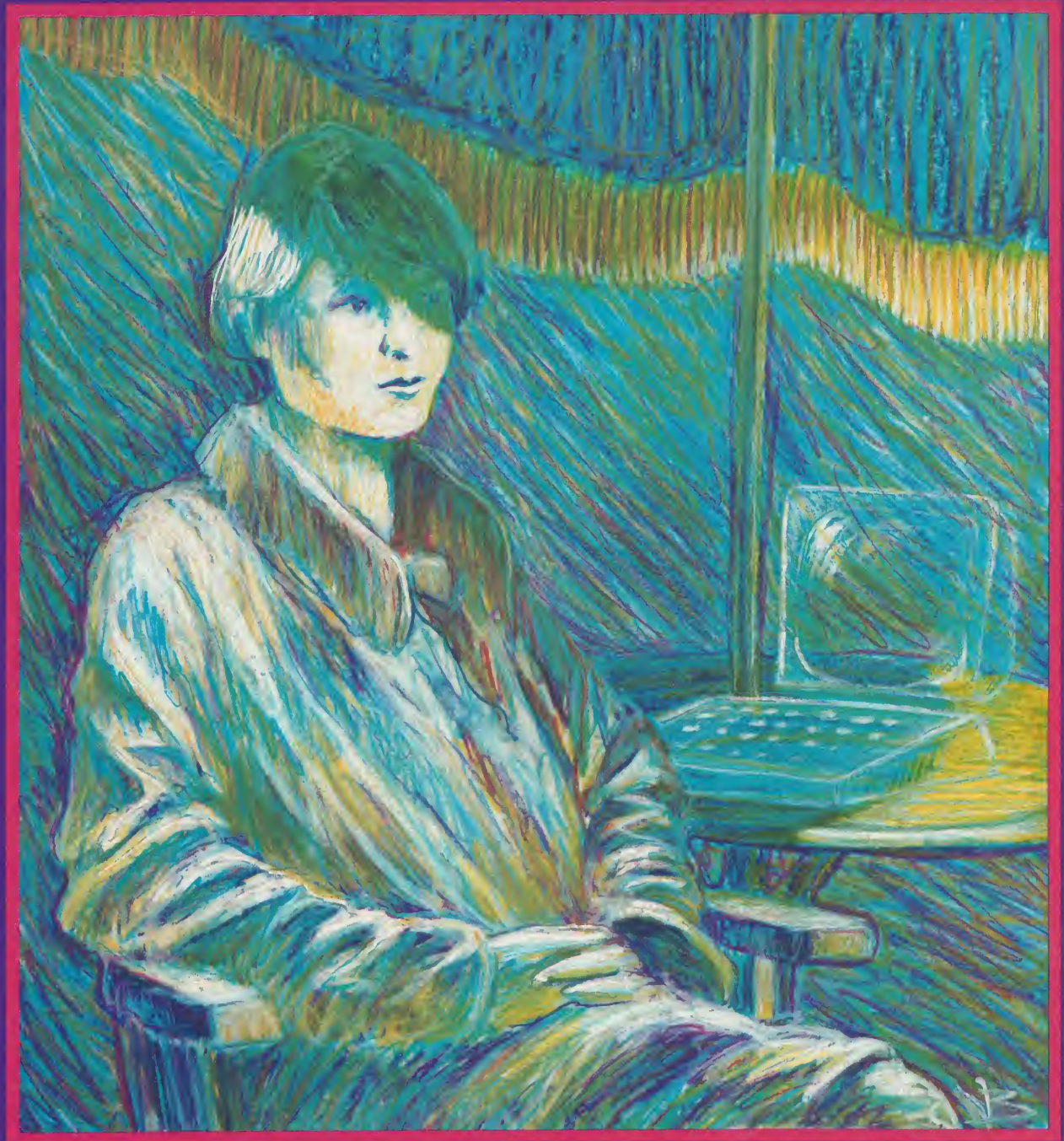


UNIX™ REVIEW

THE PUBLICATION FOR THE UNIX™ COMMUNITY

November 1984 \$3.95



USER FRIENDLY UNIX

UNIX RELATIONAL DATA BASE MANAGEMENT SYSTEM

UNIFY
CORPORATION

UNIX RELATIONAL
DATA BASE
MANAGEMENT
SYSTEM

REFERENCE
MANUAL

UNIX RELATIONAL
DATA BASE
MANAGEMENT
SYSTEM

TUTORIAL
MANUAL



UNIFY FINISHES FIRST!

In one independent competition after another, UNIFY has proved itself the fastest UNIX data base management system. No wonder it's been selected by more computer manufacturers than all other UNIX data bases combined.

UNIFY speeds you through development and expedites program execution with some of the most powerful utilities of all, including:

Fully menu-driven design.

A powerful screen handling package that helps you format screens quickly, with no coding required.

Raw I/O, that lets you bypass the UNIX file system for up to 40% faster performance in large data bases.

Built-in optimizers that select the fastest of four data access methods.

Industry standard IBM SQL query language, plus our powerful report writer, for easy access by end-users.

Ninety subroutines for advanced program development... the most complete package of its kind.

UNIFY's integrated design links program modules like screens, query language and report writer to help you quickly create complete, friendly, easily expandable applications.

Horsepower for the long run.

Unlike other data bases, UNIFY won't slow down under the weight of additional data or multiple users. It's built with the power to support new features later.

Judge for yourself.

Send for our 300-page tutorial and 500-page reference manual—yours for only \$95—that show you how to build virtually any application. Contact UNIFY, Dept. MMS-11, 4000 Kruse Way Place, Bldg. Two, Suite #255, Lake Oswego, OR 97034, (503) 635-6265, TELEX 469220.

UNIFY[®]
THE PREFERRED DBMS.

Circle No. 1 on Inquiry Card



UNIX™ REVIEW

THE PUBLICATION FOR THE UNIX COMMUNITY

Volume 2,

Number 8

November 1984

DEPARTMENTS:

- 4 **Viewpoint**
- 8 **Devil's Advocate**
by Stan Kelly-Bootle
- 14 **The Human Factor**
by Richard Morin
- 60 **Industry Insider**
by Mark G. Sobell
- 63 **C Advisor**
by Bill Tuthill
- 70 **Rules of the Game**
by Glenn Groenewold
- 76 **/usr/lib**
by Jim Joyce
- 80 **Problem Solver**
by Bob Toxen
- 98 **Recent Releases**
- 104 **Calendar**
- 108 **Advertiser's Index**

Cover art by Victor von Beck

FEATURES:

- 20 **BEYOND USER FRIENDLY**
by Jeff Schriebman
A sensible suggestion for a substitute for the hackneyed "user friendly" standard.



- 23 **DEVELOPING SYSTEM FRIENDLY USERS**
by Jim Joyce
Ways in which users can make better use of UNIX capabilities.



- 28 **SYSTEM FRIENDLY USERS**
by Jim Joyce
A rundown of the best UNIX books on the market.



USER FRIENDLY UNIX

32 MENUS: THEIR USE, ABUSE and DESIGN

by Jordan J. Mattson

Arguments for well-tempered menu-driven systems.



38 INTERVIEW WITH DON NORMAN

by Mark Compton

The system's most outspoken critic gives UNIX a right proper roasting.



51 COMPUTER AIDED INSTRUCTION

by Vanessa Schnatmeier

Three different UNIX proposals for putting courseware on the right course.



57 3B2 BENCHMARKS REVISITED

by Gene Dronek

A corrected version of earlier test results.

PUBLISHER:

Pamela J. McKee

EDITOR:

Mark Compton

ASSISTANT EDITOR:

David Chandler

ASSOCIATE EDITORS:

Ken Roberts, Scott Robin

PRODUCTION DIRECTOR:

Nancy Jorgensen

PRODUCTION STAFF:James Allen, Dan Arthur, Tom Burrill,
Cynthia Grant, Corey Nelson,
Florence O'Brien, Denise Wertzler,
S.T. Wilson**CIRCULATION/DEALER SALES:**

Tracey McKee, Barbara Perry

BUSINESS MANAGER:

Ron King

ADVERTISING**REPRESENTATIVES:**Jules E. Thompson, Inc.
1290 Howard Avenue Suite 303
Burlingame, CA 94010Lucille Dennis 415/348-8222
303/595-9299 - Colorado
312/726-6047 - Illinois
617/720-1888 - Massachusetts
713/731-2605 - TexasJules E. Thompson, Inc.
2560 Via Tejon
Palos Verdes Estates, CA 90274
Ed Winchell 213/378-8361
212/772-0933 - New York**TELE-TYPESETTING:**

Data & Staff Service Company

PRINTING:

Thomas Ogle

EDITORIAL ADVISORS:Dr. Stephen R. Bourne, Software
Engineer, Digital Equipment
Corp.Jim Joyce, President, International
Technical Seminars, Inc.**REVIEW BOARD:**Dr. Greg Chesson, Technical Staff,
Silicon Graphics, Inc.Larry Crume, Director, AT&T UNIX
Systems Far EastTed Dolotta, Senior Vice President
of Technology, Interactive
Systems CorporationGene Dronek, Director of Software,
Aim TechnologyDavid Fiedler, President, InfoPro
SystemsGeorge Goble, Systems Engineer,
Purdue UniversityBill Joy, Vice President of Research
and Development, Sun
MicrosystemsJohn Mashey, Software
Engineering Manager,
Convergent TechnologiesRobert Mitze, Department Head,
UNIX Computing System
Development, AT&T Bell LabsDeborah Scherrer, Computer
Scientist, Mt. XinuJeff Schriebman, President, UniSoft
SystemsOtis Wilson, Manager, Software
Sales and Marketing, AT&T
Technology SystemsWalter Zintz, Executive Director,
Uni-Ops**CONTRIBUTING EDITOR:**Ned Peirce, Systems Analyst, Bell
Laboratories

VIEWPOINT

What's in a name?

I can half imagine the snickers as people pick up a magazine emblazoned with the brash headline, "User Friendly UNIX."

The strengths of UNIX are many, but the system's user interface has rarely been counted among them. Compare unadorned UNIX, for instance, with what has come to be regarded as the standard for "user friendliness" — the Apple Macintosh. The difference could hardly be more striking.

Does that mean UNIX is unfriendly? It depends on how you view the term. Richard Morin (*The Human Factor*) calls "user friendly" a "clear contender for the title of "Most Hackneyed Slogan of the Year." I quite agree.

So before addressing the question of whether UNIX is user friendly or not, we decided it would be best to start off with a definition of the term. Jeff Schriebman, president of UniSoft Systems and a member of the UNIX REVIEW Editorial board, agreed to tackle the unenviable task. His eloquent proposal could serve well as an industry standard.

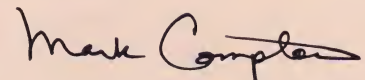
Jim Joyce follows with an assessment of the other side of the equation: system friendly users. "Instead of fighting the system, the user can convince the system to adapt in terms both can understand," he writes. The article goes on to say how.

"Well-tempered" menus for soothing the UNIX beast are the focus of a piece by Jordan J. Mattson, a UC Santa Cruz UNIX programmer specializing in menu-

driven office automation systems.

The "User Friendly UNIX" theme then concludes with a spicy interview with Don Norman, the outspoken UNIX critic who took the system's user interface to task in a celebrated 1981 article in *Datamation* titled "The Trouble with UNIX." Some valid questions — and not a few hackles — were raised by the original article. Chances are this interview will do likewise.

Should this issue of UNIX REVIEW leave unanswered questions, so be it. It's long past time that serious questions be raised about the UNIX user interface. General acceptance of the system depends on it.

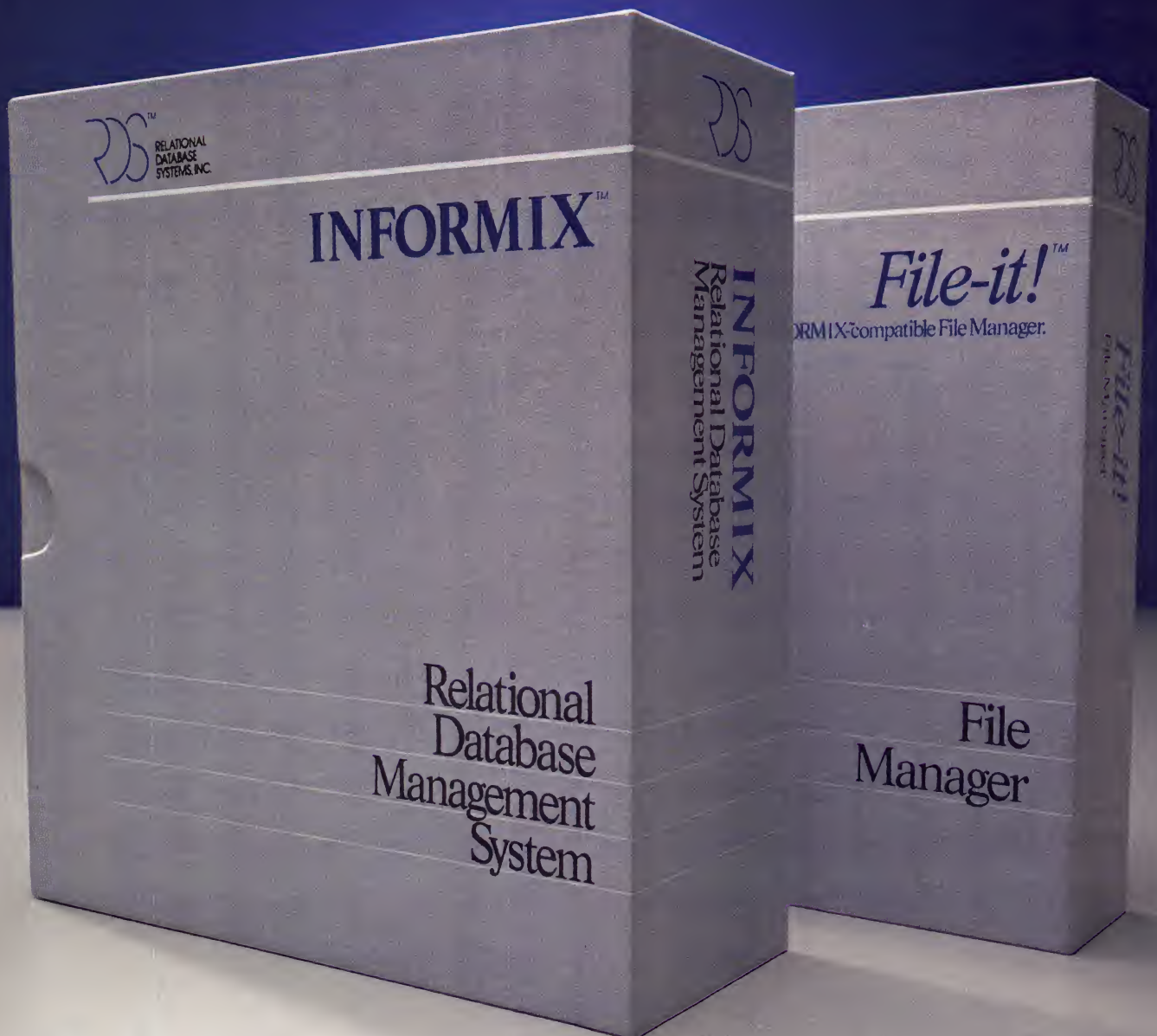


**YOU CAN'T
PREDICT
THE FUTURE.**

**BUT
YOU CAN BE
PREPARED
FOR IT.**

WITH INFORMIX AND File-it!

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



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

RDS.

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

Incompatibility is a thing of the past.

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

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

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

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

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

Simplify program development.

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

Software good enough for AT&T.

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

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

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

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

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

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

*RDS products are available for the following systems:

Altos 586, 986, 8600, 68000	Hewlett Packard 150, 9000
Apollo DN300	Series 200, 9000 Series 500
AT&T 3B2, 3B5, 3B20,	IBM PC, PC-AT, PC-XT
AT&T Personal Computer	Intel System 86/380, 286/310
BBN C machine (all models)	Masscomp NC 500
Bunker Ramo Aladdin 20	Momentum Hawk 32
Charles River Data Systems	NCR Tower
Universe 68	Onyx C8002, C8002A
Convergent Technologies	Pacific Micro Systems PM200
MiniFrame and MegaFrame	Perkin-Elmer 32 Series, 7350
Corvus Systems Uniplex	Pixel 100/AP, 80 Supermicro
Cromemco System 1	Plexus P/25, P/35, P/40, P/60
DEC 11/23, 11/34, 11/44,	Pyramid Technology 90X
11/60, 11/70, VAX 11/730,	Radio Shack Model 16
11/750, 11/780	SCI Systems IN/ix
Dual Systems System 83	Silicon Graphics IRIS 1400
Fortune 32:16	Visual Technology 2000
Forward Technology 320	Wicat Systems
General Automation Zebra	Zilog System 8000
(all models)	(all models)

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



**RELATIONAL
DATABASE
SYSTEMS, INC.**

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

INFORMIX is a registered trademark of Relational Database Systems, Inc. RDS, File-it! and C-ISAM are trademarks of Relational Database Systems, Inc. UNIX is a trademark of AT&T Bell Laboratories. MS is a trademark of Microsoft and PC-DOS is a trademark of International Business Machines.

See us at

COMDEX™/Fall '84

November 14-18, 1984

MGM Grand Hotel
Las Vegas, Nevada

Circle No. 2 on Inquiry Card

DEVIL'S ADVOCATE

Bootez Le Système, Ma'm Cloche! Je Veux Grepper Mall!

by Stan Kelly-Bootle

Most Fifth Generation watchers limit their watching to the two pennant-chasing favorites, Japan and the United States. My last trip to Paris, and subsequent study of the literature, have convinced me that there is, *sans blague* and *Sacre bleu*, a third serious contender, a *cheval noir* lurking behind *les barricades* and straining at the leash. *Oui*, indeed, the French are on the old "Marchons! Marchons!" again. And when *les grenouilles* get going, *Prenez Garde, Mes Enfants! Zair weel be much rolling of ze tête, hein?, and conseederable spillage of ze sang impur*. The new revolutionary songs echo the glorious "Liberté, Egalité, Fraternité" of the young *République*. Freedom from *Le Grande Bleu!* (*Le Perfide Oncle Sam* has replaced perfidious Albion.) Equality with *Le Soleil Levant*. Fraternity with *L'Intelligence Artificielle*.

Developing the hardware (*matériel*) and software (*logiciel*) to provide the long-awaited dawn of intelligent AI is, of course, the kernel of current Fifth Generation research. Japan's strengths in the areas of robotics, IC technology and union-bashing are well known, but many feel, seriously offset by traditional weaknesses in software. The US has unequalled pools of talent, nay, genius, in all the relevant fields. The pools, however, compete rather than



cooperate, in business, military and academic domains. In contrast, the Japanese psyche seems to encourage a unity of purpose where the pools combine to form *tsunamis* (a word I have been dying to use for some time), and the Fifth Generation computer has emerged as a national priority. Where does this leave the French?

My view is that the French combine a Japanese national fervor with an American flair for intellectual individualism. The French government has been pursuing a vigorous *hitech* policy, especially in *l'informatique*, for many years; it is not widely appreciated in this country that both the *PTT* (the French equivalent of AT&T, loosely speaking) and the *BNP* (*Banque Nationale de Paris* — which one cannot easily translate, since it is wealthier than any American Bank) together with the

major electronics and computer companies are either completely or partly nationalized concerns. While the conservative Maggie Thatcher across *La Manche* has been dismantling and "privatizing" the companies which were nationalized under the Labour parliament of 1945, the socialist Mitterand has diverted a considerable portion of his GNP toward strengthening the public sectors of the French economy, with a special emphasis on computers and automation.

Paris has become a user friendly base for many distinguished international computer scientists, including Seymour Papert. Signs of intense, high-level computation can be seen in all walks of life. *Autour de l'horloge*, from *Réseaux de Temps Partagé* in *les bistros*, to *ordinateurs portables* for *les belles de nuit*, *les claviers* are going *clique, clique*, and *les curseurs pétillants* flash on *les écrans de moniteur*. On the home front, the standards of the government-controlled TV networks have been deliberately lowered to encourage the use of Teletext and Telefax.

The outstanding symbol of this French technological renaissance is the Pompidou Center in Paris where the arts and sciences merge with no sign of mutual disdain or embarrassment. The building itself looks as



"It will now," said Alis.

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

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

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

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

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

Alis™

The next-generation office software system from APPLIX

Finally, some answers in Wonderland.

APPLIX, INC. 302 TURNPIKE ROAD, SOUTHBORO, MASSACHUSETTS 01772 (617) 481-4721

Circle No. 3 on Inquiry Card

though a large truck, taking an oil-refinery to Saudi Arabia, had broken down in the heart of the old city. Unable to budge it, the Mayor must have then commissioned Matisse and Chagall to spruce up the pipework in their favorite primary colors. Walking through narrow 17th century streets, drinking, breathing and smelling the ambience of *l'Ancien Régime*, then turning a corner to confront *Le Monstre* is one of life's most pleasantly enduring *chocs*. To the credit of the Parisians, the Pompidou Center was not stormed *à la Bastille* but, rather invaded by joyous mobs anxious to enjoy its facilities.

I saw thousands of natives of all ages jostling to get some *de première main* on the terminals and PCs which are freely and openly available. And they were not playing *Kong d'Ane* or *La Guerre des Etoiles*. Neither were they engrossed in a trivial *programme d'une seule ligne en BASIC*. Gnarled *paysan* and brash *étudiant* alike were keying in massive jobs in *Assembleur* and *ZOZOT* (LISP). LISP has become so widely used that there is now a national shortage of parentheses; a good *crochet d'occasion* will fetch 50 *francs* on the *marché noir*. Why not switch to Pascal? Well, *Monsieur B. Pascal was French, sans aucun doute, but Herr N. Wirth? Jamais!*

The well-known French xenophobia in matters linguistic is turning out to be an advantage in the AI game. It's all very well pursuing *les compilateurs des langues naturelles* with abstract theories, but sooner or later you must *mettez l'argent où se trouve la bouche* ("put your money where your mouth is").

Practical progress in Natural Language processing, according to the French, presupposes that you have a *natural* language to begin with, and that furthermore

you *keep* it that way! The Anglo and Nippophone communities are at a serious disadvantage when it comes to language purification schemes. The Japanese language, let's face it, is a typographical disaster area straining under the burden of innumerable sociolinguistic anachronisms. Their software problems, even with the

**The French combine a
Japanese national
fervor with an
American flair for
intellectual
individualism.**

normal high-level computer languages, have been well documented. Rather than simply "lomanizing" their scripts, the Japanese really need a drastic clean sweep. A good start, some say, would be switching to Finnish or Basque, then gradually moving through Proto-Indo-European to Latin, then on to French. I predict some objections to this plan, not the least being that it could delay the Japanese Fifth Generation by approximately 5000 years. But who said it was going to be easy? Certainly not me, and not, with much more authority, Hubert L. Dreyfus (*What Computers Can't Do*, Harper Colophon Books, 1979).

The English language does not have the same Japanese structural dysfunctions, but much worse in French eyes, it has degenerated into a shapeless, undisciplined pidgenese, a mere trade language like Mobilian, bor-

rowing and lending, shedding all semblance of grammar and good taste in order to curry favor — a *lingua franca* for the World's illiterate. And all this despite valiant attempts circa 1066 to bring some Gallic precision to the Anglo-Saxons.

The French view their linguistic heritage much as the Sierra Club views Mono Lake. Your average Californian, for example, will march on even days to save the darter snail, but on odd days the parade is to promote multilingual ballots. The five minority languages in France have never received any such encouragement from the French-speaking majority.

In the US, we have a few William Safires preaching gooder English in the wilderness; in France, on the other hand, there are 40 or so Safidic watchdogs, known as *L'Académie française*. They meet regularly under governmental auspices and have extensive powers to enforce *le mot juste*. The main enemy is *franglais*, the insidious use on an English word where a perfectly good French one is about to be invented! A ministerial scribe can be fined, in fact, for using *le floppy* when *le disque souple* is clearly indicated. A common trap for the unwary is "computeur," which gets you five years on *L'Ile du Diable* (Devil's Island). In France, your *données* (data) is put in order by *l'ordinateur*.

My money is on the nation that gave us Descartes, Pascal, Fermat, Laplace, Poincaré, Marat-Sade and my wife! I urge you all to pull out those dusty French schoolbooks, brush up on your Voltaire, and practice with me the paradigms of *La Nouvelle Vague*:

*J'ai perdu le registre de
ma Tante dans le fichier de
mon Oncle.*

(I have lost the record of my

CMC COMPUTER METHODS LIMITED



- Interactive Spelling
- Balance Sheet Mathematics
- Auto Footnoting
- Auto Outlining
- Phrase Glossary
- Template Assembly
- Photo Typesetter Interface
- Widow-Orphan Elimination
- Pagination
- Form Generation
- Electronic Mail
- Data Entry
- Financial Analysis

DOCUMENT PROCESSING SYSTEMS

Box 709 Chatsworth, CA 91311 (818) 884-2000 TWX910-494-1716 Int'l Tlx 292-662 XED UR

Aunt in the file of my Uncle).

Il y a une erreur dans ce boitier.

(There is a bug in this package.)

Furthermore, remember to pronounce UNIX as "oooo-neeeks," which, once and for all, will prevent those unfortunate snickers when your neighbor asks you to recommend a good *SO* (*Système Opérateur!*).

UNIX, in any language, continues to lead the way in the search for expert systems and machine intelligence. Here are two examples which I gave at the last Uni-Ops Conference, revealing that UNIX knows more than you imagined in the disparate fields of US Aviation and Labor History. I offer them *en français*, to see if you have been paying attention:

*chat amelia_earhardt
amelia_earhart pas
trouvée*

*chat jimmy_hoffa
jimmy_hoffa pas trouvé*

Glossary

En français — In French

Kong d'Ane — Donkey Kong

La Guerre des Etoiles — Star Wars

De première main — Hands on (at first hand)

Pas trouvé — Not found (masculine)

Pas trouvée — Not found (feminine)

chat — cat

Ma'm Cloche — Ma Bell

Ma Belle — Ma Bell before the breakup

Je veux grepper mal — I'm dying for a good grep

Crochet d'occasion — Second-hand bracket (parenthesis)

Le Perfide Oncle Sam has replaced perfidious Albion.

Sang impur — Foreign blood

Franc — *Quatre sous*, and falling daily

Les belles de nuit — Third shift ladies

Réseau — Network

Temps partagé — Time sharing (the French pioneer was Proust, see his *A La Recherche du Temps Partagé*)

Ordinateur portable — Portable computer

La Nouvelle Vague — The New Wave

Bistro — Fast drink shop

Autour de l'horloge — Around the clock

Le Grand Bleu — IBM

La Manche — The English Channel (in spite of many attempts at conversion)

Oui — Yes

Jamais — Never

Sans blague — No kidding! Honest Injun!

Sans aucun doute — Indubitably

Sacre bleu — Holy IBM!

Les grenouilles — The Frogs (derogatory)

La perfide Albion — The Wasps (derogatory)

Marché noir — Black market

Cheval noir — Dark horse

Le Soleil Levant — The Rising Sun (in this context, Japan — not the House in New Orleans)

L'Informatique — Post-graduate computer science

Le Hitech — Any Renault built after 1935

IA — AI

SO — OS

PTT — Postes Telegraphes Telephones

Prenez Garde, Mes Enfants! — Steady on, chaps!

Ecran de moniteur — Monitor screen

Clavier — Keyboard

Curseur pétillant — Flashing cursor

Programme d'une seule ligne — One-line program

Le mot juste — 32 bits

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

YOU can print your
XROFF manuals, proposals, forms...

right in your own shop!

Use **your** computer and our software (**XROFF**) and fonts (we have 100's) with the laser printer, typesetter, inkjet or dot matrix printer of your choice. We can provide whatever equipment you don't already have, at a price less than you would think.

Call or write for more information:

Image Network

770 Mahogany Lane, Sunnyvale CA 94806 (408) 746-3754

This ad was set using XROFF on a Xerox 2700 laser printer

The Best of Both Worlds.

If you think that you can't have your cake and eat it too, then you better take a closer look at what Convergent Technologies has to offer. At Convergent Technologies, we offer computer professionals the "Best of Both Worlds."



We know that computer professionals want the freedom to turn ideas into products that get quickly shipped out of the door, so generally, our products ship less than 1 year from conception.



We also know that computer professionals do not want the uncertainty of seeing development money drying up, projects getting scrapped or the continuing industry shake-out to start affecting them, so that is why we provide qualified people with all of the necessary responsibility and resources to continue to build technically aggressive products.

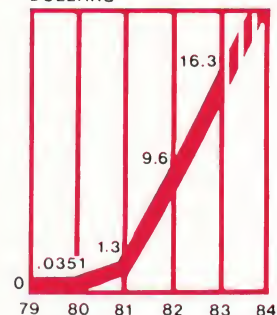


In the 5 years since our founding, we have brought 8 major products to the market and that makes us one of the fastest growing publicly held companies in California . . . but our results speak for themselves. So if you want the "Best of Both Worlds", consider the following opportunities:

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

Most available positions call for a minimum of 2 years of software development experience and a mini/micro background. BS/CS or MS/CS preferred. In addition to a competitive compensation package, we offer significant project and equity participation in a fast product turnaround environment, plus a 13 week sabbatical after just 5 years. Call Joy Mar at (408) 945-6971, or send your resume to Convergent Technologies, Dept. ur1, 30 East Plumeria, San Jose 95134. We are an Equal Opportunity Employer. Principals Only, Please.

IN TENS OF MILLIONS OF DOLLARS



Convergent Technologies

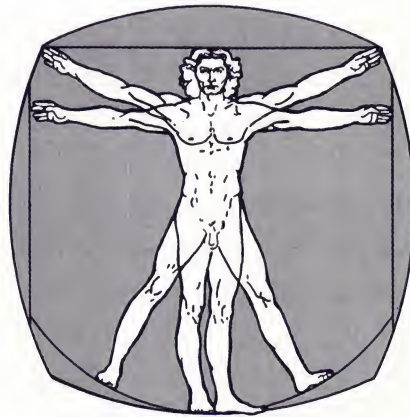
THE HUMAN FACTOR

Whaddya mean, user friendly?

by Richard Morin

"If the point of contact between the product and the people becomes a point of friction, then the industrial designer has failed. If, on the other hand, people are made safer, more comfortable, more eager to purchase, more efficient — or just plain happier — the designer has succeeded."

Henry Dreyfuss (Designing For People, Simon and Schuster, 1955)



equally well, and their performance on seemingly equivalent jobs can be radically different.

Certain common limitations and other traits are responsible for this variation. The workplace limitations promulgated by OSHA generally reflect physical limits: size and weight of objects to be handled, levels of ambient noise or lighting that can be tolerated, and any number of other considerations. The newer cash registers further reflect mental limitations by performing calculations for tax, change returned and so forth.

When limitations affect only part of the population, a designer must decide whether to deal with them. The design of a control console, for instance, must take into account the fact that about 10 percent of males are color blind, since the use of color as an indicator could easily be missed by such an operator. The design of a color television set, on the other hand, does not have such constraints, save that it should present an acceptable picture to a color blind viewer.

Some limitations are more specific to the world of computers. In general, humans are very good at pattern recognition, but poor at calculation. They are terrific at relating loosely connected factors, and terrible at considering all possible related factors. They can handle a chart or graph with ease,

"User friendly" is a clear contender for the title of "Most Hackneyed Slogan of the Year." No computer-related product can come out these days without a banner proclaiming how !!!USER FRIENDLY!!! it is.

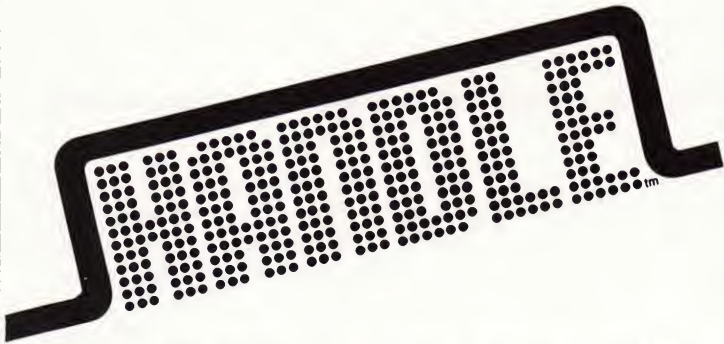
The "user friendly" label is the Madison Avenue way of saying that a product has been designed properly in terms of human factors. Only a few years ago, local technical bookstores contained almost nothing worth mentioning on human factors in computing. Now though, the topic competes with introductory UNIX for shelf space.

Your author has very mixed feelings about this. It's nice to be noticed, of course, but things are getting lost in translation. There is a set of real issues to be considered, and a set of valid criteria to be used in designing products

for use by human beings. A "user friendly" designation means nothing without an accompanying human factors analysis. In the name of enlightened consumerism, then, let's look at what is meant by human factors, and see how such an analysis should be done.

HUMANS AREN'T STANDARDIZED

Human beings are remarkably adaptable and variable creatures. They can crawl into tight spaces, use varying amounts of light, perform complicated sequences, and in general do things that a designer of robots would be hard put to replicate. Humans vary, however, in strength, speed, intelligence, tastes and so on. They cannot perform all jobs



Office Automation Software . . . Selected for Distribution by AT&T.

