

UNIX™ REVIEW

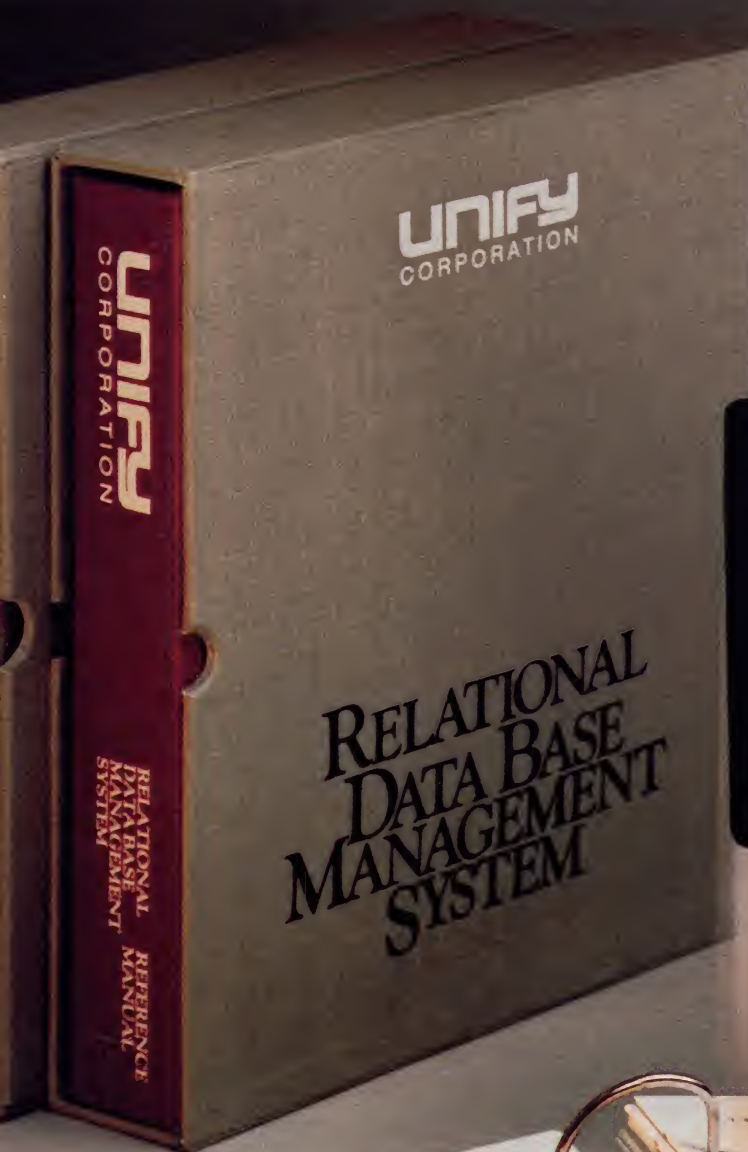
THE PUBLICATION FOR THE UNIX™ COMMUNITY

March 1985 \$3.95



PERFORMANCE

47.69



```

[system]
UNIFY SYSTEM
17 SEP 1984 - 18:81
System Menu

1. Schema Maintenance
2. Schema Listing
3. Create Data Base
4. SFORH Menu
5. ENTER Screen Registration
6. SQL - Query/DML Language
7. SQL Screen Registration
8. Listing Processor
9. Data Base Test Driver
10. MENUH Screen Menu
11. MENUH Report Menu
12. Reconfigure Data Base
13. Write Data Base Backup
14. Read Data Base Backup
15. Data Base Maintenance Menu

```

```

[student]
[INQUIRE]
UNIFY SYSTEM
25 Aug 1985 - 18:45
Student Registration Form

Invoice Number: 458

Last Name: Gordon      First Name: Richard

Company: Silicon Design Labs
        5550 Industrial Way
        Basking Ridge NJ 07898
        (201) 555-5488
Student's phone number (if different): (201) 555-5421

Class code (enrwy): CP8905      Subject: C Programming
Class fee: 395.00              Class date: 9/1/85
Deposit date: 8/15/85          Deposit amount ($): 188.00
Payment date: 8/25/85          Payment amount ($): 895.00

```

```

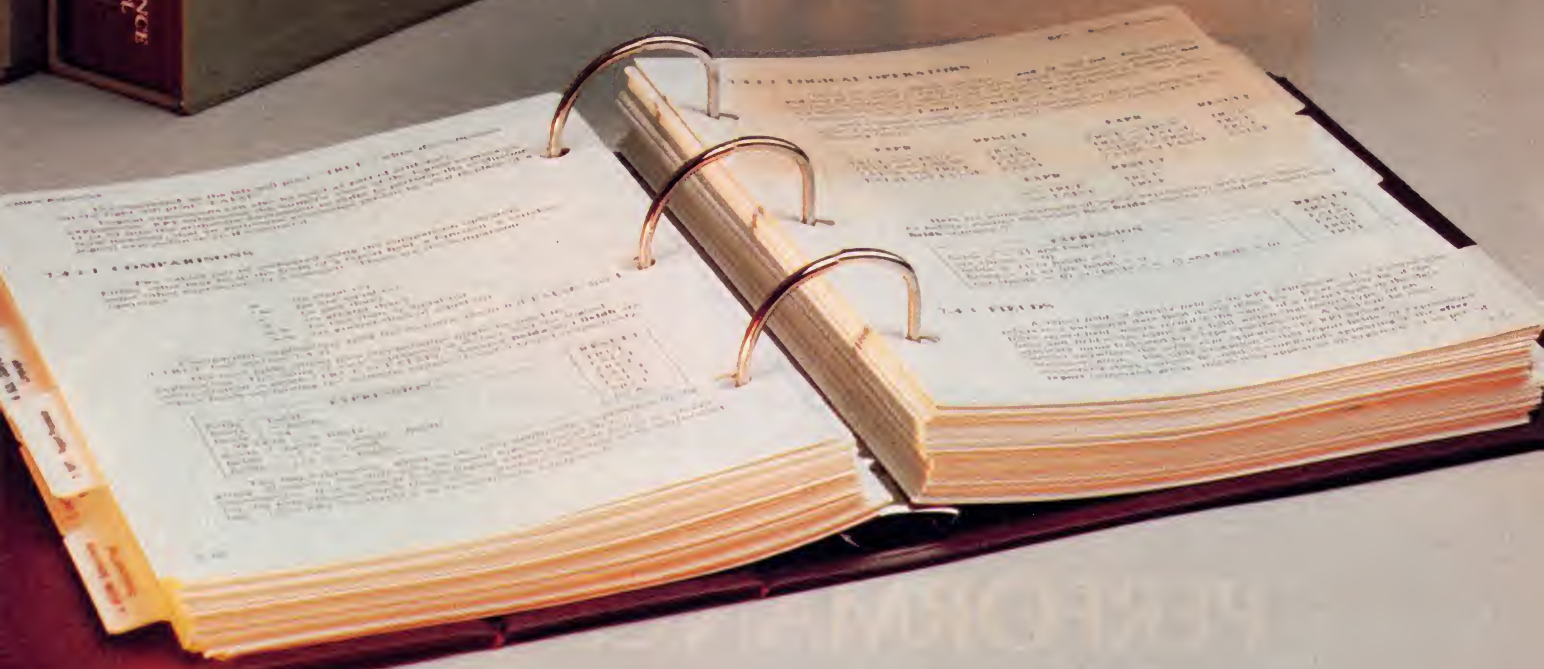
[student]
[INQUIRE]
UNIFY SYSTEM
25 Aug 1985 - 18:45
Student Registration Form

Current: 1

REPORT          TO: SCREEN PRINT FILE FILENAME
1. Student Registration Listing  (x)  ( ) (x)—listing
2. Student Billing                ( )  ( )
3. Billing Summary                 ( )

REPORT #: 1

```



UNIFY. THE MIGHTY EASY DBMS.

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

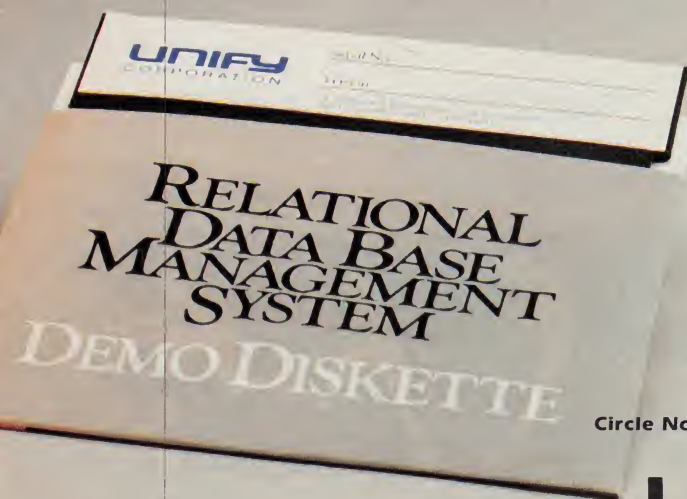
UNIFY guides the nonprogrammer through data base development with minimal steps, comprehensive menus, on-line HELP, elementary Query By Forms capability, and clear documentation.

