having to start over with new software.

Energy Plus supports the IBM PC/AT (using Xenix) and 9000, the Hewlett-Packard IPC and 9000, Fortune, Wicat, Cadmus, Callen, and others. In the future, the software will also support other supermicro and minicomputer systems, including hardware from AT&T, Data General, and Digital Equipment Corp.

The system includes features to help first-time computer users, such as complete on-line documentation and "Help" keys that enable users to get on-line assistance while in the middle of any program.

For more information, contact Creative APLications Inc., 2525 S. Wadsworth Blvd., Suite 206, Lakewood, CO 80227; 303/985-4465.

Please circle Reader Service Number 165.

HARDWARE AND SYSTEM

DEC ENHANCES ULTRIX-32 AND PRO/VENIX

Digital Equipment Corp. has announced a series of enhancements for two Unix-based operating systems, Ultrix-32 and Pro/Venix.

Version 1.1 of Ultrix-32 provides increased support for the VAX family of computers and broader compatibility with System V. The Ultrix-32 operating system is an interactive, virtual-memory, timesharing Unix system that runs on members of Digital's VAX family of computers.

Version 2.0 of Pro/Venix allows applications designed to run under System V to be ported to Pro/Venix with little or no modification. Ethernet communications and support for the Laboratory Interface Module on the Pro under Pro/Venix were also announced.

Prices on the Ultrix-32 Version 1.1 vary according to the hardware, number of users, package, and services chosen.

Pro/Venix Version 2.0 is available in the base system package, priced at $495, which includes edi-

TEXT EDITING

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

NEW RELEASE

UNIPRESS EMACS™

VERSION 2

Subject: Multi-window, full screen editor.

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

New Features:

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

Features:

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

Price:

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

**Call for terms*

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

OEM terms available. Mastercard/Visa accepted.

Please circle Ad No. 27 on inquiry card.

UniPress Software

Your Leading Source for UNIX Software.

tors, serial communications, text processing, a BASIC interpreter, and graphics utilities; and the Software Development Package, priced at $600, which includes the C, FORTRAN, and Pascal compilers; SCCS; and the more popular Unix system support interested in the software development package.

For more information, contact Digital Equipment Corp., Maynard, Mass.

Please circle Reader Service Number 166.

SERIES 8000 FAMILY OF COMPUTER SYSTEMS FROM SMS

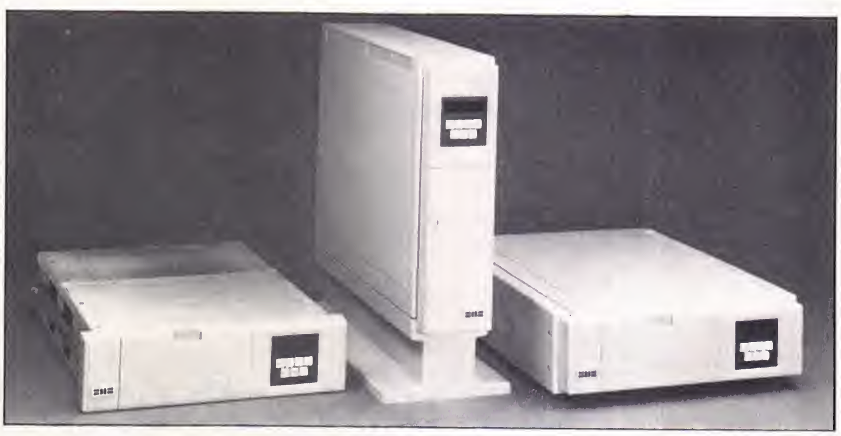
Scientific Micro Systems' SMS 8000 Model 40 is a Multibus-compatible, Winchester-based microcomputer system available in a wide variety of configurations. All versions include the system enclosure, a choice of fixed and removable peripherals, 8086/80286 processors, and up to 6 Mbytes of main memory.

The SMS 8000 will run all system and application software developed for Intel's RMX 86 or Xenix operating systems, as well as CP/M-86. A removable 10-Mbyte Winchester cartridge drive may also be included as either a system disk or backup device. In addition, a high-speed streaming tape drive may be packaged as a backup device.

The SMS 8000 Model 40 offers single or dual 5¼-inch Winchester drives with capacities of 12 to 280 Mbytes. List price for the SMS 8000 Model 40 starts at $5900 in quantity, complete with foundation module and one serial port, 12-Mbyte Winchester, 5¼-inch 7000K-byte floppy, 8086 CPU, and 512K bytes of main memory.

For more information, contact SMS, 339 N. Bernardo Ave., Mountain View, CA 94043; 415/964-5700.

Please circle Reader Service Number 167.



SMS' new Multibus-compatible microcomputer

SYSTEMS SOFTWARE

UNIX SYSTEM V FOR INTEL's 80286

AT&T's Unix System V, Release 2, operating system is now available for use with Intel's 80286 microprocessor.

The 80286 microprocessor's on-chip memory management and protection eliminate complex reprogramming of the Unix system kernel when Unix System V is transported from one 80286-based computer system to another.

Many independent software vendors support the 80286 microprocessor with enhanced versions of System V/286. OEM's can choose among the UniPlus operating system by UniSoft Systems, the IN/ix by Interactive Systems, and System V/286, Release 2, by Microport Systems (a spinoff from Digital Research Inc.).

AT&T will market the source-code product to resellers building hardware and software products around System V/286. It is priced at $43,000 for the initial host 286 system and at $16,000 for each additional host.

For more information on the System V/286, contact Intel Corp., Literature Dept. W-193, 3065 Bowers Ave., Santa Clara, CA 95051; 408/987-5084.

Please circle Reader Service Number 168.

MODULA-2 FOR THE UNIX SYSTEM

A compiler and run-time library for the Modula-2 language has been developed for Unix systems based on the Motorola 68000 series of microprocessors. With Modula-2/68, it is now possible to use the Modula-2 programming language for all types of Unix programming. It can be used by itself or with C and Pascal to implement complex system programming applications.

With Modula-2/68, program modules can be compiled separately. An executable process can then be built by linking with other previously compiled program modules or by linking with library procedures written in C or Pascal. Modula-2 procedures can be included in the standard object libraries for use by programs written in C or Pascal. For M68000 application the Modula-2 programs can be output in the standard Motorola S-record format.

Modula-2/68 is now available for UNOS and 4.2BSD Unix systems. Manufacturers of Unix systems based on the M68000 are being encouraged to add the Modula-2/68 implementation of Modula-2 to their line of supported programming languages. Modula-2/28 is also available as a cross-compiler for use with the VAX/VMS operating system. List price for UNOS version is $495.

For more information, contact Djavaheri Bros., 697 Saturn Court, Foster City, CA 94404; 415/661-6868.

Please circle Reader Service Number 169.

C68 FROM INFORMATION PROCESSING TECHNIQUES CORP.

C68 from Information Processing Techniques Corp. (IPT) is a development tool for software designers targeting development to Motorola 68000-based machines. Four elements comprise a C cross-compiler: a cross-compiler that generates 68000 code from C source, a cross-assembler to produce relocatable binary files that can be linked together with the linker, a cross-linker to link relocatable binaries that may be run on any 68000-based machine, and a librarian that produces libraries for use with the linker.

C68 also allows programmers to develop software for Motorola 68000-based machines on a variety of other systems.

C68 is available for Data General (DG) computers running RDOS, AOS, ECLIX, and AOS/VS, as well as on DEC computers running VMS or the Unix system. There is a one-time license fee of $7000.

Other C tools from IPT include a C language debugger, a C compiler for DG systems, and a Unix operating system for 16-bit DG systems.

For more information, contact Information Processing Techniques Corp., 1096 East Meadow Circle, Palo Alto, CA 94304; 415/494-7500.

Please circle Reader Service Number 170.

AT&T SUPPORTS GSS STANDARDS

GSS Graphics Standards have been selected for the new AT&T Unix PC. An integral part of the AT&T product offering is the GSS implementation of the Virtual Device Interface (VDI), being proposed by ANSI as a computer graphics standard. The VDI allows an application program to control any graphics device in a fully portable, device-independent manner. AT&T will supply the GSS VDI and peripheral device drivers on the AT&T Unix PC.

AT&T has licensed and will offer the complete line of computer graphics products from GSS. AT&T also will offer, as optional packages, the C Language Bindings for the VDI (provided with the AT&T Unix system Utilities Package), GSS-Chart, and CDI Sound Presentations. Later, the GSS-Toolkit family, which includes the Graphical Kernel System (GKS), Plotting System, and Metafile Interpreter, will be available on the AT&T Unix PC.

For more information, contact Graphic Software Systems, 25117 Southwest Parkway, Wilsonville, OR 97070; 503/682-1606.

Please circle Reader Service Number 171.

IBM SNA EMULATION SOFTWARE FROM SYSTEMS STRATEGIES

Systems Strategies Inc., a developer of IBM communications emulation software, has announced cSNA/3270, an IBM Systems Network Architecture (SNA) emulation software package.

cSNA/3270 allows a mini- or microsystem with standard ASCII CRTs and printers to emulate an IBM cluster controller operating in SNA/SDLC mode, with attached IBM 3278 CRTs and 3287 printers. It operates over dedicated or switched communications lines.

It allows manufacturers of general-purpose computer systems, PBXs, word processors, protocol converters, local-area networks, and intelligent workstations to connect non-IBM system mainframes through SNA networks.

For more information, contact Systems Strategies Inc., 225 W. 34th St., New York, NY 10011; 212/279-8400.

Please circle Reader Service Number 172.

WHITESMITHS' ENHANCED C COMPILER FOR 8086 FAMILY

Whitesmiths Ltd. has announced an enhanced version of its C compiler product for the 8086-based family of computers. The compiler offers features that enable programmers to perform more functions with the single software development tool.

Version 3.0, available in native and cross-compiler form, supports all members of the 8086 family, including 8088, 80186, 80286, 8087, 80287, and math coprocessors.

New features include the ability to support all 8086 memory models, as well as hybrid memory models that were previously available only to assembler programmers. These enhancements make Whitesmiths' Pascal suitable for implementing a wider variety of applications. All compilers also now include the systems' interface library source code as a standard feature.

Version 3.0 source code written for the 8086 now compiles under Whitesmiths' C compilers for other systems, including the DEC PDP-11 and VAX, Intel 8080, M68000, and, IBM 370.

For more information, contact Whitesmiths Ltd., 97 Lowell Rd., Concord, MA 01742; 617/369-8499.

Please circle Reader Service Number 177.