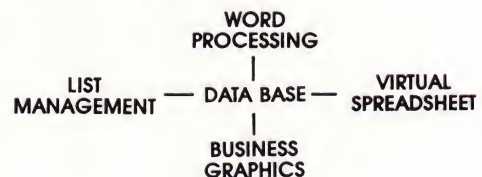
HANDLE is the number one office automation or "integrated" software system available for the UNIX™ operating system. Selected by **AT&T** Information Services for distribution on the 3B2/300™ and 3B5™ UNIX systems, **HANDLE** is the most powerful full-featured office automation system in the market today.

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

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



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



The first Office Automation System for UNIX.

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

Circle No. 6 on Inquiry Card

but often balk at a table of values or at a mathematical function.

In designing a system or simply choosing equipment, one should try to keep users from having to do things that they do poorly. This may be accomplished by having the equipment do more work. Alternatively, it may result from a simple restructuring of the task.

In a research and development environment, the nature of the work to be performed is usually not well known. Support facilities for such an environment should therefore be quite flexible. In a production environment, the tasks to be performed are more predictable. A thorough study of such tasks can often result in dramatic improvements in performance.

WHO IS THE USER?

A "user friendly" label is meaningless without a clear idea of the nature of the user. A system that seems very congenial to a systems programmer may be a nightmare for a novice, and vice versa. A novice can format text quickly and easily using a Macintosh, but would have severe difficulties with **nroff** or **troff**. The UNIX expert could use the Macintosh software, but would grow tired of repeating the same functions every time a similar document had to be produced.

One must determine not only how sophisticated the intended user is but how familiar the user is with the specific tasks to be performed. A novice typically scores low in both areas. A computer science researcher may be very sophisticated about computers in general, but unfamiliar with the use of a particular package or device. The resident UNIX guru may know every program on the system intimately.

It is very difficult to optimize simultaneously for multiple

disparate goals. Generally, the most reasonable solution is the one that works well for the most typical user, and is not too taxing for others. Ideally, of course, a

A "user friendly" label is meaningless without a clear idea of the nature of the user.

solution allows different levels of interaction for different levels of users.

It might seem this sort of analysis is very subjective at best and is not well suited to an engineering or research approach. It turns out, however, that a substantial amount of work has been done to answer design questions of these sorts. *Human Performance Engineering* (Robert W. Bailey, Prentice-Hall, 1982) is perhaps the best introductory text on the subject. It seeks to distill the human factors work done at Bell Laboratories into a form that can be used by system designers elsewhere. The author summarizes a designer's objectives as follows:

A designer should strive to ensure that work is done with the greatest possible accuracy within the shortest time. In addition, the performance itself should be satisfying, and the designer should ensure that the time to develop the necessary, unique skills is as short as possible.

The text does an admirable

job of showing how a designer should meet these objectives: through analysis of relevant factors, tradeoffs of competing solutions, and so forth. It would serve equally well as an introduction to human factors for an interested consumer.

BRING OUT THE TOYS

So much for generalities; what about the toys we all love? Many of the nifty features to be found on new graphics workstations have their origins in human factors research. Mice, touch screens, windows, black-on-white text and high resolution graphics screens are all examples of this.

In trying to justify the use of a given item, a corporate purchaser needs to show why it is better suited to an application than some other, presumably cheaper one. A human factors study may be able to supply such a justification. In any case, it may point out considerations which might otherwise be overlooked by the purchaser.

These considerations may well determine what kind of equipment is appropriate. Touch screens are useful for simulating control panels, but terrible at making fine discriminations. Even in a control panel application, intensive or continuous use raises the specter of "touch screen elbow." Mice are ideal for some applications, and are nothing if not trendy. The user should be seated to use a mouse well, however, and this is not always convenient or even possible.

With the results of a human factors study in hand, one can judge the suitability of available devices with a much greater degree of confidence. This presumes, however, that the purchaser is able to interpret the specifications of the myriad of available devices. This is particularly difficult in the field of

Available
for IBM PC

What C did for Programming

Mark Williams has done for C Programming

The C Programming System from Mark Williams

MWC86 gets your C programs running faster and uses less memory space than any other compiler on the market. Then *csd*, Mark Williams' revolutionary C Source Debugger, helps you debug faster. That's The C Programming System from Mark Williams Company.

MWC86

MWC86 is the most highly optimized C compiler available anywhere for the DOS and 8086 environment. The benchmarks prove it! They show MWC86 is unmatched in speed and code density.

MWC86 supports large and small models of compilation, the 8087 math coprocessor and DOS 2.0 pathnames. The compiler features common code elimination, peephole optimization and register variables. It includes the most complete libraries. Unlike its competition, MWC86 supports the full C language including recent extensions such as the Berkeley structure rules, voids, enumerated data types, UNIX* I/O calls and structure assignments.

Quality is why Intel, DEC and Wang chose to distribute MWC86. These industry leaders looked and compared and found Mark Williams to be best.

User Friendly

MWC86 is the easiest to use of all compilers. One command runs all phases from pre-processor to assembler and linker. MWC86 eliminates the need to search for error messages in the back of a manual. All error messages appear on the screen in English.

A recent review of MWC86 in *PC World*, June, 1984, summed it up:

"Of all the compilers reviewed, MWC86 would be my first choice for product development. It compiles quickly, produces superior error messages, and generates quick, compact object code. The library is small and fast and closely follows the industry standard for C libraries."

csd C Source Debugger

Mark Williams was not content to write the best C compiler on the market. To advance the state of the art in software development, Mark Williams wrote *csd*.

csd C Source Debugger serves as a microscope on the program. Any C expression can be entered and evaluated. With *csd* a programmer can set tracepoints on variables and expressions with full history capability and can single step a program to find bugs. The debugger does not affect either code size or execution time. *csd* features online help instructions; the ability to walk through the stack; the debugging of graphics programs without disturb-

ing the program under test; and evaluation, source, program and history windows.

csd eases the most difficult part of development — debugging. Because *csd* debugs in C, not assembler, a programmer no longer has to rely on old-fashioned assembler tools, but can work as if using a C interpreter — in real time.

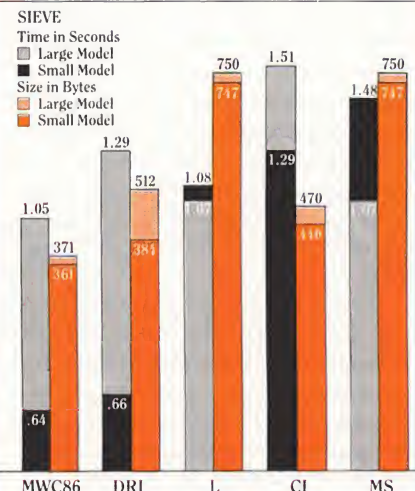
The C Programming System from Mark Williams now supports the following libraries:

Library	Company
Windows for C	Creative Solutions
Halo	Media Cybernetics
PHACT	PHACT Associates
The Greenleaf Functions	Greenleaf Software
Btrieve	SoftCraft

The C Programming System from Mark Williams

The C Programming System from Mark Williams delivers not only the best C compiler for the 8086 but also the only C source level debugger. That's why it does for C programming what C did for programming. The Mark Williams C Programming System gives the programmer the MWC86 C compiler and the *csd* C Source Debugger for only \$495. Order today by calling 1-800-MWC-1700. Major credit cards accepted.

Technical support for The Mark Williams C Programming System is provided free of charge by the team that developed it.



*Unix is a Trademark of Bell Laboratories.



Mark Williams Company
1430 W. Wrightwood Ave.
Chicago, IL 60614

computer graphics. Fortunately, some excellent help is available.

HELP ON GRAPHICS

While any effort to describe the state of the art in computer graphics is bound to be out of date in relatively short order, *Fundamentals of Interactive Computer Graphics* (James D. Foley & Andries Van Dam, Addison Wesley, 1982) is the best current attempt to cover the field, in terms of both hardware and software. In particular, the text's coverage of the Core System, produced by the ACM's SIGGRAPH, provides a good overview of the tools a designer should expect to find in a generalized graphics support environment.

In addition, there are a number of magazines covering computer graphics. They range from very technical journals to "gee whiz" slicks, and new ones are emerging almost monthly. Some time spent in a good library will give an overview of the currently available magazines. The reader should pick out one or more magazines that seem to cover topics of interest at a reasonable level. In any case, the advertisements are often the best source for information on the current commercial state of the art.

STILL FURTHER AFIELD

Topics such as programmer productivity, the sociology and psychology of computer programming, creativity and so forth are of interest to students of human factors, but may not have a direct bearing on a particular design question.

Leaving aside the scores of books on structured analysis, documentation, management, programming, et al., some introductory material is available in these areas as well. An early work,

The Psychology of Computer Programming (Gerald M. Weinberg, Van Nostrand, 1971), stands up well, as do Weinberg's texts on general systems thinking. *The Mythical Man-Month* (Frederick P. Brooks, Addison Wesley, 1972) gives an unequalled "school of hard knocks" overview of the sociology of project management.

The field of human factors has a large research component, and a great deal of this work is being done with computers. UNIX, because of its pervasive nature in

**Human factors
insights can make
an acceptable
product good and a
good product
great.**

academia, and because of its ability to adapt to changing requirements, has become a favored research environment for studies of computer-related human factors questions. We may hope that some of the results of such research will eventually enter the mainstream of UNIX usage. *Software Psychology* (Ben Shneiderman, Winthrop, 1980) gives a very thorough introduction to this world of experimental research and offers an extensive bibliography.

Given the lengthy production cycle for books, the most recent information will almost always be found in journals, conference proceedings and tutorials. Note that such papers tend to be both more demanding intellectually and less

polished editorially than those found in more formal books. For the current literature on research issues in human factors in computer systems, there are three major sources:

1) ACM SIGCHI (Association for Computing Machinery Special Interest Group on Computer & Human Interaction), P.O. Box 64145, Baltimore, MD 21264;

2) The Human Factors Society, P.O. Box 1369, Santa Monica, CA 90406;

3) IEEE CS (Institute for Electrical & Electronic Engineers Computer Society), P.O. Box 80452, Worldway Postal Center, Los Angeles, CA 90080.

So much for our introduction and reading list in human factors. Familiarity with the above material will not make one an expert in human factors. It will, however, give one the ability to filter out most of the "user friendly" slogans used by marketing and sales personnel. It will also allow one to analyze a design and (sometimes) find out how well it will meet the needs of the users.

Human factors insights can make an acceptable product good and a good product great. Human factors errors, on the other hand, can be very costly — as in the famous misplaced indicator at Three Mile Island. *Caveat emptor...*

Richard Morin is an independent computer consultant specializing in the design, development and documentation of software for engineering, scientific and operating systems applications. He operates the Santa Forda Computer Lab in Walnut Creek, CA.

■

BEAT OUR BENCHMARK.



GET THIS WATCH.

Take the Plexus Challenge.

Everybody in the UNIX-based supermicro business talks performance.

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

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

How do we dare make this offer? Simple: our multiprocessor architecture and highly-tuned version of UNIX leave the single-CPU competition in the dust. Multibus architecture and intelligent I/O processors let you expand modularly from



5 to 40 users — without changing a single byte of object code.

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

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

IPLEXUS

Supermicros built for speed.

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



BEYOND USER FRIENDLY

What should a computer offer?

by Jeff Schriebman



One of the most popular, and probably least understood, phrases used in the computer industry today is the term "user friendly." In discussing the relative merits of small computer systems, it is useful to have some understanding of what this term means or should mean.

We can assume that a person asking, "Is this system user friendly?" is not a computer expert, but rather a person hoping to be able to use the system. Whether for business or personal use, people want a computer for one reason — to get their work done easily and efficiently. All too often, however, people discover that many computers create additional work by forcing users not

only to adapt their work to the computer's abilities and limitations, but also to adapt their thinking to that of the programmers who wrote the program or system. Programmer solutions are often responses to system limitations, but sometimes they're also easier, less general ways of obtaining an effect. Regardless, the frustration involved and the time spent trying to discover the abilities and limitations of the hardware and the underlying algorithms can send the costs of buying a computer way above the suggested retail price — or worse, render the computer system useless.

It was in response to this situation that the term "user friendly" appeared on the scene. Some word processors, for example, were made user friendly by

using standard office terminology, such as "file drawer" and "folder" in place of "directory" and "file." In other cases, "user friendliness" was thought to be achieved by including verbose prompts and error messages. Current technology has developed user friendly systems with mice and bitmapped display screens. Yet, users who just want to get the job done will not find a system user friendly if, in order to use it, they must train themselves to think in some other, often convoluted way. If users must fight with the tool they're trying to use, or work around it, then the tool isn't of much use.

The computer should be *transparent* to the person using it. That is, the user should be able to concentrate on the task at hand and use the computer as a tool.



Illustration by Stephen G. Luker

The user shouldn't have to work around or fight that tool to get the job done. Just as a person using a pencil to do a math problem concentrates on the problem rather than on the pencil, so should a user be able to focus attention on the successful resolution of a problem — instead of having the tool *become* the problem!

A term such as user friendly is never used with simple tools, such as the pencil. The fact that these simple tools are convenient and easy to use is taken for granted. When the computer industry provides usable systems, the term user friendly will drop from use regarding computers as well. For example, there are already computers that are truly usable — not many, but a few. Yet these computers are not referred to as user friendly. For the most part, they serve as controllers. They perform a specific function and do that job well. We use these computers without considering them as such in cars, clocks, dishwashers, refrigerators, phones and any number of commonplace devices. They have achieved *transparency* and risen to the level of simple tools where their ease, convenience and usefulness are taken for granted.

For the future, the question is not *if* computers will become user usable, but *when*. Technology and the marketplace are already mov-

ing in that direction. Software and human engineering interfaces are being developed to make computers easier to use. Cognitive engineers are investigating how people think and reason, and how they interact with computers. Their knowledge in this area is growing and strongly affecting the directions taken by technology. The natural techniques that people use in reasoning through a problem are winding their way into computer software. Improvements in computer interfaces are assured in the not-too-distant future. In time there will be voice, written and visual input available and accessible — user usable.

The direction in the computer industry toward user usability does not negate the necessity for computer literacy. At the very least, it is important that people not be *computer-frightened*. Interfaces are being developed to make computers usable by large groups of people, but this will only have a limited impact if people don't overcome their own obstacles to using computers. People cannot effectively use computers if they're afraid of them — chances are you were not afraid of a pencil when you first started to write. Hopefully, once computers become more usable tools, the fear of them will disappear as well.

Naturally, the first part of this

educational process is understanding what a computer is. In understanding that computers are tools, it is obvious that they could not, nor should they, act like friends. Computers are machines and are thus incapable of willful action either for or against the user. Thus, the term user friendly is inappropriate for both computers and software. If a program does its job quickly, efficiently and is easy to use, it's not *friendly*, it's *useful*.

Regardless, it is my contention that people don't even *want* user friendly computers. People don't want to wake up in the morning and have their computer chat with them over coffee. They may, however, want their computer to prepare the coffee, turn on the lights or wake them with music. People need to be able to use computers, not establish friendships with them. As people become more computer literate, overcoming obstacles they may have once had to computers as tools, and as the computer industry develops better and more easily-used programs and systems, user friendly might well become a term better applied to *users* who approach computers with an open mind.

Jeff Schriebman is the founder and president of UniSoft Systems, a leading producer of UNIX ports. He is also a member of the UNIX REVIEW Editorial Board. ■



DEVELOPING SYSTEM FRIENDLY USERS

Suggestions for taking advantage of UNIX capabilities

by Jim Joyce

The phrase "system friendly users" seems a bit out of place in an issue devoted to "user friendly UNIX," but the point here is that UNIX, like any computer system, needs to be brought up on new users just as it needs to be brought up on new hardware.

If "ports" (as it were) to new users are successful, we get system friendly users. If not, users will complain the system is hard to learn because of clashes with habits learned from other systems they have known (and perhaps loved) or perhaps from pure UNIX-phobia. Perhaps they will cite the Datamation article "The Trouble with UNIX" (November 1981) and betray their fears that the system is dangerous, even hostile.

As Jeff Schriebman points out in the previous article, a truly user friendly computer system is transparent to the user. To turn this around a bit, a truly system friendly user would feel the system does not get in the way.

That is what most UNIX hackers claim about UNIX.

Before hackles start to rise on either side of the issue about UNIX users being system friendly, I would like to point out that few hackers I know use anything like vanilla (or unchanged) UNIX. The system interface is easy to modify either for the Bourne shell through **.profile** files, or for the C shell through **.login** and **.cshrc** files. These files are actually shell scripts that guide the appropriate shell in altering the environment.

This sounds user friendly, and it is *system friendly*, too, in that instead of fighting the system the user can convince the system to adapt in terms both can understand. Rather than battle about what isn't quite right, the user and the system can agree to what *is* right.

For example, the big bugaboo for many UNIX

detractors is the danger of the **rm** command, especially:

```
$ rm *
```

which removes all files (except directories and "dot" files) in the current directory.

In the C shell this is easily remedied by inserting:

```
alias rm rm -i
```

in either the **.login** or **.cshrc** files, or by habitually typing **rm -i** instead of merely **rm** to the shell as a command. The **-i** flag asks **rm** to inquire whether each file it finds is actually to be deleted. Only a **y** response will instruct **rm** to delete the file; anything else will leave the file intact.

The Bourne shell does not have the **alias** command, but there are other remedies. Users can establish local command directories where a suitably tailored shell script replaces the original **rm** command:

```
: Bourne shell version of aliasing rm to rm -i  
/bin/rm -i $*
```

The typical name for a user's local command directory is **bin**, located in the user's home directory. The full pathname is in the **\$HOME** shell variable. The shell script:

```
PATH=$HOME/bin:$PATH  
export PATH
```

will guide the search for command name resolution first to the local **bin** directory, and then the others in the search path.

A better solution yet is the **pick** command, discussed in Kernighan and Pike's *The UNIX Programming Environment* (Prentice-Hall, 1984),



```
:
# pick: select arguments
#   Adapted slightly from Kernighan and Pike,
#   The UNIX Programming Environment
#   (Prentice-Hall, 1984), p. 160.
#
PATH=/bin:/usr/bin

for i
do
    echo -n "$i? " > /dev/tty # force output to terminal
    read response
    case $response in
    y|Y*) echo $i ;;
    q|Q*) break
    esac
done < /dev/tty # force input from terminal
```

Figure 1 — *The pick command, introduced in Kernighan and Pike's The UNIX Programming Environment.*

shown in Figure 1. Its use would be in the pipeline:

```
$ pick * | rm
```

The list of files generated by the `*` is presented, one name at a time, to the user (`echo ... > /dev/tty`), who is then to reply (`... < /dev/tty`). If the reply is anything beginning with upper or lower case `Y`, the file passes along the pipeline. Upper or lower case `Q` quits the selection process. Any other response causes the shell script to go on to the next filename.

As Kernighan and Pike point out, `pick` generalizes the `-i` option of `rm` to *all* commands that can operate on several input files without adding code that will have to be tested to any of the commands.

Other examples of the benefits system friendly users can gain abound in Steve Bourne's *The UNIX System* (Addison-Wesley, 1983). Granted, Rob Pike, Brian Kernighan and Steve Bourne are *not* today's typical UNIX user, but that's one of the points of this article: how users can become system friendly.

SYSTEM FRIENDLY DOCUMENTATION

To take advantage of UNIX capabilities, users need to know what they are. Documentation and training play a major role here. Among the documentation are the books mentioned earlier, Kernighan and Pike's *The UNIX Programming Environment* and Bourne's *The UNIX System*. But these books are really not for beginners.

Introducing the UNIX System, by McGilton and Morgan (McGraw-Hill, 1983) is a justly popular introductory book, as is Sobell's *A Practical Guide to*

the UNIX System (Benjamin-Cummings, 1984). Both books are oriented toward Version 7 UNIX with Berkeley enhancements such as the C shell and the `vi` editor.

Just out is *Exploring the UNIX System*, by Stephen G. Kochan and Patrick H. Wood (Hayden, 1984), which was written in a System V environment at AT&T Bell Laboratories. This book, reviewed at greater length in this month's */usr/lib* column, has a fascinating approach starting with a non-technical discussion of what an operating system is. The tour through utilities and shell scripts culminates in a chapter on system administration. Each of the books mentioned offer help to users wishing to become system friendly.

Before so many good books burst upon the scene, the best resource for someone wishing to become a system friendly user was the *UNIX Programmer's Manual*, a collection of entries best characterized as a collection of intra-office memos shared within the Bell Labs Computer Science group and later released with copies of the UNIX system on tape.

It is a sad truism that once you know UNIX reasonably well, the manual tells all and is a wonderful resource for UNIX goodies. It used to be standard practice at Bell Laboratories to reread the manual once a month. But with the recent packaging of System V.2 documentation, where three volumes have expanded to nearly three times that number, that practice is more honored in the breach than the observance.

Still, rereading Section 1 of the manual every two or three months will bring surprising rewards



SYSTEM FRIENDLY

— such as remembering the right UNIX tool to use in a given situation. Browsing in the online manual pages is one way to do such reading while taking a break from online work. Sadly, a number of manufacturers have removed the online manual (as well as games), and this option is thus denied to more and more UNIX users.

SYSTEM FRIENDLY TRAINING

Reading about UNIX is a relatively slow way of learning the capabilities of the system. The best way is to have a guru in the room with you. As a second-best option, hands-on experience and live seminars are among the fastest ways to gain facility with UNIX. Introductory UNIX seminars are offered by, it seems, everyone in sight. Yet these rarely contain enough information for a participant to become system friendly.

The true powers of either the Bourne shell or the C shell are usually skipped over because of time, and topics such as **awk** and **sed** are often neglected entirely. Users may be willing to sit through a four-day course as an introduction to UNIX, but after that few have the time or inclination to allow for further four-day intrusions into production.

One solution is to package half-day and full-day courses on specific UNIX topics, such as "Using the C Shell More Effectively" (which fits comfortably into a half-day package), or "Problem Solving with the Bourne Shell" (a full day topic). The idea is to focus on particular aspects of the UNIX system rather than trying to create all-encompassing courses for "the intermediate" or "the advanced" user — a far too general approach. Our experience at ITS indicates this focused approach toward short courses is effective.

But it is not simply the short, intense format of the course that is effective. Examples are an immediate way of teaching, and by focusing on working examples rather than handwritten snippets of code or shell script, participants leave the class with things they can try at their own installations. By practicing with system friendly examples, the participants can become system friendly users.

Several training organizations I have talked with seem to have the opinion that *anyone* can teach a course once it is planned. But if courses adhere to the short course format and emphasize examples, the instructors need to be able to answer any questions they may encounter. The simplest solution is: ask knowledgeable, articulate technical people to be instructors. We term this approach "expert-led" teaching, and feel it is particularly effective given the example-driven course material.

For organizations that do not have access to

Let us prove how Cromemco systems can increase your satisfaction with UNIX System V.

Call, or visit, one of our Official System Centers today:

USA

In Arizona:

Artemis Computer
602/957-0469
Systems Solutions, Inc.
602/224-0026
Professional Data Systems, Inc.
602/265-6656

In California:

Quintec
213/889-4819
American Computer & Communications
415/849-0177
MCM Enterprises
415/327-8080
American Computer & Engineers
213/477-6751
Kieruff Electronics
213/725-0325
Accountability Systems
916/639-4570
Excolibur
916/972-9252
Kieruff Electronics
714/278-2112
Kieruff Electronics
408/971-2600
Cromemco, Inc.
818/346-6690

In Connecticut:

Datacraft, Inc.
203/673-6952

In Florida:

Automated Computer Systems
305/594-3819
Computer Centre
813/484-1028
Royal Data, Inc.
305/267-1960

In Georgia:

Cromemco, Inc.
404/391-9433
Systems Atlanta
404/928-0240
Kieruff Electronics
404/447-5252

In Illinois:

Schauers Office Services
312/755-2100
Kieruff/Chicago
312/640-0200
Commercial Data Systems
309/797-9401
Alpine Computer Center, Inc.
815/229-0200
Cromemco, Inc.
312/934-0650
Alternate Computer Services
312/893-1992

In Indiana:

K.I.D. Enterprises
312/891-1064
Harbourtown Sales
317/877-4900
Microcomputer Specialists, Inc.
219/762-8541

In Kansas:

Treadwind Systems
316/624-8111
Armstrong Microcomputers
316/223-2939

In Louisiana:

Muse Data Technologies
504/293-0320
Standard Systems, Inc.
318/625-8613

In Maryland:

Dynamic Data Processing
301/589-8950

In Massachusetts:

Kieruff/Boston
617/667-8331
Cromemco, Inc.
617/938-7010

In Michigan:

United Microsystems Corporation
313/668-6806
Jepsan Group, Inc.
616/698-8700
Automated Business Consultants
313/478-0557

In New Jersey:

Kieruff Electronics
201/575-6750

In New Mexico:

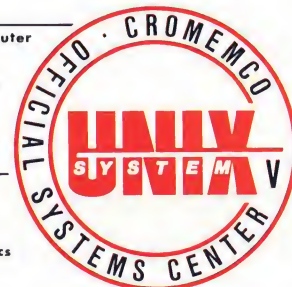
South West Computer Stores, Inc.
505/292-6568

In New York:

C.C.S., Inc.
212/986-7520
Computer Closet
914/268-5164

In Ohio:

Lucas Office Equipment & Service, Inc.
513/433-8484
Odyssey Systems, Inc.
216/526-9933



(ISIS) Innovative Systems/Integrated Software
419/531-0220

In Pennsylvania:

Modular System Design
412/521-6700

In Texas:

Kieruff Electronics
214/343-2400
Gunn Enterprises
713/781-6911
Procomp dba Integrated Computer Systems
713/226-3648
Computer Crossroads of America
214/231-6108

In Virginia:

SMS Data Products
703/827-0640
Business Communications Systems
703/344-5563
VCA Corporation
703/281-4666

In Washington:

Kieruff/Seattle
206/575-4420

In West Virginia:

Systems Support
304/766-7762

In Wisconsin:

Computer World
414/733-9547

Bay Tech of Wisconsin, Inc.
414/846-3038
Computer World
414/499-9983

INTERNATIONAL In Australia

Minicomp Software & Education
61-1/957-6800
Insystems P/L
61-3/690-2899

In Canada:

Cro-Corp Holdings, Inc.
403/286-8459
Future Electronics
610/421-3251

In Costa Rica:

Control Electronics
506/23-50-17/24-44-44

In England:

Jarogate Ltd.
44-1/671-6321
Cromemco, Ltd.
44-1/785-9822

In Greece:

Algorithm Ltd.
30-1/933-8463

In Hong Kong:

Vonda Computer & Equipment
8529 348702-5

In Italy:

C.N.I.A.
39-51/375009/359406

In Japan:

Asahi Gloss
81-3/218-5848

In Israel:

Information Systems Ltd.
03-775111

In Mexico:

Micromex, S.A./DE C.V.
905/687-8886/8913/
905/536-5503

In Mid-East:

Multi Medio Video, Inc., CA USA
408/727-1733
National Computer System
Karachi 4, Pakistan
Computer System Marketing System
Jeddah 21431, Saudi Arabia

In The Netherlands:

Rocomp B.V.
31-40/524055

In Norway:

Micro Systems A/S
47-2/41-69-76

In Scotland:

Micro Centre Complete Micro
44-31/556-7354

In Sweden:

Datrosiering Konsult AB
46-8/753-3090

In West Germany:

Cromemco GmbH
49-6196/481606
Digitronic Computer systeme
49-4103/88672
Cosy-X Computer System GmBH
2173/52071/72
Comicro Deutschland
49-2151/795577

Cromemco®

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

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

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

More capability and expandability

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

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

But we give you even more.

A complete solution

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

protocols. Mainframe interface.

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

Easy to use.

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

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

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

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

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

Main, West Germany, or Cromemco, Ltd., The Cambridge House, 178-182 Upper Richmond Rd., Putney, London SW15, England.



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

Cromemco[®]

Circle No. 9 on Inquiry Card



knowledgeable, articulate technical people, a viable alternative is to pair a knowledgeable technical person as a resource person with an articulate course leader, assuming the two have rapport and have *rehearsed* questions the technical resource person is to field. The rehearsal will give the technical resource person confidence in answering questions when they come up, and the course leader can ascertain how and when to make best use of the technical person. Importantly, the technical person should not simply be kept to bail the course leader out of difficulty. The rehearsal should include some give-and-take that allows both to present information.

These observations about how to encourage system friendly users provide a starting point for

thought rather than The Final Word on the matter. Some may still feel the shell has Got To Go, and that a more forgiving user interface is needed. So be it! The shell is, after all, simply a program, and is easily replaced by a proper entry in `/etc/passwd`. Yet for those who learn to use UNIX, shell and all, the system is a delight for its flexibility and unobtrusiveness.

Jim Joyce is a long-time UNIX advocate dating from his teaching days at UC Berkeley when UNIX first arrived there. He is now president of International Technical Seminars, Inc., 520 Waller Street, SF, CA 94117, 415/621-6415, a teaching and consulting firm specializing in the UNIX system. ■

GETTING SYSTEM FRIENDLY THROUGH READING

The best books on the market

Every month it seems there are two or three new UNIX-related publications in print. The question most asked at The Independent UNIX Bookstore is no longer "is there anything about UNIX?" but "what is the best book for me?" Here is a short guide to the best books for managers, programmers, non-programmers and more advanced UNIX users.

FOR MANAGERS

Managers, marketing and technical people who want to know something about UNIX but do not need detailed knowledge will find Paul Weinberg and

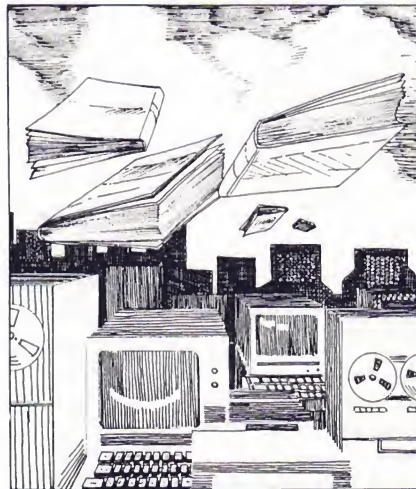


Illustration by Hyon Kim

James R. Groff's *Understanding UNIX: A Conceptual Book* (Que Corp., 225pp, \$17.95 paper) the

right book. It discusses how UNIX fits into the world of computing, what the major UNIX features and benefits are, and how the system is structured.

To illustrate just how astute the authors are, they accurately rumored in 1983 that InterActive Systems was at work on VM/IX, the IBM mainframe port of UNIX. VM/IX was released mid-July, 1984.

Technical discussions of commands and shell scripts are included in Weinberg and Groff, but not at the level of detail to be found in a book for programmers. The subtitle is "A Conceptual Guide," and that is what the book delivers.

PROGRAMMERS LEARNING UNIX

Three books tie for best choice for this audience, depending on how experienced a programmer the reader is.

For highly experienced programmers who may already know something about UNIX from articles that have appeared, Steve Bourne's *The UNIX System* (Addison-Wesley, 349pp, \$16.95 paper) is the book to read. Bourne is also the author of the standard shell. Those who thread their way through the tennis ladder example in this book will truly know the power of UNIX utilities and the shell as a programming language.

Those needing a more introductory book have a genuine choice between Henry McGilton and Rachel Morgan's *Introducing the UNIX System* (McGraw-Hill, 556pp, \$19.95 paper) and Mark G. Sobell's *A Practical Guide to the UNIX System* (Benjamin/Cummings, 428pp, \$21.95 paper). Both books are carefully done, containing clear examples of making use of the power of UNIX. Sobell's book shows what can be done with a thoughtful use of visual aids to help readers grasp concepts. McGilton and Morgan have included a chapter on system administration for readers who have one of the many micro-based UNIX systems now available.

UNIX FOR NON-PROGRAMMERS

Just which book is the best for a non-programmer depends on the meaning of the term "non-programmer" and what the reader wishes to learn.

A senior systems analyst may well find Groff and Weinberg, mentioned above, quite satisfactory as a guide to the scope of UNIX power.

Selective reading of McGilton and Morgan or Sobell can provide

a non-programmer with a working knowledge of the word processing capabilities of the system.

Non-programmers in a Berkeley UNIX environment may find *UNIX Primer Plus* by Mitchell Waite, Donald Martin and Stephen Prata (Sams & Co., 414pp, \$19.95 paper) a pleasant way to learn the system. However, the book's Berkeley 4.2 UNIX emphasis will be frustrating to those using systems that do not include Berkeley enhancements.

INTERMEDIATE UNIX

Without a doubt, *The UNIX Programming Environment* (Prentice-Hall, 357pp, \$19.95 paper) by Brian Kernighan and Rob Pike is a must-read book for anyone seriously interested in using UNIX effectively. I will say at last in print: theirs is a book of philosophy disguised as a technical book. It is technical, to be sure, but its strength lies in the thinking they do about the UNIX programming environment.

The script for the **pick** utility and the discussion of it are worth the price of the book. Programmers who add features to programs will want to study the discussion carefully to see a very different approach that is a genuine increase in functionality.

UNIX INTERNALS

UNIX source code, one soon finds, is protected by trade secret, and anyone who has had access to source code is bound by signed agreement not to reveal the code to those who have not signed such a nondisclosure agreement.

Thus, Douglas Comer's *Operating System Design: the XINU Approach* (Prentice-Hall, 474pp, \$29.95 hardbound) cannot, of course, be UNIX source in book form. That XINU is UNIX spelled backward is provocative enough, though.

The book, bottom line, is a

WE SPECIALIZE

in the Placement of

UNIX Managers & Professionals

\$40K - \$80K + Package

--You know you're in demand--
We find the **right** career move for you!

Call Frank Flanders
603-880-4900

ortune
Personnel Consultants

505 West Hollis Street, Suite 208
Nashua, N.H. 03062

Circle No. 11 on Inquiry Card

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. UR
Houston, TX 77224
713-496-6100



Scientific Placement, Inc.