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

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

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

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



Circle No. 300 on Inquiry Card

UNIFY®
THE PREFERRED DBMS.



UNIX™ REVIEW

THE PUBLICATION FOR THE UNIX COMMUNITY

Volume 3,
Number 3
March 1985

DEPARTMENTS:

- 4 Viewpoint**
- 8 The Monthly Report**
By Mark Hall
- 14 The Human Factor**
By Richard Morin
- 60 Industry Insider**
By Mark G. Sobell
- 64 Rules of the Game**
By Glenn Groenewold
- 68 C Advisor**
By Bill Tuthill
- 78 Devil's Advocate**
By Stan Kelly-Bootle
- 84 The UNIX Glossary**
By Steve Rosenthal
- 104 Calendar**
- 108 Advertiser's Index**

Cover art by J. Kelly Davies

FEATURES:

22 PERFORMANCE DEFINED

By Rob Warnock

A look at a word that is often used but rarely understood.



26 BENCHMARKS UNVEILED

By Gene Dronek

What benchmarks are—and aren't.





PERFORMANCE

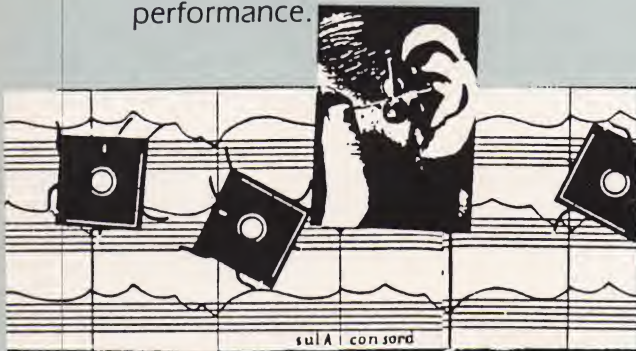
32 ISOLATING PROBLEMS

By Clement T. Cole and Ed Breslin
System diagnosis recommendations that can lead to effective performance tuning.



38 SYSTEM TUNING

By John Bass
An endorsement of some "tried-and-true" methods for optimizing performance.



42 INTERVIEW WITH GEORGE GOBLE

By Dick Karpinski
One of the UNIX community's most noted performance jocks speaks out on his favorite topic.



50 PERFORMANCE SOFTWARE

By Roger J. Sippl
Suggestions for optimizing programs for the UNIX environment.



UNIX REVIEW (ISSN-0742-3136) is published monthly by REVIEW Publications Co. It is a publication dedicated exclusively to the needs of the UNIX community. Second class postage paid at Renton, WA 98055. POSTMASTER: Please send Form 3579 to UNIX REVIEW, 500 Howard Street, San Francisco, CA 94105. Entire contents copyright 1985. All rights reserved and nothing may be reproduced in whole or in part without prior written permission from UNIX REVIEW.

UNIX REVIEW is sent free of charge to qualified individuals in the U.S. The publisher reserves the right to determine qualification. For those not qualified, there is an annual (12 issues) subscription charge of \$28 in the U.S., \$35 in Canada and \$48 in all other countries (surface mail). Correspondence regarding editorial (press releases, product announcements) and circulation (subscriptions, fulfillment, change of address) should be sent to 500 Howard Street, San Francisco, CA 94105. Telephone 415/397-1881. Correspondence regarding dealer sales should be sent to 901 South 3rd Street, Renton, WA 98055. Telephone 206/271-9605.

Letters to UNIX REVIEW or its editors become the property of the magazine and are assumed intended for publication and may so be used. They should include the writer's full name, address and home telephone number. Letters may be edited for the purpose of clarity or space. Opinions expressed by the authors are not necessarily those of UNIX REVIEW.

UNIX is a trademark of Bell Laboratories, Inc. UNIX REVIEW is not affiliated with Bell Laboratories.

PUBLISHER:

Pamela J. McKee

EDITORIAL DIRECTOR:

Stephen J. Schneiderman

EDITOR:

Mark Compton

ASSOCIATE EDITOR:

Mark Hall

ASSISTANT EDITORS:

Ken Roberts, Scott Robin

PRODUCTION DIRECTOR:

Nancy Jorgensen

PRODUCTION STAFF:

James Allen, Dan Arthur, Tom Burrill, Cynthia Grant, Tamara V. Heimarck, Corey Nelson, Florence O'Brien, Denise Wertzler

DEALER SALES:

Tracey McKee, Barbara Perry

CIRCULATION DIRECTOR:

Wini D. Ragus

CIRCULATION MANAGER:

Jerry M. Okabe

BUSINESS MANAGER:

Ron King

MARKETING MANAGER:

Donald A. Pazour

ADVERTISING REPRESENTATIVES:

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

PRINTING:

Thomas Ogle

EDITORIAL ADVISOR:

Dr. Stephen R. Bourne, Software Engineer, Digital Equipment Corporation

EDITORIAL REVIEW BOARD:

Dr. Greg Chesson, Technical Staff, Silicon Graphics, Inc.
 Larry Crume, Director, AT&T UNIX Systems Far East
 Ted Dolotta, Senior Vice President of Technology, Interactive Systems Corporation
 Gene Dronek, Director of Software, Aim Technology
 George Goble, Systems Engineer, Purdue University
 Bill Joy, Vice President of Research and Development, Sun Microsystems
 John Mashey, Software Engineering Manager, Convergent Technologies
 Robert Mitze, Department Head, UNIX Computing System Development, AT&T Bell Labs
 Deborah Scherrer, Computer Scientist, Mt. Xinu
 Jeff Schriebman, President, UniSoft Systems
 Otis Wilson, Manager, Software Sales and Marketing, AT&T Technology Systems
 Walter Zintz, Executive Director, Uni-Ops

HARDWARE REVIEW BOARD:

Gene Dronek, Director of Software, Aim Technology
 Doug Merritt, Technical Staff, International Technical Seminars, Inc.
 Richard Morin, Consultant, Santa Forda Computer Laboratory
 Mark G. Sobell, Consultant

SOFTWARE REVIEW BOARD:

Ken Arnold, Consultant, UC Berkeley
 Jordan Mattson, Programmer, UC Santa Cruz
 Kirk McKusick, Research Computer Scientist, UC Berkeley
 Doug Merritt, Technical Staff, International Technical Seminars, Inc.
 Mark G. Sobell, Consultant

CONTRIBUTING EDITOR:

Ned Peirce, Systems Analyst, Bell Laboratories

VIEWPOINT

The performance question

How do *you* spell performance? S-p-e-e-d?

That seems to be the attribute most commonly referred to when people speak of how a system delivers. In truth, performance entails a good deal more.

Rather like computing itself, the subtleties of performance are lost on most users. Apart from the vague notion that performance somehow means "bigger" or "faster", many wouldn't recognize the best machines for their needs if they fell on their heads. Little wonder that sexy packaging alone sells so many computers.

Though it's easy to see why people might regard performance as some elusive holy grail, the word *does* have meaning—despite the best efforts of marketers to dilute it. The key to unlocking the performance mystique lies in understanding that there are necessary tradeoffs.

How much do you have to give up in one area to get what you want in another? Do you get enough for the trouble? What are the resources to trade between? How do you know which resource to shore up and which resource to drain from?

Need I say that those questions—among others—are explored in this issue? Rob Warnock, the architect behind Fortune Systems' 32:16 computer, opens with an exploration of what performance entails. Among his topics is an examination of the links between performance and the forgotten art of systems analysis.

The dark art of benchmarking provides the focus of a piece written by Gene Dronek, author of the AIM Benchmark suites. Gene not only tells how UNIX tools can be used to measure performance but how benchmarks can be properly employed to facilitate purchase decisions.

Clem Cole and Ed Breslin of Masscomp follow with an article detailing how users might diagnose problems in systems they already own. Case histories drive the points home.

Since finding problems is only half the battle, John Bass, who helped start up both Onyx Systems and Fortune Systems, tells about how to obtain solutions. Software engineers should be warned, though—John's tuning article has a very distinct hardware orientation.

The software side of the story is told by Roger Sippl, president of Relational Database Systems, Inc. Roger focuses on the tradeoffs integral to optimizing software for the UNIX environment.

Dick Karpinski, the manager of UNIX Services at UC San Francisco, closes the issue with an interview of George Goble, whose ability to squeeze the last precious bit of performance out of a UNIX system has become legendary.