TOOLS FOR C AND AI PROGRAMMERS FROM CCA UNIWORKS

CCA Uniworks has announced a spate of new products, including the Safe C Family of five C software development tools, the Runtime Analyzer for automatic detection of programming errors, the Interpreter for fast program turnaround, the Dynamic Profiler for high-resolution program monitoring, and the English-to-C Translator and C-to-English Translator programming aids.

AI programmers using VAX systems can now purchase common LISP and OPS5 for VAX from CCA Uniworks and can combine them with CCA EMACS. Common LISP is a standardized and flexible version of LISP for use on VAX systems under the Unix system, Ultrix, or VMS. OPS5 is a new VAX programming language designed especially for developing expert systems.

Also new is MPROLOG, a new modular version of the logic-based programming language, Prolog.

Products distributed by CCA Uniworks are supported under the Unix system (4.2BSD, System V), Ultrix, and VAX/VMS.

For more information, contact CCA Uniworks Inc., 20 William St., Wellesley, MA 02181; 617/235-2600.

Please circle Reader Service Number 178.

REPORT GENERATOR FOR PHACT ISAM

UniPress Software Inc. has announced the Phact Report Generator (Phact-rg), a report generator that provides a high-level command language for formatting reports in conjunction with Phact databases.

Phact-rg complements UniPress' existing Phact ISAM package, which can access files either sequentially or through keyed record access. Phact-rg is available on the IBM PC, VMS, and the Unix-based systems.

Phact-rg supports multiline titles and column headings, dynamic fields and algebraic expressions, and can be run in interactive or batch mode.

The instructions for Phact-rg are written in a simple report instruction file that can be created with any editor or word processor.

Phact-rg is priced at $165 for the IBM PC/MS-DOS, $275 for small 68000s, $420 for large 68000s, and $575 for VAX machines running the Unix system or VMS.

For more information, contact UniPress Software Inc., 2025 Lincoln Highway, Edison, N.J. 08817; 201/985-8000.

Please circle Reader Service Number 179.

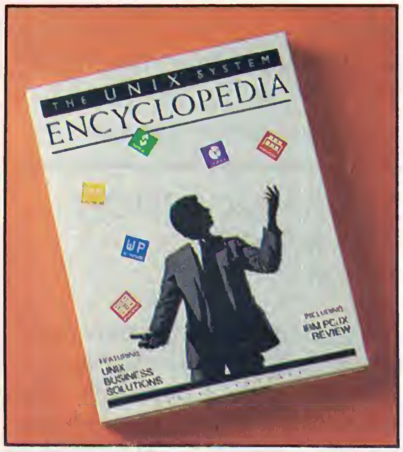
OFFICE MANAGEMENT SYSTEM FROM MED-SYSTEMS

Med-Systems has introduced a financial and medical record management system for up to six physicians. The Professional Office Management System, Version 2, will perform the following functions: patient record retrieval, patient registra-

Continued on page 108

THE UNIX/WORLD COLLECTION

COMPLETE YOUR PROFESSIONAL LIBRARY ON THE UNIX SYSTEM!

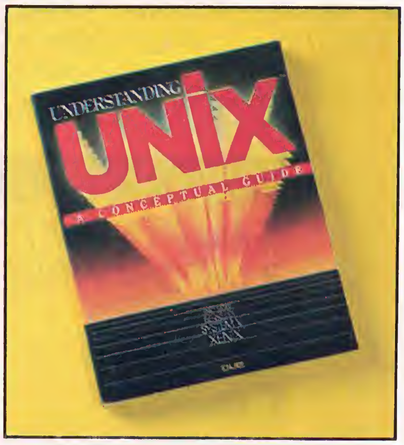


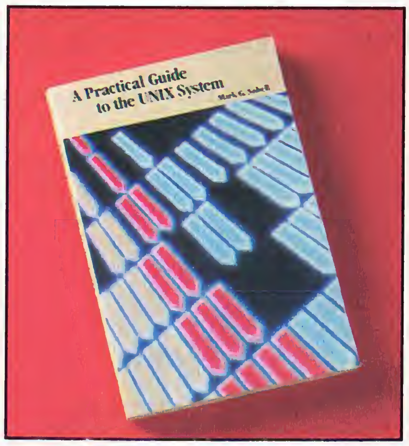
The Unix System Encyclopedia
YATES VENTURES

The UNIX System Encyclopedia is the largest available collection of information about the Unix marketplace. Over 400 pages, with a comprehensive Unix directory. Includes 100 pages of manufacturer specifications, photographs, software reviews, and articles on the Unix system.
PUBLISHER'S LIST PRICE: $34.95

The C Programming Language
Kernighan / Ritchie
PRENTICE-HALL

The C Programming Language is an in-depth guide to programming in C. A true classic, it appeals to all levels of expertise.
PUBLISHER'S LIST PRICE: $22.50





Understanding Unix—
A Conceptual Guide
Groff / Weinberg
QUE PUBLICATIONS

An excellent introduction for managers new to the Unix system, the *Conceptual Guide* introduces you to the reasons for implementing the Unix system.
PUBLISHER'S LIST PRICE: $19.95

A Practical Guide to the Unix System
Mark Sobell
BENJAMIN CUMMINGS GROUP

A Practical Guide to the Unix System is an excellent reference as well as a detailed learning tool. As an added bonus, the author had included a comprehensive overview of Xenix.
PUBLISHER'S LIST PRICE: $22.95

A User Guide to the Unix System
Second Edition
Dr. Rebecca Thomas / Jean Yates
OSBORNE MCGRAW-HILL

Takes up where *A User Guide to the Unix System* left off. The basics of the Unix System for novices, with hands-on tutorials.
PUBLISHER'S LIST PRICE: $18.95

ITEM		QTY	PRICE	TOTAL
User Guide to the Unix Syst. Second. Ed. Thomas/Yates	U200			
Practical Guide to the Unix System Sobell	U201			
The C Programming Language Kernighan/Ritchie	U202			
Understanding Unix, Conceptual Guide Groff/Weinberg	U203			
The Unix System Encyclopedia, Yates Ventures	U204			
SUBTOTAL				
SAVE! IF YOUR ORDER IS FOR MORE THAN ONE BOOK, SUBTRACT 20% HERE				
CA RESIDENTS ADD 6.5% SALES TAX				
SHIPPING AND HANDLING				$2.50
GRAND TOTAL				

SAVE ON ORDERS OF TWO BOOKS OR MORE, TAKE 20 % OFF THE SUGGESTED LIST PRICE!

Ship Order to:
Name: ____
Company ____
Add. ____
City ____ State ____ Zip ____
Signature ____

☐ Check Enclosed
☐ Please Charge to My:
☐ Mastercard ____
☐ Visa Card ____
Expires

ADDRESS ORDERS TO:

UNIX/WORLD Magazine
Tech Valley Publishing
444 Castro St., Suite 1220
Mountain View, CA 94041

SYSTEMS ADMINISTRATION: CURES FOR BUSINESS ILLS

SERIES PART 3, A FILE SYSTEM CHECKUP

BY DR. REBECCA THOMAS

If you've been with us these last two months, then you know that starting up your Unix system requires a lot more than merely the flick of a switch. For one thing, it requires a healthy file system.

In the last installment we mentioned the importance of checking and repairing the file system whenever you start up. In this installment we detail how to do this file check and repair procedure using the commonly occurring `fsck` program.

Before discussing checking and repairing the file system, however, we need first to present a brief overview of the file system parts with which you need to be familiar. In the next section we describe the three major parts to every file in the Unix file system.

First is the *inode*. Each file is described by a structure called an inode. Inodes are located in special data blocks (not used for file data), and each 512-byte block can contain as many as eight 64-byte inodes. The inode contains all the data about the file except the file name and the actual data contained in the file. Note that the disk addresses (locations on disk) for the file's data blocks are contained in the inode area. The inodes are numbered from 2 (reserved for the root directory, "/") through 65,535. This unique identifying number is known as the *inode number* or simply the *i-number.*

Then, there are the *data blocks,* which are located on another area of the disk and contain the actual data in a file. Each block can typically hold 512 characters. Some Unix system implementations (such as the Berkeley Unix system) use larger block sizes ranging from 1024 characters and up. Even if the file contains only one character, an entire data block must be allocated to hold this single character.

UNIX™ & 'C' TRAINING

MANY UNIX-BASED SYSTEMS
ONE UNIX TRAINING COMPANY

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

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

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

ONE UNIX TRAINING COMPANY MULTIPLE DELIVERY SYSTEMS

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

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

Please circle Ad No. 115 on inquiry card.

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

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

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

™ UNIX is a trademark of Bell Laboratories.

COMPUTER TECHNOLOGY GROUP

Telemedia, Inc.

310 S. Michigan Ave., Chicago, IL 60604

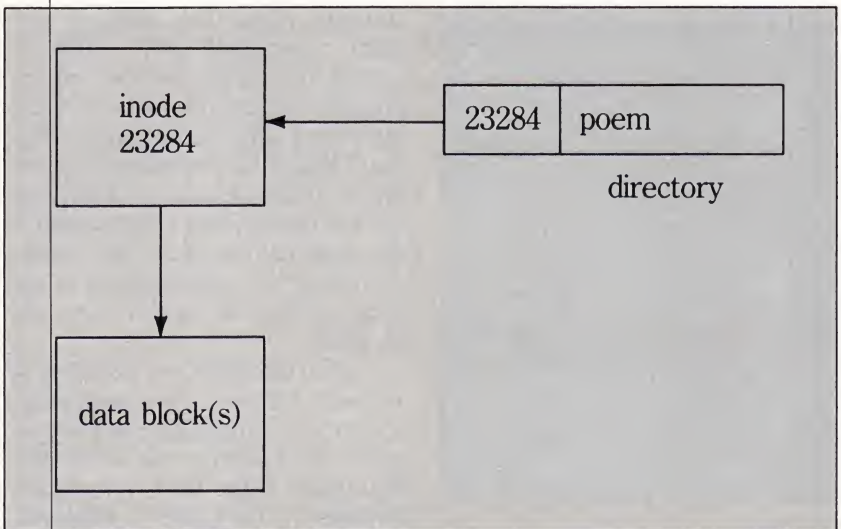


FIGURE 1: THE RELATIONSHIP BETWEEN DATA BLOCKS, THE INODE, AND A DIRECTORY ENTRY

The last part of a Unix file is the *directory,* which lists one or more file names. Each entry in a directory contains one file name and the inode number that points to the inode for the corresponding file. Note that directories themselves also have an inode structure.