*Unix is a trademark of Bell Labs

Circle No. 10 on Inquiry Card



SYSTEM FRIENDLY

book on operating system design, and XINU is a UNIX-like operating system. Someone with access to

UNIX source code could also use the book as a helpful companion for reading the code.

GUIDES TO SOFTWARE AND SYSTEMS

Three guides are available to help readers determine what UNIX offers, what software is available and what systems and software is available.

Bill Freiboth's *The UNIX Guide* (Pacific Micro Tech, 118pp, \$24.95 paper) is designed "to help readers determine what UNIX offers." Version 7 UNIX, the *de facto* standard at this time, System III, System V, 4.x Berkeley UNIX and UniSoft's UniPlus are compared and contrasted in discussion and table form. The guide is not for techies, because they don't need it. It is for those wanting to know about UNIX version differences.

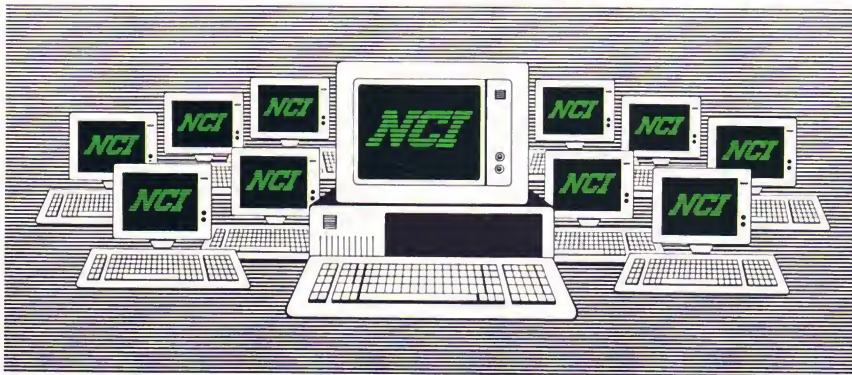
The */usr/group UNIX Catalog* (*/usr/group*, 438pp, \$30.00), compiled by August Mohr, is now in its third edition, containing summaries by 276 companies of 797 UNIX-related products and services.

Although it tries to be complete, the catalog is at the mercy of the timeliness of companies contributing their entries. Nonetheless, it is a valuable compendium of information about UNIX in a commercial environment.

Ray Jones's *The UNIX Applications Software Directory* (Onager Publishing, 198pp, \$50.00 paper) lists nearly 400 descriptions of applications packages from benchmark suites to COBOL compilers, accounting packages and graphics software. It should be used with the */usr/group UNIX Catalog* for best results in reaching companies who have software available. At the back of the directory is the promise of a hardware directory. I hope this shows up soon, as it is very much needed.

-JJ

A new breed in speed **NCI COHERENT**



The only UNIX compatible operating system that supports 11 simultaneous users on 1 IBM PC XT

Only NCI COHERENT offers powerful multiuser capability with UNIX compatibility.

NCI made COHERENT a multipurpose operating system. We extended it to suit a wide variety of applications. For example, it can now be used to turn a PC into a data multiplexer or an inexpensive database machine in a local area network. NCI COHERENT can also be used for high speed protocol conversion and process control. We've even written a real-time version of COHERENT, exclusive to NCI, which will run on an intelligent peripheral board.

NCI has enhanced the COHERENT kernel making it much faster. Then we added commands that make programming easier, including

a source code control system, plus many UNIX System V features. We also gave it exclusive support for a wide variety of hardware and PC compatibles. NCI COHERENT comes with full technical support and complete documentation, organized in the UNIX manner.

A variety of licensing options are available including runtimes, production runtimes, and driver kits.

Remember, when you license COHERENT your license includes multiuser and our low license fees

make your application far more economically feasible.

The NCI technical design team that engineered the improvements to COHERENT is also available on a contract basis.



Performance engineered into every product.
 NETWORK CONSULTING INC.
 Suite 110, 3700 Gilmore Way,
 Burnaby, B.C. Canada V5G 4M1
 Phone: (604) 430-3466

*UNIX is a trademark of Bell Laboratories. COHERENT is a trademark of Mark Williams Co. IBM PC and IBM PC XT are trademarks of International Business Machines Corp.

Circle No. 12 on Inquiry Card



EMERALD ONE™

SOFTWARE WITH COURAGE, BRAINS AND HEART

WHAT IS EMERALD ONE?

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

THE TOOLS OF EMERALD ONE

EMERALD ONE integrates your office tasks through:

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

SOFTWARE FOR THE WORK GROUP

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

EMERALD CITY, THE PEOPLE BEHIND EMERALD ONE

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

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

EMERALD
C · I · T · Y

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

See us at

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

Circle No. 13 on Inquiry Card

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



THE WELL-TEMPERED M • E • N • U

A story of use, misuse and design

by Jordan J. Mattson

With more and more new users entering the UNIX world, we have seen a proliferation of UNIX based menu-driven systems. Given this proliferation, questions arise: where should menu-driven systems be used; what are the components of a menu-driven system; how should system designers keep menu-driven systems under control; and what is a good model for the building of menu-driven systems?

Given a little thought, it is clear that menu-driven systems can be very useful in several areas and that it is worth the effort of system designers to build menu drivers for systems in these areas.

There are three primary areas where menu-driven systems can be useful. These are systems used by new/naive users; systems that are seldomly used; and systems that have commands with a very complex syntax and/or a multitude of options.

A menu-driven system can help new/naive users to quickly see that there is productive work that *they* can do using UNIX. By side-stepping the long learning curve for UNIX, users can be emboldened. They can discover that



much can actually be done using UNIX, thus whetting their appetites for more learning. They may even begin to explore.

When building a menu-driven system for new/naive users, the system designer must guard against making users too dependent on menus. To achieve a system in which users are not overly dependent, it's important to not allow the menu driver to drive all of the parts of the system. Force users to learn about what there is of UNIX outside of their

menu-driven system. (For example, in my menu-driven systems, I have never included a menu for printing output. This forces users to learn about the UNIX directory structure and how files are printed.) In some cases, users will not wish to learn how to use UNIX, but rather only how to run a specific system on their UNIX system. In these cases only, it makes sense to build a menu-driven system that will take care of all their UNIX needs. Deciding which course of action to take is a judgment call on the part of the system designer.

The second primary area where menu-driven systems can be useful is in facilitating

the use of important, but seldom-used systems. Here, menus can document the systems and guide users through them. Examples of where this approach would be appropriate might include billing programs used once every six months or system statistics used once a year. Deciding whether a system should be included in this category involves thinking through how often the system will be used versus how quickly users of the system will forget how to use it. In some cases, a menu-

driven system will be appropriate for a system used once a week, while in other cases a menu-driven system will only be appropriate if the system is used only once every six months. There are no hard and fast rules. The system designer will have to judge if a menu-driven system is appropriate on a case-by-case basis.

Another factor in favor of building menu-driven systems in the case of seldom-used systems is the generally poor quality of UNIX documentation. Menu-driven systems are essentially self documenting and if the same menu interface is used in the building of all of the menu-driven systems at a particular installation, users will find it easy to shift from using one system to another.

The third primary area where menu-driven systems can be useful is with systems that have commands with a large number of options and/or a complex command syntax. In these systems, the menu driver can greatly aid users by guiding them through a decision tree of options and then assembling the users' choices into command arguments with the correct syntax, thus ensuring that the selections make sense and are presented to the command in a correct format.

As in the case of the seldom-used system, a menu-driven system can act as documentation for a system which has a large number of options and/or has a complex command syntax. Building a menu-driven system, however, should not be used as an excuse for turning out shoddy system documentation. Rather, the menu-driven system should act as a supplement to standard documentation.

THE WELL-TEMPERED MENU

A menu-driven system can be divided into two parts: interface and actions. The interface is provided by some form of menu driver. The actions are commands that have their arguments and option flags set by the interface. A menu-driven system in which it is possible to access the actions by

either using the interface or ignoring it is a well-tempered menu system.

Under a well-tempered menu system, users will be able to decide if they wish to access the actions of the system via the interface provided by the menu driver or through standard shell commands. By building a well-tempered menu system, you will be able to get new/naive users going immediately, but still allow them to "grow into" the system on their own without the aid of the menu interface. This will avoid situations in which you have to tell once-naive users that they cannot use the programs of their system without using the menu driver that was once helpful but has since grown annoying.

The key to ensuring that you have a well-tempered menu

The system designer must guard against making users too dependent on menus.

system is to decouple the interface part of the system from the action part of the menu system. If a system has had its interface decoupled from its actions properly, then a well-tempered menu system will result.

Producing a menu-driven system in which the interface is decoupled from the actions involves writing the menu-driven system as a set of black boxes. The primary black box is the menu driver, while the rest of the black boxes are the actions. The menu driver (the interface) will in some fashion, depending on implementation, discover what the users' choices are and call the appropriate action black box with the appropriate arguments and

options. Because we have built our system as a set of black boxes, we only pass information (in the form of arguments and return conditions) back and forth between the interface black box and the action black boxes. This is the only means of communication between the parts of our system. Because of this, there is no difference between a call from the menu driver to a black box and a call from a user to a black box. Thus, we have an interface that knows how to call the actions and yet leaves a way for users to call the actions.

Building a menu-driven system as a set of black boxes with a black box for an interface and black boxes for actions has several advantages:

- 1) It allows for standardization of menu driver software at UNIX installations, thus reaping two benefits: a) It allows users to quickly move from one menu-driven system to another, and b) it makes the maintenance of the menu-driven system easier since interface code is standardized.
- 2) The isolation of all interface code within the menu driver reduces the complexity and size of code in actions.
- 3) The user is able to access the action parts of the menu-driven system without going through the interface.

BUILDING WELL-TEMPERED MENUS

Once it has been decided to build a system as a menu-driven system and to build it as a well-tempered menu system, one question remains: how to go about building it?

In three years of building menu-driven systems for office automation at the University of California at Santa Cruz, I have come to the conclusion that the best way is to work with a set of standardized library routines that implement what I will call the Menu Library. By using the Menu Library along with the `curses` and `termcap` (or `termlib` on some systems) libraries, it is possible to



implement what I will call screen-oriented, interactive menu drivers.

A screen-oriented, interactive

menu driver is one that allows the user to move to different "windows" on one of a number of "screens" to make choices. Each

of the "windows" has some text, which is displayed on the user's terminal screen, and an action associated with it.

In an implementation of the Menu Library, the "screens" are mapped to the object MENU and the "windows" are mapped to the object MENUWINDOW. An implementation of the objects MENU and MENUWINDOW is given in Figure 1 (p. 36).

Given the objects MENU and MENUWINDOW, the Menu Library needs a set of operations to allow the programmer to access and use the objects to construct menu drivers. These operations fall into the following broad categories:

- A) Operations to prepare to use the menu driver.
- B) Operations to create MENUs and MENUWINDOWS.
- C) Operations to set the various fields of a given MENUWINDOW.
- D) Operations to create a **curses** WINDOW from a MENU.
- E) Operations to "run a menu." That is, move the user's cursor on the screen and take actions when the user chooses them.

The operations for each of these categories, along with their arguments are highlighted in Figure 2 (p. 37). These operations, along with the objects on which they operate, make up the code which needs to be included in a library of menu-driven building routines.

Jordan J. Mattson is a Computer and Information Sciences student at the University of California at Santa Cruz and works as a UNIX programmer specializing in menu-driven office automation systems for the UCSC campus community. He is part of the UNIX REVIEW Software Review Board and can be reached at ucbvax!ucsc!ucsce!jais.

NEW

ONE SIZE FITS ALL

Heurikon presents Minibox – a multiuser UNIX workstation based on its powerful HK68™ single board microcomputer and Uniplus+™ UNIX System III or System V operating system with Berkeley enhancements.

Designed with the OEM in mind, *one size fits all*. Both compact and flexible, the Minibox includes within its 10.5" w x 13.9" h x 20.5" l frame a 200 or 400 watt power supply, six slot Multibus™ card cage, (4-5 available for user use!), single double density floppy disk drive, streamer tape drive, and 31 or 65 Mbyte Winchester drive (expandable to 280 Mbytes). All this within the same cabinet! System status LEDs on the front panel inform the user of CPU and disk drive activity.

With Uniplus+™, Minibox becomes a flexible and affordable tool for program development, text preparation, and general office tasks. Included is a full "C" com-

piler, associated assembler and linker/loader. Optional languages are:

Macro assembler, ISO Pascal compiler, FORTRAN-77 compiler, RM-COBOL™, SVS BASIC (DEC BASIC compatible interpreter), SMC BASIC (Basic-Four BB3 compatible interpreter), and Ada™. Other utilities include UltraCalc™ multiuser spread sheet, Unify™ DBM, Ethernet™, and floating point processor. Alternate operating systems available are PolyForth™, Regulus™, CP/M 68K™, and others.

*UNIX is a trademark of Bell Laboratories. Unify is a trademark of Unify Corp. UltraCalc is a trademark of Olympus Software. Ethernet is a trademark of Xerox Corp. Uniplus+ is a trademark of UniSoft Corp. PolyForth is a trademark of Forth, Inc. Regulus is a trademark of Alcyon Corp. CP/M-68K is a trademark of Digital Research. Ada is a registered trademark of the U.S. government. Ada Joint Program Office. RM-COBOL is a trademark of Ryan-McFarland Corp. HK68 is a trademark of Heurikon Corp. Multibus is a trademark of Intel Corp.

HEURIKON

3001 Latham Drive
Madison, WI 53713

Telex 469532

800/356-9602
In Wisconsin
608/271-8700

© 1983



High Performance Machines.

Now, from the same people who brought you the industry-leading price/performance champion 3300-Series supermicro comes a totally new dimension in high performance machines—the Contel-Codata 3400-Series. We've designed the 3400-Series specifically for the OEM marketplace and proudly bring you more versatility, dependability, and expandability than any other manufacturer. It's the Codata Difference and another major improvement in OEM microcomputers.

Whether your application is scientific or desktop, whether you need a graphics engine or a free-standing floor model, the Codata 3400-Series gives you the industry's broadest range of capabilities.

3400-Series features include 8-megabyte RAM addressability, expanded mass storage with 12, 47, 84, 168 or 320-megabytes of high-speed Winchester

efficiency, 9-track 800/1600 BPI magnetic tape, hardware floating point accelerator for Fortran/Pascal, Ethernet LAN, and, of course, the M68000 running the Contel-Codata autoconfiguring UNIX™ operating system. All 3400-Series systems are based on Multibus™ architecture giving you nearly unlimited versatility of applications.

Let Contel-Codata put you on the fast track! Test drive the 3400-Series supermicros TODAY. For details contact us at: CODATA SYSTEMS CORPORATION, 285 N. Wolfe Road, Sunnyvale, CA 94086, 408/735-1744, 1-800-521-6543. Telex 172869 CODATA SUVL. In Europe, CONTEL-CODATA, 250 Avenue Louise, Box 101, 1050 Brussels, Belgium. Telex 65942 CADO-B. MULTIBUS is a trademark of Intel Corp. UNIX is a trademark of AT&T Labs

CONTEL CODATA

CODATA SYSTEMS CORPORATION



WELL-TEMPERED MENUS

```
/* Here are the implementations of the objects MENU and MENUWINDOW
 * and their supporting data structures.
 *
 */

/* The argument family of data structures provide the facility for
 * passing different types of arguments and different numbers of
 * arguments to functions easily.
 *
 */

union Arg
{
    int    Integer;
    long   Long;
    float  Float;
    char   Character;
    char   *String;
    char   *Format;
    MENU   Menu;
};

struct Argument
{
    int    ArgType;
    union Arg TheArg;
};

typedef struct Argument *ARGUMENTPTR;

typedef ARGUMENTPTR *ARGUMENTS;

/* Next we turn to the implementation of the struct MenuWindow, which
 * is the struct to which MENUWINDOW is a pointer.
 *
 */

struct MenuWindow
{
    /* This is the pointer to the function which is used to update the
     * text displayed on this MenuWindow's "window".
     *
     */
    int    (*Update)();

    /* These are the "arguments" to be used when calling the Update()
     * function.
     *
     */
    ARGUMENTS UpdateArguments;

    /* This is the pointer to the function which is used to call the
     * action associated with this MenuWindow's "window".
     *
     */
    int    (*Action)();

    /* These are the "arguments" to be used when calling the Action()
     * function.
     *
     */
    ARGUMENTS ActionArguments;

    /* These are the variables that define the boundaries of this
     * MenuWindow's "window".
     *
     */
    int    XStart;
    int    XEnd;
    int    YStart;
    int    YEnd;
};

/* We at least define MENUWINDOW and MENU. The MENUWINDOW is but a
 * pointer to MenuWindow, and the MENU is but a pointer to MENUWINDOW.
 *
 */

typedef struct MenuWindow *MENUWINDOW;

typedef MENUWINDOW *MENU;
```

Figure 1—An implementation of the objects *MENU* and *MENUWINDOW*.

A) Operations to prepare to use the menu driver.

```
BeginMenus()
```

B) Operations to create MENUS and MENUWINDOWS.

```
MENU CreateMenu(N)
```

```
int N;
```

```
/* N is the number of MENUWINDOWS that the MENU returned by this  
 * function will have.  
 *  
 */
```

C) Operations to set the various fields of a given MENUWINDOW.

```
/* In all of these functions, N is the MENUWINDOW of the MENU M  
 * on which we wish to operate.  
 *  
 */
```

```
SetBounds(M, N, XStart, YStart, XEnd, YEnd)
```

```
MENU M;  
int N;  
int XStart;  
int YStart;  
int XEnd;  
int YEnd;
```

```
SetFunctions(M, N, Update, Action)
```

```
MENU M;  
int N;  
int (*Update)();  
int (*Action)();
```

```
/* In SetDisplayArguments and SetActionArguments, Format is a string  
 * that informs the function of how many arguments and their types are  
 * being passed in using VARGS (notice va_alist and va_dcl).  
 */
```

```
SetDisplayArguments(M, N, Format, va_alist)
```

```
MENU M;  
int N;  
char *Format;  
va_dcl
```

```
SetActionArguments(M, N, Format, va_alist)
```

```
MENU M;  
int N;  
char *Format;  
va_dcl
```

D) Operations to create a Curses's WINDOW for a MENU.

```
WINDOW *WindowMake(M, N)
```

```
MENU M;  
int N;
```

E) Operations to "run a menu".

```
RunMenu(M)
```

```
MENU M;
```

Figure 2—The operations necessary for allowing programmers to access and use the objects for constructing menu drivers.

DON NORMAN

Of all the articles and papers written about UNIX, few have stirred more interest than one written by Don Norman in the November 1981 issue of Datamation.

Titled "The Trouble with UNIX," it took the system's user interface to task, labeling it "a disaster for the casual user."

"If UNIX is to become a general system, then it has got to be fixed," he argued. Among the remedies he proposed were: 1) develop a fundamental set of design principles, 2) provide users with an explicit model, and 3) provide mnemonic aids.

Not all of Norman's readers shared his views. Many wrote to question not only his premises but his character. Some of the questions raised in this interview, in fact, follow up similar queries made in Doug McIlroy's celebrated reply to Norman. (McIlroy, one of



the architects of UNIX, lambasted Norman in that letter. Copies of the letter still make the rounds at Murray Hill.) Norman today merely shrugs, "Yeah," he says. "Who is this Don Norman and why is he saying all of those nasty things?"

Since this issue, devoted to "User Friendly UNIX," comes out three years to the month after Norman's piece first gained general circulation, UNIX REVIEW decided to send Mark Compton to Norman's office at UC San Diego to see if he's still saying all those nasty things. He is.

As for who Don Norman is: he serves as director of the UCSD program in cognitive science, a field he helped pioneer. He has authored seven books and countless papers, and has degrees in Electrical Engineering from MIT and the University of Pennsylvania. His doctorate in

SPEAKS OUT

An interview with the UNIX system's
best known critic

Photos by G. Pasha Turley

Psychology was earned at the University of Pennsylvania. His major research interests are to understand (and model) the human mind. But he also directs a large research group on human/computer interaction.

REVIEW: *Your article, "The Trouble With UNIX," has come to be regarded as a landmark document. What do you feel were the piece's key points?*

NORMAN: The interface for UNIX is simply horrid. It's a very uncomfortable system. Even experts make errors — often really serious errors. Beginners have a very difficult time learning it.

There were a number of different factors that combined to lead me to write that article. First, UNIX at the time was not widely known except among a fairly small band of people in universities and research laboratories. But among that band, wow: UNIX was *the* operating system. It was the wave of the future. It was all that a proper operating system ought to be and ever could be. Yet I had trouble with it.

So the first point was that UNIX was highly touted as the savior to our problems. The second point was that I continually made mistakes — serious mistakes — and I'm not a beginner. I have been using computers since the Univac I. Those were the days when you could walk inside the CPU. We had a thousand words of memory in the machine, and I programmed in the machine language numbers. Anyway, I've been around machines since then. I worked for Remington-Rand Univac in Minneapolis and helped design some of the very first transistor computers. So I'm not a

novice, and yet I was making very serious errors, including the famous `rm *` error. What's more, I found I wasn't alone. Even the UNIX gurus were making these of errors, but they would shrug them off. They would laugh. Even the manual shrugs them off.

The third point is that I'm a cognitive psychologist, and what I'm really interested in is how the mind works and the theory of how the mind works. Most of my efforts have been developed towards understanding mental processes and trying to build a comprehensive theory. But I'm an engineer also, a graduate of MIT in electrical engineering, with a background in engineering, and I've always been interested in how things work and in the devices we build. I got interested in accidents—industrial accidents, aircraft accidents, nuclear power plant accidents, and through examining these, I came up with an understanding of how people make errors. To my surprise, the major cause of error proved to be bad design.

REVIEW: *Not operator error?*

NORMAN: Oh, it's called human error. But almost always, it is really bad design.

I said there were three factors: 1) UNIX is highly touted; 2) yet I observed that I was continually making mistakes, and it wasn't just that I was making mistakes — even the manual commented on them; and 3) I was particularly interested in the mistakes, and had discovered already in the examination of nuclear power plant accidents that design is often faulty. I found the design of UNIX to be faulty. I was a bit tired of everyone telling me how wonderful it was. So, one weekend when I should

have been doing something better, I made yet one more error with UNIX, and I rattled off this piece. It only took a few hours. That's how "The Trouble with UNIX" began.

REVIEW: *So, had it not been for that fit of pique, the article may never have been written, despite the fact that the observations had been rumbling in your mind for some time.*

NORMAN: That's absolutely right. I wrote the article in one night, but then what was I going to do with it? I sent it around to some of my friends and colleagues and UNIX gurus to see what they thought. All the people at my laboratory thought it was just fine. It captured their thoughts exactly. Of course, I'm director of the laboratory, so their comments may not have been unbiased. Some of the other people on campus, though, had much stronger ideas. One person said, "Well, if you can't use UNIX without all those errors, you don't deserve to be using it."

Then somebody sneaked into my files. We run an open system and I've always encouraged people to look through my files, to find the papers I'm writing and read them, and even send me comments on them. This person went into my files and took the UNIX article and circulated it across *netnews*. Obviously ashamed of what had been done, they disguised the sending location. That's how I became well known, because now suddenly this electronic version of my paper was circulated all around the country.

REVIEW: *What channels had you initially intended to circulate it in?*

NORMAN: No channel. One of the

UNIX*

OPERATING SYSTEM

- Advanced Full-Screen Editor
- PC DOS File Transfer
- 8087 Co-Processor Support
- Queuing System
- IBM Support

Now you can have a complete UNIX System III implementation on most models of the IBM PC.

The IBM Personal Computer Interactive Executive[†] (PC/IX) offers you tools like the C language, programmer's workbench, communication facilities, a text processing system and much more. It's also a multitasking system that offers you the same facilities found on larger UNIX operating systems.

And PC/IX incorporates many significant enhancements. For example, a new full-screen editor helps you program more effectively. It offers such advanced features as windowing, function-key editing, the ability to

execute commands from the editor, and automatic file backup. Beginning users of the editor can rely on a variety of "help" screens.

If you've been using PC DOS, you'll be glad to hear that it can co-reside with PC/IX. And that you can transfer files between the two systems.

PC/IX is specifically designed to take full advantage of the architecture of the IBM PC.

It transparently supports the 8087 Math Co-Processor. And PC/IX automates the management of input/output streams through a multipurpose queuing and spooling system.

IBM has other software for PC/IX

IBM'S PC/IX

available, too, including INfort (a FORTRAN compiler with programming tools) and INmail/INnet/FTP (an electronic mail and file transfer facility)†

And because PC/IX is from IBM, you get IBM's quality documentation, as well as IBM service and support.

PC/IX requires 256KB of memory on an IBM PC with fixed disk or an IBM PC XT, PC XT/370 or PC AT.

You'll find PC/IX at your nearest IBM Product Center or your Authorized IBM Personal Computer Dealer. To order, or for more information and the latest list of PC/IX application software, call IBM toll free at 1 800 IBM-2468, and ask for the PC Software Department,

Ext. 393. Or call your IBM marketing representative.



IBM Direct PC/IX 11-84
PC Software Dept. 3E/393
1 Culver Road, Dayton, NJ 08810

Please send me more information on PC/IX.

Name _____

Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Phone _____

*UNIX is a trademark of AT&T Bell Laboratories. PC/IX is based on UNIX System III, which is licensed to IBM by AT&T Technologies, Inc.
†Developed for IBM by INTERACTIVE Systems Corp.

Circle No. 24 on Inquiry Card



things I do is write. I publish a lot. I've written a number of books. I make a point of writing an average of two hours a day, a couple of pages a day. I just wrote this. I had no particular channel in mind.

REVIEW: *You were going to just put it online and show it, perhaps, within the lab?*

NORMAN: I wrote it because I should have been doing something else, and this was more fun. It just came, just flowed. That's how I write most things. I don't worry about what to do with it. Things like that suggest themselves later. Then I said, "OK, I have it. What'll I do with it?" So I showed it around the lab. Then it went around the country. It wasn't until several months later that I thought that maybe I should publish it. Then I finally figured out where to publish it.

REVIEW: *The response was a bit vocal...*

NORMAN: The response was a *bit* vocal? That's a mild understatement.

REVIEW: *Did that surprise you?*

NORMAN: A number of my students had graduated and were in places like Bell Laboratories, and one student working for John Mashey reported that he had to disguise himself, and would go around saying, "Don Norman, who? I haven't heard of Don Norman. Is he a psychologist?" Amazing.

REVIEW: *The article had a very interested audience at Bell Labs.*

NORMAN: Yes. In fact, I started receiving computer mail. A few messages at first, and then a lot of messages. When I printed it all out, it was 35 single-spaced pages of messages.

REVIEW: *That included the response from Doug McIlroy?*

NORMAN: No. McIlroy never sent his response to me. I have never

seen the entire response. I managed to see a little section here and there. From what I can gather, it says, "Who is this Don Norman, and why is he saying those nasty things about me?"

REVIEW: *Or at least, "Why is he saying those nasty things about my system?"*

NORMAN: When I wrote the article, I tried to say, "Look. Just try to understand what the issues



One weekend when I should have been doing something better, I made yet one more error with UNIX.

are." If you look at the history of UNIX, you can understand what happened. First of all, nobody designed UNIX to have a lousy interface; it just happened. Second of all, it was designed as a personal operating system for a few people, and as a result, the computer

science and the philosophy of design is truly elegant. Last, it was designed on teletypes, 10 character-per-second teletypes, and a lot of the terseness of the interaction is a result of that. You sure didn't want to wait any more than you had to. One extra character was a pain. In fact, in my recent work on the design of interfaces, I have developed a quantitative system for examining design tradeoffs. These are principles of design that show how a system will degrade or improve from the user's point of view, as you change the technology.

One clear tradeoff is information versus speed. As the baud rate goes down, it's essential to cut the message size — to zero if you can. That's basically UNIX. It started at 110 baud, but by the time UNIX was circulated, we had 9600 baud screens, screens that had character addressing and bit-mapping. UNIX couldn't support this.

One more point: the power of the UNIX operating system is a weakness. The power is that it's extensible. Anybody can add a program. Anybody can change the shell. Anybody can add anything they like. But the user cannot distinguish something that some arbitrary person added from something that's fundamental to the kernel. There is no way the user can tell the difference. As a result, hundreds, maybe thousands of people have added new things to UNIX, each person having their own idea about how to do it, which makes the system completely inconsistent.

The naming structure, the abbreviation, the flag structure, the special characters, the editing commands — everything is unique for each separate program; That's one of the things that makes it so difficult. The famous **rm *** error occurs in part because, one part of the system, the shell,

expands the star, but another part of the system, the command, uses it. And since the thing that does the expansion is separate from the thing that uses it, there is no way to say, "Hey, did you realize you're zapping every file?"

Although I applaud the designers, they made at least one serious error — a lack of thought about the interface. They failed to provide tools for the interface. They provided tools for lots of things, but not for the interface. When you write a program, you have to write all these calls to handle the arguments and to handle the escapes. First of all, you hate it. That's not creative programming. It's one of these things you have to do. It's like writing documentation. Who wants to do it? Second, because there's no tool that enforces a standard, each time you do it, you make up something that seems sensible at the moment, but may be completely incompatible with everything else. With proper tools, you can get standardization. Bell Labs and AT&T (and all of the other organizations that have spun off, with new names every week) recognizes this problem and tries to standardize. But they can't. You want a standard to be upward compatible, and that means that you're stuck with most of what's already there. It's too late to impose standards. So I think we're stuck forever with horrible UNIX.

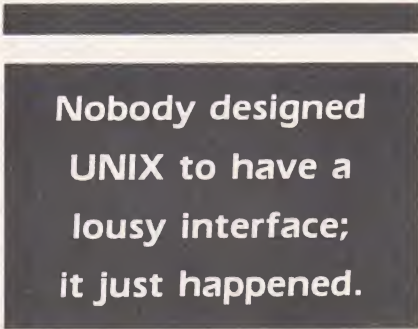
REVIEW: *With no hope of any improvement or with hope for only modest improvement?*

NORMAN: Let me quote a standard policy of writing software. If you write software, you figure out where the problems are and you figure out what expansions you'd like, but you don't patch and patch and patch. If you do, you end up with a program that nobody can understand, not to mention one that doesn't really

work very well. The proper thing to do is to throw away what you had before and start over. Redo it. Because then you can do it elegantly and cleanly. I think that's what has to happen with UNIX. It has been a very valuable learning lesson. But now we should throw it away and do it right.

REVIEW: *Or front-end it, perhaps?*

NORMAN: Front-ending is just another patch. You can make patches look alright as long as you want a small environment. If



Nobody designed
UNIX to have a
lousy interface;
it just happened.

you'd like a big environment, an environment as rich as UNIX, with all the rich possibilities, no single front end will suffice.

UNIX suffers from at least two serious flaws. In any sensible world, these would kill it. Of course, the world isn't sensible. What flaws, you ask? Well, to simplify: 1) UNIX is a kludge — it's time to start over; and 2) UNIX is based around the technology of the 1960s — it is now the 1980s. Let me explain.

First, although UNIX had an elegant start, it has been worked on by so many people, over such a long period of time, with so many discrepant views, that today it is a mess. As Bill Joy said in his interview in the August *UNIX REVIEW*, "You start out with a clean concept and then sort of accrete guano. It lands on you and sticks and you can't do anything about it, really." He was talking

about the **vi** text editor, but the same principle applies to UNIX.

Second, UNIX was designed for an earlier era, with an earlier technology, the technology of the 1960s. **Termcap** and a few other enhancements got it up to the 1970s. But, in case you haven't noticed, it is now the 1980s. The technology of today is large, bit-mapped screens, with windows, pop-up menus and pointing devices (note I didn't say this is the correct or best way, just today's way — but it is certainly better than your old 24-line by 80-character terminal). Again, to quote Bill Joy, "The fundamental tension in UNIX that I think AT&T doesn't understand is that everyone is going to have a bit-map." I agree.

UNIX has aged gracefully, but it is now time to retire. It was a valuable tool and a worthwhile experiment. Now it is time to launch a worthy successor. Alas, there are only pretenders, no real contenders. There is nothing to take its place. Not yet. UNIX, with all its warts and surface flaws is still the best general purpose operating system around. Too bad. And worse, everyone wants UNIX today, especially those who don't know what it is.

REVIEW: *Let me get back, if I might, to the response to your article. Would you be able to characterize the response?*

NORMAN: I think I would divide it into three parts. First of all, there are those who derided me, who said I had no business using machines if I was that incompetent — that if anybody ever made a mistake, it was their own fault, etc. Second, there were the defensive responses. Those came from people who were intimately involved with the development of UNIX, and in many ways I don't blame them.

Third, there was a group of



NORMAN INTERVIEW

people who applauded me. I made a lot of money out of that piece, in the sense that it brought me a fair amount of consulting. I was flown here and there to look at systems so that I could nod wisely and say, "It's a disgrace."

I got invited by Bell Labs to make a grand lecture tour of about four labs. They all came to boo, but actually it was interesting. The response was very good from the audience. On the whole, they appreciated what I said, and there was always a question period afterwards in some private room, where people would get together and talk to me. I thought I was going to be mobbed. But in fact, all they did was tell me about their problems and the difficulties of the system. So it turned out there was a tremendous amount of support, even at Bell Labs.

REVIEW: *In addition to raising hackles, it sounds as though you're saying a certain awareness was raised. What are the manifestations of that heightened awareness? What have you seen resulting from the article?*

NORMAN: The real point I want to get across is that you need to worry about the person using the system, not about the system programmers. Although, you know, for that matter, systems programmers make the same kinds of errors, too — or make their own special kinds of errors. You need to worry about the users, from the users' point of view, and provide them with proper tools. My message has been said before. I was certainly not the first person to say it. But perhaps the way in which I said it brought it to people's attention.