Without further ado, then . . . strap on your seatbelts, performance jocks—we're going for a ride.

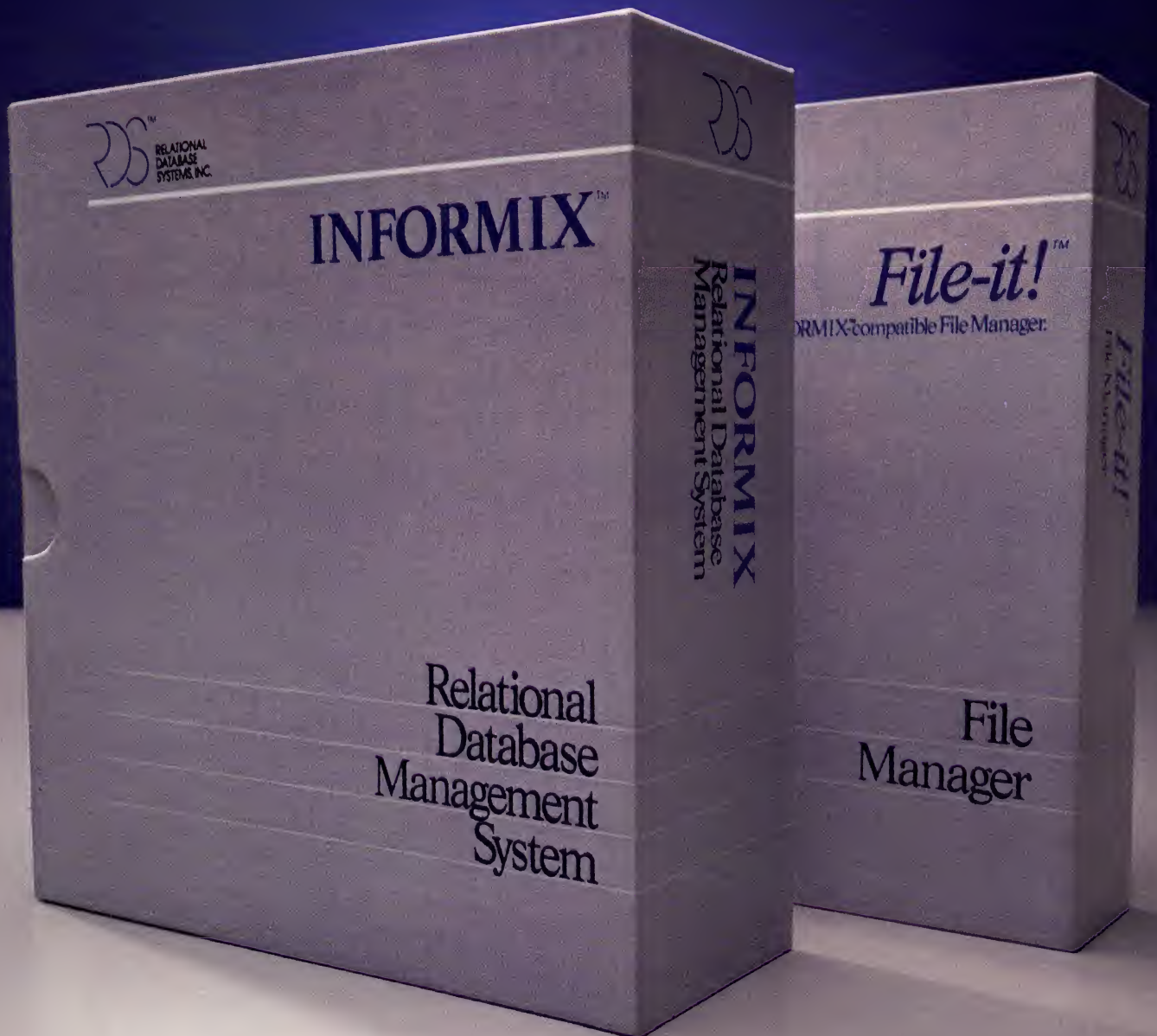
Mark Compton

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

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

WITH INFORMIX AND File-it!

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



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	Ornyx C8002, C8002A
Convergent Technologies	Pacific Micro Systems PM200
Miniframe and Megafame	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.



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.

THE MONTHLY REPORT

System V Interface Definition

by Mark Hall

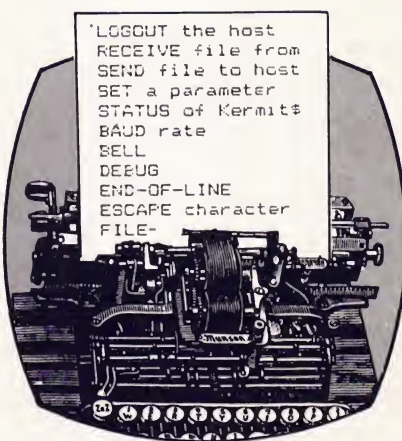
AT&T BREWS A TASTY STANDARD

A number of announcements at the 1985 UniForum Conference in Dallas, TX, underlined AT&T's efforts to gather support for UNIX System V. Before a large audience at a plenary session in the Loew's Anatole Hotel, Bill O'Shea, executive director of the Computer Systems Software Division at Bell Labs, outlined the situation from the perspective of AT&T.

"The (UNIX) business is rapidly maturing. There's growth among VARs (value-added resellers) and explosive growth in ISVs (integrated system vendors)," claimed O'Shea. "An infrastructure is starting to come together around System V and we're trying to make a real standard—not a paper standard."

To buttress this support, AT&T released its *System V Interface Definition*. "Its intent," O'Shea said, "is to help software developers write programs. The document," he said, "is in line with the /usr/group standard."

"In line," of course, doesn't mean an exact replica. But Bob Marsh, founder of /usr/group and chairman of Plexus Computers, noted that the folks at AT&T "have gone a long way to make it compatible." Marsh understands AT&T's approach to be a "superset" of the /usr/group standard.



He said, "Every step they've taken has basically been a good one."

Jeff Schriebman, who was on the /usr/group committee for validation of the System V standard, pointed out that /usr/group standard activity is now in the hands of the Institute of Electrical and Electronic Engineers (IEEE) which hopes to work out the "minor" discrepancies between the AT&T and /usr/group versions.

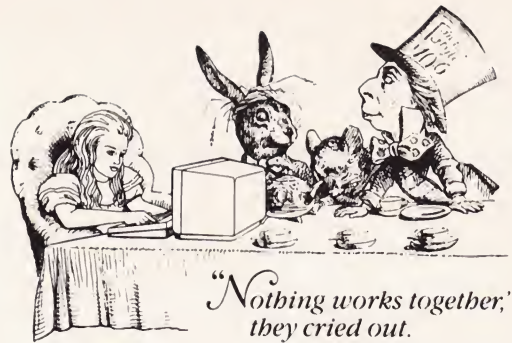
O'Shea admitted "that there are some differences. But there's no question where the issues are and that the differences are minor." One of the stumbling blocks he pointed to was the network interface. "It's clear," he said, "that we need to provide a common networking environment." The goal, O'Shea suggested, "is to provide the capability for any two UNIX systems to talk to each other" as well as to other non-UNIX

systems. He said he believed that there's currently a "fundamental technology missing" on this issue.

O'Shea did not, however, see this obstacle as insurmountable and contended that Dennis Ritchie's paper, "A Stream Input-Output System" (*AT&T Bell Laboratories Technical Journal*, Vol. 63, No. 8 Part 2, October 1984) went a long way toward addressing the conundrum. In his paper, Ritchie observed, "Network connections require protocols more ornate than are easily accommodated in the existing (UNIX) structure."

Ritchie, whose work was originally completed in the Fall of 1983, stated that a modular approach would bring better performance. The use of what he calls *streams* made the introduction of processing modules effective at a Bell Labs test site. His work necessitated rewriting the drivers for terminal and network devices, as well as all protocol handlers, "but only minor changes were required elsewhere in the system." Ritchie modestly concluded, "The improvement in modularity is hard to measure, but seems real."

Although not swearing to the inviolability of *streams* technology, O'Shea said it can be used as "a general basis to do networking work." The AT&T executive considered it critically important to



*"Nothing works together,"
they cried out.*

"It will now," said Alis.