SHOWING THE RELATIONSHIP

Figure 1 shows the relationship between the data blocks, the inode, and a directory that references an ordinary file. In this figure we show that the directory associates the file name `poem` with the inode for the file (we show 23284 as the inode number). We also show the inode pointing to a data block. The data block contains the actual characters in the file.

Besides these file system structures, we need to mention two other components important for administration of the file system.

The superblock is a very special block in the file system. It contains global information about the file system such as the file system name, the total size of the file system, the total number of free data blocks and free inodes, and so on.

A copy of the superblock is kept in memory at all times for easy access by the Unix system kernel. To update the disk copy of the superblock, you must perform a `sync` operation. Most Unix system implementations run an `update` daemon process in the background, which executes `sync` periodically to update the disk copy of the superblock.

The *free-block list,* the final component of the file system you need to know about, is a chain of blocks that contain the disk address of all the data blocks in the system that are available for use by a file. Part of this list is maintained in the superblock structure.

Now let's learn about operating the `fsck` command, which can check the integrity of the file system and can direct any needed repairs interactively.

OPERATING THE `fsck` PROGRAM

The file system check utility is an interactive program that will prompt you for yes or no decisions on whether to correct certain error conditions. It is a good idea to run `fsck` one time through, answering no to all queries, in order to assess the scope of any problems before you direct `fsck` to repair any error conditions. Figure 2A shows one possible command line format for invoking `fsck`.

Here *filesystem* is the name of the file system to be checked. In general, you don't have to supply this argument because `fsck` will obtain the names of the file systems to be checked from the ordinary file `/etc/checklist` (Bell) or `/etc/fstab` (Berkeley).

The `fsck` program will automatically answer no to all queries if you invoke it with the `-n` option. One reason for this conservative approach would be that the problem lies with the hardware and not with the data on the system disk. In this case, you wouldn't want `fsck` to attempt repair of the file system data structures.

Immediately after invocation, `fsck` enters an initialization phase. If `fsck` cannot get enough memory to store its intermediate results, it may query you to name a file to be used temporarily as a "scratch pad" during the checking procedure. There need be nothing special about the particular name you choose, but if possible you should select a path name so that the scratch file is not on the file system being checked.

Figure 2B shows an example of output from `fsck` if there are no errors. The program may first print the name of the file system it is about to check, and then it scans that

a. A Sample Command Line Format:

/etc/fsck [-y] [-n] [*filesystem*]

b. Sample Output:

```
# /etc/fsck
/dev/root

** Phase 1 — Check Blocks and Sizes
** Phase 2 — Check Path Names
** Phase 3 — Check Connectivity
** Phase 4 — Check Reference Counts
** Phase 5 — Check Free List
414 files 9857 blocks 7560 free
# □
```

FIGURE 2: OPERATING THE `fsck` PROGRAM

```
# /etc/ncheck -i inodenumber... filesystem
# □
```

FIGURE 3: USING `ncheck` TO DETERMINE FILE NAME

file system several times, as shown by the different "phases." Finally, some statistics about the file system are displayed.

Now we'll discuss operation of the `fsck` program in more detail, examining each phase of `fsck`'s operation and some commonly occurring informational and error messages and how to deal with them. You need to study this material closely because the correct operation of the `fsck` program is critical to repairing and maintaining a file system. Before we discuss the phases of `fsck`, you need to know how to clear a corrupted inode.

ZAPPING INODES

Many of the `fsck` queries ask you to "CLEAR?" an inode because it is corrupted. If you answer yes, the inode is cleared by writing all zeros into it, and the corresponding file is destroyed. Sometimes you can save the data in the file by first copying the file contents to another location in the file system.

In saving a file, first answer no when the `fsck` program displays the "CLEAR?" query on a bad inode. The `fsck` program will also display the inode number (indicated after "I=") and the size of the file (indicated after "SIZE="). You should record these figures because you will need them later. Continue with the `fsck` program, answering no to any "CLEAR?" prompt on an inode you intend to save.

One deficiency of the file check program is that you cannot determine the name of the file from the bad inode because `fsck` simply reports the inode number. This is a situation where the `ncheck` program comes into play. After returning to the shell, use the `ncheck` program to determine the file name associated with the bad inode. Figure 3 shows the command line syntax for using `ncheck` to determine the file name. Here *inodenumber* is the inode number of the file you are searching for, and *filesystem* is the name of the file system you are checking.

If no file name was reported by `ncheck`, the file is unreferenced (by a directory entry), so you can't access it for a backup with a Unix system utility. If the file is referenced, however, a file name is displayed. You can use a utility such as `cat` to examine the contents of the file. If the data is intact, you should back up the file contents, say with `cp` or `tar`, to a different file system.

After saving the contents of any bad file, rerun the `fsck` program, but this time answer yes to the "CLEAR?" queries. After the file system is clean (no errors reported during a `fsck` run), retrieve the file that was saved earlier.

THE `fsck` PROGRAM GOES THROUGH PHASES

Now we will discuss the remaining operational phases of the `fsck` program. In the phase numbered 1, each inode in the file system is checked for its own data integrity, and then the disk blocks pointed to by the inode are checked. Several error messages at this stage are possible. "UNKNOWN FILE TYPE" means the inode is corrupted and should be cleared after the file contents are backed up.

If the message "BAD" occurs, the block address in the inode is invalid. Another common error message is "DUP" (for duplicate), which occurs if a block has already been claimed by another inode. You will not be prompted to remove "BAD" or "DUP" blocks until phase 2 or 4. A large number of "DUP" and "BAD" blocks result from a damaged *indirect block,* which is a data block containing addresses of other data blocks for the file.

Another warning message is "POSSIBLE FILE SIZE ERROR," which occurs if the estimated number of blocks for an ordinary file does not agree with the actual number of blocks claimed by the inode. The "DIRECTORY MISALIGNED" message occurs if the size of a directory file is not a multiple of 16 bytes.

If these last two error conditions aren't cleared during the interactive repair in later phases of this program, you should back up the file or directory contents. Manually erase the file (with `rm`) or directory (with `rmdir`), and the corrupted inode will be de-allocated. After that, run `fsck` again to be sure the file system is now clean.

If any "DUP" blocks were discovered in phase 1, this error will initiate another scan of the inodes, as indicated by the message "Phase 1b—Rescan for More DUPS."

If a block that has been claimed by the inode being examined has already been tagged as a "DUP," the "DUP" warning message is repeated, and you will be prompted "CLEAR?" If you answer yes, the inode will be marked for possible removal in a later phase.

ANOTHER ERROR MESSAGE

Another possible error message during phase 1 is "PARTIALLY ALLOCATED INODE." This means that the inode is neither allocated nor unallocated. You should answer yes to the following "CLEAR?" prompt, but you may want to back up the associated file before clearing the inode.

In phase 2, `fsck` inspects all directory inodes in the file system. First the inode for the root directory is examined. If `fsck` reports it "UNALLOCATED," `fsck` will abort because this means the root inode is corrupted. If you get the message "DUPS/BAD IN ROOT INODE (CONTINUE)?" answer yes, but expect a large number of related file system errors as a result of this last error condition.

`fsck` can fix most file system errors, except for a corrupted superblock or inode. In some cases, even these structures may be repaired by someone experienced with the file system and the file system debugger program, `fsdb`.

Continuing with phase 2, the inode (and each block for that inode) corresponding to each directory entry in the file system is examined. One error condition that could occur here is "OUT OF RANGE," which means that the inode number of the directory entry is invalid. If you elect to remove the inode in response to the "REMOVE" prompt, the inode field of the directory entry will show zero.

An "UNALLOCATED" message means the inode is possibly unallocated, but it should be allocated because it is referenced by a directory entry. In this case, answer yes to the "REMOVE?" prompt so that the directory entry doesn't point to an unallocated inode.

At this point you will be given the opportunity to clear the inodes that had a "BAD" or "DUP" block discovered during phase 1 or 1b of the check procedure. The `fsck` program may supply you with a file name, or you can use `ncheck` to determine it from the inode number. If the file is of no value to you, answer yes to clear the inode. The "DUP" blocks are frequently shared between a file and the free-block list, so in this case don't clear the inode. Instead, wait for an additional phase 6, which rebuilds the free-block list.

GLOSSARY/COMMAND SUMMARY

`cat`	Display the contents of one or more files
`cp`	Copy the contents of a file
`fsck`	Check consistency of file system and repair interactively
`fsdb`	File system debugger; can be used to patch a damaged file system
`mklost+found`	Shell procedure used to create the `lost+found` directory
`mv`	Move (by renaming) a file
`rm`	Remove an ordinary file by removing a directory entry (or link) to the file
`rmdir`	Remove a directory file
`ncheck`	Generate a path name from an inode number
`sync`	Update the disk copy of the superblock
`tar`	Tape file archives; may be used to copy one or more files (or directory contents) to another part of the on-line file system or backup media

```
# cat mklost+found
echo 'Mount point (pathname) for file system? (RETURN) for /'
read rootdir
cd ${rootdir}/lost+found
for file in 1 2 3 4 5 6 7 8 9 0 a b c d e f
do
     >$file
done
rm *
# □
```

FIGURE 4: A TYPICAL `mklost+found` SHELL PROCEDURE

PHASES 3 AND 4

In phase 3 all the allocated inodes are scanned for unreferenced directories; that is, directories where the inode corresponding to the parent directory entry (`..`) does not exist. In this case you will be prompted to "RECONNECT?" any orphaned directories. If you answer yes, a link from the orphan directory to the special directory `lost+found` will be made. The name of the link to the `lost+found` directory will be the inode number from the orphan, now newly connected, directory. After the `fsck` program is finished, you can examine the entries in `lost+found` and move them (by renaming with `mv`) to their appropriate place in the file system.

If a `lost+found` directory doesn't exist in your file system, you can easily create it if your system has the `mklost+found` shell script. This command creates the `lost+found` directory by making a large number of directory entries and then deletes them to give a large, but empty, directory. If you delete an entry (file or directory) within a directory, the size of the directory isn't changed.

If you don't have the `mklost+found` shell script, you could simulate the procedure by changing to the root directory of the file system and entering `mkdir lost+found` and then `cd lost+found`. Next, create several dummy files and then remove them (with `rm`). Figure 4 shows a typical `mklost+found` (Bourne) shell procedure to accomplish this task.

Phase 4 is concerned with the link count or reference count information that was accumulated in phases 2 and 3. In phase 1 the reference count is first set to the link count value stored in the inode. The link count is the number of links (or file name aliases) for that physical file. Then, in phases 2 and 3, the reference count is decreased each time a valid link is found while scanning the file system. So the reference count value should be zero when phase 4 begins.