I don't want to take credit for any new movement, but a new spirit has certainly arisen that acknowledges that the user interface is a very important part of the system.

The interface is *the* most impor-

tant part of the system. It's the part the user sees and interacts with, and as far as the user is concerned, that's all there is — the interface. I think the design of a system ought to start with the interface. You need a specialist for the interface who is half-psychologist, half computer scientist. The interface should be the first thing to be designed. This philosophy hasn't completely caught on yet, but it is surprising



Everyone wants
UNIX today,
especially those
who don't know
what it is.

how sympathetic the world is becoming. We have groups at Bell Laboratories, at Xerox Parc, at Apple and at other major companies thinking about how to design from the interface down into the system.

So, what is the result of that article? My name has become

known, yes. That isn't very critical or important. I was already a well-known psychologist. There have been very few changes made. I think the world is going to change. I think we're going to see computer systems that are really usable, that are good tools for all levels of users, from the novice to the wizard.

Today the very best tools are available on the LISP machines, machines that have working environments. It's not the windows, it's not the high resolution bitmap screen, and it's not the mouse that makes for a livable system. What's important is that on one system you have the interpreter, the compiler, the debugger, the editor, the operating system, the mail system, the utilities — all using a common language, a common mode of interaction, and all available at any instant. A large amount of the credit for that goes to Warren Teitelman for making Interlisp a real environment where you want to live and do everything.

REVIEW: *Given that you feel that the design of the system should start with the interface and that UNIX doesn't pass muster, I find it surprising that you continue to use UNIX. Two questions stem from that. One is, why? Secondly, you obviously have some affinity for UNIX. In fact, you spoke about it in the article itself, saying that you would think of using no other system. Does that mean that the language you used in the article was largely rhetorical?*

NORMAN: I will admit to a certain amount of rhetorical overkill. But what are the alternatives to UNIX? There aren't any. Much as I dislike many of the features, and much as I think a much superior job can be done, there aren't any alternatives. I'm a scientist. I use VAX-en and Suns. UNIX is clearly the preferred operating system.

I think one of the best operating systems of the early era

was TENEX and Tops Twenty, which was derived from TENEX. It had a very effective command completion and help system. It allowed you to control how much information the system would give you. But the Tops Twenty system and the TENEX system were available only on the DEC 10 and 20 computers. We didn't have that. We went the DEC 11 route. And there, the best operating system clearly was UNIX. Now with our Sun workstations, we again have the choice of UNIX or nothing. There really isn't an alternative.

REVIEW: *In McIlroy's response to your article, he charges that your points "apply only to the hoary old 1975 edition of UNIX." Do you think most if not all of your points would have applied equally well to Version 7?*

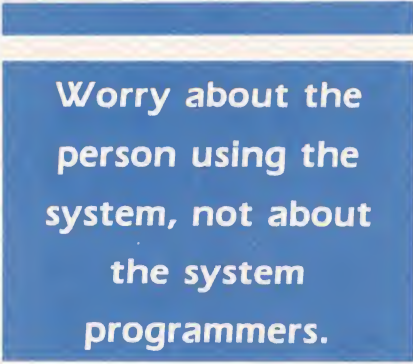
NORMAN: I don't agree with that part of his response. Yes, the details of what I said may indeed differ from system to system, but the basic philosophy and the incompatibilities and the difficulties with UNIX remain to this day. I'm not familiar with Version 7. We use 4.2 BSD, but from all I know, the difficulties are there in Version 7 as well, and I think they are fundamental.

REVIEW: *Despite your feelings about the interface of UNIX, you must make do with it. Does that suggest that you have made a number of modifications to improve the interface for yourself and your comrades? If so, what sorts of modifications have you made?*

NORMAN: I hang my head in shame. I must admit we have done precious little to improve this fundamental aspect of UNIX despite the fact that we have a crew of talented people, and moreover a fairly large research project whose entire function is to understand how to build better

human computer interfaces. But I guess we just felt it wasn't appropriate for us to spend all of our time redoing the UNIX system. We'd rather try to understand what interfaces are about, understand the principles, so that in the future these principles might guide design.

One of our students, Gary Perlman, did devise a very interesting interface for UNIX, *menunix*, a menu-driven UNIX with some clever characteristics. It allows you to explore the space



Worry about the person using the system, not about the system programmers.

fairly quickly. There were always two menus simultaneously available, one serving as a guide to the set of commands in the workbench, and the other allowing you to move up and down the file structure. But it was really a demonstration program. Although most of us used it for an hour or two, nobody really used it seriously. Gary moved on to Bell Labs, where he worked on developing UNIX tools, and produced a number of things useful for standardization, such as a standard argument handler. But again, these are patches on top of the system. Gary is now at the Wang Institute, where I hope he can teach the next generation of software designers about the principles of interfaces.

We have done a large number of small things, like inventing a fair number of things to aid the user, such as standard basic login shells, standardized **aliases** and

something we call a cliché generator that reminds you how to open your file, how to check the flags, how to do a check for signals. You ask for a cliché, and into the editor will pop a standard way of doing things with some comments, so that you can either accept it or modify the existing code. In a sense, we're programming by example. That's just one of a fairly large number of very small tools.

REVIEW: *When I walked into your office you were playing with a program generated by your group called "Notepad." Can you say something about that?*

NORMAN: It's part of our general philosophy to understand how people are really using things, and then to try to build systems that support that. We're not out to improve UNIX, but to try to improve people's interactions in general. One thing we observed was that people don't stick to a task for long; they get interrupted, or they go work on something else for awhile. We are trying to build systems that allow you to be interrupted, to find the new information you need, and then remind you of where you were. By the way, reminding you of where you were isn't just about to, say, open the editor to exactly the same spot. The system also has to show the person what he was thinking, what was present, what the whole state was. The **vi** editor, for example, hides lots of critical variables, like all of its buffers, and this means you're really missing a lot of the context unless the system makes this visible as actual reminders of what you were doing. No past system was designed to allow you to interrupt what you were doing instantly and quickly get what was essentially a new notepad on which you could write your thoughts. Don't worry about what the name of your thoughts should be or where they should be



filed, or how they should be structured. Just get them down. Then, when you're finished, you can take the time to organize them properly. In the meantime, you have a list of the things you had been doing, which you could easily resume. This, too, is a demonstration project, developed by Allen Cypher.

REVIEW: *Is it a demonstration project of how machines can emulate the human mind, or of how they can facilitate the mind?*

NORMAN: We don't need machines to emulate the human mind. We don't need more human minds. We have enough human minds. What we need are machines that can complement the mind.

REVIEW: *Nevertheless, your system very much depends on*

how the human mind works.

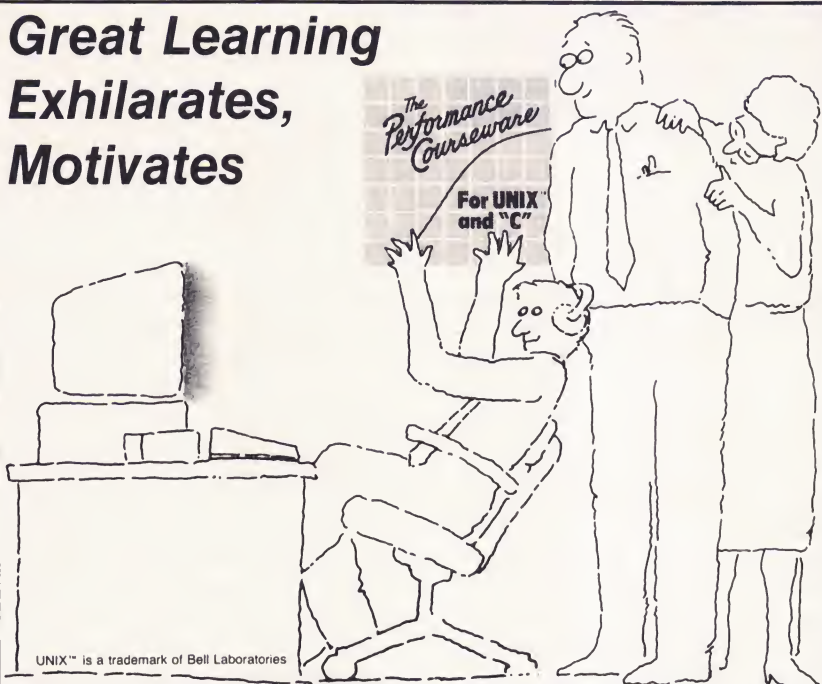
NORMAN: Of course. Absolutely. But what I want to build is tools. We had a debate in our laboratory for a while: tools versus intelligence. Some groups feel that tools should build an interface that is artificially intelligent, that read your mind, read your intentions. This simply won't work. Ever. It's not that we don't know enough, it's that we will never know enough. Half the time, I don't know myself what I'm trying to accomplish. How is a computer supposed to know? Take Bill Budge's pinball construction set. That's the way I think programs ought to be — tools for constructing whatever you wish to do. With that construction set my four-year-old son and I have constructed a variety of games. Some

of them have nothing to do with pinball. We were provided with a powerful tool for constructing things that, note, never griped about syntax errors. It is impossible to make a syntax error with a construction tool. Can you imagine using a UNIX operating system where syntax errors were not possible?

REVIEW: *Or where the system told you what the syntax error was?*

NORMAN: No! Better yet, a system shouldn't have to explain what you did wrong. The system should be so natural to use that you couldn't possibly provide a syntax error. Let's say, for instance, that you're sitting in a restaurant, eating with a knife and fork. Can you make a syntax error with a knife? No. I want the

**Great Learning
Exhilarates,
Motivates**



UNIX™ is a trademark of Bell Laboratories

And it saves you money.

USER TRAINING CORPORATION

CALL NOW (408) 370-9710

591 W. Hamilton Ave. • Campbell, CA 95008

Circle No. 14 on Inquiry Card

UNIX Support



UNIX Telephone Helpline

Help with:
General UNIX Questions
Emphasis on **Nroff**, **Troff**,
ed, **vi**, **sed** and **csh**.

We also offer
Consultant Referral Service.

(415) 731-2978

UNIX Helpline, from Technical Type & Composition

Circle No. 15 on Inquiry Card

UNIXTM & '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 customized to your system and to specific groups within your organization.
- Video-Based Training for consistent training that is always available at your location.
- Interactive Videodisc Training, which dynamically tailors courses to the individual—from novice to expert programmer.

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

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

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

TM UNIX is a trademark of Bell Laboratories.

COMPUTER TECHNOLOGY GROUP

Telemedia, Inc.

310 S. Michigan Ave., Chicago, IL 60604



operating system to be my tool, and indeed I'm engaged in a project to develop a system of programming through direct manipulation that will allow me to move objects around on the screen and interconnect them by drawing lines. That is how I will program. There will be no syntax errors.

REVIEW: *You indicate in the Datamation article that the user interface must be improved if UNIX is to appeal to the casual user and become a general system. Much of what you said earlier in this interview, though, suggests that you're not sure that that's possible without trashing the system and starting from scratch.*

NORMAN: Or restricting its use. You can clearly do a really superior interface for a limited domain of operations. I'm not certain you can for a large one.

REVIEW: *In any event, how might user interface improvements service experienced users, who don't really care about whether or not the system gains general acceptance?*

NORMAN: Experienced users do care deeply about whether or not they can use the system efficiently. I want to make one point perfectly clear. I am for systems that are usable by everybody. I want the experts to be comfortable since most users of systems are experts in the sense that they use the system frequently. But I also want to support the inexperienced user, the casual user and the experienced user who goes away for a month and comes back and has trouble. I am sometimes accused of trying to make systems really easy for beginners at the expense of the expert, and that's just not true. UNIX is also a problem for the experts. Take a look at any UNIX wizard. What do they have beside them? A stack of manuals

a foot high. And it isn't because they have to use some unknown program here and there, and have to understand its workings. It's because they have to remind themselves of the arguments and the syntax. That should not be necessary.

People claim that I'm trying to build user friendly systems. I would love to say yes. Unfortunately, the term "user friendly" has been taken away from us. What is meant by user friendly today is a perversion of that term.

REVIEW: *What does user friendly mean to you? Or what is a better term?*

NORMAN: Today, user friendly seems to mean excessive hand-holding. We built a system once, a message system, that tried to give you help and advice. One of my colleagues once sent me a note that said, "Once upon a time I had a girlfriend, and this girlfriend used to follow me around all the time, all day, all night, and she used to tell me how much she loved me, and she would call me up to tell me she loved me, and she would come by at all hours of the day or night to tell me how much she loved me. I couldn't take it anymore. I finally got rid of this girlfriend. Well, when I use this message system, I am reminded of the girlfriend. If you don't change the message system, I will get rid of the message system just as I got rid of the girlfriend." Yes, that's the problem of the user friendly system. It's sometimes useful to have heavy prompting the first time through a process. But that's it, and I don't call that user friendly. I call it perverse. It violates all my principles.

I don't have a good term to substitute for "user friendly." We call our research project at UCSD "UCSD," for User Centered System Design. We want the focus to be the user and the usability at

all levels. That's probably going to mean a system, by the way, that is modular, expandable, extensible, and available in several versions. I don't think you can make a single system that's unchangeable, that everybody can use with equal efficiency.

We have a bunch of slogans. I like slogans. One of the slogans is, "There are no correct answers. There are only tradeoffs." When I'm asked what the proper design of an interface is, I trot out that slogan. There is no proper design. Each particular design, each feature you put in, is a tradeoff. It'll have some virtues, it'll have some deficits, and what the designer has to do is be aware of those, and then make choices. Ideally, you provide alternatives so that users aren't stuck with a single way of doing things. You make the system extensible. You try to make it easy to go from one alternative to another and for people to remind themselves of what is available. If you want a pointing system with menus, well, remember that pointing is sometimes awkward — whether you use a light pen, touch the screen, or use a mouse. It's not always a desirable solution. You should also provide a command language. Many systems do this.

REVIEW: *When you say flexibility, is that synonymous with lots of alternatives? One of the complaints Bill Joy has about vi, for instance, is that...*

NORMAN: Aah! The vi editor? No, that's not flexibility. That's the kitchen sink.

Vi and emacs are editors that have the kitchen sink thrown in. Everything you can imagine is there, in some archaic command or another, such that it's impossible to learn them all or use them all. I don't know of any expert who uses all of the command set; you establish your own private com-

mand set.

The problem with *vi*, though, is not so much that it has so many commands as that they are un-systematic and have no good underlying model. Here's my point about design. The user is going to make up a mental model of what he thinks is going on in the system. That mental model is used to predict what will happen. Just like when you pour cream from a creamer, you can see what's going on — you can see the level, you can see the angle. Can you imagine how it would be to use a creamer that wasn't visible? Let's say you simply pushed a button that caused the creamer to rotate. You would have no idea of how far it was rotating, or of how much cream had been poured. It would all be invisible to you. You

can't use devices like that. There has to be some visible operation taking place. Most of today's computers are invisible. You don't know what's going on. You can't see anything happening.

REVIEW: *Sometimes it doesn't even have the courtesy of telling you what's happening.*

NORMAN: Right. And that makes it very hard to develop a good mental model of the system. If you can develop a good mental model, not only can you understand what is happening, but you can predict new things that you can do, and you can understand problems when they crop up because you can derive what must have happened. That's all well and good; but where does the user get this model? You get it from the system

because you interact with the system. And that system, therefore, must make visible what's present. I call what's visible in the system the "physical image," or the "system image."

How does the system get built? Well, you have a designer. The designer starts off with what I call a conceptual model. Sometimes it's a neat, clean, systematic model. The whole goal is to get the user to use that same model. But how does the user get it? The designer doesn't personally instruct the user. The user interacts with the system and tries to figure out what is going on so the designer had better always have the user in mind. The system should make visible what it is doing in a way that is consistent with

Continued to Page 94

TRAINING

SYSTEM V

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

WHY AT&T FOR UNIX SYSTEM TRAINING?

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

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

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

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

AT&T COURSES OFFER:

The same training and methods we use to

teach the UNIX System to our own people.

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

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

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

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

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





Illustration by Heda Majlessi

WRITING A NEW SCRIPT

UNIX and computer-assisted instruction

by Vanessa Schnatmeier

It's no simple task to write training programs, with or without the power of UNIX. For one thing, trainers must always consider their audience, which may be comprised of people who are less than enchanted by the topic at hand. Consider, for example, the plight of business hostages forced by their supervisors to probe into the dark secrets of a new word processing system. And what if the trainer isn't producing a high-powered course for mass distribution, but is "merely" trying with limited time and funds to develop a simple tutorial for the shop or office?

Enter the concept of computer-aided instruction (CAI), also known as computer-managed training (CMT) or computer-based training (CBT). This plethora of acronyms is a clue that this area means different things to different people.

Educational software, or courseware, often suffers in translation from the teacher's mind to the programmer's code. We've all seen tutorials that have proved to be glorified page-turners, electronic "books" we could have bought more cheaply in hardcover. No one wants to be responsible for writing such dull material. The ideal solution would

be a system for generating software that is easy for educators to write in and equally easy for students to use.

The idea of authoring languages and systems is to help CAI authors avoid drudgery and reinventing the wheel. Authoring languages are really sets of instructions that deal with programming at an easy-to-learn level. Authoring systems are packages of tools that aid instructors in preparing, scheduling, administering and grading a computer-supported course, an editor to maintain files, an interpreter to present the course material to the student, and information management tools that record student performance and help the instructor manipulate background information and records.

Not everyone agrees with this approach. For instance, Paul F. Merrill stated in the July 1982 issue of *Creative Computing* that CAI languages and authoring systems were oversimplified and too restrictive. Not only that, he said, but you have to call in routines from an underlying language, such as BASIC, Pascal or C, to do some of the tricky work. Why not write using C and eliminate the middleman?



Well and good, but believe it or not, not every educator or trainer is eager to learn a complicated computer language in order to construct a training program. The advantage of the CAI languages and authoring systems to be discussed here is that they sharply reduce the number of commands and concepts a computer novice must grasp initially, yet leave a window open for educators to fully utilize UNIX capabilities.

BASIC PREMISES

Barry Moritz, president of Measurement Concept Corporation, recently explained to a panel at UNIX Systems Expo '84 his concept of the basic requirements for a good CAI system. His first requirement is that the system be able to take advantage of various teaching models. Any good system will have a flexible interface, so that everyone involved, from courseware author to teacher and student, can participate.

The second requirement is adherence to standards, currently a major concern among people working in the UNIX environment. Standards mean commonality among distant and disparate groups: a person in Japan should be able to talk to a person in Europe, or a West Coaster to an East Coaster. Standards also loom large from an economic perspective, since courseware can be very costly to produce. "If it's possible to produce courseware within a well-defined standard that is generally acceptable, then it will find much wider use, no matter whose box it's being run on," Moritz explained.

From standards, portability naturally develops. This means not only portability with respect to software, but portability of systems across audiences. Moritz gave the example of a set of tools

appropriate not only for a foreign-born person trying to learn English, but also for an Anglophone who wishes to learn French as a second language.

A good CAI system must permit dynamic interaction. Readers are familiar with looking at a page or frame, absorbing its contents, and moving on by turning the page or performing a programmed learning task. "Is this the right way, or is this a leftover from the printed word?" asked Moritz. We may have all grown up in that environment and therefore know best how to produce material for that environment, he emphasized, but it should not necessarily be the only way. CAI specialists are

It's no simple task to write training programs.

closely examining how to use computer tools to produce more interactive and dynamic information displays and keyboard input.

A crucial question for an authoring language or CAI system is how many levels of language complexity it can encompass. When most languages get complex, they also become very difficult to use. The standard solution, of course, is to point at the special group of people called programmers and let them handle the complex logical languages. However, walking that path removes from the loop some of the most important people in the educational process: the knowledge area expert and the actual teacher. "How do we get these people into the loop? And to what extent can they use the facilities

of the language?" Moritz asked. It's clear that that level of user can operate a frame system and produce some type of courseware, but a much better goal would be to develop a growth-oriented and open-ended language that would permit inexperienced users to grow along with the system and expand their abilities as courseware writers.

PILOT

Authoring languages aren't exactly new on the CAI scene. Pilot, one of the ancestral CAI languages, was developed in the late 1960s by Dr. John Starkweather, who was working at the time at the University of California at San Francisco. Thos Sumner, author of a Pilot derivative called C-Pilot, told about his first reaction to Pilot years ago: "When I first saw it, I said, 'Gee, how can you do anything with that?' It was so limited and so simple. And then he started showing me some of the things he could do with it. It was amazing how powerful it was — the careful selection of what it could do with only eight primitive commands accomplished a great deal."

Pilot and its derivatives typically provide about a dozen commands for instructors to prepare dialogues with students. Some Pilot commands include: TEXT, to write text at the student's terminal; ACCEPT, to read the response the student types; MATCH, to compare the student's response with a list of incorrect (or correct) responses and set a flag if there is a match; and JUMP, to transfer control to the statement following a specified program label. Commands like these are all an instructor will need to produce a lesson — to put text on the screen, offer a response, hint or prompt if the student requires it, and branch on to another part of

the lesson when necessary.

Starkweather first brought Pilot up on an IBM 360 Model 50, and Pilot's first micro implementation was on a Datapoint computer. Pilot spread quickly, and by 1973, enough versions of Pilot had cropped up on systems around the country that a standards meeting was convened. The meeting produced a set of standards called Core Pilot, the touchstone for determining whether a CAI language could be considered a true child of Pilot.

In 1978, two Western Washington University researchers, Larry Kheriaty and George Gerhold, expanded Pilot's range so that it could handle floating-point arithmetic and standard mathematical functions, as well as the original integers and text. This version became known as Common Pilot. "They did a lot of work in moving Pilot in the direction of generality," said Sumner.

Some installations now use Pilot in place of shell scripts, according to Sumner, because its control structures are easier for some things, and large, complicated dialogues take less time to write with Pilot than with shell scripts. Another advantage of Pilot is that it's two to three times as fast for things requiring a lot of shell processing.

"One of the things it's used for where I'm located at UCSF is producing the help system for providing quick online aid for use of the system," Sumner said. "The Pilot program administers a hierarchy of textfiles, other Pilot programs and so forth, to respond to help requests."

Sumner's own version of Pilot, C-Pilot, is a strict superset of Common Pilot. It runs only on UNIX systems, ranging from the IBM PC/XT running PC/IX to the AT&T 3B2, and others using AT&T and Berkeley versions of UNIX.

C-Pilot comes in handy for more than just CAI applications, he said — it's used for anything relying on "dialogue programming" or interaction programming. C-Pilot front-ends programs that are too complicated to instruct, presenting menus and gathering data. Questionnaires are another area where C-Pilot has shown its worth, Sumner said. "Just as you customize the path of a student through lesson material, you can customize the questionnaire being administered for research purposes to avoid saying, 'Well now, don't look at the next six questions, go to page five.' He's at page five."

CAST

CAST, Computer Assisted Self Training, is also a superscript of Pilot. CAST is a product of Measurement Concept Corporation, which decided to use UNIX as the environment for developing courseware in the late 1970s — though the company only recently brought its work to the commercial marketplace.

CAST is billed as a language, but it's actually an authoring system. CAST is implemented in C and, like C-Pilot, runs under several versions of UNIX on microcomputers and minicomputers. Therefore CAST can use any UNIX facilities by calling the operating system and using it in specialized areas — computation, statistics, text editing, graphics or file management. A chemistry instructor, for instance, could write a basic course in CAST and dive into special graphics software to draw a picture of a complex molecule.

Dr. John M. Morris, a senior associate at Measurement Concept, noted in an article on CAST for /usr/group that the language is designed to be terminal independent. This somewhat limits the use of features unique to specific

Announcing . . . PROFLEX™

A comprehensive integrated multi-user accounting and management information system. Written entirely in the Progress® database management language from Data Language Corporation. Easily modified source code provided.

Accounting Modules: Cash Management, General Ledger, Accounts Payable, Accounts Receivable, Job Costing, Fixed Assets, Orders and Inventory, Payroll.
Management Modules: Administration and Operations (including tasks, documents, scheduling, dispatching, activities, communications), Sales, Rentals, Personnel, Investments, Trusts, Taxes.
Professional Modules: Medical, Legal, Accountant.

Look for engineering, scientific, and statistical modules to come.

Also available: NIAL AI language on Fortune 32:16. Better than LISP or Prolog.

STARWOOD GROUP, 17707 Blanco, San Antonio, TX 78232, 512/592-7735

Circle No. 18 on Inquiry Card

UNIX™ OPPORTUNITIES

CAMBRIDGE - BOSTON
RT. 128

"The secret of getting ahead is . . .
Getting Started" — Mark Twain.


(EQUITY)

Start-Up and established organizations are offering chance of a lifetime — Professional — Financial — Personal Growth opportunities

• Call (617) 848-5188 •

Jim Barry

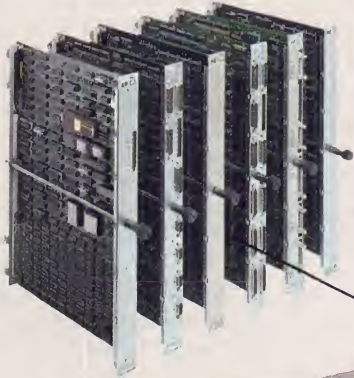
To discuss your goals or
send your resume

 ADVANCED ERGONOMIC
MANAGEMENT

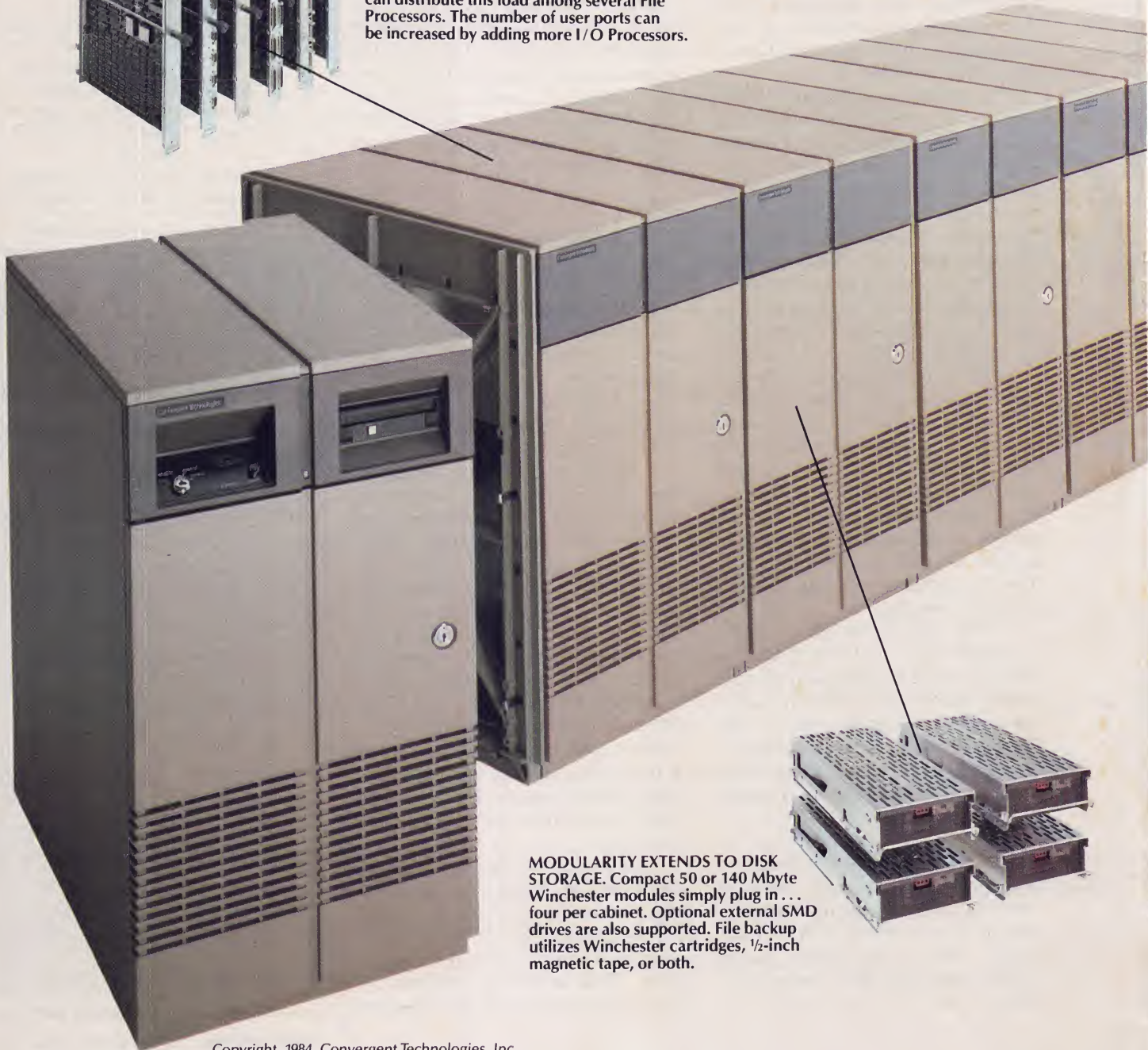
Software & Engineering
Personnel Consultants
872 Massachusetts Ave.
Suite 1-3
Cambridge, Ma 02139

Circle No. 17 on Inquiry Card

IT WILL GROW ON YOU.



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



MODULARITY EXTENDS TO DISK STORAGE. Compact 50 or 140 Mbyte Winchester modules simply plug in . . . four per cabinet. Optional external SMD drives are also supported. File backup utilizes Winchester cartridges, 1/2-inch magnetic tape, or both.

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

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

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

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

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

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

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

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

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

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

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

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

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

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



Convergent Technologies

Where great ideas come together

MiniFrame™: the entry-level multiuser UNIX system.

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



MiniFrame and MegaFrame are trademarks of Convergent Technologies, Inc. UNIX is a trademark of Bell Telephone Laboratories, Inc., VAX is a trademark of Digital Equipment Corp.



terminals, but still permits the user to take advantage of general features such as a split screen capability. Version 2.0 of CAST allows the author/instructor to define up to 10 individually addressable screen areas; each area can contain either graphics or text.

CAST, like C-Pilot, can use other programs (called "scripts" in the CAST lexicon) as co-processors to accomplish a specific extension, such as controlling a special device. CAST lets authors define function keys, produce animation and lineprinter graphics, and mark a place in a lesson to return to later.

INSTRUCTIONAL WORKBENCH

As one might have expected, AT&T hasn't neglected this section of the UNIX market. There are economic reasons for this, of course, but CAI also fits remarkably well into AT&T's need for training in hundreds of highly specialized jobs.

Bob Frontini, who works in training for AT&T Technologies, in describing one virtue of AT&T's CAI product, Instructional Workbench, said, "The term that people at Bell Labs like to use is that Instructional Workbench is 'isomorphic' to C and shell, which basically means that I can hand a listing from Instructional Workbench to one of our C pro-

grammers and he can read it without having to read any documentation."

Like CAST, Instructional Workbench is an interpreter, a delivery system, an administration system and an authoring system. Under the authoring system, users create not programs but "topics." There are two ways to prepare courseware with this system. The hard way, said Frontini, is to call on executable UNIX commands, C commands, BASIC, Fortran or shell commands. The easy way is to use CAST's authoring system, called Compose. Compose has two levels of prompts, either terse screen prompts for experienced users, or more verbose prompts for the less experienced.

Compose also lets the author select among the available UNIX text editors to create topics and offers building blocks for courseware construction called templates. "A template is a prompting script that helps you construct something that can be reduced to a particular formula," such as a multiple choice question, a menu or a random question, said Frontini.

Authors can create Compose topics or free topics, which are UNIX files that contain topic language code; the two types of topic can be freely intermixed. As a result, he said, AT&T can develop a good 95 percent of its

training courses using Compose.

AT&T Technologies is currently using Instructional Workbench to support UNIX-based office automation products released by AT&T, as well as on its own network products, where the switching machines run software based on UNIX.

A VISION OF CAI

The goal of all these authoring languages and systems is to make individualized computer-assisted instruction easy for the teacher and the student. Students or trainees should be able to work with a lesson *when* they need it, going at their own pace, with access to patient, appropriate feedback. The teacher or instructor should be able to keep tabs on each individual, examining test scores and comparing them with other class members' work. The instructor should also be able to compare the class record with that of other classes. Another ideal situation would be one in which students could learn from a CAI language and eventually write their own programs/scripts/topics in the same language, in effect pulling themselves up into knowledge by their own bootstraps.

UNIX, already well suited to providing the programming environment, can provide these same online facilities to distant researchers, or to students studying on terminals at home, because of the operating system's communications resources. UNIX could thus be the key to putting courseware on its best course.

Vanessa Schnatmeier is an Oakland-based writer specializing in women's issues and computer marketing/education. She serves as a contributing editor for A+ magazine and is frequently published in PC and Computers & Electronics. ■

Image Network's XROFF, *right now*, prints *troff/ditroff* documents on:

- VAX
- Pyramid
- Plexus
- PDP 11's
- Integrated Solutions
- Amdahl
- IBM-PC
- 3B20
- System 5
- System III
- Berkeley 4.2
- IS/WB
- V 7
- UTS
- MS/DOS
- Xenix
- Xerox 2700
- Xerox 8700
- Xerox 9700
- Diablo ink jet
- Diablo thermal
- Dec LNO1s
- Compugraphic 8400
- APS-5 typesetter

at leading organizations from Berkeley to Murray Hill.

Call or write with *your* requirements!