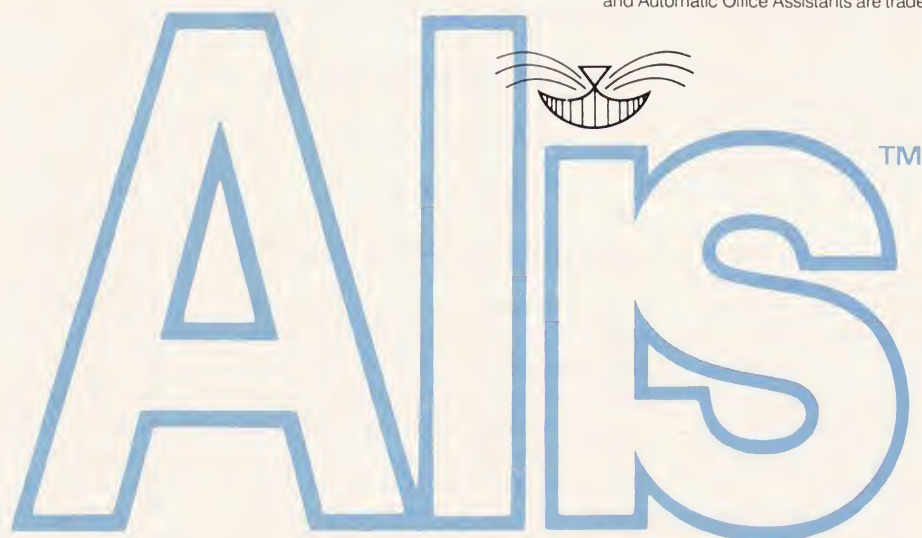
With Alis™ everything works together. Text, spreadsheets, drawings, business graphics and database information work together in a single, always editable document.

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.

 TM

The next-generation office software system from APPLIX

Finally, some answers in Wonderland.

APPLIX, INC., 112 TURNPIKE ROAD, WESTBORO, MASSACHUSETTS 01581 (617) 870-0300

solve the networking puzzle if UNIX is to become commercially viable. Working from the common ground of the *System V Interface Definition* will avoid "ad hoc solutions," O'Shea claimed.

PUTTING TEETH INTO A STANDARD

The release of AT&T's *System V Interface Definition* was in itself a significant move, but the broad range of support System V received at UniForum gave AT&T's pet UNIX version still more clout.

UniSoft Systems of Berkeley announced that it would provide a verification suite that would allow software developers to test their products for compatibility with UNIX System V. Microsoft Corp. informed the industry that it was developing XENIX System V, thus entering into the precarious position of being a key supplier for both AT&T and its rival from Armonk, NY.

Further endorsements for System V came from chip manufacturers. Motorola revealed that it would continue to port System V to future generations of its 68000 microprocessor. Intel said it had ported System V to its 80286 chip. And National Semiconductor likewise announced a System V port to its 32000 series of microprocessors.

Higher up in the hierarchy of system architecture, it was announced that Amdahl Corp.'s UTS UNIX mainframe derivative will support System V. Immediately after that announcement, speculations were raised throughout the Dallas InfoMart as to when AT&T would begin to purchase Amdahl mainframes as an OEM. Such a move, it was reasoned, would make AT&T a complete system vendor, able to offer UNIX from micro-to-mainframe, pitting it across the product gamut with IBM.

THE NON-ANNOUNCEMENTS

While ballyhooing the broad support for System V, AT&T did fail to talk openly about two issues at UniForum. The first unmentionable was the Safari or 7300 microcomputer which is widely rumored to be a UNIX machine loaded with applications software appealing to business users disinclined to the vagaries of UNIX. The 7300, designed and built by AT&T's Santa Clara, CA, ally, Convergent Technologies, is believed to be in final debugging. Sources outside Convergent and AT&T have also suggested that integration of the personal computer into AT&T's network architecture may be causing further delays.

AT&T was equally mum on the demise of its agreement to create a library of System V software with

While ballyhooing the broad support for System V, AT&T did fail to talk openly about two issues at UniForum.

Digital Research Inc. of Pacific Grove, CA. Had everything gone according to plan, DRI would have provided end users with a stamp of approval for programs written to UNIX System V specifications. The software was to be placed on a list of products deemed workable and supportable by DRI.

With the *System V Interface Definition* and its own ISV program already stimulating software development, AT&T contends that the DRI deal was

unnecessary and that it was mutually dissolved. Others have speculated that AT&T's arrangement with DRI was originally a knee-jerk reaction to IBM's relationship with Microsoft. The agreement with DRI, these people conclude, amounted to little more than the addition of a big name software house to the AT&T fold. Besides questioning who finally "pulled the plug" on the deal, some are wondering what capabilities AT&T once believed DRI would bring to its strategy. Perhaps we'll never know.

LOW-END UNIX

Apple Computer, Inc. of Cupertino, CA, has re-christened the Lisa 2/10 as the Macintosh XL. The new high-end Mac micro comes with a 512K RAM expandable to one megabyte, meaning that it has enough memory to run UNIX. And for the last six months, UniPress Software, Inc. of Edison, NJ, has been delivering UNIX System V on Lisa 2s with 10 MB hard disks. Using the UniSoft port of System V, UniPress has brought a single-user configuration with text processing, utilities, and a C compiler to market for under \$1000. According to company spokesman Phil Ruff, a multiuser software package can be purchased for just under \$1500. Added to the \$3495 Mac XL pricetag, users can put together a low-end multiuser UNIX system for less than \$5000. The multiuser software can handle up to eight users, but UniPress recommends only three because of the hardware limitations of the latest addition to Apple's Macintosh product family.

UniPress offers a healthy list of software packages for the Macintosh XL. Along with database programs from RDB and Unify, there is word processing, and the multi-window, multi-screen editor **emacs**. UniPress also has

INTEGRATED OFFICE SOFTWARE

®

XED

Full Screen Interactive • Document Processing • Legal & Accounting • Telephone Mail • Typesetting
Calendar Scheduling • Legal & Accounting Applications • Spreadsheet
Data Base & Forms™ Executive Mail • Telephone Directory

XED®

INTEGRATED OFFICE SOFTWARE

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

compilers for Fortran, Pascal, BASIC, COBOL, and Ada.

The Macintosh XL receives additional UNIX support in the form of a XENIX derivative implemented by the Santa Cruz Operation of

Santa Cruz, CA. With it comes all the utilities of XENIX and one popular applications program, an accounting package from Open Systems, Inc. of Minneapolis.

Elsewhere on the low-cost

front, Hewlett-Packard's has come out with the Integral, a transportable computer with up to one megabyte of RAM running UNIX. The amber-screen unit includes a mouse and a built-in Think Jet printer. Like the Macintosh, the Integral is a closed system; that is, add-on peripherals such as modems and disk drives have to be connected via cable. Without a hard disk, the device costs less than \$5000. Numerous application programs are currently available for the Integral, including Memo Maker and Multiplan. Numerous windows can be sustained on the screen simultaneously, allowing users to move back and forth between operations without losing their place in a program.

Another UNIX-like operating system, VENIX, is available on the Data General One, a briefcase-sized portable computer. The DG One provides a LCD display, weighs less than 10 pounds, and costs under \$3000.

Moving UNIX onto even smaller machines is becoming feasible in light of recent announcements from Atari and Commodore. Atari's new ST line of personal computers, amusingly called Jackintosh, provides up to 512K of RAM and uses the M68000 microprocessor found in the Macintosh XL. The Atari machines, however, retail for less than \$600 (not including disk or monitor). Commodore reportedly is developing a comparable machine, the so-called "Amiga". No one has ventured to put UNIX on the Atari line, but Mark Williams is said to be porting Coherent to the Commodore machine.

Whether "UNIX in the home" is a market worth pursuing is a matter yet to be explored, but soundings are clearly being made.

Mark Hall is the Associate Editor of UNIX REVIEW. ■

Circle No. 295 on Inquiry Card ►

The Software Professionals' Job Fair

SOFTFAIR™

85

SoftFair is different—specifically designed to help you search in new career directions. For the broadest possible view of your local software marketplace, SoftFair brings together technical managers and personnel representatives from 50 or more of each area's most desirable employers. No need for inconvenient phone calls, lengthy cover letters or work-time interviews. Just join us at the site nearest you and see what your experience has earned you at SoftFair.

SoftFair will be held on these days (Monday & Tuesday)
from 4 PM to 9 PM at each location:

Waltham, MA
March 4 & 5
Hillcrest Function Ctr.

San Francisco
Bay Area
March 4 & 5
Sunnyvale Hilton

Dallas, TX
March 11 & 12
Hyatt Regency at
Reunion

Anaheim, CA
March 18 & 19
Sheraton Anaheim

Washington, D.C.
March 18 & 19
Sheraton
Washington (NW)

Northern, NJ
March 18 & 19
Hilton Inn,
Woodcliff Lake

Exciting career opportunities are available for experienced software professionals in a variety of **Software Engineering, Scientific and Business Applications** areas related to **development, support, implementation, training, quality assurance, technical writing, sales, marketing and management.** A minimum of two years' on-the-job experience is required.

If you can't attend SoftFair, send a resume indicating the location in which you are interested for distribution to attending companies. Software Career Link, Dept. UR 3, 67 South Bedford Street, Suite 400W, Burlington, MA 01803. (617)229-5813.