If the reference count for an inode was actually zero to begin with in phase 1 (indicating a corrupted inode), you may clear the inode at this point. To do so, answer yes to the "CLEAR?" query for the unreferenced file "UNREF FILE" or directory "UNREF DIR."

If the reference count for the inode is not zero at the start of phase 4, not all the links were discovered in phases 2 and 3. The `fsck` program distinguishes between two cases at this point: (1) none of the claimed links were discovered, in which case the inode refers to an "orphan" file or directory; and (2) not all of the claimed links were discovered, in which case the `fsck` program can correct the link count field in the inode.

In the first case, `fsck` can link the orphaned file or directory to the `lost+found` directory. You should answer yes to the "RECONNECT?" query to accomplish this. If this linkage operation was not successful, or if you decided not to link, then answer yes to the subsequent "CLEAR?" query associated with the next "UNREF FILE" (or "DIR") message.

In the second case, `fsck` will adjust the link count if you answer yes to the "ADJUST?" query that follows the "LINK COUNT FILE" (or "DIR") and "COUNT *X* SHOULD BE *Y*" message.

In the last stage of phase 4, you are given the chance to clear the inodes of unreferenced files and files containing bad or duplicate blocks. Finally, if you get the error message "FREE INODE COUNT WRONG IN SUPERBLK," you should answer yes to the "FIX?" query.

PHASE 5

Phase 5 checks the free-block list. Any bad or duplicate blocks in this list are flagged, and later you will get the message "BAD FREE LIST" and be prompted to "SALVAGE?" (to which you should answer yes). After you have elected to salvage the free list, a phase 6 is initiated, which reconstructs the free block list.

If a file system was corrupted and then fixed, you will get a message like "****** BOOT UNIX (NO SYNC!) ******" or "****** FILE SYSTEM WAS MODIFIED

******" just before `fsck` exits. This message means that `fsck` had to modify the file system in order to repair it. To prevent the work done by `fsck` from being "undone," the system should be rebooted without performing a `sync` operation (see last month's installment for more about `sync`).

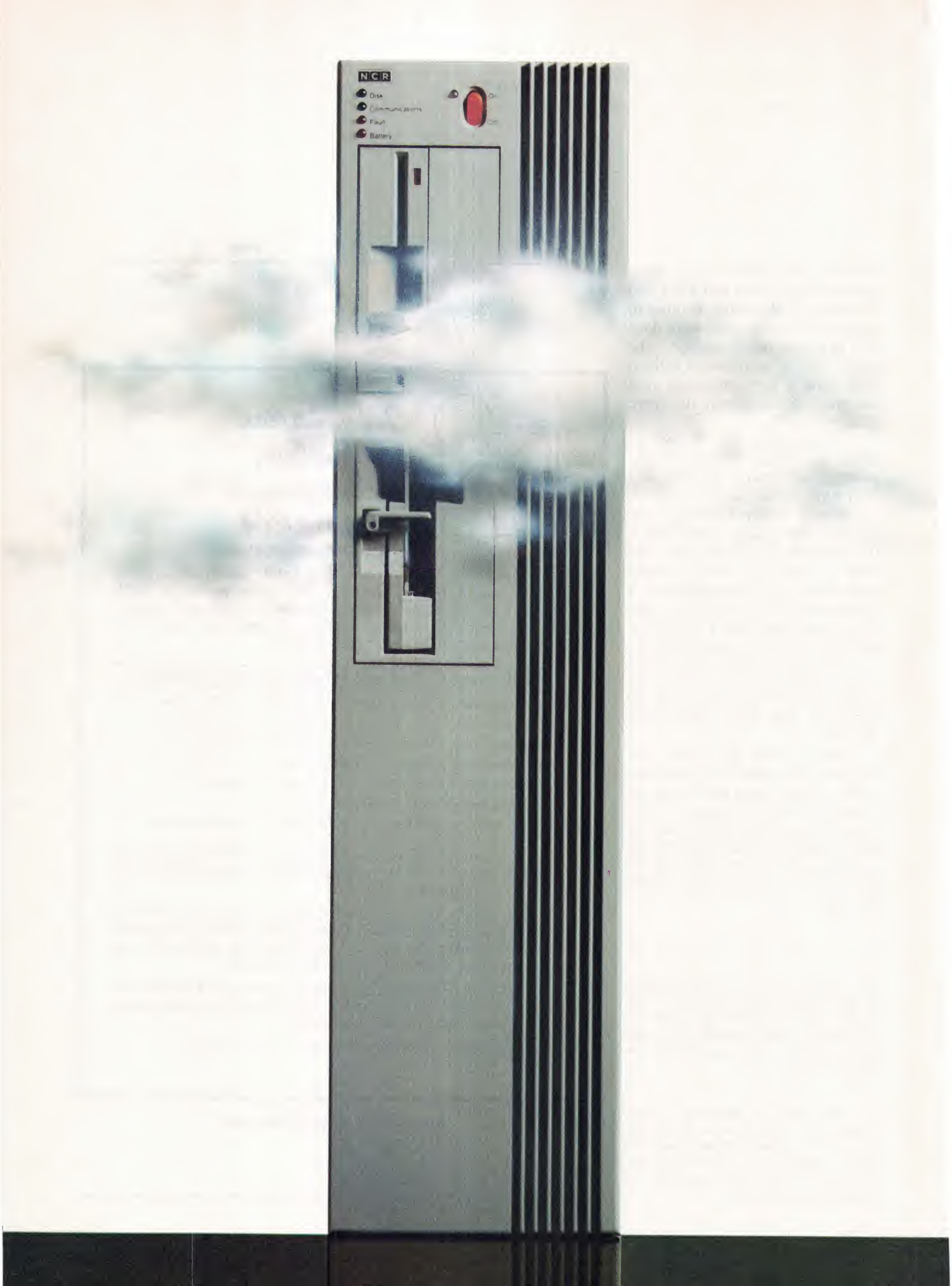
At this point, shut the system down by simply turning off the power, wait several seconds, and then turn on the power (or press the "Reset" switch, if available) and bootstrap the system again. This is the *only* time you should ever shut down or reset the system without performing a `sync` command first. If there were errors on the last run, you should repeat the `fsck` program until no more errors are reported.

If you follow the steps outlined in this installment for checking and repairing your file system, you will help keep your Unix system alive and well.

Next month our "Cures For Business Ills" turns to a discussion of the general steps you would use to create a new file system. Then, because your Unix system can't be used when the system disk fills up, we'll discuss several utilities and approaches for managing this valuable resource. ☐

Dr. Rebecca Thomas, UNIX/WORLD's Editor Emeritus, is an author of A User Guide to the Unix System, *the second edition of which is now available. She is currently writing a book on Unix system administration.*

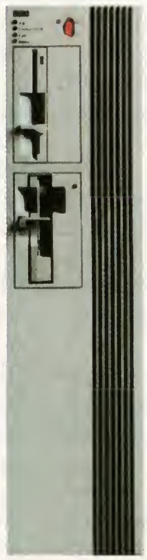
This article is based on A User Guide to the Unix System, *second edition, by Dr. Rebecca Thomas and Jean Yates. Copyright (C) 1985 by McGraw-Hill Inc. Used with permission of Osborne/McGraw-Hill.*


NCR
Disk
Fault
Battery

THE MORE YOU KNOW ABOUT UNIX,* THE TALLER THE TOWER XP STANDS.

The engineers at NCR are now convinced that UNIX is the best operating system in the industry.

Of course, to convince these reliability fanatics, UNIX had to endure thousands of hours of grueling tests. But that's simply the price any component must pay to get into an NCR product.



And it's a better UNIX for the experience.

The NCR-enhanced UNIX in the Tower XP is the first UNIX of true, commercial reliability.

It virtually never PANICs. Consequently, neither do your customers.

So that this UNIX might be all things to all users, we employed the Tower XP's menu builder to create the five faces of UNIX. A remarkable case of multiple menu personalities. One for each of five user levels, from sophisticated user to data duffer.

And our UNIX speaks a language for each of five programmer types: Business BASIC, Pascal, Fortran, C, and Cobol.

Team this powerful operating system with the Tower XP's other prodigious features; consistent error logging, extensive remote and in-service diagnostics, Automatic Power Failure Recovery, and more, and you've got a monumental mixture. Capped by NCR's towering commitment to OEMs.

The more you know about Tower XP, the better NCR looks.

TOWER XP. BUILT FOR SYSTEMS BUILDERS BY NIT-PICKING FANATICS.

NCR
OEM Systems Division

NCR Corporation, U.S. Data Processing Group, OEM Systems Division, USG-1, Dayton, Ohio 45479. Nationwide (800) CALL NCR. In Ohio (513) 445-2190. In Canada (800) 268-3966.

*UNIX is a registered trademark of AT&T.

Please circle Ad No. 26 on inquiry card.

able interface that should satisfy both user groups.

If the facility does favor one group of users over the other, it is probably biased toward helping beginners. This is appropriate because beginning system users are the ones who presumably most need an on-line help facility. Even so, several features have been added that allow more experienced users to retrieve the information contained in the system more quickly; these features should become evident as the user interface is described.

Basically, the `help` facility is a menu-driven system. The simplest way to enter the facility is to type "help" at shell command level. This yields a menu that allows the user to select either the `starter`, `glossary`, `usage`, or `locate` modules of the facility. Each of these four modules can also be entered directly from shell command level. For example, the `usage` module can be entered by typing either "usage" or "help usage" at shell command level.

Once inside the `starter`, `glossary`, `usage`, or `locate` module, users can obtain specific information. In `starter`, the information is accessed by making a selection from a menu. In `glossary`, entering the name of a term or symbol retrieves its definition. In `usage` and `locate`, the system requests either a specific command name or a set of keywords, respectively.

After processing the input, either the requested `starter` information, the definition, the command information, or a list of appropriate Unix system commands, respectively, is output to the user's terminal.

The `glossary`, `usage`, and `locate` modules can be entered directly from shell command level by typing (for `glossary`) "glossary term" or "help glossary term" (where "term" is a Unix-related term or symbol), (for `usage`) "usage [-d/-e/-o] command_name" or "help usage [-d/-e/-o] command_name," or (for `locate`) "locate keyword1 [keyword2]" or "help locate keyword1 [keyword2]".

In the `usage` module, the option "-d" retrieves the syntax summary and description of the specified command and is the default. The option "-e" retrieves the examples, and the option "-o" retrieves the option descriptions.