Image Network, 770 Mahogany Lane, Sunnyvale CA 94086
(408)746-3754

This ad was set using Xroff on a Xerox 2700 laserprinter

Circle No. 80 on Inquiry Card

THE TRUTH ABOUT THE 3B2

Revised benchmark results

by Gene Dronek

In the August issue of *UNIX REVIEW*, we concluded an in-depth, two-part analysis of the AT&T 3B2 computer with a compendium of benchmark results. Almost immediately, two unrelated readers wrote us saying the benchmark figures were incorrect. They pointed out the 3B2 machine uses a 100 Hz clock rather than the 60 Hz clock we previously assumed.

Dave Kruffusz, a member of technical staff at AT&T Bell Laboratories, confirmed the 100 Hz clock rate, and helped us rerun the Aim Benchmarks with Hz = 100. In Figure 1, we supply the correct performance numbers.

As expected, all performance readings came up 67 percent ($100/60 = 50/30 = 1.67$) from the August runs.

Also mentioned in the article were comparative point scores for several systems, comparing overall performance. Based on the new results, the 3B2 earns 2099 points, versus the Convergent Technologies Megaframe's 2323 points, the DEC VAX 11/730's 1710 points and the Plexus 3000's 1621 points.

We thank the two alert readers for responding so quickly, and also wish to credit Ram Chelluri, Senior Engineer at AT&T for assisting with this errata.

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

ARITHMETIC INSTRUCTION TIMES (microseconds per op)				ARRAY SUBSCRIPT REFERENCES (microseconds)		
	short	long	float	double	short[]	long[]
+ add	2	1	3906	2128	7	6
× multiply	7	4	9259	7407		
/ divide	16	13	5348	3650		
MEMORY LOOP ACCESS TIMES (microseconds per byte)				FUNCTION REFERENCES (microseconds/ref)		
	read	write	copy	0-parameters funct()	1-parameter funct(i)	2-parameters funct(i,i)
CHAR type	944 ns	5	6	19	24	30
SHORT type	874 ns	3	4			
LONG type	448 ns	651 ns	2			
INPUT/OUTPUT RATES (bytes/sec)				PROCESS FORKS 22 per second		
	read	write	copy	SYSTEM KERNEL CALLS (calls per second and microseconds per call)		
DISK	39K	36K	36K	getpid() calls:	3 Kcalls/sec or	345 microseconds/call
PIPE			155K	sbrk(0) calls:	2 Kcalls/sec or	507 microseconds/call
RAM 1-byte			154K	create/close calls:	158 pairs/sec or	6329 microseconds/pair
RAM 4-byte			517K	umask(0) calls:	2 Kcalls/sec or	422 microseconds/call

Figure 1 — Corrected benchmarks for the 3B2.

The Power of a Mini, The Cost of a Micro, The Clout of UNIX.



Introducing The Perfect System Builder

With the VISUAL 2000, we've designed the perfect tool for system builders and integrators. A multi-user computer system that supports more terminals, offers more expandability, and gives you more configuration flexibility than any other system in its price range. Truth is, it outperforms a lot of higher priced systems as well.

What that means for you is an important competitive edge in delivering integrated business solutions. An edge that comes not just from the VISUAL 2000's surprisingly low price. But from its higher technology. Technology that gives you . . .

Power to support more on-line users in demanding applications—up to 16 independent terminals or workstations.

Flexibility to accommodate a remarkable variety of application requirements, thanks to an open architecture, compatibility with mainstream industry standards, and a full spectrum of configuration options.

Expandability that can take you further than any other system in its class—an unrivaled upgrade path to protect your system investment.

Put the Full Power of Intel's 286 in Your Systems

The Intel 286 is today's chip of choice for UNIX-based systems. Only the 286 gives you: on-chip memory management; an instruction set optimized for multi-tasking; pipelined architecture; and an optional 287 numeric coprocessor which can speed up floating point by a factor of 10. What do these features mean to the end user? Faster response time, more users supported, and lower system cost.

A "Mainframe" on a Single Board

The VISUAL 2000's single-board base-level design is the key to higher performance, greater reliability and lower cost. A single high-density board includes the 286 CPU; 512KB-2MB of RAM; controllers

for Winchester, floppy and streaming tape; an intelligent communications processor; six RS-232 ports; and a parallel printer port. There's even a real-time clock with battery backup. A few short years ago a system with these features might have been called a mainframe.

From a Multi-User System to a Multi-System Network

With the VISUAL 2000 you never pay for more system than you need, and never have to settle for less. Up to 6 megabytes of RAM. 4 Winchester disk drives. Floppy. Streaming tape backup. And mainframe communications. All in a small stand-up enclosure which looks right at home next to a desk.

Use it as a central processor for inexpensive video terminals, or as a database manager for a cluster of intelligent workstations or personal computers, including VISUAL's own IBM®-compatible COMMUTER™.

And if a fully expanded VISUAL 2000

See us at COMDEX, booth #1836.



VISUAL 2000

isn't enough, you can connect up to 254 systems in an ARCNET® local area network.

In all configurations, the VISUAL 2000 achieves an exceptional ratio of price/performance.

Software: A Complete System Builder's Tool Kit

Ultimately, the strength of a system is the strength of its software. And there too, the VISUAL 2000 really shines. Start with XENIX™, Microsoft's enhanced version of UNIX™, which gives you everything you expect from UNIX *plus* greater speed, reliability and ease of use in business applications. Then add all the tools you need to deliver end-user applications quickly and easily. Languages such as C, SMC BASIC, RM/COBOL™, TOM BASIC™, SOFTBOL™, and MicroFocus Level II COBOL™ to give you instant compatibility with hundreds of proven business applications. Plus sophisticated system foundations such as INFORMIX™ database

management and RealWorld™ modular accounting. And professional productivity software such as 20/20™ spreadsheet modelling and XED™ word processing.

The Bottom Line

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

And when you consider that VISUAL has long been a leader in video terminals, and can supply a full range of compatible terminals and workstations, we think you'll agree that VISUAL is the system builder's system supplier.

The VISUAL 2000.

It began with a better design. And it's available *now* to help you build better systems. But don't take our word for it.

See for yourself.

Call VISUAL today.

VISUAL

 See for yourself®

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

REGIONAL OFFICES:

Northwest:	(415) 490-1482
Southwest:	(213) 534-0200
North Central:	(513) 435-7044
South Central:	(214) 255-8535
Northeast:	(201) 528-8633
Southeast:	(301) 924-5330



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

Circle No. 60 on Inquiry Card

INDUSTRY INSIDER

The end user's dilemma

by Mark G. Sobell

End users are being torn between the friendliness of MS-DOS software and the power of UNIX and its associated hardware. Until recently, you could only have one or the other. But now, with the advent of several LANs (local area networks — see below), you can run friendly DOS programs on a PC at your desk while communicating with a host UNIX system down the hall. You can use the host system for its printer, hard disk and backup facilities while the DOS system remains dedicated to running your programs. If you need more power, you can always rely on the serious, albeit less friendly, computing power of the host system.

THE MICROSOFT VISUAL SHELL

Now Microsoft, the company that worked with IBM to produce MS-DOS for the original IBM PC and XENIX (Microsoft's UNIX port) for the IBM PC/AT, is adding another way of dealing with the UNIX/DOS power/friendly dilemma. It designed a user interface, the Visual Shell, based on the design of several of Microsoft's MS-DOS products, including Multiplan and Words.

The Visual Shell presents a naive user with a more intuitive interface to XENIX without sacrificing any of the functionality of XENIX. To better help the naive



user, the Visual Shell includes context sensitive help at all levels.

As a bonus for the system integrator or someone who simply needs a specific set of menus, a sophisticated user can customize the Visual Shell interface by modifying, adding or removing menus.

THE IBM PC/AT AND C-MERGE

By next spring, the IBM PC/AT will be able to run both MS-DOS and XENIX on the same machine. Although it won't run both concurrently, you will be able to store DOS and UNIX programs on the same hard disk and reboot the machine to run either operating system at will. Thus, XENIX with the Visual Shell will co-reside with DOS and its friendly programs on a single machine: what could be better?

One improvement would be the ability to run friendly DOS programs under UNIX in a multiuser, multitasking environment, side-by-side with powerful UNIX programs. Enter C-Merge — Microsoft's C compiler that IBM is offering with the PC/AT.

As the second source for Microsoft, the Santa Cruz Operation (SCO) has been working extensively with the new compiler. I spoke with SCO Vice President Doug Michels about C-Merge and what it means to both the software developer and the end user.

"C-Merge runs under XENIX and, using an MS-DOS library, cross-compiles code that runs under MS-DOS," Michels explained. "You get code that runs under both XENIX and DOS with the same amount of work that it takes to produce code for XENIX alone. C-Merge provides a set of libraries that are functionally equivalent to most of `stdio` and `libc` so that many sophisticated UNIX applications can easily be ported to DOS. In addition, by developing DOS code in a XENIX environment, you can take advantage of all the XENIX utilities.

"What this means is that the software developer can use the UNIX Source Code Control System (SCCS) to keep track of code as it is being developed, UNIX **make** to speed up compiling, as well as UNIX **Lex** and **yacc** to

speed and enhance program development. All this while developing MS-DOS software.”

What this means to the end user is that he or she can have the same program running under MS-DOS and UNIX. This is the ideal migration — the same program on personal computers and larger microprocessors.

In fact, this is the kind of migration path that major software suppliers, such as Relational Database Systems (INFORMIX) have been promoting for years. This setup allows you to develop programs and enter data on larger or smaller machines, whichever is practical, and run the programs with the data, again on either machine.

You can see the power in a network of DOS machines attached to a host UNIX system, all able to run the same programs.

Locus Computing, the company that manufactures the AT&T PC interface that connects to AT&T's 3B series, announced the first of what Jerry Popek, president of Locus, called, “an impressive series of agreements with major manufacturers.” The latest notch on Locus' belt is the Sequent Balance 8000, a National 16032-based UNIX system. The Locus PC-Interface allows AT&T PC 6300s, IBM PCs and PC look-alikes to run local MS-DOS programs that access UNIX files stored on the host Sequent machine. It also enables PC users to execute programs on the host UNIX system, even while running an MS-DOS program locally. These are the same functions that PC-Interface performs for the AT&T 3B series.

WHO'S ON FIRST

AT&T is touting its low-end PC 6300 as IBM PC compatible to capture some of the IBM market. IBM is running AT&T's UNIX on several PC models to go after the

UNIX market. And now Unisource reports that AT&T is buying its own UNIX back from Unisource

IBM and AT&T are dancing with each other, but they're holding each other at arm's length.

(as VENIX) for in-house use on the AT&T PC 6300 IBM-compatible machine.

A solution to the end user's dilemma appears to be unfolding. IBM and AT&T are dancing with each other, but they're holding each other at arm's length. Both are offering each other's software, and IBM, through C-Merge, has brought compatibility between UNIX and MS-DOS one step closer.

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

XENIX Communications Available NOW!

Put your computers on speaking terms.



Introducing
TERM. Communications Software

Everyone from the beginning computer user to the expert finds TERM easy to learn and powerful to use. Just plug it in and go! In a few keystrokes you can access a remote database or send a group of files to another system.

TERM allows your computer to perform efficient, error-free exchange of binary or text files, over phone lines or hard-wired circuits at speeds of up to 9600 baud. Available options allow you to include or exclude a group of files for transfer in a single command.

TERM's "data capture" feature allows saving transcripts of sessions with remote mainframe and minicomputers to disk for later editing or printout, if desired.

- Pre-installed and ready to run
- Automatic error checking and re-transmission
- Wildcard (*.*) file send/receive, capability
- Xon/Xoff, Etx/Ack, Ascii protocols for communications with non-TERM systems
- Full/half duplex emulation mode for remote systems
- Modem7 protocol for remote bulletin boards
- Auto-dial/Answer and Hangup supported on Hayes Smartmodem 300/1200 and compatibles
- Programmable batch file capability
- Unattended file transfer/auto logon
- Translation tables for input and output

TERM is available now on the Altos 586 and Tandy Model 16, along with more than 35 CP/M—80, MSDOS, and CP/M—86 systems; IBM PC*/XT, Kaypro, Osborne, Televideo, Victor, Apple* II—CP/M, Heath, Vector, Sanyo, Eagle, Molecular, Altos, and many others.



CALL OR WRITE FOR FREE PRODUCT CATALOG
CENTURY 9558 South Pinedale Circle
S O F T W A R E Sandy, Utah 84092
We make it easy for you. (801) 943-8386

CP/M is a registered TM of Digital Research

Circle No. 20 on Inquiry Card



FUSION™ is the only connection.

In an industry fragmented by diversity, FUSION makes connections.

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

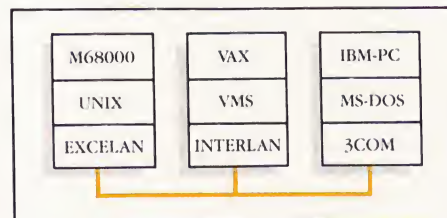
FUSION frees you to buy the best computer for the job—without worrying about compatibility. Your VAX mainframe can now communicate with your IBM-PC. You can even add M68000 work stations or a communications server. FUSION links them all—and more. Regardless of operating system, protocol, or network hardware.

FUSION™

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

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

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



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

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

FUSION is a licensed trademark of Network Research Corporation.



C ADVISOR

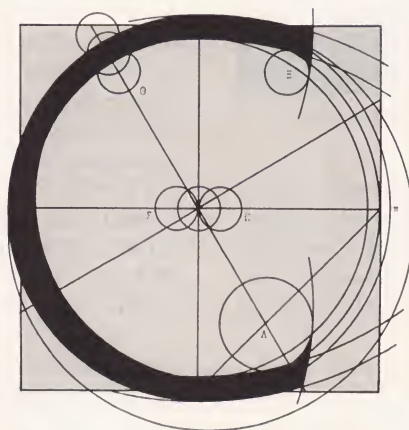
Conjuring curses

by Bill Tuthill

The C library package **curses** performs cursor motion and screen updating on fixed-character video display terminals. Unfortunately, the package provides no support for bitmap display workstations. The **curses** library should be considered a medium-level interface to the terminal. It is not as low-level as the **term lib** routines discussed in the August issue — **curses** provides windowing and an improved **refresh**. But it is not as high-level as something like IBM's Display Management System (DMS), which allows you to edit panels that control display and input regions on the screen. In general, you must be a C programmer to use **curses**, and manipulating the screen is much harder than editing a DMS description.

One nice design feature of **curses** is that the algorithms for updating the screen are very efficient at low baud rates. Rather than redrawing the whole display, they redraw only what's necessary. At 1200 baud, this is extremely helpful, but on large screens at 9600 baud or higher, waiting for CPU cycles to perform optimization often takes longer than would waiting for the entire screen to be redrawn, particularly on loaded machines.

In the past, **curses** has been used mostly to write games. The infamous **rogue** program on Berkeley VAX/UNIX (now available on the IBM PC) is based on **curses**, and was the first major program to exercise it. Some programmers have written serious applications based on **curses**, but often encountered more trouble than they expected. The features required for screen editors (such as forward and reverse scrolling, insert-line and delete-line) do not exist.



There are other problems as well. As a matter of fact, I have never seen a bug-free version of **curses**. But this is probably true of all complicated pieces of software.

There are two major strands of **curses**: one by Ken Arnold, distributed on 4.1 and 4.2 BSD, and one now maintained by Mark Horton, distributed on System V Release 2. Arnold's version is in the public domain, but Horton's version is superior — if you can get it. It makes use of hardware support for insert-line and delete-line, and has improved display algo-

rithms. Mark Horton has done an excellent job of maintaining **vi** for many years now, so his participation ensures that **curses** will receive the kind of support it deserves. Of course, Mark Horton is being paid for this work — Ken Arnold never was.

There is probably a market right now for a UNIX-based display management system. Many database packages provide forms editors, but they are often not general enough for many applications. Difficult for even casual C programmers, **curses** is nearly impossible for someone who doesn't know C. Unfortunately, any program based on **termcap** or **curses** is solving a problem of the seventies, not the eighties. Display terminals with fixed character sets will gradually give way to bitmap display terminals with interfaces like Apple's Macintosh.

In all the years **termcap** has been around, making it possible to write screen-oriented software, only three major UNIX programs have appeared that actually use cursor movement: **vi**, **rogue** and **emacs**. This indicates two things about UNIX. First, teletype mentality persisted for a long time, particularly within AT&T; only market pressure forced it to in-

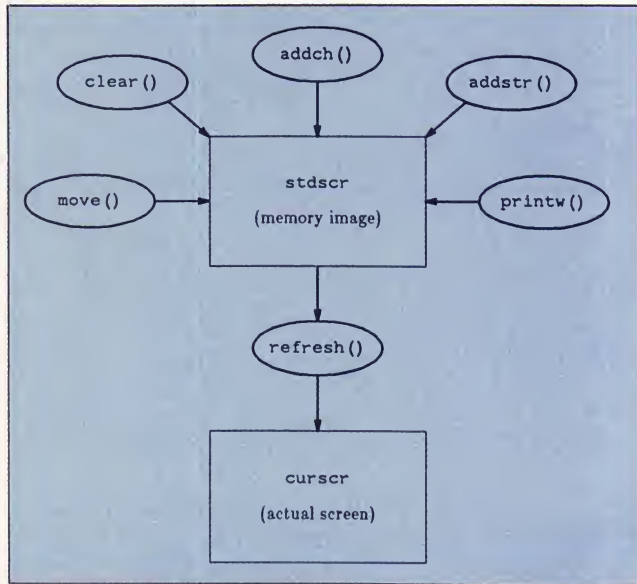


Figure 1—Standard screen and current screen.

clude **vi** on System V. Second, although many screen-oriented programs appeared (such as a visual shell), their user interfaces were so bad that they never caught on. Full screen software is not necessarily user friendly, but is more likely to be so than line-oriented software.

THEORY OF CURSES

The theory behind **curse**s is that there is an image of the screen in memory, called the standard screen (**stdscr**), and the physical screen of your terminal, called the current screen (**curscr**). The most common routines perform output to **stdscr**, which acts as a buffer to **curscr**. When the programmer wants the physical screen to correspond exactly with the standard screen, the **refresh()** routine is called. The **curse**s package will determine an efficient algorithm for updating the terminal screen. Changes will be made using the quickest operations (such as linefeeds and character output), not in the order in which the standard screen was updated. (See Figure 1.)

To initialize the package, the routine **initscr()** must be called before any of the other routines that deal with windows and screens are used. The **initscr()** routine reads the appropriate **termcap** description, initializes the terminal, and allocates memory for a new window the size of the screen. The routine **endwin()** should be called before exiting, in order to reset the terminal to its normal state.

Character output is done with the **addch()** routine, string output is accomplished with **addstr()**, and formatted output is done by means of **printw()**.

Cursor motion is done with the **move()** routine, which should be handed the y and x coordinates of the desired position on the screen. After any output or movement occurs, the **refresh()** routine must be called before anything will change on the terminal screen.

Programs that use **curse**s must include the file **<curse.h>**, which in turn includes the file **<stdio.h>**. Programs then should be compiled as follows:

```
cc prog.c -o prog -lcurse -ltermli
```

where **prog** is the name of your program. The **curse**s library should be loaded before **termli**, because **curse**s requires certain **termli** routines.

A SAMPLE PROGRAM

The C program shown in Figure 2 on pp. 66-67 is an example of an unsophisticated editor implemented with **curse**s. The program, called **med**, paints the screen with two musical staves and allows the user to move around the screen, placing notes on the staves, and words beneath. The CTRL-W command will write what is currently on the screen to a file, while CTRL-X will write and then quit. The interrupt character (whether DEL or CTRL-C) will quit without writing.

Note that we need not include **<stdio.h>**, because **<curse.h>** does this for us. We also include **<signal.h>** because we need to trap interrupts. For the sake of clarity in this program, functions we define, other than **main()**, have an underscore in their name, while library functions do not.

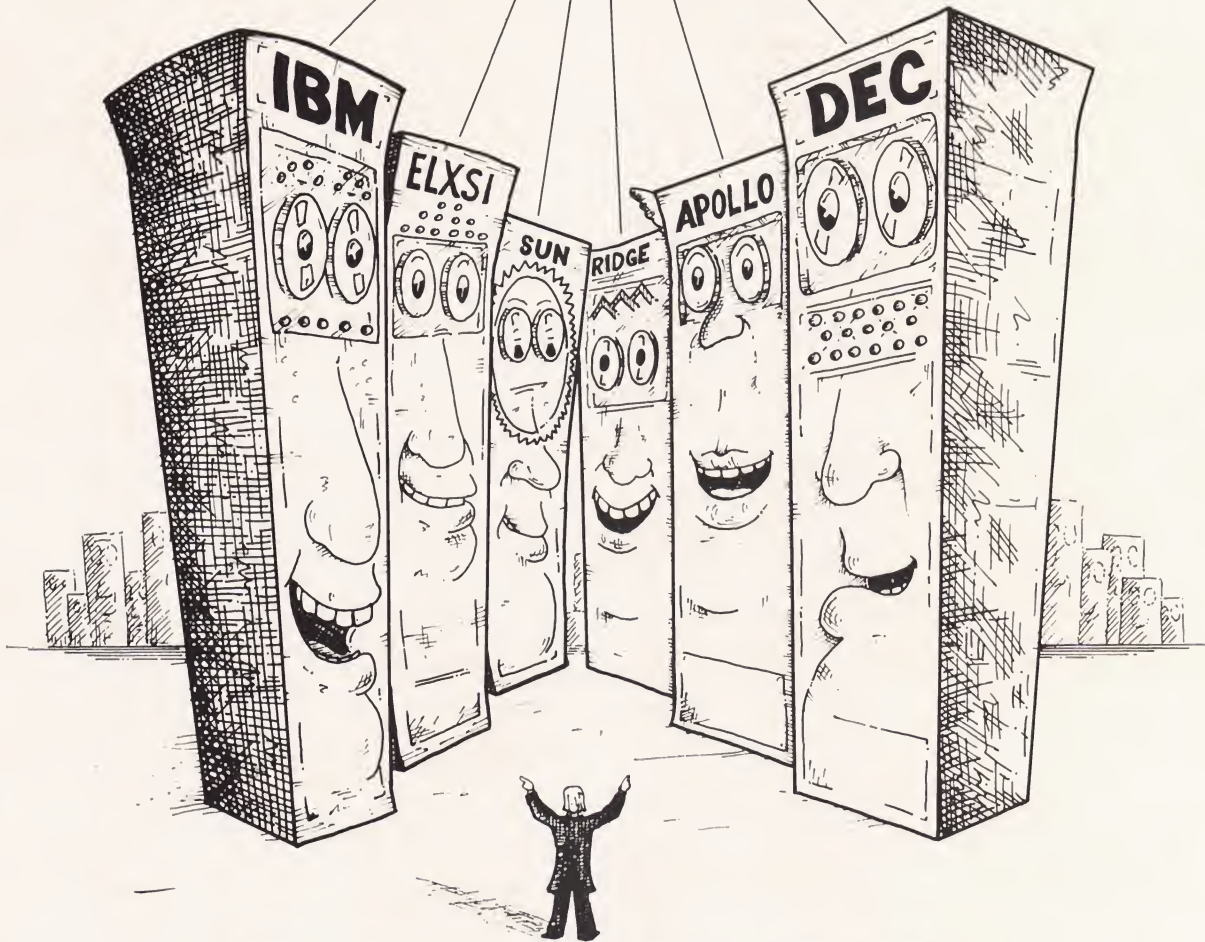
The **initscr()** function reads the terminal description and allocates memory for a window data structure the size of the screen. The next call to **refresh()** will clear the screen. Because they operate in full-screen mode, most **curse**s programs call **noecho()**, **nonl()** and **crmode()**, which is equivalent to this command line (based on Version 7 UNIX):

```
stty -echo -nl cbreak
```

This is because full-screen software must perform some action for certain command characters, rather than simply echoing these characters on the screen. Also, we don't want newline mapping. Finally, the system must read and respond to each character as it is typed, rather than only after each line is sent.

We trap the interrupt signal to execute the **go_away()** function, so that DEL or CTRL-C (whichever your system uses) will cause the program to exit gracefully. Then we place the musical staves on the screen with the **paint_scr()** routine, and position the cursor at the bottom left of the top staff.

MAINSAIL™



SPEAK MAINSAIL. BECAUSE NOT EVERYONE SPEAKS UNIX.

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

That's where MAINSAIL comes in.

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

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

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

XIDAK

Circle No. 22 on Inquiry Card

```

/*      compile with: cc med.c -o med -lcurses -ltermLib
*/
#include <curses.h>      /* which includes <stdio.h> */
#include <signal.h>

main(argc, argv)      /* med - a full-screen music editor */
int argc;
char **argv;
{
    register x, y;
    int c, go_away();

    if (argc != 2)
        fprintf(stderr, "Usage: %s outfile\n", argv[0]), exit(1);
    initscr();
    crmode(); noecho(); nonl();
    signal(SIGINT, go_away);
    paint_scr();
    while (TRUE) {
        getyx(stdscr, y, x);
        switch((char)(c = getchar())) {
            case '\r':          /* CRLF */
                move(y + 1, 0); break;
            case '\t':          /* TAB */
                move(y, x + 8); break;
            case '\b':          /* ^H left */
                move(y, x - 1); break;
            case '\012':        /* ^J down */
                move(y + 1, x); break;
            case '\013':        /* ^K up */
                move(y - 1, x); break;
            case '\014':        /* ^L right */
                move(y, x + 1); break;
            case '!':           /* ! staff */
                add_n_ch(c, 3, y, x); break;
            case '|':           /* | bar */
                add_n_ch(c, 8, y, x); break;
            case '\022':        /* ^R redraw */
                clearok(curscr, TRUE); break;
            case '\027':        /* ^W write */
                write_scr(argv[1]);
            case '\025':        /* ^U undo */
                paint_scr(); break;
            case '\030':        /* ^X exit */
                write_scr(argv[1]);
                go_away(); break;
            default:
                if (c < ' ')      /* control char */
                    fputc('\007', stderr);
                else
                    addch(c);
                break;
        }
        refresh();
    }
}

```

Figure 2—An example of an editor implemented with curses.

The heart of the program is an infinite loop containing a giant **switch** statement. Note that we control the infinite loop with the Boolean value **TRUE**, defined in **<curses.h>**. The **switch** statement processes RETURN, TAB, and what we use as the four directional keys, CTRL-H (left), CTRL-J (down), CTRL-K (up) and CTRL-L (right). CTRL-L sometimes is used to redraw the screen, but this program uses CTRL-R for this instead. We would have to extend the program to support labeled arrow keys on ANSI

standard terminals, which usually transmit multi-character escape sequences.

After every iteration of the loop, we call **getyx()** to obtain the current y and x coordinates, which is easier than manually resetting the coordinates. This routine fills in its second and third parameters with the current y and x positions. Why don't we have to pass these parameters by address? The reason is that **gettyx()** is a macro, defined in **<curses.h>** as follows:

```

paint_scr()          /* place treble and bass staves on screen */
{
    register int co, ln;
    int staff = 0;

    move(0, 0);
    while (staff++ < 2) {
        addch('\n'); addch('\n');
        for (ln = 1; ln <= 5; ln++) {
            for (co = 0; co < COLS-1; co++)
                addch('_');
            addch('\n');
            for (co = 0; co < COLS-1; co++)
                addch(' ');
            addch('\n');
        }
        move(10, 0);
        refresh();
    }

    int status;

write_scr(name)      /* write screen image to named file */
char *name;
{
    FILE *fp, *fopen();
    register x, y;

    if ((fp = fopen(name, "a")) == NULL)
        perror(name), status = 1, go_away();
    for (y = 0; y < LINES && y < 25; y++) {
        for (x = 0; x < COLS-1; x++)
            putc(stdscr->_y[y][x], fp);
        putc('\n', fp);
    }
    fclose(fp);
}

add_n_ch(c, n, y, x) /* make vertical line of n chars on screen */
char c;
int n, y, x;
{
    while (n-- > 0) {
        move(y-n, x);
        addch(c);
    }
}

go_away()           /* restore terminal in case of interrupt */
{
    signal(SIGINT, SIG_IGN); /* ignore 2nd interrupt */
    mvcur(0, COLS-1, LINES-1, 0); /* no refresh required */
    endwin(); /* uninitialized curses */
    exit(status);
}

```

```
#define getyx(win,y,x) y = win->_cury, x = win->_curx
```

Since the C preprocessor substitutes the assignment statement above directly into our code, we wouldn't want to employ **&y** or **&x**, which would constitute an illegal left side of an assignment statement.

The vertical bar will draw an eight-character-high bar to separate measures, while the exclamation mark will draw a three-character-high stroke to represent musical notes. CTRL-R redraws the screen

in its current state (in case it gets messed up by system messages). CTRL-W writes the screen to a file, and then paints a new set of staves on the screen. CTRL-U returns the screen to its original state, as if nothing had happened. CTRL-X writes the screen out to a file (as would CTRL-W), and then exits. All printing characters besides | and ! appear on the screen, so you can place directions and lyrics alongside the music.

The `paint_scr()` routine is fairly straightforward

```

struct _win_st {
    short      _cury, _curx;
    short      _maxy, _maxx;
    short      _begy, _begx;
    short      _flags;
    bool       _clear;
    bool       _leave;
    bool       _scroll;
    char       **_y;
    short      *_firstch;
    short      *_lastch;
    struct _win_st *_nextp, *_orig;
};
    
```

Figure 3—The window data structure, declared in < curses.h >.

ward. Two staffs are painted, separated from each other by two blank lines. Each staff is composed of alternating lines of underscores and blanks, out to the number of columns on the terminal. If you compile and run the program, you will notice that only underscores are sent to the screen — the blanks are optimized out, because the screen has already been cleared.

The `write_scr()` routine is a bit trickier. Opening the output file for appending, using `fopen()`, is

← Δ φ ε / < > ≡ " →

MIPS SOFTWARE PROVIDES

THE APL - UNIX® SOLUTION:

DYALOG APL

UNIX® based - fully functional commercial APL including nested arrays, upper and lower case data support for the UNIX® environment, dynamic workspace size, external functions (callable subroutines written in other languages) full screen editor, error trapping, commercial formatter and a host of other desirable features.

DYALOG APL is available for a variety of UNIX® computing environments including VAX®, PE, Gould, 3 B Series!, NCR Tower, Zilog, Fortune, Perq, Ridge!, Pyramid!, and Sun!. For further information about DYALOG APL on your 68000, 16032 or 8086 base system, call or write today.

MIPS SOFTWARE DEVELOPMENT, INC.
 31555 West 14 Mile Road
 Suite 104
 Farmington Hills, MI 48018
 313-855-3552

UNIX® is a trademark of Bell Laboratories.
 VAX® is a trademark of Digital Equipment Corporation.
 ! Call for Availability

n α □ □ τ ω _ o p α l ε | o

Circle No. 63 on Inquiry Card

easy. But what about the screen data structure? The window data structure is declared in < curses.h > as shown in Figure 3.

Since `stdscr` is a pointer to a window data structure, the characters making up the standard window are accessible through `stdscr->_y`, which is a pointer to a pointer to a character. If we want the 10th character on the first line, we could get it by specifying `stdscr->_y[0][9]`. So we can proceed through the entire screen by having a loop traverse the columns, inside a loop traversing the lines. Since newlines are not part of the window, we have to add them explicitly at the end of each line. It would also be possible to use the `inch()` routine to retrieve characters at particular (y,x) coordinates, rather than traversing the data structure. However, this would involve a `wmove()` for each position on the screen, which would incur much function call overhead.