SoftFair '85 is sponsored by Software Career Link
serving equal opportunity employers.

Circle No. 296 on Inquiry Card

HCR/PASCAL, WIRTH ITS WEIGHT IN C

PASCAL Originally designed by Niklaus Wirth, is now available for a wide range of UNIX™ processors. HCR/PASCAL conforms closely to industry standards, passes all conformance tests in the PASCAL Validation Suite. Supports multiple module programs, a dynamic string package, and direct random file access.

C is the standard language of UNIX, HCR/PASCAL is written in C and translates PASCAL into C producing efficient optimized code. This approach allows direct interaction with the UNIX environment and offers a high degree of portability.

UNIX is a powerful yet flexible operating system environment. HCR/PASCAL is available today on a diverse range of UNIX hardware: AT&T 3B™ series, the NCR Tower,™ DEC PDP-11/VAX,™ and others. HCR has a growing line of UNIX software including business applications. We back up all our software with full support. To find out how we can put HCR/PASCAL, C, and UNIX together for you, call or write:

Human
Computing
Resources
Corporation



10 St. Mary Street, Toronto, Ontario, Canada M4Y 1P9 (416) 922-1937

THE HUMAN FACTOR

Random thoughts on UNIX system performance

by Richard Morin

System performance is generally considered to be a very serious topic. Conferences are held that focus solely on the measurement of it. System tuning is regarded as an even more arcane art. Pooh.

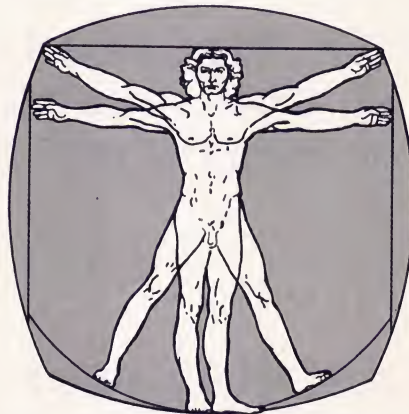
Some aspects of system performance are, indeed, rather mystical. Others, however, remain pretty simple. Let's talk about these for a while.

FIRST, MEMORY

Lack of sufficient memory is probably the most common UNIX system bottleneck. UNIX is a memory hog, requiring anywhere from one to several megabytes. Insufficient memory will cause the system to swap processes and/or pages.

This produces lots of expensive disk I/O. The process involved is affected directly, since it can't run until it is in memory. The entire system is also affected by the effort expended in doing the I/O. This is not to say that you should simply install as much memory as possible. A given computer may not need more memory. When a machine bogs down, however, the amount of available memory should be examined first.

Even when memory is available, it is not always wisely used. UNIX systems that rely on swapping are forced to copy entire processes in and out of memo-



ry. Operating systems, such as 4.2BSD, which move only the needed pages, use memory more efficiently.

The same copy of a program can serve several users simultaneously. Only the data areas required by each copy must be duplicated. This reduces the amount of memory required by each user.

Similarly, a program can be used sequentially by several users. UNIX provides a "sticky bit" that makes programs stick around in memory. Programs that are invoked frequently are good candidates for this.

NEXT, DISKS

Disk space is generally a scarce resource, and the system won't perform well if no free disk storage is available. There's an unwritten law of computing that dic-

tates that when you install a new disk, it will fill up before the warranty card has been sent in. Kirk McKusick of UC Berkeley recently had a 400 MB student disk fill up in three days.

Still, when a user community can be persuaded to police disk usage, the situation usually stays under control. Some core files disappear. The remaining files usually have some reason for their existence.

Persuading users can be difficult, however. Charging them isn't very effective, unless the users are directly responsible for paying the bills. Disk quotas ("Fascism comes to UNIX") implemented under 4.2BSD, seem to help somewhat.

Having said all that, there are occasional reasons to "waste" disk space. Keeping UNIX reference materials online, for example, results in less contention for physical manuals. If you do this, it makes sense to use **catman** to run some of the materials through **nroff**, keeping the resulting files around. (This speeds up the **man** command considerably.)

Other space/time tradeoffs may show up in your users' applications. Inverted indices take up room but allow fast searches. The main thing is to maintain a balance between the cost of disk storage and the response time of your system.

MODULAR. INTEGRATED. NOW.

Handle™ presents
a modular software
series for office
automation.

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

Handle Office Automation Software Series product modules include:

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



AT&T 3B2/300 & 3B5



CONVERGENT TECHNOLOGIES
MINIFRAME



DURANGO POPPY II



To be
announced?

HANDLE
TECHNOLOGIES, INC.

850 NORTH LAKE BOULEVARD / PO BOX 1913 / TAHOE CITY, CALIFORNIA 95730 / 916-581-5227

TM— HANDLE, HANDLE WRITER/SPELL, HANDLE CALC, HANDLE GRAPH & HANDLE LIST ARE TRADEMARKS OF HANDLE TECHNOLOGIES, INC. UNIX IS A TRADEMARK OF AT&T BELL LABORATORIES. XENIX IS A TRADEMARK OF MICROSOFT CORPORATION. MINIFRAME IS A TRADEMARK OF CONVERGENT TECHNOLOGIES. POPPY IS A TRADEMARK OF DURANGO.

Disk activity should also be balanced, since a single disk drive can only work so fast. If all of your disk activity is occurring on a single drive, consider moving some files or directories to other disks.

Finally, a note on disk layout maintenance: an empty file system has one big pool of free space. Files written on it are allocated nearly optimally. All of their blocks lie close to each other, allowing efficient access.

Now the disk is used for a while. Files are written and deleted, written and deleted, and the free space becomes increasingly fragmented and disordered. New files are scattered across the disk, causing the drive to waste a great deal of time.

Running **fsck -S**, if your system supports it, will reorder the free

When a user community can be persuaded to police disk usage, the situation usually stays under control.

list. This will allow the system to allocate new files more efficiently.

Existing files will remain scattered on the disk, however. This

can be solved by performing a complete **dump** and **restore** of your user file systems. Besides, an occasional backup is a good idea....

FINALLY, TERMINALS

Another bottleneck has to do with terminal I/O on timeshared systems. Many systems use terminal interface boards that interrupt the processor for every character handled. Well, terminals are slow and processors are fast, so no problem. Right?

Wrong. Let's assume we have a number of 9600 baud terminals on a system. Each terminal can accept a maximum output rate of around 1000 characters per second. Now assume that the servicing of an interrupt takes 100 mi-

SYSTEM V

TRAINING

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.



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

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

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

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

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

Available now. For additional information, call or write,

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



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

Circle No. 293 on Inquiry Card

croseconds, or 1/10,000th of a second.

Let any 10 terminals ask to type at the same time. The terminal interrupts alone will use up all of the CPU time. Many systems have far more than 10 ports in use. It is quite possible some mid-afternoon blahs result from this problem.

A cheap and dirty solution is to crank down the baud rate. At 1200 baud, it takes eight times as many terminals to produce the same number of interrupts.

A better solution is to switch to Direct Memory Access (DMA) interfaces. A DMA interface is told by the processor where to find the output data. It then goes after the data directly, interrupting the processor only when it is done. Since the data could be a number of lines of text, this cuts down the number of interrupts by orders of magnitude.

Finally, there is an approach taken by Plexus. Give all the terminal handling duties to a separate processor. This gets rid of low-level interrupt handling, as well as a great deal of other I/O processing.

IS IT CONSISTENT?

It might seem from the foregoing discussion that speed of execution is all-important. Peculiarly enough, it isn't. Studies show that people are much more sensitive to consistency of response time than to speed.

If it always takes five minutes to compile a particular piece of code, the user will adjust to it. If, on the other hand, a compile can take anywhere from one to 10 minutes, the user will get very frustrated and upset.

Workstations are comfortable in part because one knows how long they will take to do something. The typical timeshared computer, by way of contrast, is utterly unpredictable. One knows

that response time will degrade as more users try to work, but the degree of degradation remains unknown

Unfortunately, there is very little that can be done about this. An

A cheap and dirty solution to the mid-afternoon blahs on timesharing systems is to crank down the baud rate.

attempt to normalize response time by consistently slowing the system would not be well accepted by the user community. Keeping large numbers of people off the system at peak demand times would also cause howls.

About the most that can be done is to even out the demand by assorted administration policies. Prime time can carry a higher cost than other periods. Batch jobs can be relegated to off hours or run at low priority.

IT'S NICE TO BE NICE

Sometimes a few monster programs bring a system to its knees. This can be helped in a number of ways. System hogs can often be optimized to use less resources. Special purpose hardware can be purchased to make them run faster.

Additionally, the impact of these programs can be reduced by adjusting their runtime priorities. While **cs**h automatically lowers the priority of programs run in

background, **sh** does not. Neither shell does anything to the priority of programs run in foreground.