Every screen, whether it is primarily a menu screen or not, contains a list of options available to the user at that point. Obviously these options vary from screen to screen, and there is not enough space here to describe every screen in the system. Nevertheless, we can describe the options in general terms and provide a reasonably thorough understanding of how the facility works. At every point, the user has the following set of options:

- The user can proceed with entry to a module of the facility, or with some processing within a module.
- The user can always restart the current module.
- The user can always "advance" to the next screen.
- The user can always exit directly to the Unix shell.
- The user can always return directly to the initial `help` menu, if that is where the facility was entered.
- The user can always escape to the Unix shell to execute a command and then return to his or her current location in the facility.

Aside from the rules given in the last paragraph, the only general principle we used in designing the facility was to make sure that the information it contains is densely packed. Quite justifiably, most users strongly object to retrieving screen after screen of menus and intermediate information before reaching the information they need.

Certain advantages do accrue for beginners if they are given only a small amount of information in each display, but we believe these advantages to be far outweighed by the irritation this sort of procedure causes in experienced users and by the disorientation that nearly everyone experiences after traversing a long series of screens. Conceptually, the `help` facility described here is never more than three layers (or screens) "deep." This results in a quite dense packing of information.

In summary, the `help` facility is an interactive utility available at shell command level that provides a variety of information on the Unix system. Although the facility is menu driven, it allows direct access at two interior levels from the shell so that experienced users can retrieve information quickly. In addition, the facility's data can be expanded and customized using a set of software tools provided with the facility.

We spent much time designing the flow of control through the facility, and we believe we have achieved our goal: a terse, but easy-to-use structure and command language. □

Mr. Butler holds a doctorate in psychology from Brown University. He joined AT&T Bell Labs in 1978 and currently works in a group developing new features for the Unix System kernel. Ms. Kennedy is currently completing an M.S. in computer science at Rutgers University. She joined AT&T Bell Labs in 1979 and currently is a Unix System utilities systems engineer.

A POSTMAN AND A PHILOSOPHER

BY DR. REBECCA THOMAS

In this month's installment of "Wizard's Grabbag," we have several reader contributions to share with you, including a shell script for formatting mailing labels and some thoughts on Unix system philosophy.

Dear Dr. Thomas:

Here's a shell script useful for formatting records of a mailing list into mailing labels. [*Doctor's note:* Figure 1A lists this Bourne shell script, `mklabel`.] You may use this script with the C shell if it begins with a "#" and if every "Newline" is escaped with a backslash.

The input data file for the `mklabel` program must consist of name and address fields (and possibly other data) separated by a particular field-separation character (one not used within the field itself). For this example, each record in the data file consists of six fields—name, company name, street address, city, state, and zip code—separated by commas. All records should contain this same number of fields. Also there should be no extra "Newlines" at the end of the data file.

[*Doctor's note:* Figure 1B lists a sample data file, and Figure 1C shows the shell script being used in a typical application. Here the shell is directed to pipe the output of `grep`, which selects only people in California, into `sort -n`, which then orders the selected records by zip code, and then into `mklabel`, which in turn formats the data into labels—one across, six lines per label, including one line between each. Finally, the result is sent to `lpr` for spooling on the system line printer. You can view a copy of that same data stream on your terminal because we inserted `tee /dev/tty` into the pipeline.]

Ray Swartz
Berkeley Decision/Systems Inc.
Santa Cruz, Calif.

SOME UNIX PHILOSOPHY AND OTHER THOUGHTS

Dear Dr. Thomas:

I'm submitting some thoughts from a Unix manager's guide that I developed in high school, when I had to teach younger students how to manage our Unix Version 7 on a PDP 11/34:

Unix systems tend to encourage the programmers that use them to improve their system and personalize it. This is what keeps Unix alive and well. I do not claim that Unix is, as it stands, a complete system. This is probably why everyone ends up personalizing and modifying

a. The `mklabel` Bourne Shell Script

```
$ cat mklabel
awk ' BEGIN {FS=","}
      { for (count = 1; count <= 3; count = count + 1)
              print $count
        printf "%s, %s %s\n\n\n", $4, $5, $6
      }'
$ □
```

b. The sample input data file

```
$ cat database

Ray Swartz,BD/S Inc.,150 Belvedere Terrace,Santa Cruz,CA.,95062
Fred Unix, AT&T Technologies Inc.,P.O. Box 25000,Greensboro,NC.,27420
Rebecca Thomas,UNIX/World Magazine,444 Castro St.Suite 1220,Mountain View,CA.,94041
$ □
```

Continued

c. A typical application for using mklabel

```
$ grep CA database ¦ sort -t, +5 ¦ mklabel ¦ tee /dev/tty ¦ lpr
Rebecca Thomas
UNIX/World Magazine
444 Castro St. Suite 1220
Mountain View, CA. 94041

Ray Swartz
BD/S Inc.
150 Belvedere Terrace
Santa Cruz, CA. 95062

$ □
```

FIGURE 1: THE mklabel BOURNE SHELL SCRIPT AND ITS USAGE

it. In my four years as a Unix system programmer/manager/wizard, I have come to the following conclusions:

(1) The installation of a Unix system is never complete.

(2) The Unix system is dynamic and must stay that way if it is to keep up with the rest of the software world.

(3) Copyrighting/protecting software/ideas only hinders the progress of Unix development in general.

(4) The concept of "software toolboxes" should be introduced to the entire Unix community. These toolboxes should be relatively free of copyright shoestrings and other such development hindrances. By "toolbox," I refer to a complete set of routines which performs a specific sub-task within a given system.

For example, all programmers (if they are worth their salt) have written, possibly on multiple occasions, list processors of some sort. A package that does all sorts of list processing, such as the one that I have, saves gobs of time and money. It is public domain, and the only restriction is that if you pass it on to someone else there will be no money changing hands on the deal. Neat, huh? and *very* helpful. (I must give credit for this idea to Bob Zimmerman, a Carnegie-Mellon University programmer/wizard.)

(5) Many people think of the operating system as sacred territory. This is a gross misconception. Unix must advance on all levels, including the operating system, libraries, "bin" programs, etc. Those without source licenses of some sort have a definite problem with modifying the operating system and "bin" programs. But adding libraries and even replacing "bin" programs with better ones is still possible, and many debuggers make it possible to modify the code in its compiled form anyway. You need not be a systems person to make changes as long as you understand what you are doing before you do it. Caution is a programmer's best friend.

Regarding the "Grabbag" contribution in Vol. 1, No. 5:

Adding a shell script front-end for message transmission seems to me to be overdoing it a little . . . . The mail program can easily be modified to report the fact that mail has arrived for a user if he is logged in. [*Doctor's note:* Some systems are already set up to do this. For these, simply initialize the MAIL environment variable to the path name for the recipient's mail file.] The user then reads the message (memo) when he or she feels like it.

This requires a small and straightforward modification to the mail subroutine that actually writes the message to the user's mail file. When any message is written, this routine should also (1) check if the user is logged on, and if so (2) get the name of the terminal being used, and then (3) send to that terminal a couple of "Newlines," whom the message is from (sender/user ID), the subject of the message, and perhaps a couple of "Bell" characters ("Control-G").

If you don't have the source code for the mail program, here is an alternate suggestion with slightly more overhead:

Rename the original mail program to, say, do-mail. Write a "driver" program, naming it mail. The driver looks at its invocation arguments, and if you are sending a message to another user, the driver sends the message (without the subject field) to the recipient's terminal (if logged in).

The driver then "fires up" `domail` with the same invocation arguments. This approach has one obvious flaw—if you send someone a message and then abort, the recipient will still get the message (on the screen, at least).

Obviously, these alternate solutions are not complete because you would rather have our messages delivered immediately.

can't specify a time for message delivery, but some of us (1) don't have the `at` program on our systems, (2) use `calendar` for our personal reminder service, or (3)

One copy of it needs to sit somewhere, with all the programs that you wish to account for linked to it. Any call for those programs (that is, the driver) would do several things. First, it would record the program name and arguments, date,

Have you ever wished to find out quickly, without going through the system accounting files, how many times anyone uses a program that exists on the system? How about accounting for the usage of *any* `/bin` or `/usr/bin` program? Again, a driver is the answer.

time, user ID, or anything else you want into a log file (either one entry in a file for all programs or one entry in a log file for each different program).

Second, it would use the `execve()` system call (or equivalent) to execute the real program—with the first argument being the concatenation of a hidden directory and the actual program name, the second argument being the `argv` pointer, and the third being a pointer to the *unaltered* environment. After all, you don't want to pass the wrong PATH value to the actual program.

Paul B. Reiber, Jr.
American Robot Corp.
Pittsburgh, Pa.

A PORTABLE VERSION OF `smail`

After going to press with our April installment of "Wizard's Grabbag," Michael Poppers submitted another version of his `smail` Bourne shell script. The version published in April was specifically for Interactive's IS/3 Unix system on a VAX 11/780. Here are some changes he suggested to make this script run on most Unix systems that have the necessary utility programs, such as `whois` or `cut`.

First change lines 23 and 77 to `echo >>$file`.

Then in line 37 eliminate the pipe to the second instance of the `cut` command. Change the arguments to the first `cut` command to be `-d: -f1`, which "cuts out" the contents of the first colon (":") delimited field from the password file (the user or log-in name) for storage in the variable, `sender`. Use `-d: -f1,5` if you wish to include the contents of the fifth (comment) field, too. □

Erratum for April 1985 "Wizard's Grabbag":

The contents of line 7 in Figure 2 should be all on one (long) line.

"Wizard's Grabbag" is a regular feature of UNIX/WORLD. Submit your shell scripts, C programs, or tips and techniques that ease the burdens of system administrators and programmers to "Wizard's Grabbag," UNIX/WORLD, 444 Castro St., Suite 1220, Mountain View, CA 94041. Authors of published entries receive $50 for shell scripts, `awk` scripts, `sed` scripts, `lex`, `yacc`, and C programs, or tips.

Please follow these guidelines for reader contributions: Write your shell scripts, C programs, and other code so that it is portable across different versions of the Unix system. If possible, it should run without change on Bell Version 7, Systems III and V, and Berkeley 4.x. Thus, you should use "universal" Unix utilities such as `who am i` (all systems) in lieu of `whoami` (Berkeley only), and the Bourne shell, if possible, when coding shell scripts.