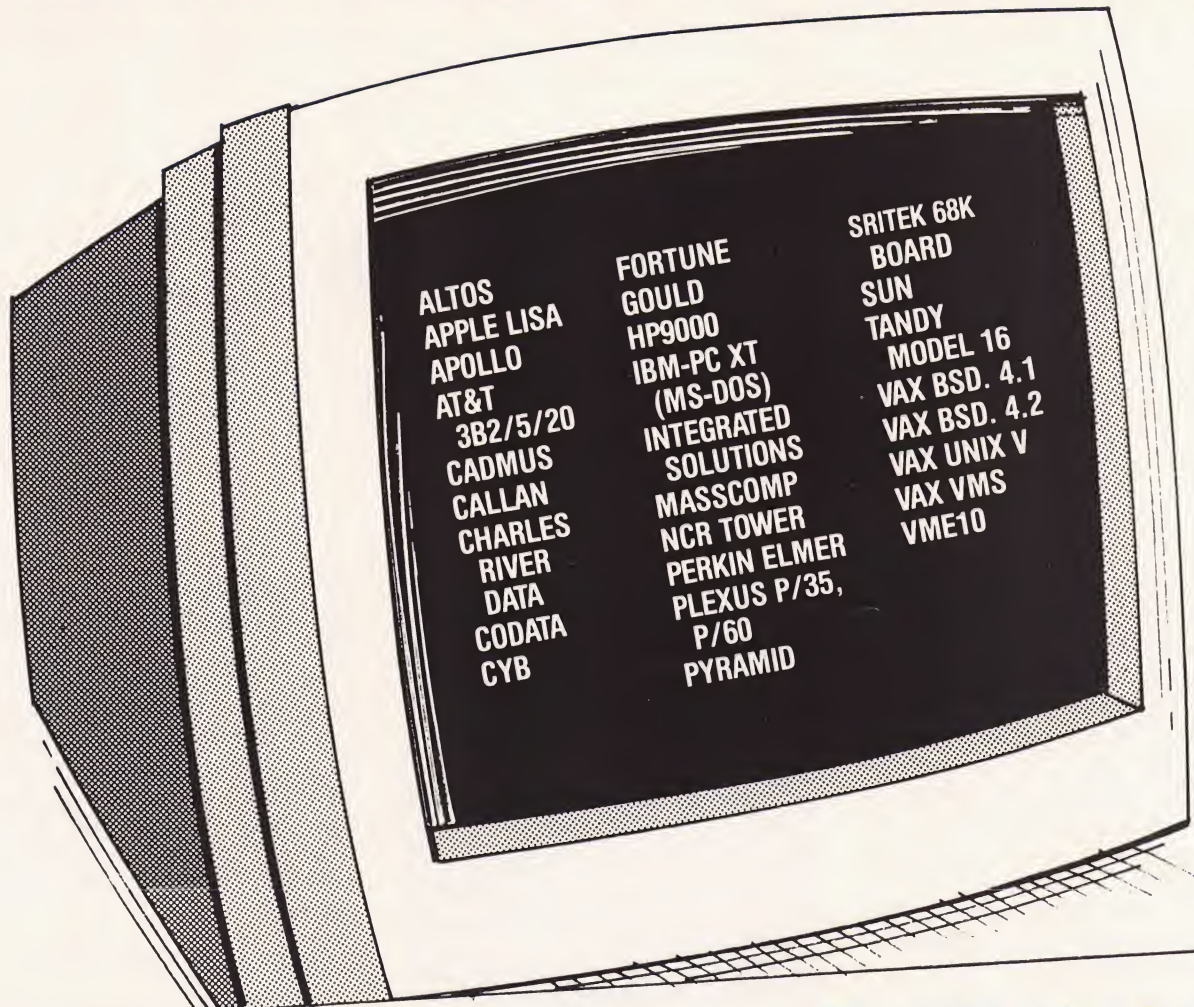
The `go_away()` routine is typical of `curses` cleanup functions. We use `mvcur()` instead of plain `move()`, which would require an additional call to `refresh()`. The parameters passed to `mvcur()` are the origin and destination coordinates. The origin should be the upper right corner of the screen, and the destination should be the lower left corner, so that `mvcur()` will always move far enough. Finally, we call `endwin()` to reset terminal modes. Then we exit with a status of 0 if everything went fine, a status of 1 if the output file couldn't be opened for appending, and a status of 2 if the program was interrupted (2 is the signal number for interrupt).

The main virtue of this program is that it fits on two pages. It is not of production quality, since it was intended to be only a demonstration of `curses`. It needs to be more robust before it can be a useful program. Bounds checking should be done before all calls to `move()`, since `curses` does not do this for you. Arrow keys need to be added, which would take about a page of code. Most importantly, there must be a `help` facility to make it simple to learn and use.

Of course, `curses` has many features not demonstrated here. For example, it can allocate multiple windows of varying sizes, and overlay them as required. Our sample program uses only the two default windows, `stdscr` and `curscr`. There are `curses` routines to draw boxes, erase portions of the screen and put text in reverse video. The new V.2 version of `curses` has many wonderful new features, particularly ones to support the writing of editors and to make use of arrow and function keys. One incompatibility to watch out for, however, is that the `crmode()` function has been renamed `cbreak()`.

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

Whatever System You Use . . .



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

PRODUCTIVITY AIDS

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

APPLICATIONS

- QCalc Spreadsheet
- Lex Word Processing
- Leverage List Processor

LANGUAGES

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

SUPPORTED O.S.

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

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

Call or write our trained staff for more product information

UniPress Software, Inc.

2025 Lincoln Highway, Edison, NJ 08817

201-985-8000 • Order Desk: 800-222-0550 (outside NJ) Telex: 709418

Mastercard and Visa

Trademarks of: UniPress Software, Inc.; Lattice, Inc.; Altos Computer; Apple Computer; Apollo Computer; Western Electric; Callan; Cyb; Charles River Data Systems; Fortune Computer; Gould; Hewlett Packard; International Business Machines; Integrated Solutions; Masscomp Computer; NCR; Pyramid Technologies; Perkin Elmer; Plexus Computer; Sun Microsystems; Digital Equipment Corp.

Circle No. 23 on Inquiry Card

RULES OF THE GAME

Stalking the derivative copyright

by Glenn Groenewold

A creation of the mind is an elusive thing. To enforce the rights and obligations stemming from such a creation, the thing to be protected must first be defined. One of the approaches the law has taken to simplify this task is to require that intellectual property be reduced to tangible form before it can be accorded legal protection.

This is the condition explicitly laid down in copyright law and is also essentially the approach taken in patent law. However, there are other concepts of intellectual property protection that don't require this; the law of trade secrets, discussed a few columns back, is an example.

But where the "tangible form" prerequisite is a requirement of law, confusion can reign since the same intellectual creation may often exist in a number of different forms.

This can occur when the creator has revised his or her own work. If the process of revision is more or less continuous, it may be impossible to say which version of a work is the definitive one, as is the case with Mozart's *Don Giovanni*.

Or it can occur when other people have altered the work, as when a stage director changes dialogue in a play because the original lines don't sound right. Taking this one step further, a work may change to the point



where it becomes an essentially new one. This happens when a motion picture script is made from a play, or, as occurred with the Berkeley enhancement to UNIX, when a work has had modifications and additions grafted onto it.

When new copyrightable material emerges from changes or additions to a copyrighted work, the result is a *derivative work* in the eyes of the law.

CHANGES IN A WORK YOU OWN

The owner of a copyrighted work has the right to use it as he or she pleases. This includes the right to change it. Revisions and additions to either a computer program or documentation that is *unpublished* and not registered with the Copyright Office present no

copyright problem; protection for the changes will automatically arise.

If the work has been published by being released to others, or if it is unpublished but has been registered, changes can become significant. When the new version has strayed very far from the original, it may be advisable to assert a new copyright for the new material.

HOW FAR IS VERY FAR

Frustratingly, illustrations that the law provides for our guidance come almost entirely from the traditional field of literature. These tell us that minor changes such as the addition or deletion of punctuation or the substitution of one word for another will not necessitate a new copyright. On the other hand, an adaptation of a play into a motion picture comprises a new work, which must be protected by its own copyright.

The test is simple to state but often hard to apply: if the revision or addition results in copyrightable material, which is to say if it in itself constitutes an original expression of a idea, it can be separately copyrighted — and in most instances should be.

As with so many of the legal concepts that have been hauled in to fill the void that originally constituted the field of computer law,

A NEW DIMENSION IN DATABASE MANAGEMENT



Booth #1996-7

See us at

COMDEX™/Fall '84

November 14-18, 1984

Las Vegas Convention Center
Las Vegas, Nevada

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

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

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

RHODNIUS Incorporated

10 St. Mary Street, Toronto, Ontario, Canada M4Y 1P9

(416) 922-1743

Telex: 06-218536 TOR.

Circle No. 26 on Inquiry Card

this is an imprecise fit. For instance, there was initial uncertainty whether the same program in the form of source code and the form of object code represented two independent expressions of an underlying idea, or were merely

two copies of the same expression. Though this confusion has since been resolved in favor of the latter interpretation, it can still be difficult to use the applicable standards to determine when a change in a computer program con-

stitutes a new expression of an idea.

While theoretically it should make no difference whether a program takes the form of binary code or high-level language, the literary paternity of copyright law dictates that the more closely the representation resembles traditional language the easier it will be to apply the legal principles.

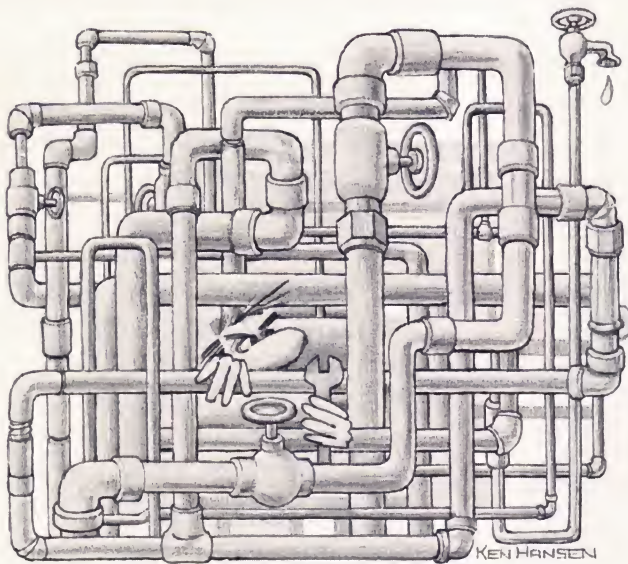
A conservative approach might be to regard any substantial quantitative change in Assembly language as a new copyrightable expression. If the program is in the form of C or some other higher-level language, a workable rule of thumb might be to view the substitution of a series of statements — or of even one crucial statement — as copyrightable, while ignoring minor alterations in language or format.

Documentation is much easier to deal with because it resembles traditional literature. An entirely new section added to existing documentation should be treated as new copyrightable material. Otherwise, only a revision that amounts to a rewriting, or that alters the substantive content of the document, need normally be the subject of a new copyright. When in doubt, there's no harm in asserting and registering a new copyright, aside from the 10 bucks you'll be out.

Whether the new or revised material consists of software or documentation, it legally gives rise to an entirely new copyright, *not* an amendment of the earlier one, which is unaffected. The year of publication of the new version is simply added to the copyright notice and shown along with the original date. You register the derivative work with the Copyright Office on Form TX, just as with the original registration.

WORKS OWNED BY ANOTHER

In the UNIX community, it's



Without SoftShell,[™] learning to live with UNIX[™] could be a real trap.

SoftShell is a convenient interface which guides you gracefully through UNIX. If you're new to UNIX, or have the task of training new users, SoftShell simplifies its complexity. If you're already a UNIX fan, you'll love SoftShell because it augments your system. You'll find yourself further exploring the great depth and versatility of UNIX. Available for UNIX System-V, Berkeley 4.2 and IBM's PC/IX, SoftShell is an invaluable tool for users at all levels of expertise. For information, contact Logical Software, Inc., 17 Mount Auburn Street, Cambridge, MA 02138, (617) 864-0137.

UNIX is a trademark of AT&T Bell Laboratories.



Logical Software Inc.

Circle No. 27 on Inquiry Card

commonplace to work with systems and programs owned by someone else who has granted a license for their use. As is true of any contract, the provisions of a licensing agreement generally define lawful use of the licensed material.

Once again, however, we come up against the fact that in the area of traditional literary works there's virtually no counterpart to the functions of computer software. The user of a computer system ordinarily is expected to create applications programs tailored to his or her own needs. After all, that's why the user got a computer in the first place. This generally causes no problem so long as such programs are limited to in-house applications.

But what if the user wishes to exchange programs and databases with another licensee, or with someone who is not a licensee, or has a different category of license? Or what if the user wishes to market them for use by the general public? And what will be the copyright status of this new material?

Clearly, not all licenses — or licensees — are equal. It's crucial that you know what type of license you're operating under and what the agreement says about your use of the copyrighted systems and programs licensed to you.

Ordinarily the copyright owner can impose restrictions on who may use copyrighted material, the extent to which it can be used, and the ability to create derivative works. Moreover, even when alterations and additions are permitted, the copyright owner can specify the terms under which any new versions can be licensed, and can even require that the copyright on any derivative work be assigned to the owner.

Some license agreements

cover these points specifically, permitting both modification and sublicensing. Others can be quite user hostile, requiring inferences to be drawn from their general

A creation of the mind is an elusive thing.

provisions. Typically, source licensees have greater latitude than others, but in interpreting a licensing agreement it should never be assumed that actual or apparent silence indicates consent. If you're in doubt as to what

the agreement permits, check with a lawyer.

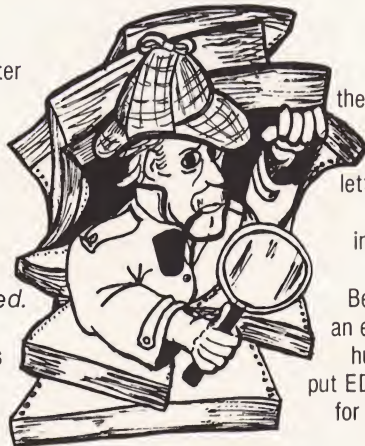
Copyright law is quite clear in at least one regard: you can obtain no copyright on a revision or addition to someone else's material if your use of that material was unlawful. This doesn't necessarily mean that the Copyright Office won't register a copyright if you apply. In accepting a copyright for registration, the Copyright Office goes no further than to determine that the type of material involved appears to be copyrightable.

So if you intend to adapt *The Magic Mountain* as a musical for the Broadway stage, you had certainly better talk to the owner of the rights to Thomas Mann's works before you begin, because

Even Holmes Couldn't Find His Listings

You don't have to be a master detective to know that unmarked listings can be criminally hard to find. That's why Beach Software developed EDGEWRITER™, a software product that labels all your listings, as they're being printed.

EDGEWRITER™ determines the size of your listings, and then prints any



message you want on the front (or folded) edge. And it prints large, easy-to-read block letters . . . which means you'll spot your listings quickly and easily.

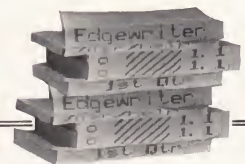
Beach Software has put an end to the great listing hunt. To find out how to put EDGEWRITER™ to work for you, contact us today.



EDGEWRITER™ by

Beach Software

1905 Carnegie, No. 1
Redondo Beach, CA 90278 (213) 379-3085



EDGEWRITER is a trademark of Beach Software

Circle No. 61 on Inquiry Card

if you produce a derivative work (which this clearly would be) without first obtaining the consent of the owner of the original copyrighted work, any purported copyright is invalid, whether registered or not.

But once again the analogy to traditional literary works fails to provide definitive answers for situations in the computer world. It becomes difficult to say when a computer program is a derivative work and when it isn't. There's no

equivalent in the literary world to the transportable program that can be used in conjunction with various operating systems. It's difficult to conceive of anyone bothering to write a "Chapter 14A" that can be plugged into the novel of a reader's choosing, but in a sense, this is what is done when computer programs are written.

Of course an applications program is *dependent* on an operating system in the sense that it can accomplish nothing without it. And ordinarily such programs are written within the context of a specific operating system. But does this make the program a derivative work?

Unfortunately we do not yet have a definitive answer to this question. Certainly the legal argument can be made that if a program actually is independent of the system for which it was created, in that it can run equally well on competing systems protected under different copyrights, it should not be considered a derivative work. And as we know, look-alike systems intended to be compatible with UNIX software *do* exist.

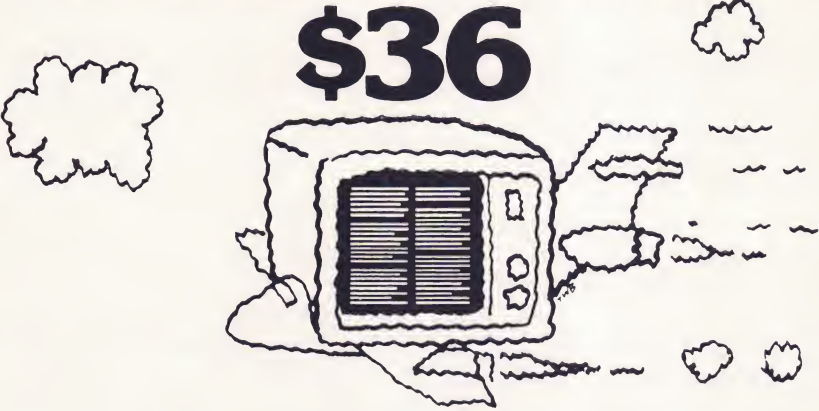
What it comes down to is that if you wish to market a program you've created for a system such as UNIX, you'd be well advised to first review the situation with your lawyer. Keep in mind that in addition to copyright considerations, as a licensee you'll be under various contractual obligations with respect to your use of such things as trade secrets. You'll want to make sure that you don't unintentionally violate these.

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

/USR/TROFF™

ROUND TRIP NEW YORK/ PORTLAND

\$36




From the Big Apple to UNIX™ Paradise and back. 5 minutes out, 48-hour return. TYPESETTING at 15½ feet-per-minute—wider than this entire UNIX™ Review page (about 1½" wider).

Telecommunication of documents, including mathematical formulae, gets TYPE SET and RETURNED on first class ACCUCOM-ODATIONS. No waiting at the TYPESETTING counter, and no reservations required. We also "book" passage for magnetic tape and raw text.

Call for details—1-800-ACCUCOM (222-8266)

UNIX is a registered trademark of BELL LABORATORIES



Circle No. 28 on Inquiry Card

(503) 684-2850

9730 SW Cascade Blvd. / Suite 200 / Tigard, Oregon 97223

CrystalWriter™

Complete Word Processing for the UNIX™ System

"...one of the best kept secrets
in the world of UNIX System
application software..."

"...a truly superior package..."

David D. Coleman
UNIX/WORLD Vol. 1 No. 3

AVAILABLE NOW, CrystalWriter
from SYNTACTICS is the complete
word processing system designed
to support the first time user—
without ignoring the
experienced word processor.

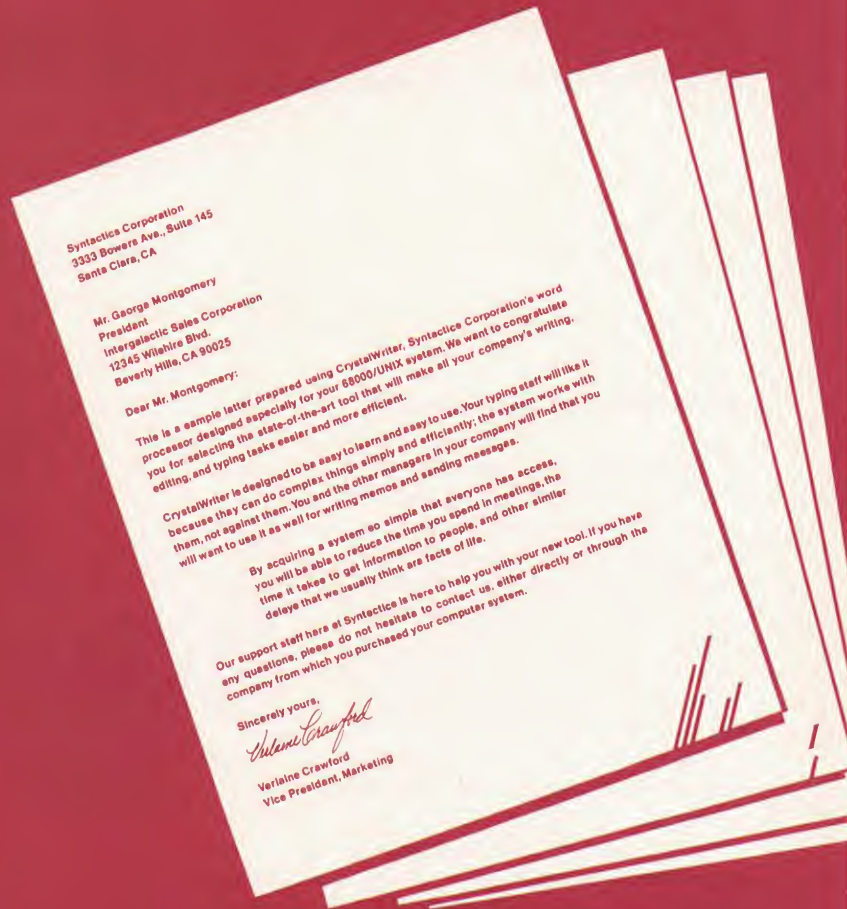
For more information on
CrystalWriter, call today

800-626-6400
(In California (408) 727 6400)

SYNTACTICS
3333 Bowers Avenue, Suite 145
Santa Clara, CA 95054



SYNTACTICS™



UNIX is a trademark of Bell Laboratories
SYNTACTICS and CRYSTALWRITER are trademarks of Syntactics Corporation

Circle No. 29 on Inquiry Card

/usr/lib

System V Unveiled

by Jim Joyce

EXPLORING THE UNIX SYSTEM

Stephen Kochan, author of *Programming in C* (see the June column for a review) has teamed up with Patrick H. Wood to produce a most impressive introduction to Unix System V (Hayden, 1984, 371pp, \$18.95). Where most books start out with logging in—granted a reasonable enough place—these authors start in Chapter 2 with what an operating system is, with UNIX as the example.

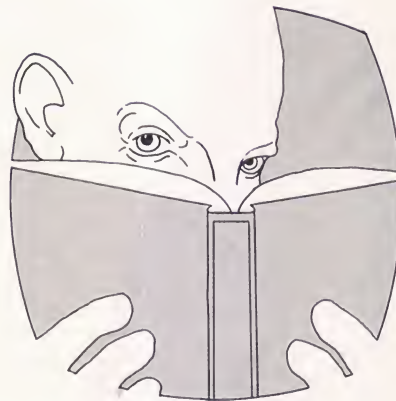
They actually discuss process swapping, and manage to do so in a way that is accessible to any intelligent reader. Their clear diagrams nicely illustrate swapping and how user processes are time-sliced.

Humor abounds, some of it being quite broad. Some of it, though, borders on being in-jokes. For example, the use of **xyzy** as an encryption key may puzzle readers who have not played adventure and do not know it is the secret word that transports an adventurer between the underground cave and the well house.

A more readily accessible form of humor discusses system startup as it used to be: "In the dark ages (of UNIX), [startup] entailed a lot of switch setting, button pushing, typing, and praying." (p. 272).

In fact, the chapter on system administration is packed with real goodies for anyone with System V. Too bad AT&T's administration documentation isn't as clear or as helpful.

There is a somewhat misleading suggestion under the discussion of **fsck** that "you can respond to



any message that ends in ? with a **y** with confidence." (p. 291) The cautionary "if you find that **/etc/fsck** has lost too much data when you mount the file system and look at it, you may have to restore the file system from a backup" simply is understating the risk one takes in blindly replying **y** to **fsck**.

Another minor quibble is that the **echoe** option to **stty** is used in examples but is not explained. The option is not really that difficult to explain, and its omission seems strange.

I like the way Kochan and Wood make use of *before* and *after* parallel screens in their presentation of **vi** commands. It is a neat solution to a difficult problem.

Overall, this is a WONDERFUL book, and comes right when manufacturers are busily converting to System V. Oh, yes. There was one typo I found, on page 94, line 2: the word *you*, when it should be *your*. That was the only typo, though, that I saw. The book was written and typeset on a UNIX system, and demonstrates that, in the proper hands, UNIX typesetting can be pleasing to the eye.

The Table of Contents follows:

Table of Contents for *Exploring the UNIX System*

1. Introduction	4
2. What Is an Operating System?	7
3. The UNIX File System	7
4. Getting Started	36



January 21 - 25, 1985
INFOMART • Dallas, Texas

OFFICE SYSTEMS

EXPERIENCE THE WORLDS OF UNIX*

UniForum '85 is your passport to the fascinating "Worlds of UNIX." You'll examine the growing impact of UNIX in Office Systems, Personal Computers, Engineering/Programming, and Market Trends at UniForum '85, the largest UNIX event ever held.

PERSONAL COMPUTERS

More than 200 major vendors, in 850 booths, will display and demonstrate all that's new in UNIX products and applications.

An extensive conference and tutorial program will expand your UNIX database. This program, organized by /usr/group, will include

14 all-day tutorials on user-specific aspects of UNIX...40 in-depth and informative conference sessions...four nationally-known plenary speakers.

In addition, a number of introductory courses in UNIX will be presented throughout the event.

UniForum '85 will be your total UNIX experience. Whether you're just getting started... or are a seasoned UNIX veteran... UniForum '85 is *the* UNIX event of the year.

Sponsored by



For Complete Information, Call:

1-800-323-5155

(In Illinois, Call: 1-312-299-3131)

Or Write:

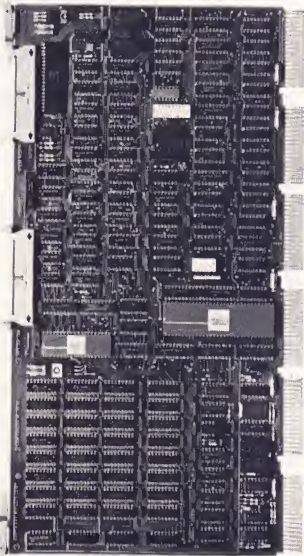
UniForum
Suite 205
2400 East Devon Avenue
Des Plaines, IL 60018

MARKET TRENDS

ENGINEERING/
PROGRAMMING

*UNIX is a registered trademark of AT&T Bell Laboratories.

COMBOARD/HASP DEC → IBM INTERCONNECT



Your DEC computer has better things to do than be a processor for your IBM communications. Save valuable computing capacity by handling your interconnect workload with COMBOARD/HASP from Software Results.

COMBOARD/HASP is a single board 32 bit computer that plugs into your DEC Unibus. COMBOARD/HASP handles all the real-time interrupts and protocol for data communications.

COMBOARD/HASP teamed with Comboard software, is a fast, cost-effective solution for troublesome communications problems. Models are available for transfer rates from 2400 to 56,000 bps.

For further information call or write Software Results, the leader in DEC to IBM communications.

COMBOARD™

Communications Results from
**SOFTWARE
RESULTS
CORPORATION**

Call Toll-free
1-800-SRC-DATA
(1-800-772-3282)

In Ohio call collect, 1-614-267-2203

2887 SILVER DRIVE • COLUMBUS, OHIO 43211 • TELEX 467-495 SRC DATA CI

European Subsidiary

SRCommunication GmbH Kaiserswerther Str 45 D-4000 Duesseldorf 30, FRG
Telephone (0211) 48-10-98 Telex 8 587 466

SEE US AT...

DEXPO™-WEST 84, Anaheim, CA, December 12-15, Booth #711
UNIFORM™, Dallas Infomart, Jan. 22-25, 1985, Booth #1285.

COMBOARD is a trademark of Software Results Corporation

DEC, UNIBUS are trademarks of Digital Equipment Corporation

Circle No. 31 on Inquiry Card

5. Using the UNIX System	45
6. The Old Shell Game	59
7. Screen Editing with vi	29
8. UNIX in the Office	38
9. Program Development	16
10. UNIX Security	13
11. Communications	11
12. Administrating your System	45

Appendixes

A. For More Information	4
B. Overview of Commands	25
C. Complete Command Summary	9
D. Administrative Commands	6
E. Comparison of sh and csh	3
F. Adding New Users	4

Index	6
-------	---

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

C COMPILER

- FULL C
 - UNIX* Ver. 7 COMPATABILITY
 - NO ROYALTIES ON GENERATED CODE
 - GENERATED CODE IS REENTRANT
 - C AND ASSEMBLY SOURCE MAY BE INTERMIXED
 - UPGRADES & SUPPORT FOR 1 YEAR
- C SOURCE AVAILABLE FOR \$2500⁰⁰

HOST	6809 TARGET	PDP-11* LSI-11* TARGET	8080/(Z80) TARGET	8088/8086 TARGET
FLEX* UNIFLEX* OS-9*	\$200.00 <small>WITHOUT TARGET</small> \$350.00 <small>WITH TARGET</small>	500.00	500.00	500.00
RT-11* RSX 11* PDP-11*	500.00	200.00 <small>WITHOUT TARGET</small> 350.00 <small>WITH TARGET</small>	500.00	500.00
CP/M* 8080 (Z80)	500.00	500.00	200.00 <small>WITHOUT TARGET</small> 350.00 <small>WITH TARGET</small>	500.00
PCDOS* CP/M86* 8088/8086	500.00	500.00	500.00	200.00 <small>WITHOUT TARGET</small> 350.00 <small>WITH TARGET</small>

*PCDOS is a trademark of IBM Corp. MSDOS is a trademark of MICROSOFT. UNIX is a trademark of BELL LABS. RT-11/RSX-11/PDP-11 is a trademark of Digital Equipment Corporation. FLEX/UNIFLEX is a trademark of Technical Systems consultants. CP/M and CP/M86 are trademarks of Digital Research. OS-9 is a trademark of Microware & Motorola.

408-251-6653

TELECON SYSTEMS

3577 Chablis Circle
San Jose, California 95132

THE INDEPENDENT UNIX^{*} BOOKSTORE

OVER 60 UNIX AND C ITEMS IN STOCK INCLUDING

■ UNIX BOOKS

- The UNIX System by Stephen R. Bourne
(Addison-Wesley)
- Operating System Design: The XINU Approach
by Douglas Comer (Prentice-Hall)
- The UNIX Programming Environment by Brian
Kernighan and Rob Pike (Prentice-Hall)
- A Practical Guide to the UNIX System by
Mark G. Sobell (Benjamin/Cummings)

■ C BOOKS

- The C Puzzle Book by Alan R. Feuer
(Prentice-Hall)
- The C Programming Language by Brian W.
Kernighan and Dennis M. Ritchie (Prentice-Hall)
- Learning to Program in C by Thomas Plum
(Plum-Hall)
- C Programming Guide by Jack Purdum
(Que Corp.)

■ C and vi REFERENCE CARDS

■ PERIODICALS

- UNIQUE
- UNIX Review
- World UNIX & C

■ vi POSTER, UNIX SHELL POSTER

■ T-SHIRTS

- "UNIX is a Trademark of Bell Laboratories"
- "-rwxrwxrwx"
- "grep for it"
- "awk: bailing out near line 1"

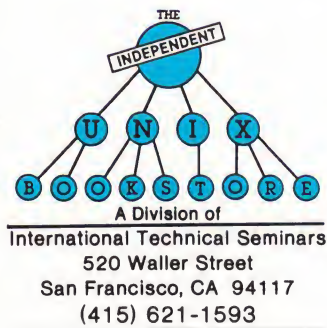
■ NEW ITEM

- UNIX Coffee Mug
- "UNIX" in blue on outside
- "is a Trademark of Bell Laboratories"
- inside lip imprint
- Porcelain mug, kiln-fired permanent color

Call or write for a complete catalog

Mail and phone orders only.
We ship anywhere in the U.S.
Contact us for shipping elsewhere.

* UNIX is a Trademark of Bell Laboratories



PROBLEM SOLVER

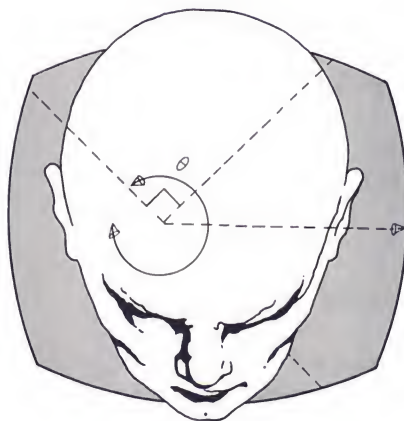
Unraveling the mysteries of System V's terminal interface

by Bob Toxen

As anyone who has dealt with System V (or System III) knows, the System V terminal interface has a completely different means for getting and setting terminal modes than does Version 7 or BSD 4.2. The names of the system calls, the content of the manual page entries — even the sections the manual entries are in — have been changed. While the new system calls take some time to learn, they offer more power and versatility than Version 7's I/O control and are more systematic and easier to use than Berkeley's 4.2 distribution because they were built from scratch rather than on top of Version 7, which in turn was built on top of Version 6.

The system call for getting and setting terminal modes is called `ioctl()` and takes three arguments. The first is an integer specifying the *file descriptor*; the second is the *request*, an integer that specifies what operation is to be performed; the third argument is either a pointer to a `termio` structure or an integer, depending on the request.

The first group of requests, called *primary ioctls*, expect the third argument to be a pointer to the `termio` structure. This structure and some useful constants are defined in `<termio.h>`, which should be included in your program. (This structure will be discussed later.) There are basically two requests. The first, `TCGETA`, reads the current parameters into the user-supplied structure pointed to by the third argument, which I will call *arg*. The second, `TCSETA`, immediately sets the terminal parameters according to the values stored in the user-supplied structure specified by *arg*. These are similar to `getty()` and `setty()` in functionality.



Additionally, the `TCSETAW` request is identical to `TCSETA` except that it waits for the output buffer to drain before taking effect. Typically, it should be used if output parameters are being changed. Lastly, there is the `TCSETAF` request that waits for the output to drain before flushing the input queue (throwing away typed characters that have not been read by any program yet) and changing the parameters.

There are three secondary `ioctls` that are useful for communications; they all take an integer *arg*. The first, `TCSBRK`, waits for the output to drain (be sent to the terminal). Then, if *arg* is 0, it will send a break signal to the device (zero bits for 0.25 seconds). This feature is used by `cu` and `uucp` to cause remote computers to change baud rates. In `cu`, one generates a break by entering the sequence:

```
~%b
```

or:

```
~%break
```

at the beginning of a line, followed by a newline. This is not documented in `CU(1)`.

This second secondary request is `TCXONC`, which is used to start and stop (suspend) output or input transmission. This is useful for implementing protocols other than *X-on/X-off*. If *arg* is 0, then output is suspended; if it is 1, then suspended output is restarted. If *arg* is 2, suspend input (send a `XOFF` to device); if 3, restart suspended input (send a `XON` to device). The fact that *arg* may be 2 or 3 is also undocumented.

The last secondary request, `TCFLSH`, may be used to flush the input or output queues. If *arg* is 0, the

4.2BSD

FROM NOW ON, CONSIDER IT SUPPORTED

When it comes to Unix™ systems, “standard” isn’t always good enough.

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

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

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

- Fully supported 4.2BSD-based binary licenses (MORE/bsd) for VAX™ computers.
- 4.2BSD source support and source updates for current 4.2BSD source licensees.

- Enhanced 4.2BSD-based source software for new sites, with or without redistribution rights.
- Full support for a wide variety of DEC™ and non-DEC peripherals.
- Assistance for OEM’s and hardware manufacturers developing 4.2BSD-based products.