Large background **sh** programs and all large foreground programs should therefore be run under the **nice** command. System managers who don't trust their users to do this may wish to run **renice** on any program that has been running for, say, a minute.

Programs that don't need to run immediately should be scheduled to run later, by use of the **at** command. Service bureaus have been running large programs at night for years. UNIX users should consider doing likewise.

Remember, adjusting priorities does not affect the total throughput of the system. It merely improves the response time of the higher priority programs.

RESPONSE TIME vs. THROUGHPUT

It is important to distinguish between response time and throughput. Response time is a measure of how long the system will take to handle a particular request. Throughput is a measure of how much work the system can perform in a given period of time.

In a batch environment, throughput is the key measure. In an interactive environment, response time is critical. Most systems fall somewhere in between.

Pick your organization strategies to concentrate on the measure that is most important in your environment. No strategy is perfect, but if you understand your goals, you stand a far better chance of achieving them.

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



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

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

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



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

much disk space as they need in 10MB increments.

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

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

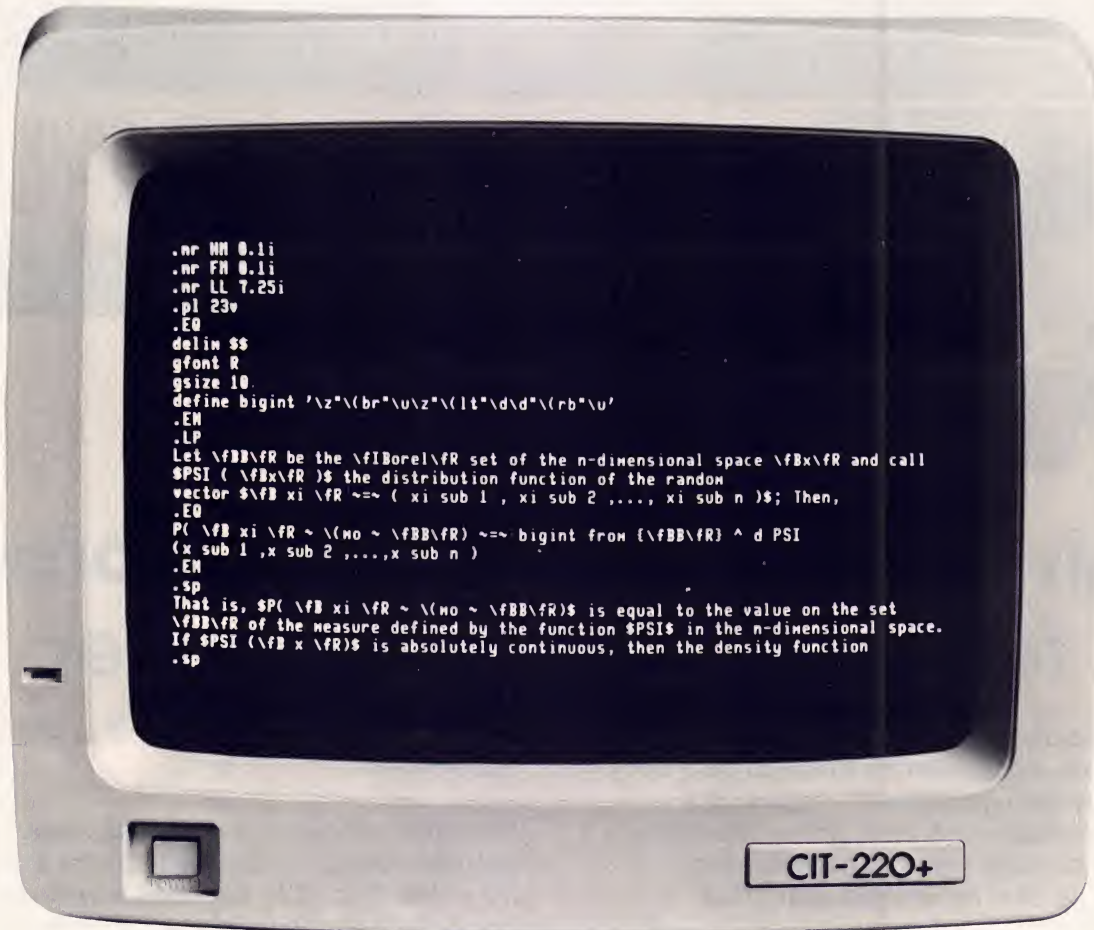


Absolute Computers, Inc.

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

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

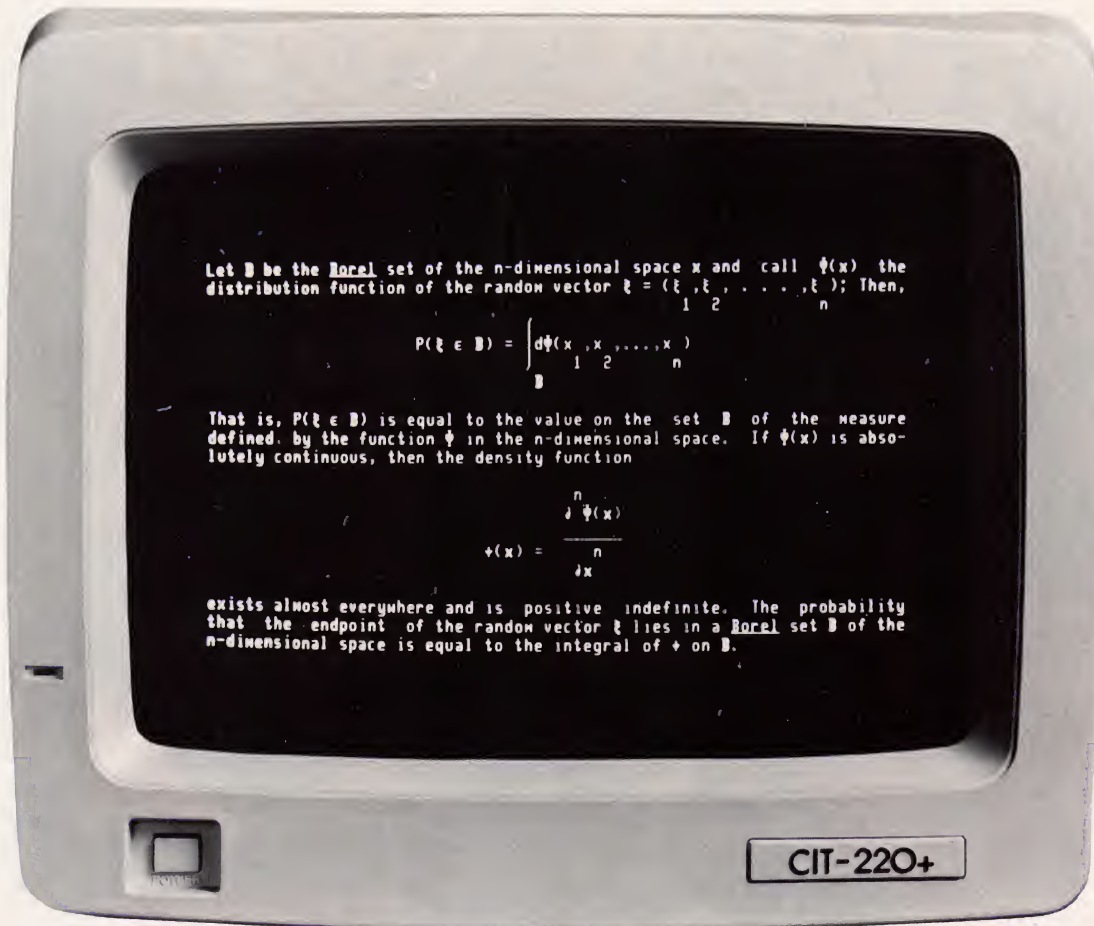
NROFF and TROFF users:



What will this look like?

A BREAKTHROUGH IN UNIX SOFTWARE FOR NROFF AND TROFF DISPLAY

PreView™ shows it for less!



**NOW
COMPATIBLE
WITH**

DEC VT-100
DEC VT-220
DEC VT-240
CIT-101e
CIT-220
Wyse-WY-85

Check with
your local
representative
for an updated
list of
compatible
terminals.

Are you still struggling with your NROFF and TROFF documents without PreView™?

PreView™ provides the finest in highly enhanced previewing of formatted NROFF and TROFF documents on a variety of terminals. Special features of NROFF and TROFF are displayed quickly and easily before you request a printout.

Complete NROFF and TROFF character sets are displayed,

including mathematical symbols, the greek alphabet and many other special characters.

Equations and tables can now be previewed with **eqn** and **tbl** preprocessors.