However, C shell scripts are also welcome because most of our readers now have access to this popular command interpreter. Use the standard I/O library when writing C code. In addition, use the lint syntax checker to eliminate nonportable constructions, and compile the code with a portable C compiler such as `pcc` to help ensure portability. Hardware dependencies, such as terminal control sequences, should be eliminated or at least minimized and isolated to one code region or to a separate module. Keep your example as short as possible, say under 100 lines of code.

tion, appointment system, daily transactions, billing, and insurance processing.

Reports of a transaction record and payment register (bank deposit slip) are automatically printed at the end of each day. In addition, the total fees collected are automatically entered at the end of each day into the checking account balance in the general ledger.

The complete Office Management System will operate with a Tandy model 6000 microcomputer, a 12- or 15-Mbyte hard-disk drive, and any type printer. The complete program software costs $600.

Up to five additional work stations may be added to this system. The programs are written in COBOL and use the Xenix (Unix) operating system.

For more information, contact L.E. Olney, M.D., 1932 Brookside Rd., Kingsport, TN 37660; 615/246-4155.

Please circle Reader Service Number 180.

TERMINAL EMULATOR SOFTWARE FOR HP 9000 SERIES 200

Hewlett-Packard Company now offers terminal emulator software that allows HP 9000 Series 200 computers to act as interactive, asynchronous, alphanumeric terminals to HP, DEC, Data General, CDC, Prime, and other mainframes.

Three of HP's Pascal 2.1-based emulators for asynchronous, alphanumeric terminals have been consolidated into one Pascal 3.0-based product. The HP 2392A and DEC VT 100 terminal emulator (HP 98791B) replace the asynchronous-terminal emulator, the HP 2622A terminal emulator, and the VT 100 and HP 2622A terminal emulator.

The Series 200 computer retains all of its engineering and computational capabilities, while the emulator provides datacomm and terminal-configuration menus that are equivalent to the HP 2392A and the VT 100 terminals. When used as

a bridge from engineering to another facet of a company's operations, the emulator offloads the host computer by performing tasks on the Series 200 workstation.

The emulator supports the entire Series 200 (except Model 226) in either a stand-alone configuration or under the Pascal 3.0 operating system. It also supports Pascal 3.0 peripherals and execution.

For more information, contact your local HP sales office. □

Please circle Reader Service Number 181.

UNIX/WORLD's "New Products" section is provided as a service to our readers, and our selection criteria are based solely on the needs of our readership. If you would like to have your product news considered for publication, please address your correspondence to UNIX/WORLD Magazine, New Products Editor, 444 Castro St., Suite 1220, Mountain View, CA 94041. Because of the large number of press releases we receive, UNIX/WORLD cannot verify the accuracy of claims made by a product's manufacturer. We advise that you thoroughly test any product before buying.

THE C STANDARD IS COMING

PART 2

BY STEVE HERSEE

Last month I discussed the reasons behind the current C standardization efforts, as well as the progress made in that endeavor at the June and September 1984 meetings of the American National Standards Institute (ANSI) X3J11 C standards committee meetings.

In this month's installment, I'll bring you up to date on current status of the draft of the proposed C standard, as well as the technical issues discussed, debated, and (hopefully) resolved at the committee's December 1984 meeting.

That meeting was hosted by Motorola Inc. in Tempe, Ariz., from December 3-7. At this meeting, 29 voting members were present, perhaps the largest number of voting members ever present since I started going to the meetings. Several new people, who plan to become voting members, also attended this meeting, so it looks as though support for the committee is still growing.

This meeting was the first at which all three parts of the standard document (environment, language, and library) were put together in final form. When the committee was first formed and a calendar set, the December 1984 meeting was to approve a draft standard document to go out for informal comments. After the informal comments were received, the next step would be to send the draft standard out for formal comments. At that point, the journey toward a standard for C would be under way.

At the start of the meeting many people were concerned that the document would be railroaded out of the committee before it was ready. Some were worried that they had had only seven days to review the document before this meeting and that the volume of new material was too large to be able to approve during the meeting. Others felt that the committee vacillated on many issues; that is, a particular issue would go one way at one meeting and another way at the next.

I felt that we should publish the document after this meeting because all of the arguments in favor of delaying the release (not ready, issues still open, and so on) would be just as valid at following meetings.

TWO-THIRDS MAJORITY NEEDED

We decided that the committee would need a two-thirds majority to publish the document. Any no votes would be listed with a reason why the voter felt the draft was not ready for publication.

We all worked hard at the meetings, from 8 A.M. to 6 P.M., to prepare the document for publication. But as time ran out the number of changes made members uneasy about publishing a document they would not have a chance to review in its final form. We finally voted on Friday, about 20 minutes before the meeting was to adjourn. The vote was close—13 to publish and 12 not to publish. This was better than I expected, considering all the talk I had heard. The vote did not carry the necessary two-thirds majority, though, and the committee finally accepted a proposal to prepare the final document and send it, along with a letter ballot, to every committee member.

If, during the two-week voting period, the draft standard received a two-thirds majority, then the document would be published and sent to the CBEMA (Computer Business Equipment Manufacturers Association) Secretariat. Once CBEMA had the draft, interested parties wanting to submit comments to the committee could obtain a copy from CBEMA for a nominal fee.

The draft will say that C compilers should not be implemented from this draft because additional changes might be made. The committee is aware that many compilers might be changed to add features published in this draft. Please, if you are going to modify your compilers and are not part of the X3J11 committee, then join the committee. Because many of the committee members are also vendors of C compilers (like myself), those members will have the inside track on changes to the draft after it has been published.

If you cannot attend the meetings, you can sign up to receive mailings from the meetings. The committee does act upon the many written submissions it receives, and we would like more people to comment on the work we are doing. If you are interested, please feel free to contact Jim Brodie of Motorola at 602/438-3456. Now I'll discuss some specific issues raised at the last meeting.

A CHANGE IN THE USE OF THE & OPERATOR

The C language understands the operator `&` to mean "address of." So if you say `a = &b;`, this means let `a` equal the memory address of the variable `b`. In current C, this does not hold true if `b` is an array. If `b` were declared as `char b[5];`, then the statement `a = &b;` would cause an error in some C compilers.

In Kernighan and Ritchie's book, *The C Programming Language* (also known as "K&R"), the authors clearly point out that using the `&` operator with arrays is illegal (page 94). Funny, but I did not know that any laws had been passed about this issue—but then, I have always preferred the use of the word *invalid* to *illegal*. K&R is specific: If you want the address of an array and also want to use the `&` operator (so that when you read the code you can say "address of"), you must say `a = &b[0];`.

We on the committee had long arguments about common practice. How many people have been taught that, in order to get the address of an array, they *must* say just the array name or take the address of the first element of the array. I prefer to have the `&` in front of the array to remind me when an address is needed. At the September meeting, the committee voted not to allow the `&` in front of array names.

I figured I had lost on this issue, but to my surprise I found that I was saved (at the December meeting) by the COOKIE. The what? The COOKIE, which I learned about while on the committee, is an item that library functions pass around to keep items that may change from computer to computer. A good example is the time-of-day function. The standard defines a `typedef` type of `time_t`, which defines the storage needed to keep the time of day for a particular computer, as in the example in Figure 1.

If `time_t` says that `time1` is a long type, then the old K&R rules are fine. But what if the `time_t` is an array of `ints` to hold the time in microseconds? Because the standard allows the compiler vendor to define any type to `time_t`, you cannot write a portable program that passed the address of an item defined with a supplied `typedef`. Since you don't necessarily know whether such items are arrays, the standard now allows you to use the `&` operator with array names and function names.

PREPROCESSOR ARITHMETIC

If you have a program that you want to be as portable as possible, you will find yourself becoming concerned about how the preprocessor computes values. Some programs use the preprocessor to tell them the size of an "int." (You can see an example of this in Figure 2.)

Does the code in the example

ADVERTISERS INDEX

If you are interested in products or services advertised in UNIX/WORLD, our advertisers are pleased to send you additional information.

Simply fill out the reader service cards and circle the numbers of advertisers you are interested in. Then mail the card and expect prompt, pertinent information.

The ad index is published as a service. The publisher does not assume any liability for errors or omissions.

Publisher: Robert A. Billhimer

Advertising Manager: Paul B. Fertal **Advertising Assistant:** Jeanne Ketchum

444 Castro Street, Mountain View, CA 94041 415/940-1500

ADVERTISING SALES

New England, New York, New Jersey, Pennsylvania, and Eastern Canada: Charles E. Lynch, Nancy Wood, Merrie Lynch, 61 Adams St., Braintree, MA 02184, 617/848-9306. **Midwest:** Jeff Edman, Louise Grauel, The Pattis Group, 4761 W. Touhy Ave., Lincolnwood, IL 60646, 312/478-4000. **Denver:** R.H. "Bud" Foster, 14 Inverness Dr. East, Bldg. A, #232, Englewood, CO 80112, 303/790-8181. **Dallas:** Pam Watkins, 5801 Marvin D. Love Freeway, Suite 308, Dallas, TX 75237, 214/941-4461. **Los Angeles:** Gordon T. Sutton, Carole Ann Philips, 524 W. Commonwealth, Suite G, Fullerton, CA 92632, 714/525-6686. **Miami:** Ray Rickles, Ray Rickles and Company, 3783 Pinetree Dr., Miami Beach, FL 33140, 305/532-7301. **Atlanta:** Scott Rickles, Ray Rickles and Company, 699 Carriage Dr., N.E., Atlanta, GA 30328, 404/252-4567. **San Francisco:** Morton McDonald, Ginger McDonald, Ian McDonald, 265 Baybridge Office Plaza, 5801 Christie Ave., Emeryville, CA 94608, 415/653-2122; 415/967-6400. **Oregon:** Frank M. Eaton, 510 S. First, P.O. Box 696, Hillsboro, OR 97123, 503/640-2011.

```
time_t time1;

time(&time1);
```

FIGURE 1: EXAMPLES OF TIME-OF-DAY TO DEFINE STORAGE NEEDED

```
#if(0xffff + 1)
.
.
.
#endif
```

FIGURE 2: PREPROCESSOR DEFINES `int` AND SIGNED OR UNSIGNED CHARACTERISTICS OF A PROGRAM

```
case 1 .. 6:     a = b;
case 200 .. 204: a = c;
```

FIGURE 3: PROPOSED SYNTAX OF CASE RANGE ADDITIONS

```
case 1:
case 2:
case 3:
case 4:
case 5:
case 6:   a = b;
case 200:
case 201:
case 202:
case 203:
case 204: a = c:
```

FIGURE 4: POSSIBLE EXAMPLES OF CASE RANGE CODE

compile or not? If the preprocessor does its arithmetic in 32-bit blocks, the expression in the `if` is nonzero; if the arithmetic is 16 bits, then `if` has a zero value.