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

4.2BSD. It’s always been better than just “standard.” Now, with MT XINU, consider it supported.

“We know UNIX™ Backwards and Forwards”



Circle No. 33 on Inquiry Card

UNIX™ SUPPORT FROM BERKELEY

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

MORE/bsd and MT XINU are trademarks of Mt Xinu Inc. DEC and VAX are trademarks of Digital Equipment Corp.. UNIX is a trademark of Bell Laboratories.

input queue is flushed; if it is 1, the output queue is flushed; and if it is 2, both queues are flushed. In this regard, *arg* is similar to the second argument to `open()`.

The `termio` structure is the user-supplied structure whose address is passed to the primary `ioctl`s as *arg*. It is defined in `<termio.h>` and an example appears in Figure 1.

The first four members are each 16-bit unsigned integers. Each is logically split up into bit-fields, most of which are a single bit long. If the bit is on, the feature is enabled; otherwise it is disabled. Some of these features interact with others. These fields may be easily accessed with defined constants supplied in the `<termio.h>` `include` file.

INPUT MODES

The `c_iflag` member is for input modes. The first of these is `IGNBRK`. When this field is set, the `BREAK` condition is ignored. A `BREAK` may be generated by pressing the `BREAK` key on the terminal. It is commonly used to switch the baud rate when initially connecting to a computer since `BREAK` is not affected by differing baud rates. `BREAK`s are also generated when Alpha Centauri preempts your data with cosmic rays. The latter method is the reason for the `IGNBRK` feature.

On the other hand, you may want use the `BREAK` key to interrupt programs instead of the more popular `CTRL-C` or `INTerrupt` keys. The `BRKINT` feature does this. It causes `BREAK` to generate an interrupt signal and flush the input and output queues (throw away all unread typed characters and output characters that have not yet been printed) — just as `CTRL-C` or `INTerrupt` can.

`IGNPAR` causes the system to throw away characters with parity errors. It will also cause characters with framing errors other than `NUL`s to be ignored. A `NUL` with a framing error is interpreted as a `BREAK`. A character with a framing error is one without a stop bit and is generated either by the `BREAK` key or those nice folks from Alpha Centauri.

The `PARMRK` is not usually used. It will cause any characters with framing errors to be preceded by both a byte with all eight bits on and a `NUL` byte. This is only useful if you are using advanced error

recovery techniques. If `PARMRK` is on, a legitimate byte made up of 8 ones on will be entered in the input queue twice. Thus when such a byte is read and the following byte is the same, you can be assured you have received legitimate data. But, if the second character is `NUL`, a character with a parity or framing error follows.

`INPCK` will cause all the parity handling discussed for `IGNPAR` and `PARMRK`. If `INPCK` is off, a `NUL` byte is returned for characters with parity or framing errors. The `ISTRIP` flag will strip the eighth bit (zero it).

The `INLCR` flag causes typed newlines (linefeeds) to be changed to carriage returns. `IGNCR` causes carriage returns to be ignored and is useful for terminals that generate both a carriage return *and* a newline when the `RETURN` key is pressed. The `ICRNL` flag causes typed carriage returns to be converted to newlines and is usually enabled since most terminals have a big `RETURN` key and a small `linefeed` key.

Readers with TTY33s can contact the author about the meaning of `IUCLC`. (Hint: UC means Upper Case.)

The `IXON` flag allows you to use a `CTRL-S` (`X-off`) or `CTRL-Q` (`X-on`). `CTRL-S` allows you to stop output so you can read it before it flies off the screen; `CTRL-Q` allows you to continue output. In case you can't remember that `CTRL-Q` allows you to continue output, you can use the `IXANY` flag to allow any character (except a `CTRL-S`) to continue output. Whatever character is entered to continue output is supplied to the program reading the input queue — unless it is a `CTRL-Q`.

Lastly, there is the `IXOFF` flag which causes the *system* to send a `CTRL-S` to the input device if the system's input queue is getting full, and a `CTRL-Q` after space has been made free in the input queue. The `IXOFF` flag is primarily used when the input device is not a terminal but another computer.

That's it for the input modes! The rest of it is not so bad.

OUTPUT MODES

The `c_oflag` member is for output modes. The first flag, `OPOST`, determines whether or not output processing is done. That is, whether output characters destined for the device should be diddled with, delayed to allow for carriage movement, or manipulated in any other way. The `OLCUC` and `OCRNL` flags are the counterparts of the input flags with similar names. `ONLCR` maps newlines to return newline pairs on output. This flag is on for most CRTs and printers. The `ONOCR` flag prevents a carriage return from being sent if the cursor (or printer mechanism) is already at column 0. This is useful for some terminals and printers and can help speed transmission at low speeds.

```
#define NCC 8
struct termio {
    unsigned short  c_iflag;    /* input modes */
    unsigned short  c_oflag;    /* output modes */
    unsigned short  c_cflag;    /* control modes */
    unsigned short  c_lflag;    /* line discipline modes */
    char            c_line;     /* line discipline */
    unsigned char   c_cc[NCC];  /* control chars */
};
```

Figure 1—The `termio` structure defined in `<termio.h>`.

How To GET SMART ABOUT UNIX™

1. Spend three years running a Unix computing resource center.
2. Listen to someone who has.

Here at BASIS in Berkeley, a major center of Unix development, we have distilled our three years of experience selling Unix systems and software applications into two series of concise papers, all of them FREE for the taking.

These papers cover the Unix operating system, timesharing, hardware and software, database design and the like. We have purposely avoided jargon and computer theory. We want to share our Unix "street smarts" with you.

So if you want to buy smart, first be smart: either spend the next three years running your own full-service Unix computing resource center...

or talk to the people
who already have:

B.A.S.I.S.

- Hardware
- Software
- Peripherals
- Consulting

Series I: For Newcomers to Unix.

Brands of Unix
User Friendliness
Learning Unix
Assessing Computer Needs
Personal Computers and Unix
Timesharing Services
True Costs of Computerizing
Unix Text Processing
and Phototypesetting
Software Portability Under Unix

Series II: For Those With Some Unix Knowledge.

Unix System Administration
Networking Under Unix
Benchmarks I: Software
Benchmarks II: Hardware
Fourth Generation Languages

If you're in a hurry, write. Tell us a little about your interests and needs and also jot down the titles of papers you'd like to receive. Or if you prefer, circle our number on the reader service card, and we'll send you all the titles (though it will take a little longer).

B.A.S.I.S.

Bay Area Shared Interactive Systems
1700 Shattuck Avenue
Berkeley, California 94709
(415) - 841 - 1800

UNIX is a trademark of AT&T Bell Laboratories

Circle No. 35 on Inquiry Card

We've Built Unix Into More Micros Than Anyone Else... *and that's standard at UniSoft.*



UniSoft has built Unix™ into 90 microcomputers (and the list is growing) from 60 manufacturers. That's more Unix porting and supporting than anyone else. And more manufacturers who have saved time and money, because of our experience. UniSoft can typically do the job for one fifth of the in-house time and expense. When the port is over, we don't vanish. Our standard family of services includes on-line updates, maintenance, software checks on our mainframe, and documentation (the clear, understandable kind). We also offer application and communications software.

Doing more. Being faster. Offering complete support. We know that this is exceptional behavior from a systems software house. But the exceptional is standard at UniSoft. (415) 644-1230, 739 Allston Way, Berkeley, CA. 94710. *Building System V now.* Call or write for Your Building Plan.

UniSoft

Circle No. 36 on Inquiry Card

Unix is a trademark of Bell Laboratories.

The **ONLRET** flag means that the terminal will perform a carriage return as well as a linefeed upon receipt of a newline (linefeed). This affects delays and future expansion of tabs. **OFILL** causes fill characters to be transmitted to cause delays for carriage movement such as returns, linefeeds, tabs, et al. This feature is not often used since most terminal drivers actually delay transmission of further characters for the prescribed time. **OFILL** is useful if output is filtered through another computer whose buffering would destroy delay timings but not fill characters. It may also be useful to simplify drivers that use **DMA** or other techniques. The **OFDEL** causes **DE**lete characters (all seven bits ones) to be used as fill characters (if **OFILL** is on) instead of **NUL**s. Some terminal devices or printers will immediately throw away **NUL**s but buffer **DEL**s.

All of the rest of the fields in the **c_oflag** member select the amount of delay to be used for various operations. These operations are: newline (**NL**), carriage return (**CR**), TAB (**TAB**), backspace (**BS**), vertical tab (**VT**), and form feed (**FF**). The defines ending **DLY** are used for masking delays for the respective functions. The defines ending in digits select a style of delay; those ending in zero mean no

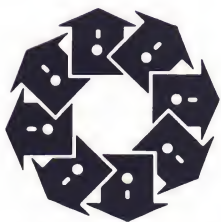
delay. The manual entry documents the actual timings for the different styles of delays.

The **c_cflag** member is for control modes, mainly baud rate and parity. The first define, **CBAUD**, is a mask for the bits that select the baud rate. The entries starting with **B** are for the various baud rates. Selecting a baud rate of **B0** will break most connections and drop **DTR**, hanging up modems. **EXTA** is commonly used for a baud rate of 19200. **EXTB** is usually not defined. **CSIZE** is a mask for the number of bits transmitted per character. It is usually **CS8**. If **CSTOPB** is on, two stop bits will be sent (to support good old TTY33 at 110 baud). This flag is rarely used otherwise.

If **PARENB** is set, parity will be generated on output characters and detected on input (though the **IGNPAR**, **PARMRK** and **INPCK** flags can also determine how input parity errors are handled). If **PARENB** is set then if **PARODD** is set, then odd parity will be used; otherwise parity will be even.

When **HUPCL** is set and all the processes (programs) that have a terminal device open have closed their respective file descriptors for it (perhaps by exiting), the data terminal ready (**DTR**) signal will be dropped (made false). This will cause most correct-

UNPARALLELED
PERFORMANCE
 and **PORTABILITY**
 in an **ISAM PACKAGE**
 at an **UNBEATABLE**
 PRICE



c-tree™
 BY FAIRCOM

2606 Johnson Drive
 Columbia MO 65203

The company that introduced micros to B-Trees in 1979 and created **ACCESS MANAGER™** for Digital Research, now redefines the market for high performance, B-Tree based file handlers. With **c-tree™** you get:

- complete C source code written to K & R standards of portability
- high level, multi-key ISAM routines and low level B-Tree functions
- routines that work with single-user and network systems
- no royalties on application programs

\$395 COMPLETE

Specify format:
 8" CP/M® 5¼" PC-DOS 8" RT-11

for VISA, MC or COD orders, call toll free
 1-800-232-3344

Access Manager and CP/M are trademarks of Digital Research, Inc. c-tree is a trademark of FairCom.

© 1984 FairCom

ly cabled modems to hang up the line (hang up the phone). This feature is usually used if there is actually a modem connected. Many microcomputers do not support this feature, however.

If **CLOCAL** is on, the terminal driver won't require the terminal to assert the **DTR** signal before allowing the **open()** system call to complete. Otherwise **open()** will hang until the **DTR** signal is present (**DTR** is sometimes referred to as *the carrier*). Of course, before you can do an **ioctl()** system call to set **CLOCAL**, you must do an **open()** to open the device file — which will hang until **DTR** is present, in which case **CLOCAL** is unneeded. The **O_NDELAY** flag to **open()** may be used to open the device without waiting for the carrier.

The **c_lflag** member is for the last set of features. The first feature, **ISIG**, will enable the typing of certain characters to generate the interrupt or quit signals.

The **ICANON** flag enables erase and kill processing and accumulation of input data into lines. This means that input characters are accumulated in an internal buffer. When a newline (linefeed character) is received, this accumulated data, including the newline, is made available to any program doing a **read()** system call. If **ICRNL** is on, a **RETURN** entered at the keyboard has the same effect as a linefeed.

It is immaterial whether the **read()** is invoked before or after the newline (or any of the other characters) is received. The user may specify that another character may also be used to terminate lines in the same way that the newline does. This will be discussed later. If one types the **EOF** character, usually a **CTRL-D**, then the characters are also available for reading but the **EOF** character itself (the **CTRL-D**) is thrown away.

If there are no accumulated characters when the **CTRL-D** is entered (because it immediately follows a previous **CTRL-D** or newline character), the **read()** returns a count of zero, i.e. no characters to be read. This is usually interpreted as *end of file*. If the program does another **read()** instead of exiting, this subsequent **read()** will be treated as any other read of a terminal device. That is, it will not return until the accumulation of characters is completed with a newline or **CTRL-D**. This implementation is used by **vi** for scrolling and by **csch's ignoreeof** feature.

If **ICANON** is on and the erase character is entered, the last character accumulated is sent off to Alpha Centauri. The erase character is also thrown away. Likewise, if the line kill character is entered, all characters accumulated are thrown away. The kill character is also thrown away. The **XCASE** feature supports upper case-only terminals and is of interest only to museums that have **TTY33s**.

If **ECHO** is set, characters will be echoed in the output as they are received. If **ECHOE** (echo erase

character, cleverly) is set as well as **ECHO** and **ICANON**, the erase character will be echoed as a backspace followed by a space followed by a second backspace. On CRTs this has the effect of making the mistyped character disappear — a major departure from Version 7, where the same action would merely position the cursor over the character.

Guess what **ECHOK** does? **WRONG!** The UNIX Support Group implemented it to echo a newline after echoing the kill character itself, which is frequently set to be a control character. Maybe in V.3... If **ECHONL** is set then when a kill character is typed, the newline following it is echoed even if **ECHO** is turned off.

If **ICANON** is *not* set, then we play by completely different rules. Specifically, no erase or kill processing is done and a newline (or **EOF**) character does not have to be entered in order for a program to read the received characters. The program does, however, have to wait for a set number of characters to be received or for a set number of tenths of seconds after the **read()** call for it to return. These two thresholds may be set by the user. If the count character threshold is set to one, the **read()** will return after every character. If the count threshold

EVALUATING UNIX* /XENIX** SYSTEMS?

FREE
BENCHMARK MANUAL
FROM



AIM TECHNOLOGY
California

(800) 672-3470 x:815

All Other Areas

(800) 538-8157 x:815

4655 Old Ironsides Drive
Suite 390
Santa Clara, CA 95054
(408) 727-3711

*UNIX is a trademark of Bell Laboratories

**XENIX is a trademark of Microsoft Corporation

Circle No. 38 on Inquiry Card

is set to seven and the time threshold is set to 13, the `read()` will return when seven characters are received or after 13/10 (or 1.3) seconds has elapsed. Even if 1.3 seconds has elapsed, the `read()` will not return until at least one character has been received unless the `O_NDELAY` flag was supplied when the device file was opened, in which case the `read()` will return with a count of zero.

Lastly, if `NOFLSH` is set, the accumulated input and output characters will not be thrown away when an interrupt or quit signal is sent from the terminal.

The line discipline may be specified with `c_line`. A line discipline of zero, the only one supported, supplies `ISIG`, `ICANON`, `ECHO`, etc. if these bits are set.

The `c_cc[]` member of the `termio` structure is a character array that specifies the characters that are used for the erase character, kill character, `EOF` character, etc. A "character" of -1 disables the respective feature. There is a defined symbol, `NCC`, that specifies the number of elements in the `c_cc[]` member, currently eight.

The interrupt character `c_cc[VINTR]` defaults to `DE`lete. The character `c_cc[VQUIT]`, which terminates a program with a core dump, defaults to `CTRL-\`. For programs in development, any character will suffice. The erase character `c_cc[VERASE]` defaults to `"#"` or `backspace`, depending on the port. The character `c_cc[VEOF]` specifies the end of file and defaults to `EOT` (`CTRL-D`). The character `c_cc[VEOL]` may be used to terminate lines and will default to newline. The character `c_cc[VEOL2]` may be used to terminate lines instead of newlines and will default to `NUL`.

The last two bytes are reserved, which is unfortunate since they should be used for specifying the count and time thresholds if `ICANON` is off. Instead, `c_cc[VMIN]` is used as the count threshold and `c_cc[VTIME]` is used as the time threshold. These are treated as 8-bit unsigned integers for this purpose. Unfortunately, `VMIN = VEOF` and `VTIME = VEOL`.

This means that whenever `ICANON` is turned on, you will want to set `c_cc[VEOF]` and `c_cc[VEOL]` to be the `EOF` and `EOL` characters. When `ICANON` is turned off, you will want to set `c_cc[VMIN]` and `c_cc[VTIME]` for the thresholds. If you want to set the threshold counts with the `stty` program in a standard implementation, you must specify the character whose ASCII value is that of the number you want. In other words, specify `CTRL-E` for a count of 5. Specifying `"5"` will give you a count of 53. I learned this the hard way and subsequently fixed the `stty` program and documentation for that port.

Figures 2-7 summarize which mode bits and bytes are stored in each structure member.

So now that you know about all these modes, what do you do with them? When you are logged in-

Input modes		
c_iflag		
IGNBRK	BRKINT	IGNPAR
PARMRK	INPCK	ISTRIP
INLCR	IGNCR	ICRNL
IUCLC	IXON	IXANY
	IXOFF	

Figure 2—Input modes.

Output modes		
c_oflag		
OPOST	OLCUC	ONLCR
OCRNL	ONOCR	ONLRET
OFILL	OFDEL	NLDLY
NLO	NL1	CRDLY
CRO	CR1	CR2
CR3	TABDLY	TAB0
TAB1	TAB2	TAB3
BSDLY	BS0	BS1
VTDLY	VTO	VT1
FFDLY	FFO	FF1

Figure 3—Output modes.

Control modes		
c_cflag		
CBAUD	B0	B50
B75	B110	B134
B150	B200	B300
B600	B1200	B1800
B2400	B4800	B9600
EXTA	EXTB	CSIZE
CS5	CS6	CS7
CS8	CSTOPB	CREAD
PARENB	PARODD	HUPCL
	CLOCAL	

Figure 4—Control modes.

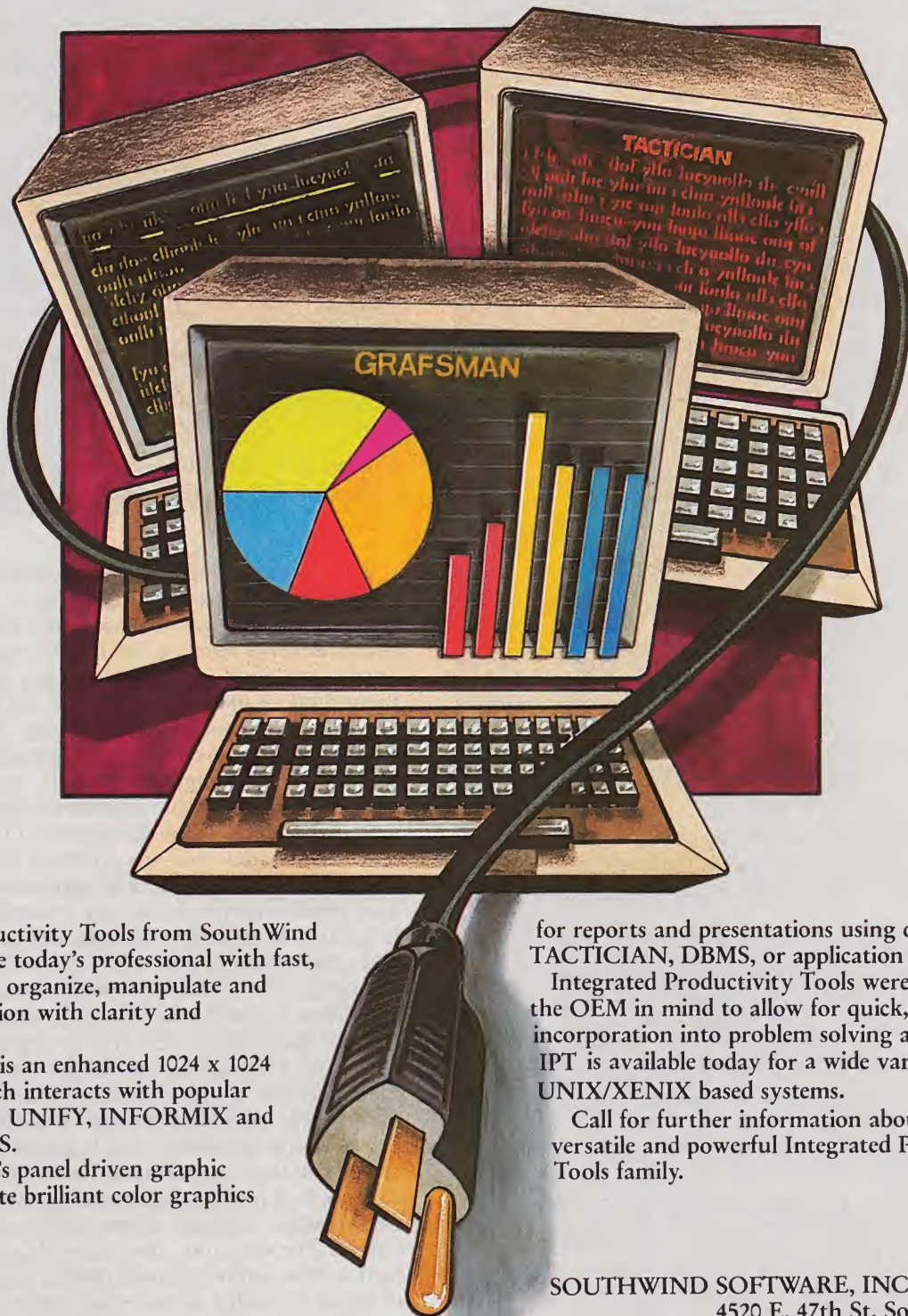
Line modes		
c_lflag		
ISIG	ICANON	XCASE
ECHO	ECHOE	ECHOK
ECHONL		NOFLSH

Figure 5—Line modes.

Control characters		
c_cc		
VINTR	VQUIT	VERASE
VKILL	VEOF	VEOL
VEOL2	VMIN	VTIME

Figure 6—Control characters.

Fast...Versatile ...Powerful Tools



Integrated Productivity Tools from SouthWind Software provide today's professional with fast, versatile tools to organize, manipulate and display information with clarity and effectiveness.

TACTICIAN is an enhanced 1024 x 1024 spreadsheet which interacts with popular UNIX DBMS ie. UNIFY, INFORMIX and MICROINGRES.

GRAFSMAN's panel driven graphic editor helps create brilliant color graphics

for reports and presentations using data from TACTICIAN, DBMS, or application files.

Integrated Productivity Tools were designed with the OEM in mind to allow for quick, easy incorporation into problem solving applications.

IPT is available today for a wide variety of UNIX/XENIX based systems.

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

Integrated Productivity Tools, IPT, TACTICIAN, and GRAFSMAN are trademarks of SouthWind Software, Inc. UNIX is a trademark of AT&T Bell Laboratories. UNIFY is a trademark of the Unify Corporation. INFORMIX is a trademark of Relational Database Systems, Inc. MICROINGRES is a trademark of Relational Technologies, Inc. XENIX is a trademark of Microsoft, Inc.

SOUTHWIND SOFTWARE, INC.

4520 E. 47th St. So.

Wichita, Kansas 67210

316-788-5537

1-800-346-3025 EXT 234



Circle No. 64 on Inquiry Card

Alphabetized modes			
Mode	Type	Mode	Type
B0	control	FFDLY	output
B110	control	HUPCL	control
B1200	control	ICANON	line
B134	control	ICRNL	input
B150	control	IGNBRK	input
B1800	control	IGNCR	input
B200	control	IGNPAR	input
B2400	control	INLCR	input
B300	control	INPCK	input
B4800	control	ISIG	line
B50	control	ISTRIP	input
B600	control	IUCLC	input
B75	control	IXANY	input
B9600	control	IXOFF	input
BRKINT	input	IXON	input
BS0	output	NLD	output
BS1	output	NL1	output
BSDLY	output	NLDLY	output
CBAUD	control	NOFLSH	line
CLOCAL	control	OCRNL	output
CRO	output	OFDEL	output
CR1	output	OFILL	output
CR2	output	OLCUC	output
CR3	output	ONLCR	output
CRDLY	output	ONLRET	output
CREAD	control	ONOCR	output
CS5	control	OPOST	output
CS6	control	PARENB	control
CS7	control	PARMRK	input
CS8	control	PARODD	control
CSIZE	control	TAB0	output
CSTOPB	control	TAB1	output
ECHO	line	TAB2	output
ECHOE	line	TAB3	output
ECHOK	line	TABDLY	output
ECHONL	line	VTO	output
EXTA	control	VT1	output
EXTB	control	VTDLY	output
FF0	output	XCASE	line
FF1	output		

Figure 7—An alphabetic listing of the mode bits along with the structure members they appear in.

*

```
speed 9600 baud; line = 0; intr = ^c; quit = ^|;
erase = ^h; kill = DEL; eof = ^d; eol = ^a
-parenb -parodd cs8 -cstopb -hupcl cread -clocal
-ignbrk brkint ignpar -parmrk -inpck istrip
-inlcr -igncr icrnl -iuclc ixon -ixany -ixoff
isig icanon -xcase echo echoe echok -echonl -noflsh
opost -olcuc onlcr -ocrnl -onocr -onlret -ofill -ofdel
```

Figure 8—Typical System V modes.

*

```
speed 9600 baud; line = 0; intr = ^c; quit = ^|;
erase = ^h; kill = DEL; eof = ^a; eol = ^a
-parenb -parodd cs8 -cstopb -hupcl cread -clocal
-ignbrk brkint ignpar -parmrk -inpck istrip
-inlcr -igncr icrnl -iuclc ixon -ixany -ixoff
isig icanon -xcase echo echoe echok -echonl -noflsh
opost -olcuc -onlcr -ocrnl -onocr -onlret -ofill -ofdel
```

Figure 9—An example of Cbreak mode.

```
speed 9600 baud; line = 0; intr = ^c; quit = ^|;
erase = ^h; kill = DEL; eof = ^a; eol = ^a
-parenb -parodd cs8 -cstopb -hupcl cread -clocal
-ignbrk -brkint -ignpar -parmrk -inpck -istrip
-inlcr -igncr -icrnl -iuclc -ixon -ixany -ixoff
-isig -icanon -xcase echo echoe echok -echonl -noflsh
-opost -olcuc onlcr -ocrnl -onocr -onlret -ofill -ofdel
```

Figure 10—An example of raw mode.

* ioctl differences			
Feature	Cooked	Cbreak	Raw
eof	^d	^a	^a
eol	^^	^a	^a
brkint	on	on	off
ignpar	on	on	off
istrip	on	on	off
icrnl	on	off	off
ixon	on	on	off
isig	on	on	off
icanon	on	off	off
opost	on	on	off
onlcr	on	off	on

Figure 11—Some of the differences between the three "standard" terminal modes of operation.

to a System V computer, your modes will typically be like those shown in Figure 8.

This is known as "cooked" mode. Note that some features such as the kill character, speed and parity can vary. A dash indicates that a feature is turned off. A caret is shorthand for CTRL, as in ^C or CTRL-C. A ^| means CTRL-\ and ^^ means CTRL-@ (NUL). The latter two are, technically, caused by a bug in **stty**'s algorithm that expands control characters to printable representations. Speaking of **stty** bugs, if **stty** is invoked without the **-a** flag, it will list those modes that are different from "standard" cooked mode. If **CREAD** is turned off, **stty** will list it as **cread** when it should say **-cread**. When **CREAD** is turned off, all read requests will return 0 — giving the appearance of end of file (lots of CTRL-Ds). This is usually the result of a programming error.

Cbreak mode looks like the example in Figure 9. Raw mode looks like the example in Figure 10.

Since I do not like to torture people, Figure 11 details some of the differences between the three "standard" terminal modes of operation.

It would be worthwhile to pause at this time and study the lists and table of **ioctl** modes for cooked, Cbreak and raw modes. Some modes are not operational when certain other modes have particular values. For example, the value of **onlcr** is unimportant in raw mode because **opost**, which must be on in order for **onlcr** to have any effect, is off. Because of these "don't care" states, one person's official

*Do you want to run C or
Pascal Software Packages on the 370?*

**Then get the IBM 370 Compiler...
from Whitesmiths, Ltd.**

Now you can...

- Port existing C or Pascal applications to the 370 environment
- Develop new C or Pascal applications on a mini or micro for use on the 370
- Develop applications on the 370 for later migration to minis and micros

***With Whitesmiths'
C/370 compiler you get...***

- Maintenance and support
- Unlimited use of libraries in binary form
- Full implementation of the C language for the IBM 370
- Full implementation of ISO Pascal Level 0 as well
- Optional cross support for the 8080, 8086 and 68000
- Interface to both VM/CMS and OS
- System Interface Source code to facilitate local modifications and enhancements

For more specific information, contact Don Watson, at Whitesmiths, Ltd., (617) 369-8499.



Whitesmiths, Ltd.

97 Lowell Road Concord, MA 01742
TLX 750246 SOFTWARE CNCM

DISTRIBUTORS: Australia, Fawnray Pty. Ltd., Hurstville, (612) 570-6100; Japan Advanced Data Controls Corp., Chiyoda-ku, Tokyo (03) 263-0383; United Kingdom, Real Time Systems, Douglas, Isle of Man 0624-26021; Sweden, Unisoft A.B., Goteborg, 31-125810.

See us at
COMDEX™ / Fall '84
Booth # H 8740



UNIX[™] operating systems. An ideal has been

If you've been waiting for an ideal operating system, your wait is over. Now there's HP-UX. It is Hewlett-Packard's enhanced version of the industry-standard UNIX operating systems. And it's available right now on a wide range of HP computer systems.

Yes. It's running on our MC68000-based machines and our powerful 32-bit systems, so

you can pick the right computer for the job.

There are extra features such as graphics and networking. Plus there's a growing array of applications software available for you to take advantage of.

And the HP-UX operating system is backed by our full service organization. As with each of our high-powered systems, we're ready



realized.

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

Sound interesting? Call your local HP sales office right now about the HP-UX operating system. Or write to Hewlett-Packard, Attn. Pat Welch, Dept. 100201, 19447 Pruneridge Ave., Cupertino, CA 95014. In Europe, con-

tact Henk van Lammeren, Hewlett-Packard, Nederland B.V., Dept. 100201, P.O. Box 529, 1180 AM Amstelveen, The Netherlands.

Productivity. Not promises.



ARTIFICIAL INTELLIGENCE

SMART EXPERT EDITOR(sm)

An Expert System for writing PLAIN ENGLISH instructions. An on-line reporter shows errors in grammar, syntax, vocabulary, readability, translatability, etc. Easy to use. Reads texts directly from VI or word processing.

SMART TRANSLATOR(sm)

An Expert System that automatically translates **English** into **French, Spanish**, etc. Produces fast, accurate technical translations, inter-active or batch. Typeset or laser output.

Programs written in "C" for UNIX.

Request Brochures:

SMART COMMUNICATIONS, Inc.



655 Third Ave., PH
New York, NY 10017
Tel: (212) 486-1894
Telex: 220883 TAUR

Circle No. 58 on Inquiry Card

For Sale

PLEXUS SUPERMICROCOMPUTERS

Available Immediately

2-P35s, 1MB CPU, 72MB Disk,
20MB Cartridge Drive
8 Serial & 1 Parallel Port

1-P60, 4MB CPU, 430MB Disk,
Mag Tape Drive,
40 Serial & 5 Parallel Ports

Contact
(313) 358-5188
Carl Dettinger

Circle No. 57 on Inquiry Card

NORMAN INTERVIEW

Continued from page 49

a single conceptual model, which allows the user to build up the proper mental model.

One of the best examples of this I have is the early Hewlett-Packard stack calculator. The documentation and the instruction manual taught that mental model, and taught every operation through it, thus giving the user a very good understanding of what was going on. That's the prototype of good design.

REVIEW: *In your Datamation article, you talked not only about the need for consistency and explicit modeling in system design — but about the need for mnemonic aids. Could you explain your view of mnemonic aids?*

NORMAN: Let's say you walk up to a terminal that is running almost any conventional operating system. If you're not familiar with that system, no matter how expert you are at computers, there's essentially nothing you can do. You can try entering a question mark or **help**, and just pray that something will come out. Walk up to a Macintosh, though, and as long as you know how to use the mouse, you can experiment with the icons and as you touch each one, they will give you a list of possibilities that you can explore. Rapidly, you can discover what the alternatives are. It's amazing how quickly you can learn a program that way. The Macintosh difference comes in the mnemonic aids it offers.