Documents can be previewed on CIE Terminals' CIT-220+ and CIT-101e with

the peripheral systems upgrade. PreView™ also supports any terminal with downloadable character set capability as well as many low-cost hard copy devices.

Available for UNIX Version 7, BSD 4.1 and BSD 4.2 and XENIX 68000 or 80X86. All PreView™ software is distributed as C language source code. A complete installation script, manual and users guide is included with each package.

All software and manual updates are provided to you **free** for one year from the date of purchase. You may purchase additional support on an annual basis from Peripheral Systems, Inc.

Circle No. 291 on Inquiry Card

PreView™ is a trademark of Affine Sciences, Inc.
UNIX is a registered trademark of AT&T Bell Laboratories, Inc.
XENIX is a trademark of Microsoft, Inc.

YES, I'd like to find out more about the PreView™ package

Name _____

Company _____

Address _____

City _____

State _____

Zip _____

Telephone _____

Type of computer and operating system _____

PSI

Clip coupon and mail to:
Peripheral Systems, Inc.
8107 Orion Avenue
Van Nuys, CA 91406
For immediate information
call (818) 902-0791



47.89

GOING FOR THE GOLD

A look at the meaning of "performance"

by Rob Warnock

When buying computers—or using them—one of the magic qualities we can never seem to get enough of and yet pay dearly for is "performance". Why is it that we are so often dissatisfied with the performance of our computer systems? In large part, it may be because we don't actually know what "performance" means, and therefore inadvertently confuse it with other attributes of a system.

In light of this problem, let's define the term before proceeding any further. The following definition, while taken from the field of human factors, is equally applicable to computer systems (with a few anthropomorphisms), and will form the basis for our study:

Performance is defined as the result of a pattern of actions carried out to satisfy an objective according to some standard. The actions may include observable behavior or nonobservable intellectual [or electronic] processing.

Robert W. Bailey, *Human Performance Engineering: A Guide for Systems Designers*, (Prentice-Hall, 1982) page 4.

Immediately, we can see some important things about this definition. First, performance is always defined in terms of a goal or objective. If you don't know where you are going, you'll never know when you get there.

Second, the goals or objectives (and therefore, performance) are dependent on the context in which the system (electronic or human) is expected to perform. In light of that and to narrow the focus, the rest of this article will concentrate on the small business or office automation environments, though a similar ap-

proach might also be taken in the software development or scientific "number-crunching" cases.)

As an example of contextual performance, even a person who has won an Olympic gold medal in the decathlon and has placed first in that event's 100 meter dash has no guarantees of "superior performance" when running the dash against someone who competes exclusively in sprints. Conversely, the sprinting specialist would probably perform poorly in the decathlon. (C programmers may substitute **sprintf(3)** for "sprint".)

As you can see from this example, performance is tied to *results* rather than *actions* (behavior). This distinction is important because, as we will see later, actions and results often cannot be compared directly. Although some action is usually needed to achieve an objective, the actions are not the result itself.

However, inappropriate actions may impair performance. Coaches will sometimes notice behaviors or actions that may appear quite energetic but actually bring negative results. One example might be swinging one's arms wide of the body while running. Seemingly minor adjustments in such behavior can result in significant performance improvements. Similarly, computer systems often perform unnecessary actions, such as recalculations of complex expressions or rereads of disk blocks where the values of expressions or data blocks have not changed since they were last used.

BEFORE THE EVENT: SETTING THE STANDARDS

The most difficult task in selecting or designing a computer system is deciding exactly what

you expect it to do. This requires looking carefully and critically at your total business process (both manual and automated functions). This is formally known as *systems requirements analysis*.

Distributing computers to departmental, work cluster, or individual users has brought numerous advantages in terms of local autonomy, decreased bureaucracy, predictable availability, and cost. But far too often, the positive qualities of centralized systems have been lost in this distribution. In particular, one can no longer assume the availability of systems analysts and programmer analysts able to perform requirements analysis for each new user application. Maintaining or hiring a professional analyst is seen today as too expensive for smaller installations. Conversely, managers who wish to perform this function for themselves have been faced with a scarcity of training resources or tutorial texts.

Purchasers of small business computers have thus tried to get by without a detailed analysis of their environments. In some cases, they've even been successful. When a personal computer is viewed as an appliance—like one's telephone or typewriter—and is more than equal to the tasks it's asked to perform, it can successfully be employed without any serious attention to cost and performance.

When a larger work group or departmental system is needed, the issues of systems analysis cannot be safely ignored, though most organizations still do so. To quote Bailey again:

Unfortunately, in some systems the designers allow standards to simply



evolve Under these conditions, there is no way to determine if performance is acceptable Often it is mistakenly assumed that designers, users, and user management all have a common standard or expectation and that any deviation from this "common standard" will be quickly recognized and corrected. This naive approach frequently results in considerable disappointment with human [and computer] performance levels of new systems. [Ibid, page 5]

Nothing can remove the ultimate responsibility from user management to see that system requirements are properly documented and agreed upon by all parties. Fortunately, a few sources of help are beginning to appear, such as Paul T. Ward's book, *Systems Development Without Pain: A Users's Guide to Modeling Organizational Patterns* (Yourdon Press, 1984), and (for programmers) *Structured Analysis and System Specification* by Tom DeMarco (Prentice-Hall, 1979).

THE WIDE WORLD OF SPURTS

Having analyzed what you want the computer to do, you will then have to define a reference workload that can be used as the basis for performance comparisons. Typically, this will consist of scenarios (or scripts) of the various tasks your target computer system is expected to complete, and will include the standards of performance you require in the completion of those tasks. There are two main dimensions that are important to measure: *responsiveness* and *throughput*.

Responsiveness measures the

ability of a system to respond to a given request in a given amount of time. It is inversely related to *response time* (or *latency*), measured as the period from when a

Computer systems often perform unnecessary actions.

request is entered until the response is completed. Even in a single-user environment, the responsiveness requirement will vary from application to application, and will even vary within a given application. Bailey states:

Users seem willing to wait varying amounts of time for different types of requests. The amount of time a user is willing to wait appears to be a function of the perceived complexity of the request and the time the request is made—people will wait longer for "hard" requests or those made at the end of a series of commands, or "closure point", than during the interaction. [op. cit., page 320]

Bailey goes on to say that psychological *continuity* of a give-and-take exchange is essential. Pauses longer than a second or two may give the appearance of terminating the interaction. If the computer is processing each character as it is typed, even short pauses of echoing or word wrapping may cause the human operator to "break stride". Here are some of the maximum acceptable response times recommended by Bailey [ibid, page 322]: key de-

pression to echo, 200 milliseconds; input to new menu, 500 milliseconds; simple inquiries, about two seconds; complex queries, four seconds; and executions of a new program, 10 seconds. Note that these are recommended *maximums*.

Even more distressing to users is *variability* of response time. Sometimes it may even be useful to enforce a *minimum* response time if that decreases the variance.

Throughput measures the total amount of work performed over a given (usually fairly long) period of time. It is often measured by executing a given set of tasks, and is inversely related to the time spent from the first request of the first task to the last response of the last task. Multiuser systems can have many tasks going at once, which can improve the throughput if tasks are able to overlap their idle periods. Unfortunately, configuring a system for maximum throughput also tends to increase both response time and the variance of response time. So it is important in establishing performance standards to make sure that your system meets its throughput goals *without* violating response time constraints.

GOING THROUGH THE MOTIONS

As if maintaining a throughput and response time balance wasn't difficult enough, it is seldom possible to test a prospective vendor's system in advance for response time and throughput under normal working conditions, (unless you happen to be buying very large systems or very large *numbers* of systems). Such tests are lengthy, costly to design and run, and difficult to judge (accurate measurements require external test-driver computers). Most users therefore settle for us-

Continued to Page 94

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”



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



THE STANDARDS OF SYSTEM EFFICIENCY

Benchmark hurdles to cross

by Gene Dronek

During a chat over lunch recently, several computer performance jocks debated the question of when micro UNIX machines would routinely top the "MIPS" (million instructions per second) barrier. The general flavor of the discussion was: "Only a few machines, manufactured by the likes of Cray and Amdahl, have ever run faster than the 1.0 MIPS level. How can micros come close?"

One of the debaters bravely offered that the workstation his company manufactures had already broken the MIPS barrier—

and, what's more, that he had actually measured it running 1000 iterations of an assembly language loop containing 1000 no-op instructions. The machine, he boasted, had chomped through in less than a second every time.

The argument, though, was far from over. A no-op isn't *really* an instruction, the others at the table shouted. They spent much of the rest of the meal chewing on the notion of just what is "real". Branches? Nope. Increment instructions? No. Tests? No. Eventually, having had their fill,

they agreed to let the matter pass. Clearly, they weren't going to agree that day to any single way of developing MIPS ratings. And, besides, it was time to order dessert.