You might say this is a nice trick that allows the programmer to find out information not otherwise available. The problem comes when you use a C cross-compiler that uses the native preprocessor, leaving the programmer wondering which arithmetic to use, the target or host. The standards committee voted to make the arithmetic appear to be done in 32 bits. This allows all preprocessor answers to be the same, and the standard provides several `include` files that give programmers the information they need.

At the September meeting the committee voted 21 for, 5 against, and 4 undecided to add case ranges to the language. The proposed syntax is shown in Figure 3.

At the December meeting the committee voted case ranges out of the language. This shows that the committee can reverse its stand on issues and, more important, it shows that if people can make a good case for their position, they can have an effect on the committee. Personally, I was in favor of case ranges because I think that the code in Figure 3 is more readable than that in Figure 4.

Some of the committee members have a problem using the "`..`" operator from Ada to specify a range. When the compiler sees `1..5`, it could interpret this either as a range from 1 to 5 or the real number 1, followed by the real number .5. I think the issue of ranges in case statements is not dead yet and may just be waiting for the proper syntax. Anyone who would like to comment on whether case ranges are necessary (and who has suggestions for a good syntax) is welcome to contact me.

FORTRAN ISSUES

With the number of C users growing each day, it is not surprising that some users have come over from the FORTRAN camp to C. The committee has had some input from FORTRAN users concerning changes they would like to see in the C language that would make C a better replacement for their old favorite. It seems that FORTRAN users have two major problems with C. Their first objection is that the current C language requires all real-number arithmetic to be done at the double-precision level. They expect an expression done entirely with variables declared as float, to be evaluated using float precision. Second, C does not allow programmers

CALENDAR

Informix Relational Database: July 5–12, Cincinnati, Ohio. Contact Information Technology Development Corp. 513/741-8968.

Unix Concepts: July 8–10, Boston, Mass. Contact Intelligent Training Solution, 916/321-5300.

Software Engineering and Quality Assurance: July 8–12, Washington, D.C. Contact Dick White, 800/424-9773 or 202/676-8516.

Unix—A Hands-On Introduction: July 9–12, Los Angeles, Calif. Contact Ruth Dordick, ICS, 800/421-8166.

16-Bit Microprocessors: July 9–12, Washington, D.C. Contact Ruth Dordick, ICS, 800/421-8166.

Local-Area Network Seminar: July 10–12, San Francisco, Calif. Contact CSI, 408/725-1568.

World Computer Graphics: July 10–12, New York, N.Y. Contact World Computer Graphics Association, 202/775-9556.

Unix System Administration: July 11–12, Boston, Mass. Contact Intelligent Training Solution, 916/321-5300.

Unix Perspectives: July 14, Boston, Mass. Contact Information Technology Corp., 513/741-8968.

National Computer Conference of '85: July 15–18, Chicago, Ill. Contact Molly Finney, 703/620-8938.

Unix For End-Users: July 15–19, Cincinnati, Ohio. Contact Information Technology Development Corp., 513/741-8968.

Programming in C—A Hands-On Workshop: July 16–19, Los Angeles, Calif. Contact Ruth Dordick, ICS, 800/421-8166.

Introduction to Datacomm and Networks: July 16–19, Los Angeles, Calif. Contact Ruth Dordick, ICS, 800/421-8166.

IBM Futures Forum: July 18–19, Chicago, Ill. Contact Lynn Bentley, Gartner Group Inc., 203/967-6723.

To further the education of Unix systems users and professionals, UNIX/WORLD each month provides a calendar of events as a service to our readers. Please send information about your seminar, trade show, or course to Calendar Editor, UNIX/WORLD Magazine, 444 Castro St., Suite 1220, Mountain View, CA 94041. All items should include a title, a brief description of the course, seminar, or trade show, the date of the event, the city and state where it will be held, the name of the sponsoring company or organization, and an address and phone number for further information. Please submit all items at least four months in advance. □

```
a = (b * c) * (d * e);
```

FIGURE 5A: SAMPLE EXPRESSION FOR EVALUATION

```
1. b*c
2. d*e
3. (1) * (2)
```

FIGURE 5B: HOW EXPRESSION WOULD BE EVALUATED IN FORTRAN

```
temp1 = b * c;
temp2 = d * e;
a = temp1 * temp2;
```

FIGURE 6: FORCING EVALUATION ORDER WITH C

```
a = +(b * c) * +(d * e);
```

FIGURE 7: PROPOSED C SYNTAX FOR FORCING EVALUATION ORDER

to specify the order in which parts of an expression are to be evaluated; programmers must assign expression elements to temporary variables to ensure a specific order (see Figure 5A).

If the example in Figure 5 were in FORTRAN, you would know that the order of evaluation would be as shown in Figure 5B. In the C language, the order of evaluation is not known—the compiler is free to do the multiplication in any order it wishes. If C programmers are worried about overflow or underflow, they now must code the example as shown in Figure 6.

FORTRAN programmers say, "If C is so great, why doesn't it allow us to specify the order of evaluation?" To address this issue, the committee is trying to modify the C language in order to give programmers control over the order of evaluation—as well as the evaluation of all float expressions—so that all arithmetic is done using float precision. The current proposal for forcing the order of evaluation is to add a unary plus or minus to the expression, as shown in Figure 7. This tells the compiler that it must fully evaluate the expressions inside the scope of the unary operator before going on. I feel that the entire issue of helping FORTRAN programmers is going to elicit much comment from the current C programmer base—I expect there to be a lot of changes in this area before all parties are in agreement.

In future installments I will continue to bring UNIX/WORLD readers up to date on the developments at the X3J11 committee meetings, as well as on the overall progress of the proposed C standard. Until then, write and let me know your feelings on the issues I've covered, as well as any others you might feel are pertinent to the C standardization efforts. □

Steve A. Hersee is currently the vice president of marketing at Lattice Inc., a developer of the Lattice C compiler for the IBM PC and cross and native compilers for the Unix system. Mr. Hersee is a member of the ANSI X3J11 C standards committee. Readers can send their comments to Mr. Hersee at the following address: P.O. Box 3072, Glen Ellyn, IL 60138.

CAREER OPPORTUNITIES

Sales

Technical Support Analysts

Sun Microsystems, Inc., is a leader in the computer industry today, due to the development our general-purpose workstations.

We are looking for individuals to provide pre and post software sales support. You will need a strong knowledge of UNIX*, with internals experience (notably device drivers) strongly recommended. Experience with 68010-based workstations would be helpful.

We offer an excellent salary, comprehensive benefits and opportunities for personal recognition and professional growth. Please send a resume to: **District Sales Manager, Mail Stop #341, Sun Microsystems, Inc., 3803 E. Bayshore Road, Palo Alto, CA 94303.** An equal opportunity employer.

*UNIX is a trademark of Bell Laboratories.

sun microsystems

Readers who have computer merchandise or services to sell, buy, or trade and who want to be included as a classified advertiser in UNIX/WORLD Magazine should write to: Advertising Department, Tech Valley Publishing, 444 Castro Street, Suite 1220, Mountain View, CA 94041. 415/940-1500. The rate for the classified section is $12 per line. The deadline for ads is the 20th of the 2nd month prior to publication. Please include complete name and address in every ad.

UNIX C FLORIDA ——If you have 3 years experience in Unix C (Informix a plus), we need you. Join one of the nation's leading consulting firms. CGA Computer, Inc. has a consistent need for people like you. Top pay and excellent benefits package. Call collect (305) 857-8884. SUBCONTRACTORS WELCOME. CGA Computer, Inc. 7040 Lake Ellenor Drive, Suite 126, Orlando, FL 32809.

WMZ Consulting Group—System Consultants for total implementation of information systems. WMZ provides design, implementation, training, and ongoing support. We have experts in UNIX on staff and we are not committed to any software or hardware. We sell solutions not iron. Call (415) 945-2070, or write WMZ Consulting Group, 1483 Ashwood Drive, Martinez, CA 94553.

UNIX* JOBS REGISTRY

National registry of candidates and jobs in the Unix field. Please give us a call; send a resume; or request a free Resume Workbook & Career Planner. We are a professional employment firm managed by graduate engineers..

800-231-5920
P. O. Box 19949, Dept. UW
Houston, TX 77224
713-496-6100

Scientific Placement, Inc.

*Unix is a trademark of Bell Labs

Please circle Ad No. 126 on inquiry card.

PROGRAMMERS/ANALYSTS/DESIGNERS

UNIX*/C Opportunities In Central & Northern New Jersey

SDA, one of America's most sought after DP consulting firms, is seeking professionals with at least 2 years experience developing systems using UNIX*/C. We offer excellent salaries and top benefits. Relocation negotiable. Please CALL COLLECT (201) 687-5910 or send your resume and salary requirements to:

Sharon A. Diller
Manager, Human Resources

SDA SOFTWARE DESIGN ASSOCIATES, INC.
1620 Rt. 22, Union, NJ 07083
An Equal Opportunity Employer M/F/H/V
*UNIX is a trademark of ATT-Bell Labs.

Please circle Ad No. 12 on inquiry card.

TOWER™ POWER

Give your Tower more Power!

up to 1.2 gigabytes of disk and tape expansion capacity; fixed and removable disk drives.

- DIBOLIX™ DIBOL™ for UNIX™
- Personal Secretary™ word processor
- ***profit•maker***™ accounting system

Contact: SHA Computers, Inc.
RD#3, Box 51, Tait Road
Saratoga Springs, NY 12866
(518) 587-5886

Trademarks: Tower NCR Corporation; DIBOLIX C/DIBOL Ventures; UNIX AT&T; DIBOL Digital Equipment Corporation; Personal Secretary Finished Software; profit-maker SHA Computers Inc

Please circle Ad No. 54 on inquiry card.

AN ON-LINE CALCULATOR

BY BILL TUTHILL

In many ways, the arbitrary-precision calculator bc typifies Unix system software. To begin with, it is neither well known nor widely used, being described only in a research paper and not in beginning documentation. Second, it comes from Bell Labs and has been on every Unix system release since 1978; Berkeley gets no credit for it. Third, it is taciturn: There is no prompt, the only output being error messages and the answer. Its two-letter name has little mnemonic significance. Fourth, and perhaps most importantly, aside from the user interface, bc is superior to more glamorous software running on systems such as CP/M and MS-DOS.