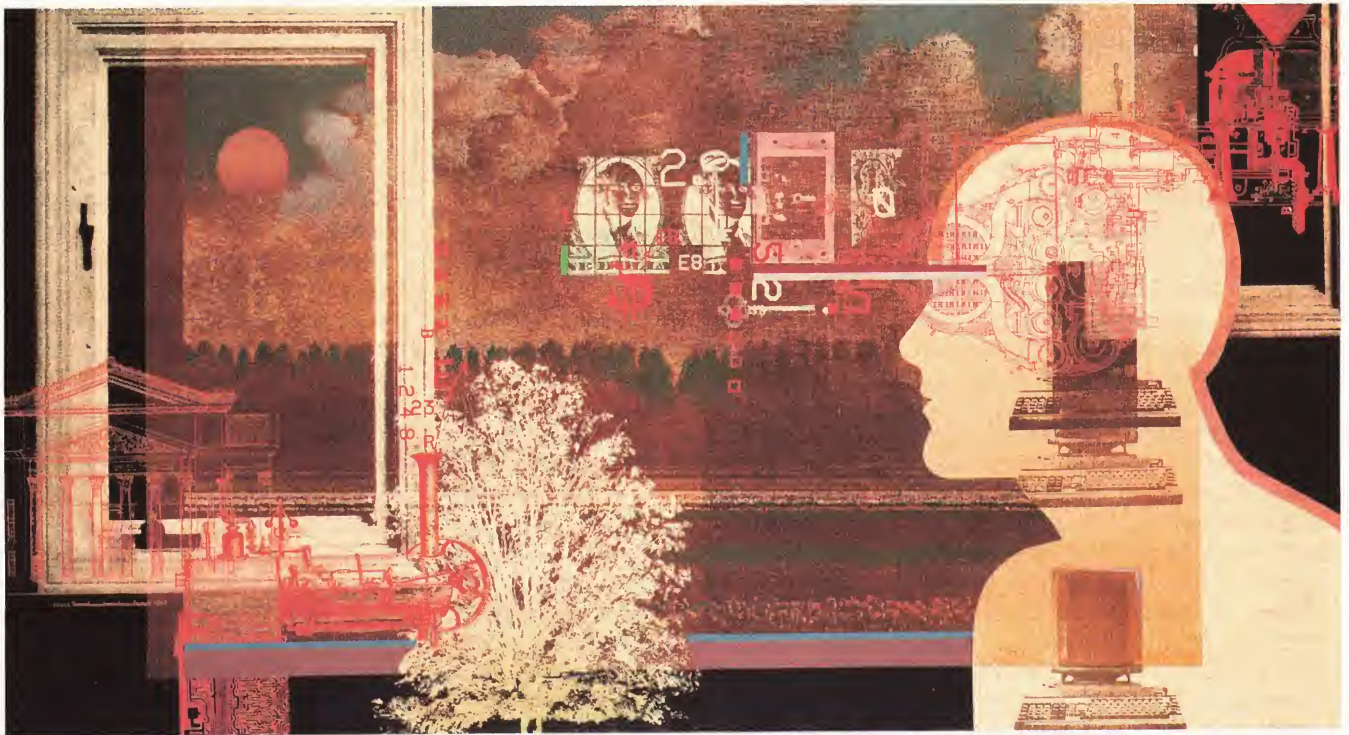
REVIEW: *But there is a great difference in functionality between a Macintosh and a typical UNIX-based system. Should — or could — the two look alike to the user?*

NORMAN: What I was trying to do there was simply contrast the blank screen philosophy that of-

fers no assistance with one that makes it possible quickly to get some notion of the alternatives. Obviously the Macintosh is not as rich or powerful an environment as is needed with a general-purpose system. UNIX offers almost no assistance. Macintosh offers a complete range of assistance, and clearly it must be possible to do similar things on larger systems. Certainly the LISP machine is a good example: a fairly powerful system that offers a lot more assistance to the user. Although it too has its flaws — and I think the people who designed the LISP machine are actually quite deficient in their understanding of the user — the machine goes a long way in the right direction because of its window technology and pop-up menu capability.

You asked about the need for mnemonic aids. I think there are at least two essential things that we should provide in that vein: 1) some physical reminder of the operations that are possible; and 2) spatial layout. We really remember things well if we can put them in a physical place. If I could put my files in a place, I could find them. Not the hierarchy of files as I have in UNIX: that is too abstract. But suppose the hierarchy was real in some sense, such that I could place them on the screen physically, and when I called up a file directory, the same file always came to the same place on the screen. I might not even have to name the file. I might just have to put it in a certain place. I might, say, be working with the file in the bottom left hand corner of the screen while making reference to the file in the upper right hand corner where I keep my notes. That way, I can physically manipulate the location. That's something we haven't exploited well thusfar. Mnemonics, after all, are not just words, but pictures

WHY BUY A DATABASE SYSTEM?



TO BUILD APPLICATIONS!

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

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

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

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

Once your application is built, PROGRESS offers:

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

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

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



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

PROGRESS

A NEW APPLICATION ENVIRONMENT

Want to make PROGRESS in a hurry?
Call Area Code 617-663-5000 today.
Or just fill out and return the coupon below.

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

- Yes! Please send me a demonstration disk and documentation for PROGRESS. I enclose \$295.
- Please send more information about PROGRESS.
- I am interested in PROGRESS seminars.

Name _____ Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Telephone (_____) _____

4

PROGRESS is a trademark of Data Language Corporation, developers of advanced software technology for business and industry.

UNIX is a trademark of AT&T Bell Laboratories.

Circle No. 42 on Inquiry Card



and shapes and locations as well.

I went on a tour of Austria recently on an extremely fancy tour bus. I have never seen a bus that was so elaborate. I sat behind the driver and counted the controls. When I reached about 80 switches and controls, I gave up. There was clearly a lot more than that. That's a large number of things to learn. So I had my companion ask the driver (in German) how hard it was to learn the controls. The driver thought it was one of the strangest questions he had ever heard. "Why should it be hard to learn?" he asked. "They're just there, and you use them. It's obvious what everything does. Hard to remember? What is there to remember? All the switches are where they ought to be, and all the ones that are related to one another are near one another."

What is the difference between these normal kinds of environments, which are really very crowded and very rich, and the computer environment? The difference is space. Space and physical existence. These things are located in a certain spot in space, and they're always there — they're physically present. That's a dimension I think should be explored in computer systems.

What we see all too often is another sort of phenomenon altogether. Take, for example, a modern aircraft, like a 747 or a 767. There are thousands of controls throughout. People are worried about that. They take too much space, making it too hard to see everything at once. In commercial aircraft, the windows are awfully small because the plane's designers have to make enough room for the controls. You can barely see out. Do you realize that when you land in a commercial aircraft, the pilot cannot see the ground where the airplane is hitting? So what are they going to do about it? It's likely that they'll replace the instruments with

video displays. Do you know what's going to happen? The pilots are going to lose all the benefits of spatial layout. Now, if you want to know what the oil pressure is, you look to the left. You want to know how much fuel is left, you look to the right. You want to see the altimeter reading, so you look in front of you. For air speed, you look here. To verify flight path, you look there. With these video displays, this won't be true anymore. Everything will be on top of each other. You change the display and get fresh information, displayed physically right where the old information screen was. That's why I think we're going to be in real trouble. The tremendous power of spatial information is about to be lost. That isn't the answer.

REVIEW: *So how do we apply these lessons to computer design?*

NORMAN: Part of the problem is that cognitive science — understanding the mind, understanding how people work — is truly a new science. I can't give you any real answers. It's going to be a while. The best thing we can do now is at least have sensitivity, and be patient with a lot of experimentation and false starts.

REVIEW: *In McIlroy's response to your article, he states that a coherent user interface is neither desirable nor possible in a universe as broad as that of UNIX. I expect you probably have a response to that.*

NORMAN: I think that he and I must be talking at cross purposes. I have too much respect for McIlroy's ability as a designer to think that he means what I think he means. The UNIX system philosophy is consistent across all systems. There is essentially the notion of filters and pipes for interconnecting programs and input-output redirection mechanisms for manipulating them. There's a notion of basic signals. There are

also the common motions of the control mechanisms among programs and among data structures and the notion of a standard I/O package called by everybody. There's a lot of consistency in UNIX. There just isn't consistency at the surface level. I see no reason why that same kind of consistency couldn't be expanded to the surface. There is no reason why we have to have different characters meaning the same things, where each program has to have a different syntax, or where the flags are arbitrary and different from program to program. UNIX runs into a real problem, as I've said, because it's so large, covers such a broad area, and is so expandable by anybody. Surely McIlroy and I must be thinking of different things when I say that consistency is important and essential and can be done, while he says it can't.

REVIEW: *Another point McIlroy makes is that the features, options, hooks, and tailorability associated with friendly interfaces are just so much "building material for a tower of Babel." Do you feel user interface features are necessarily tantamount to creeping featurism?*

NORMAN: Blech! The tower of Babel is what we have now. That's what I'm trying to avoid. What I want is an *Esperanto*. What I want is a common language, a common style for doing anything. The current system is a tower of Babel. Creeping featurism! I just told you about **vi** and **emacs**, about what happens when every time you get a new idea you add a new feature. That leads to a tower of Babel. What I want is a consistent conceptual model on the part of the designer, a consistent physical image that the system is built to convey, and as a result a consistent mental model on the part of the user. I would actually prefer to forego features, if necessary, to keep that consistent, to keep it clean and keep it usable. ■

UNIX™ Program For Profit

```
-10  FOR UNIX. SALES = SOFTWARE. SOLUTIONS
-20      SELECT UX. BASIC
-30      CASE APPLICATIONS
-40      CASE PORTABILITY
-50      CASE PRODUCTIVITY
-60      CEND
-70      NEXT UNIX. SALES
-80      PRINT ( PROFIT = UNIX. SALES ) ^UX. BASIC
RUN (Other Basics may not give you the same result).
```

- UX-Basic™ Provides Applications

- From OASIS™ Basic, accessing existing applications base
- Through UX-Link™ interfacing to DBMS and graphic libraries
- Via UX-Translate™ tapping applications from other Basics

- UX-Basic Provides Portability

- Language is available across a range of UNIX machines
- Programs will execute on any machine running UX-Basic
- Device independence allows program use on any terminal

- UX-Basic Provides Productivity

- With a full featured editor, development and debugging tools
- With easily readable, highly structured, modular style code
- With business features including: C-ISAM™ and BCD arithmetic

UX-Basic®

UNIX is a Trademark of AT&T Bell Laboratories.
UX-Basic, UX-Link and UX-Translate are Trademarks of
UX Software, Inc.
OASIS is a Trademark of Phase One Systems, Inc.
C-ISAM is a Trademark of Relational Database Systems, Inc.

See us at
COMDEX / Fall '84
Las Vegas Convention Center
Booth 1196

UX-Basic is available directly from equipment
manufacturers and selected software distributors.
UX Software, Inc.
10 St. Mary Street, Toronto, Canada M4Y 1P9
(416) 964-6909

Circle No. 52 on Inquiry Card

RECENT RELEASES

RAPITECH INTRODUCES FORTRIX

Users operating in a UNIX/C environment can now use FORTRIX-C, Rapitech Systems' new program that converts old Fortran programs and files to C code.

FORTRIX-C is a complete package configured to meet the requirements of various UNIX environments. Included are integer-character string converters, space allocators, string parsers and other string manipulators not included in standard C libraries.

FORTRIX-C is designed to make as few run-time assumptions as possible. During execution, it solicits the information required to run the program from the user. Usually, responses are either yes or no; when yes or no answers are not appropriate, a menu of possible answers is given.

In operation, FORTRIX-C converts Fortran flow control statements (**do**, **goto**, **if else**, **else if**, **continue**, **return**, **end**, **call**) to functionally equivalent sets of C instructions. For example, a **do** loop is reconstructed in C as a **for** loop; **continue** statements are reduced to null statements; I/O statements are expanded into fully parallel sets of I/O instructions. Read and write field widths are retained and the C code spacing requirements are exactly those of the parent Fortran program. Output streams for both the original and the new code look the same since all Fortran functions and subroutines are recast as C functions.

Rapitech has designed

FORTRIX-C to produce C code that retains the essence of the original Fortran source. The generated C code is fully structured and parallel to the Fortran source code. There is ready identification of program elements, which — combined with the sequential nature of the conversion — make the comparison of the Fortran and C programs straightforward. As a result, entry level programmers can use the FORTRIX-C program's output as a self-instruction aid in understanding the architecture of C code.

FORTRIX-C's conversion process also provides a measure of error validation for the input Fortran code.

FORTRIX-C executes at a typical conversion rate of 600 lines of code per minute.

FORTRIX-C is priced at \$2750. The program is available on disk or tape and includes an instruction manual as well as telephone "Hot Line" support. Delivery time is two weeks after receipt of order.

FORTRIX-C works with most major computers. Upon request, it will be configured to work with any manufacturer's proprietary system. For further information about FORTRIX-C conversion programs, contact Rapitech Systems, 565 Fifth Avenue, New York, NY 10017; 212/687-6255.

Circle No. 43 on Inquiry Card

SCO ANNOUNCES "XENIX NOW!" PROGRAM

The Santa Cruz Operation (SCO) has launched "XENIX NOW!" The program is designed

to promote and heighten awareness of the XENIX Operating System for IBM Personal Computers among dealers, vendors, OEMs and end users. SCO's XENIX packaged product for the IBM PC-XT and compatibles — including AT&T's new PC — is available. IBM PC XENIX for the AT and SCO PC XENIX for the XT are fully compatible.

Although the IBM PC is based upon the 8088 microprocessor while the PC-AT is designed around the new 80286, the portability of XENIX ensures that the user interface of the operating system and its compatible applications on each machine is identical. The line of SCO application software for the 8088-based PC has been ported to the 80286, so that each SCO-supported program is now available for the PC-AT and other 80286-based systems as well. These include the Lyrix Word Processing System, the Multiplan Electronic Worksheet, the Informix Relational Database Management System, Level II COBOL and its powerful development tools, Animator and Forms-2.

SCO offers XENIX for the IBM PC at \$595, the XENIX Development System at \$595 and the XENIX Text Processing System at \$495. All three systems are available as the Complete XENIX System at \$1350 list. (XENIX and all applications are also available for the Apple Lisa II.)

For more information call the Santa Cruz Operation at 408/425-7222.

Circle No. 44 on Inquiry Card

Circle No. 45 on Inquiry Card ►

Gould... Innovation and Quality in UNIX-based systems



Our Firebreathers are scorching old performance standards.

Gould's PowerNode™ 9000 blasts through UNIX* benchmarks at 4.5 times the speed of the VAX™ 11/780. Sound impossible? Give us your real production code or benchmarks and let us prove it.

Firebreathing Performance.

Now you can run software development and production at the same time, with highly responsive performance. Tightly coupled dual processors nearly double throughput and virtual memory accommodates large programs. Hardware fixed point and floating point accelerators retain high performance in heavy number-crunching situations. The PN9000 handles mainframe jobs in a multi-user UNIX system or serves as a backend processor in a widely distributed network.

Unique UNIX Software.

Gould's own high performance UNIX-based operating system (UTX/32™)—a unique combination of Berkeley 4.2 with special Bell System V features—makes it easy for you to use your VAX-based UNIX software. This allows easy conversion from your system to the increased power of a Firebreather.

Compatible Family.

Gould's Compatibility Suite is a collection of application software packages that are compatible across the entire PowerSeries™ product line. Use C, Cobol, BASIC, or Pascal languages intermixed. This close-knit processor family offers all the advantages of

a dedicated system plus the lower-cost-per-user option of sharing resources with Gould's standard networking capabilities including Ethernet™. The Firebreathers are the high end of the widest range of UNIX-based systems in the industry.

Gould's Firebreathers are scorching the UNIX market.

**Gould Inc.,
Computer Systems Division**
Distributed Systems Operation
6901 West Sunrise Boulevard
Ft. Lauderdale, Florida 33313
(305) 797-5459

*UNIX is a trademark of AT&T Bell Labs
™PowerNode, PowerSeries and UTX/32 are trademarks of Gould Inc.
™Ethernet is a trademark of Xerox Corporation
™VAX is a trademark of Digital Equipment Corporation



GOULD
Electronics

LANTECH DEVELOPS UNIX LOOK-ALIKE

uNETix, a multitasking operating system compatible with UNIX software and designed for networking of microcomputers, has been developed by Lantech Systems, Inc.

Lantech will offer three versions of uNETix — a standalone and two networking products. uNETix already operates on many 8086/8088-based microcomputers, including the IBM PC look-alikes. Projects are under way to port it to systems based on the Motorola 68000 and NSC 10632 microprocessors.

Available now to OEMs and end users, standalone uNETix features an MS-DOS emulator to support both MS-DOS and UNIX

applications software. Offering the personal computer user a versatile, multitasking operating system, uNETix allows display of up to 10 active windows on the screen, processing a different task in each window simultaneously. Data or text can be transferred directly from one window to another.

Lantech also offers uNETix-DFS, the first networking version of uNETix, which utilizes a distributed file system to allow transparent remote file and device access. uNETix-DFS is the result of joint development efforts between Lantech and Plexus Computers.

uNETix transfers data from one window to another directly through the system's internal memory, without requiring that

the user have Winchester disk storage.

As an option to standalone uNETix, uNETix-DFS permits the user to interconnect uNETix-based computers with a distributed file system. In addition, since uNETix-DFS is fully compatible with the UNIX System III Network Operating System developed by Plexus, the Lantech product gives the user the capability to build a transparent micro/minicomputer network. Computers in the network also can tap the processing capabilities of other systems through a virtual terminal capability.

With uNETix-DFS, portions of file systems on remote computers can be attached to the local file hierarchy. Once a path to the remote file is established, using a

UNIX™ VALIDATION



Validation Suite versions available

- UNIX System III and System V (tm Bell Labs)
- BSD 4.1 and BSD 4.2
- /usr/group Standard
- XENIX (tm Microsoft)

Validation Suites Include

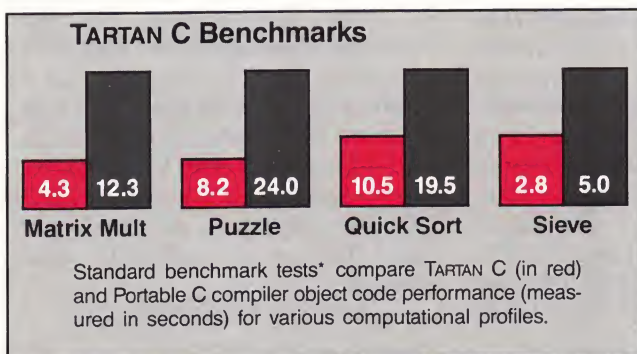
- Source code for entire suite
- Template for adding tests
- Installation support service
- User documentation
- Audit trail for all tests
- Test driver control program
- System exercisor—stress tests
- Automatic and manual tests
- Touch tests of all utilities
- Functional tests for all system calls

PERENNIAL SOFTWARE SERVICES GROUP

3130 De La Cruz Blvd. • Santa Clara, California 95054 • (408) 727-2255
SOFTWARE DEVELOPMENT, TESTING, MAINTENANCE AND SUPPORT SERVICES

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

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



Get More Performance Into Your Applications.

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

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

Get More Productivity Out of Your Programmers.

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

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

There's More.

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

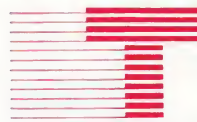
Call Today.

Order The C Compiler That's Better Than Free.

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

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

We want to be your favorite software company.



TARTAN

TARTAN Laboratories
Incorporated
477 Melwood Avenue
Pittsburgh, PA 15213

TARTAN C Error Messages

C Program with Syntax Errors:

```

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

Portable C Compiler Error Messages:

```

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

TARTAN C accurately pinpoints errors and recovers:

```

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

*A complete report on these and other benchmarks is available on request.
UNIX is a trademark of AT&T Bell Laboratories
VAX is a trademark of Digital Equipment Corporation.
©1984 TARTAN Laboratories Incorporated

remote **mount** command, the file is accessed exactly as if it were part of the local file system hierarchy. The user does not have to learn new commands to access remote files.

uNETix-DFS operates through a higher-level communications protocol, independent of the physical network medium. For local area networks, it supports Ethernet, with other popular interfaces to be added soon.

All three versions of uNETix will be marketed through OEMs and systems integrators, as well as be available to end users. All versions of uNETix, which Lantech can customize to run on any hardware system, are compatible with standard UNIX software, including the UNIX shell, file system and interface for disk drives, displays and keyboards.

More information on Lantech

and its products is available by contacting the company at 9635 Wendell Road, Dallas, TX 75243; 214/340-4932.

Circle No. 49 on Inquiry Card

SYDIS ENHANCEMENTS

With the introduction and shipment of Release 2.0, Sydis Inc., has added six major enhancements to its VoiceStation System. The system can now communicate with IBM mainframes and other computers that support asynchronous dial-up terminals. Applications development tools, relational database management software, laser printer capability and an electronic calculator have also been added.

The rollout of these enhancements follows Sydis' move to volume production after signing OEM pacts with GTE,

Italtel Telematic and Compath National.

Sydis added IBM 3270-type terminal capabilities to the system by using the Micom Micro 7400 Protocol Converter as a low-cost replacement for the IBM control unit. Four, eight and 12-port options are available that allow the VoiceStation System to perform IBM 3274 BSC or SNA emulation. A dual-host option that supports two independent mainframes is available as well.

The VoiceStation 1's keyboard and softkeys are "mapped" to perform 3270 functions and to control the size of the 3270 window on the VoiceStation screen.

In addition, the VoiceStation 1 can emulate a VT-100 or ANSI X3.64-compatible terminal, and gain access to corporate databases.

Circle No. 50 on Inquiry Card

CCA EMACS. THE MOST POWERFUL SCREEN EDITOR FOR UNIX AND VAX/VMS.

No other text editor gives you so much power, speed, and functionality as CCA EMACS™. Or makes editing so easy. Close to 400 built-in commands let you do any task with only a few keystrokes. Even things that are impossible on other editors. And with our full extension language, Elisp™, you can customize CCA EMACS to meet your program requirements.

Multiple windows is another CCA EMACS plus. So you can manage concurrent processes and move information from one window to another. And



CCA EMACS is supported by a full online documentation package that includes a novice tutorial. So any user can quickly utilize all the power of CCA EMACS.

CCA EMACS runs on Berkeley Unix™ (4.1BSD and 4.2BSD), Bell Unix (System III and System V), and VAX/VMS™ and requires 500 K of address space.

Prices for a full source license range from \$350 to \$2400.

For more information, or to find out how to get a trial copy, call Gwendolyn Whittaker at (617) 492-8860.

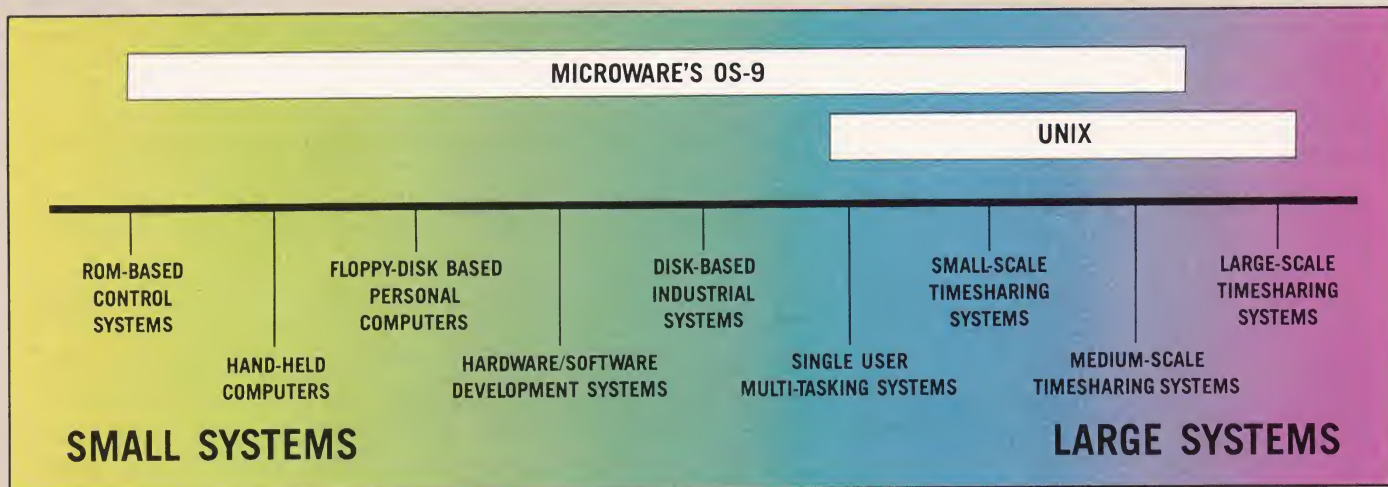
CCA Uniworks, Inc.

‡ A Crowntek Company
Four Cambridge Center, Cambridge, MA 02142

Unix is a trademark of Bell Laboratories
VAX and VMS are trademarks of Digital Equipment Corporation
CCA EMACS and Elisp are trademarks of CCA Uniworks, Inc.

Circle No. 48 on Inquiry Card

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.

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.

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

microware[®]
OS-9[™]

MICROWARE SYSTEMS CORPORATION
1866 NW 114th Street
Des Moines, Iowa 50322
Phone 515-224-1929
Telex 910-520-2535

Microware Japan, Ltd
3-8-9 Baraki, Ichikawa City
Chiba 272-01, Japan
Phone 0473(28)4493
Telex 299-3122

OS-9 is a trademark of Microware and Motorola. Unix is a trademark of Bell Labs.

Circle No. 51 on Inquiry Card

CAPE (see November 7-9).

December 10-14 Bunker Ramo Information Systems, Trumbull, CT: "Advanced C." Contact: Bunker Ramo (see November 5-9).

December 10-14 Computer Technology Group, Los Angeles, CA: "UNIX Internals." Contact: CTG (see November 6).

December 10-14 Structured Methods Seminar, New York, NY: "UNIX System Workshop." Contact: Structured Methods (see November 5-9).

December 11-14 Integrated Computer Systems Seminar, Palo Alto, CA: "Programming in C: A Hands-On Workshop." Contact: ICS (see November 6-9).

December 12-13 Intelligent Solution Seminar, Boston, MA: "Programming in C." Contact: Intelligent Solution (see November 2).

December 12-14 Computer Technology Group, Boston, MA & Washington, DC: "Using Advanced UNIX Commands." Contact: CTG (see November 6).

December 12-14 Digital Seminar Program, Los Angeles, CA: "The C Programming Language." Contact: Digital Educational Services (see November 14-16).

December 14 Intelligent Solution Seminar, Boston, MA: "UNIX Overview." Contact: Intelligent Solution (see November 2).

December 17 AT&T Technologies, Dublin, OH: "UNIX System V Administration." Contact: AT&T (see November 5).

December 17 AT&T Technologies, Sunnyvale, CA: "Internal UNIX System Calls and Libraries Using C Programming." Contact: AT&T (see November 5).

December 17 NCR Education Seminar, Dayton, OH: "UNIX

System Administration." Contact: NCR (see November 5).
December 17-19 Structured Methods Seminar, New York, NY: "UNIX System Internals." Contact: Structured Methods (see November 5-9).

December 17-21 Bunker Ramo Information Systems, Trumbull, CT: "C Programming." Contact: Bunker Ramo (see November 5-9).

December 17-21 Computer Technology Group, Boston, MA & Washington, DC: "UNIX Internals." Contact: CTG (see November 6).

December 18-20 Computer Technology Group, Los Angeles, CA: "UNIX Administration." Contact: CTG (see November 6).

December 19-21 CAPE Seminar, Tampa, FL: "A User-Oriented Evaluation Three-Day Seminar: UNIX." Contact: CAPE (see November 7-9).

December 20 AT&T Technologies, Lisle, IL: "Overview of the UNIX System." Contact: AT&T (see November 5).

December 20-21 Structured Methods Seminar, New York, NY: "Using Lex and yacc." Contact: Structured Methods (see November 5-9).

Please send announcements about training or events of interest to:

UNIX Review Calendar

520 Waller Street

San Francisco, CA 94117.

Please include sponsor, date and location of event, address of contact and relevant background information.

ACUITY® business software is compatible with any budget, and all these systems:

UNIX-based Micros	VAX, VMS or UNIX
PRIME	Convergent
IBM-PC	Harris

With prices from \$700 to \$6500 for a fully supported package, any size company can afford our general accounting and specialized project cost software.

Packages available include project management, labor/ODC forecasting, work breakdown structure, customer order processing, bill of materials processing and inventory management.

Plus a complete set of accounting software including general ledger, payables, receivables, payroll and fixed assets.

Call (619) 474-2010 for details.

See us at COMDEX at the PLEXUS Booth #'s 2392, 2388, 2493, 2489.



225 West 30th Street, National City, California 92050

Circle No. 54 on Inquiry Card

If you must have
Lotus 1-2-3 on Unix-Try Q-CALC

Q-CALC offers
full Lotus 1-2-3 capabilities
including graphics

+

Translation of 1-2-3 models into Q-CALC

For more information write/call
QUALITY SOFTWARE PRODUCTS

**348 S. CLARK DRIVE
BEVERLY HILLS, CA. 90211
213-659-1560**

*Lotus 1-2-3 is a trademark of Lotus Corp.
Unix is a trademark of Bell Labs

Circle No. 59 on Inquiry Card

Your schedule needn't be carved in stone.



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



FEATURES

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

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

MICROTRAK™

APPLICATION SOFTWARE FOR PROJECT MANAGEMENT

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

ADVERTISER'S INDEX

Accucom Data	74	MIPS Software Development, Inc.	68
Advanced Ergonomic Management	53	MT XINU	81
Aim Technology, Inc.	85	Network Consulting, Inc.	30
Applix, Inc.	9	Network Research Corp.	62
AT&T Technologies, Inc.	48	Perennial Software Services	100
B.A.S.I.S.	83	Plexus Computers, Inc.	19
Beach Software	73	Prescience, Inc.	90
CCA Uniworks, Inc.	102	Quality Software Products	106
Century Software	61	Relational Database Systems, Inc.	5, 6, 7
Codata	113	Rhodnius, Inc.	71
Computer Cognition	106	Santa Cruz Operation	Cover III
Computer Methods, Ltd.	11	Scientific Placement, Inc.	29
Computer Technology Group	47	Smart Communications	94
Convergent Technologies	13, 54, 55	Softrag Systems	107
Cromemco	95	Software Results Corp.	78
Data Language Corp.	95	Southwind Software, Inc.	87
Emerald City, Inc.	31	Starwood Group	53
Faircom	84	Syntactics	75
Fortune Personnel Consultants	29	Tartan Laboratories	101
Gould Electronics	99	Telecon Systems	78
Handle Corp.	15	Turn-Key Data	94
Heurikon Corp.	34	UniForum	77
Hewlett-Packard	92, 93	Unify	Cover II, 1
I.B.C.	Cover IV	Unipress Software, Inc.	69
IBM	40, 41	UniSoft Systems	83
Image Network	12, 56	UNIX Helpline	46
I.T.S.	79	UX Software, Inc.	97
Logical Software, Inc.	72	User Training Corp.	46
Manx Software Systems	25	Visual Technology, Inc.	58, 59
Mark Williams Company	17	Whitesmiths, Ltd.	89
Microware Systems Corp.	103	Xidak	65

Coming up in December

Personal Low-cost UNIX

- The UNIX/Hardware Connection
- Different flavors of UNIX
- UNIX on the IBM PC-XT
- Products on the horizon
- Technology futuristics

BACK ISSUES AVAILABLE

- | | |
|---------------------------------|--------------------------|
| June/July 1983 | <input type="checkbox"/> |
| August/September 1983 | <input type="checkbox"/> |
| October/November 1983 | <input type="checkbox"/> |
| December/January 1984 | <input type="checkbox"/> |
| February/March 1984 | <input type="checkbox"/> |
| April/May 1984 | <input type="checkbox"/> |
| June 1984 | <input type="checkbox"/> |
| July 1984 | <input type="checkbox"/> |
| August 1984 | <input type="checkbox"/> |
| September 1984 | <input type="checkbox"/> |
| October 1984 | <input type="checkbox"/> |

All back issues are \$4.95, including postage and handling. Enclose payment or credit card information or call 206/271-9605.

Name _____
 Company _____
 Address _____
 City _____ State _____ Zip _____
 M/C or VISA _____
 Exp. Date _____

“A UNIX™ TO BE PROUD OF!”

—Kaare Christian, PC Magazine



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!



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

©MCLXXXIV 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.

Circle No. 55 on Inquiry Card



UNIX™ HORSEPOWER!

There are a lot of UNIX based systems on the market today claiming to be "SUPERMICROS". But do they really have what it takes to run multi-user UNIX well? The IBC ENSIGN™ does and here's why:

FAST MEMORY: No computer running at **any** clock speed can run faster than it's overall memory design. The ENSIGN has up to 8MB of 120nsec memory with dual bit error correction. With IBC's proprietary memory management, **all** of this memory runs with no wait states as fast as the 68000 CPU will go. Compare this to other systems running only small cache memories at full speed. Other multiple user systems cannot load all their programs into a small cache memory. Their systems slow down considerably under a heavy multi-user load.

INTELLIGENT SERIAL I/O CONTROLLER: Even the fastest CPU will slow down when it's trying to handle interruptions from multiple on-line users. The ENSIGN provides slave serial I/O CPU's **and** FIFO buffering for both input and output. The result is the ENSIGN's ability to support up to 32 users, with heavy serial I/O demand, while leaving the main 68000 CPU free to run with little serial I/O overhead.

INTELLIGENT DISK CONTROLLER AND HIGH PERFORMANCE DISK DRIVES: The ENSIGN has a slave CPU to handle all disk operations, **plus** 16K of disk buffering. IBC's proprietary disk DMA allows high speed data transfer to main memory without slowing down the main CPU. Further, the ENSIGN

supports SMD type 8" hard disks with much faster seek times and transfer rates than 5¼" hard disks usually found in personal desk top computers.

THE RESULTS: The IBC ENSIGN runs multi-user UNIX at performance levels not attainable by other supermicros.

Call IBC and get a copy of IBC's multi-user benchmarks—benchmarks that test 8 users running large CPU programs, with heavy disk I/O **and** heavy serial I/O simultaneously. You'll find that nothing can compare to the ENSIGN.



If you want to run multi-user UNIX on a high performance system with up to 32 users, 8MB memory, and over 1,000MB disk storage, see the IBC ENSIGN.

IBC / Integrated Business Computers

21621 Nordhoff Street
Chatsworth, CA 91311
(818) 882-9007
Telex No. 215349

UNIX is a trademark of Bell Laboratories **Circle No. 56 on Inquiry Card**