All commercial desk calculators overflow at some point. You can verify this on your best calculator by continually multiplying a number by itself. Spreadsheet programs and BASIC interpreters, too, can only handle numbers of a certain size. The bc program, by contrast, does not overflow until memory is exhausted. In practice, two 500-digit numbers can be multiplied to produce a 1000-digit number in about 10 seconds. Division is equally fast and is accurate up to 99 decimal places. Aside from normal arithmetic, bc can calculate powers, square roots, and trigonometric functions. In fact, it is a complete programming language with a C-like syntax.

To begin using the calculator, just type bc after the shell prompt. Use "+" to add, "−" to subtract, "*" to multiply, and "/" to divide. Division normally yields no decimal places, but you can change this. The modulo operator "%" gives the remainder after division, and the exponentiation operator "^" takes a number to a power. To get back to the shell, type quit, as shown in Figure 1.

```
% bc
16 + 33
49
49 - 87
-38
1024 * 4
4096
100 / 3
33
100 % 3
1
2 ^ 32
4294967296
quit
%
```

FIGURE 1: SOME CALCULATIONS USING bc

As you can see, bc lops off decimal places after dividing. If you want more precision, set the scale to however many decimal places you want, as long as this value is less than 100. Figure 2 shows divisions and square roots when the scale is set to 5 and 99.

Generally, bc does not want to print more than 70 characters per line, so it prints a backslash and continues on the following line if there are more than 70 digits in the answer.

A CONVENIENT FEATURE

One convenient feature of bc is that it converts decimal numbers to octal or hexadecimal, and vice versa. Actually, it can convert any base to any other base and can perform

```
% bc
scale=5
100 / 3
33.33333
sqrt(2)
1.41421
scale=99
100 / 3
33.333333333333333333333333333333333333333333333333333333333333333333\
3333333333333333333333333333333
sqrt(2)
1.414213562373095048801688724209698078569671875376948073176679737990 73\
247846210703885038753432764157 2
```

FIGURE 2: DIVISION AND SQUARE ROOTS WITH THE SCALE SET TO 5 AND 99

```
% bc
obase=8
8
10
9
11
10
12
obase=16
10
A
11
B
12
C
13
D
14
E
15
F
16
10
```

FIGURE 3: CONVERTING DECIMAL NUMBERS TO OCTAL AND HEXADECIMAL NOTATION

```
% bc -l
a(1) * 4
3.14159265358979323844
define p() {
        return (a(1) * 4)
}
p()
3.14159265358979323844
```

FIGURE 4: A FUNCTION TO CALCULATE PI BY MULTIPLYING THE ARCTANGENT OF ONE BY FOUR

```
define f(n) {
        auto i, x

        for (i = x = 1; i <= n; i++) {
                x *= i
        }
        return(x)
}
```

FIGURE 5: A FUNCTION THAT COMPUTES *N* FACTORIAL

```
bc() {
/usr/bin/bc $HOME/.bcinit
}
```

FIGURE 6: CHANGING `bc` TO YIELD GREATER DECIMAL PRECISION FOR THOSE USING THE BOURNE SHELL ON SYSTEM V.2

arithmetic in bases besides decimal. The built-in variable `obase` sets the output base, whereas `ibase` sets the input base. Figure 3 shows how you can convert decimal numbers to octal and then to hexadecimal notation.

Programmers find it helpful to be able to convert octal and hexadecimal numbers into decimal notation. You can do this by setting `ibase` to 8 and 16, respectively. You can also convert binary numbers into decimal by setting `ibase` to 2.

The `-l` option loads a math library that includes functions for calculating sine, cosine, arctangent, natural logarithm, the exponential, and Bessel functions of integer order. For example, it is always possible to calculate pi by multiplying the arctangent of one by four. You can even package this up as a function, as shown in Figure 4.

Because the `-l` math library sets the `scale` to 20, the function `p()` above returns pi to 20 decimal places. More complicated function definitions are possible, using C language control-flow constructs such as the `for` loop. Functions need not involve the math library. Figure 5 presents a function that computes *N* factorial.

Most of the time, users want more decimal precision than `bc` gives by default. Fortunately, because `bc` reads the file specified as a command-line argument before reading from the terminal, you can easily customize `bc` to yield greater decimal precision. First, place the following line in a file named `.bcinit` in your home directory: `scale=5`. If you're using the C shell, create the following alias: `alias bc bc ~/.bcinit`. If you're using the Bourne shell on System V.2, create the function shown in Figure 6.

After you have made this change, division will always yield five decimal places of precision. If you want fewer or more decimal places, just change the value of `scale` interactively. □

Bill Tuthill is a member of the technical staff at Sun Microsystems, Mountain View, Calif. He was previously a systems analyst at Imagen Corp. and a programmer at UC Berkeley.

THE EFFECTS

The problems that hard-core computer nerds might face as they grow older are pretty obvious. When people have spent most of their time in front of the computer CRT/keyboard, they won't have had time to learn how to deal effectively with people, both in working and personal relationships. Sooner or later, such deficiencies will make themselves known, with unhappy results almost a certainty.

But there is another issue as well. Not only do such deficiencies affect the individuals themselves, they often affect society at large. There are a variety of ways in which this can occur. Persons who have not learned how to interact effectively with others may have a difficult time in most working environments, where effective teamwork (even when the task is programming!) is frequently of crucial importance. Not everyone working with computers has the sometimes dubious privilege of being his own boss and not having to interact closely with co-workers.

One might also wonder about the "ethical" sense that develops within people who have most of their primary interactions with computers rather than people. The twisted "ethics" of some computer "crackers," who get their kicks from breaking into other people's systems but who don't think twice about the rights of those people, might be one example. There are a variety of other scenarios that could be brought up as well, but let's not belabor the point here—you get the idea.

WHAT TO DO?

At this juncture, I'm supposed to come up with some brilliant solution for the problems I've outlined. Well, it's not terribly brilliant, but at least one aspect of a way to help avoid some of the problems does seem pretty clear—it's pretty simple, too. There's no reason to discourage children from learning about, enjoying, and spending time with computers. But efforts should be made to detect when children are beginning to engage in such activities to the exclusion of most other extracurricular and interpersonal behavior.

If such a situation is detected, it's important that adults help to steer such children in the direction of activities that will require some more significant person-to-person interaction than sitting at the computer can provide. Certainly we don't want to exclude the computer from the palette of children's free-time choices; it can indeed be a useful way to spend some of their time. But neither should we allow them to become so totally absorbed by this one activity that they exclude other useful activities that can also contribute to their intellectual and social development.

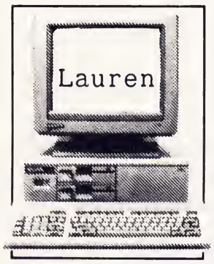
We want children to become computer-aware, and we certainly should help encourage their interest if they show particular aptitude toward working with computers. But by helping them to avoid becoming *totally* oriented toward such technology, we are doing both them, and ourselves, a favor in the long run. Do we need more computer nerds? I think not. □

```
--Lauren--
UUCP:
{decvax, ihnp4,
seismo, clyde,
bonnie} !vortex!
lauren
```

Lauren Weinstein is a computer/telecommunications consultant living in Los Angeles. He has been involved in an array of projects that range from the mundane to the bizarre. He has particular expertise in the fields of computer networking, the Unix system, microcomputer technology, and telecommunications systems.

THE NEW COMPUTER NERDS

BY LAUREN WEINSTEIN



Beware! A new generation of computer "nerds" is on the way. Many have already arrived, and their numbers will be increasing rapidly in the near future. It's not a laughing matter—it could affect our lives and culture in ways we would never imagine, many of which may well be quite negative.

The first important point to realize is that you can't necessarily identify computer "nerds" by their looks. Well, not always, anyway. Of course, we're all familiar with the popularized image of the (virtually always male) computer nerd, so there's no point in describing him here. To be sure, such images were based on a certain archetypical variety of person who indeed did (and does) exist. But don't be misled into believing that *all* computer nerds can be so easily pinpointed.

In reality, computer nerds are ultimately identified not by the way they look, but by their actions and value systems. To my mind, a computer nerd is a person whose primary focus of life, including both work and play, both job time and spare time, focuses on computers to the exclusion of virtually all else, particularly other people.

Computer nerds, by this definition, can be of either sex (though males still greatly predominate) and may in some cases show none of the appearance-related traits that have come to be associated with such personalities.

We can find computer nerds of widely varying ages, but until fairly recently the characteristics of computer nerdism didn't extend back to very young children. This is completely understandable since it is only in recent years that computers have become available to many younger people. Individuals of my generation didn't usually have any significant direct access to computers until we reached college.

That's all changed now, of course. Comparatively sophisticated computers can be found even in the smallest of primary schools, not to mention elementary/secondary schools, colleges, and universities. Children are exposed to these computers from the very earliest ages, and that doesn't even include the children whose parents have bought computers for the home and encourage their young children to play with the units.

It is from the ranks of the young children, the children who are growing up with computers practically from birth, that the new generations of computer nerds will evolve. And just as there are incredibly more computers available now than, say, 10 or 15 years ago, so there will be more computer nerds as well, as I defined them above.

THE PROBLEM

Well, why should we be concerned about computer nerds? Shouldn't we just let them alone to live as they wish and let them merrily type their lives away? No, we should not. When one's life is totally absorbed with machines, particularly complex machines like computers, it is very difficult, perhaps impossible in many cases, to develop a healthy understanding of, and respect for, people. And like it or not, we live in a world not only of machines, but of people as well!

The handwriting is already on the wall . . . er . . . the characters are already on the screen. Talk to many school teachers and you'll hear stories of youngsters who spend *all* their free time clustered around the classroom computers. After school, you can often find these same children over at the nearby Radio Shack or similar establishment, where they spend the remainder of the afternoon with the computers down there! When asked, many of these children will tell you that "their best friend" is the computer.

I'm not saying that it would be better for these kids to be in street gangs, but I *am* suggesting that for many of these children, their single-minded infatuation with computers could well lead toward significant personal problems down the line, problems that could last indefinitely into adulthood. Remember, children can now get "hooked" by computers at an incredibly young age. And the machines can indeed be seductive. They can provide a sense of power over another (albeit nonhuman) entity. They can be manipulated and controlled, and (most of the time, anyway) they don't threaten you or place you in "uncomfortable" situations, unlike the sorts of things that can arise when dealing with other children out on the playground.

In many cases, teachers and parents are perfectly thrilled to see their children spending so much time in what is viewed as a "productive" manner. Often they don't seem to realize that even a "hobby" with so much potential for constructive learning can be something of a narcotic when it blocks out all else, and it's all the more powerful for being introduced at an early age.