Measures of performance on UNIX systems have become a popular topic elsewhere, too [1,3]. It's not surprising. Starting with simple benchmarks and advancing to complex terminal emulators, systems programmers have been lured into wondering just how well their systems are performing.

Information isn't hard to find.



Illustration by Ivy Nichols

Casual estimates of performance on any running UNIX system can be obtained as easily as entering the **time** command. Furthermore, it is extremely easy to add software counters to system activities if you have source. And **cron** can be made to run background monitors that will capture periodic performance statistics. No wonder UNIX performance is studied so much!

UNIX BUILT-IN PERFORMANCE ASSESSMENT TOOLS

Before we touch on other kinds of measurement software, let's look at the standard commands built into UNIX. Besides the shell **time** command, many UNIX systems include a few commands for displaying system-level statistics. If you haven't already memorized the */etc* directory, a brief overview of the most popular tools should be interesting. The formats vary from UNIX version to UNIX version, so formats often vary from release to release.

*The **time** command.* The **time** command is the best known and most widely used performance tool, and is available on every UNIX system. Although most UNIX clocks are "noisy" for many reasons, (see the sidebar on **time**) you can still get good results with this helpful command.

*The **iostat** command.* Though it goes by different names on different UNIX versions, **iostat** is a fascinating command to watch. Version 7 and System III know it as **iostat**, but on System V, look under **sar**, and on Berkeley versions, look for **vmstat**. To use the command, start it up in background with a short interval like five or 10 seconds, and watch its output for a while. You will still be able to enter commands and yet observe the resource utilizations on subsequent **iostat** lines. Useful as the information is, **iostat** is also relentless, and will continue to report **tty** chars, disk blocks, and CPU utilization line after line until you enter a **kill 0**.

*The **accton** command.* Most

UNIX systems will keep process-by-process statistics if you turn accountability on. The file generated by **accton** grows quickly, and its description is lost deep in Section 4 of the *UNIX User's Manual*, but System V gives you a few tools to automatically boil out summaries. The most popular commands used on a system can be gleaned from this accounting, and the per-process data can be useful to characterize a system's workload.

*The **profile** command.* For C programmers, **profile** reveals where the program counter (**pc**) spends its time in any module. Users get an execution "profile" representative of time spent, thus allowing important, expensive inner loops to be found and optimized. The Berkeley variant, **gprof**, accumulates statistics recursively for all functions called within a function, and produces more graceful output.

*The **perfmon** command.* Finally, for those with Sun systems, a package called **perfmon** provides the ultimate in real-time re-



source displays. It can be quite a treat to watch **perfmon**'s bit-graphic icons trace out load information (essentially like a graphic **iostat**) in real time. The only catch is that the package is not portable, and is essentially limited to Sun systems.

MODELS FOR PERFORMANCE

A not-well-informed modeling critic once said that UNIX is merely a development environment short on performance and bloated with attention. Modeling performance on a non-performing system, he argued, was nothing more than a sterile, overdone academic exercise. Not true. First of all, he was wrong about UNIX performance. What's more, he failed to see that "models" are actually a very convenient way of identifying and talking about the assumptions built into performance testing. It is sometimes useful to organize performance methods by the complexity of the operating software. Figure 1 illustrates this point. You cannot describe performance very accurately without first entering into system workload modeling. The simplest performance measurements entail models—even single-task/single-user benchmarks.

Benchmarks, the backbone of traditional performance measuring methods, have implicit built-

in models. Hopefully, since benchmarks are by definition easy to get, you should be able to gather a dozen or so running programs that can collectively approximate your current computing needs. Portability problems,

Multiuser performance is slightly more complex to determine than is multitasking performance.

sheer volume of code, and long running times, however, may suggest that just the innermost loops—the "kernel" code if you will—be used to make handier and speedier benchmarks. In either case, not a great deal of effort need go into constructing formal performance models. As a result, it is often difficult to convincingly project benchmark timings for different loads.

Synthetic benchmarks that are specifically constructed for timing but still rely on only single-task/single-user tests have more

explicit workload models. Some synthetic programs, like the well-worn Whetstone test, have locked, unchangeable workload models, while others, such as Dhrystone[4] or Qbench[2], have parameterized models. It is more convincing to project the latter two synthetic results to different loads.

MULTITASK AND MULTIUSER PERFORMANCE

Performance models for single-process computing do not need to be complex, and therefore are not usually given much study. However, multitask and multiuser software is more complex to program, debug, and run efficiently, and thus is also more complex to measure. As an example, consider two similar systems that were measured for one to 15 users. The two systems, both from the same vendor, used identical CPU chips; only the bus structures differed. The idealized performance "scores" that were obtained are charted in Figure 2. Machine A cost somewhat more than machine B (\$10,000 versus \$6000), yet if you look only at single-user performance (vertical line 1), machine A seems to perform slower than machine B. This is very misleading, because when loaded to 10 users (see line 2), machine A can clearly out-perform the other. The vendor said A's performance advantage was due entirely to its more sophisticated (and more expensive) bus structure, which was designed especially for handling multiuser loads.

Apart from subtle differences of the sort illustrated by Figure 2, multitasking software and multiuser packages are likely to be module-synchronous, making overall performance difficult to model. Not surprisingly, the way in which multiple processes *communicate* greatly affects overall package "performance". Clearly,

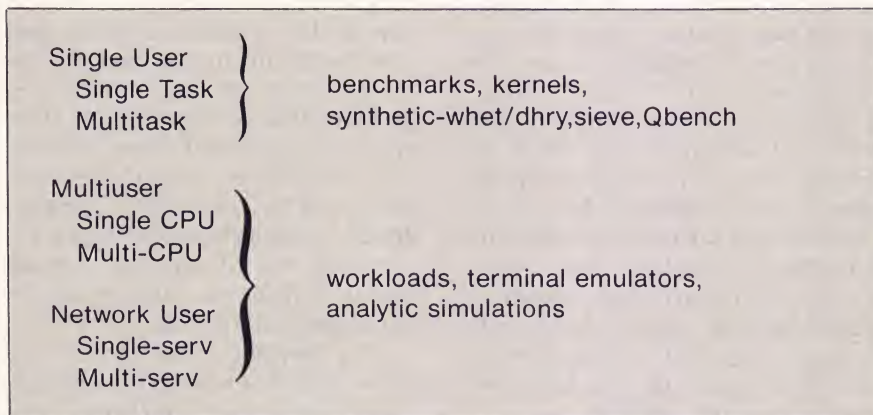


Figure 1 — Categories of performance measurement techniques.

it would be nice if benchmark suites had a few simple components that could predict multitasking performance, but unfortunately, machine load characteristics would have to be known to do so.

Response time is becoming another dominant UNIX performance issue as more customers demand the sort of consistent sub-second response they can generally get on PCs. This is more than a marketing issue, since the use of full-screen editors like **vi** can be downright dangerous if system response lags over a half-second from keystroke to keystroke.

Measuring response time can be tough since, in the hierarchy of system construction, multiuser performance is slightly more complex to determine than is multitasking performance—for at least two reasons. Users must be kept separate and protected in a multiuser environment and this shows up as increased context switching overhead. Users, moreover, generally output to CRT screens instead of files, thus incurring additional I/O overhead. UNIX systems are notorious for outrageous terminal driver overhead costs.

Standard benchmarks are often re-run with one or two **tty** ports running in background in order to determine terminal overhead. This is important since support for a single remote 1200 baud link can eat up 20 to 30 percent of a typical 68000-based or 286-based system. (Incidentally, this method, coined "displacement" benchmarking by John Bass, was first described publicly at the UNIX Systems Expo/84 Conference in Los Angeles. The displacement method allows the resource requirements of arbitrary applications to be deduced.)

Although desirable to know, response time is probably the single

most difficult indicator to measure. The only practical way to measure multiuser keyboard response is to use a second computer running a terminal emulator package. Terminal emulation requires fairly strong interest in

Generally speaking,
performance is *caused*
by components, but
limited by their
connections.

evaluation and seems to appeal to manufacturers more than consumers. Emulators normally submit script files and commands that are chosen to model workload, and then continuously time-

stamp every transaction. After testing is done, the time stamps are analyzed and response times are calculated.

Somewhere within AT&T exists an emulator program called *Quartz* that can emulate one to 16 ports on a 3B2, but it is not available as a commercial package.

FINDING BOTTLENECKS

Why do systems have bottlenecks and how do you find them? What causes "performance" and what limits it? Obviously, performance relates at least in part to the hardware. In terms of graph-theory, cut-sets, and some special hardware instrumentation, there is an algorithmic way to locate hardware bottlenecks, given that one knows the underlying design. But the operating system adds a layer of complexity before the end task can run. Software systems, including UNIX, simulate "components" for users, and communicate data in and out of those

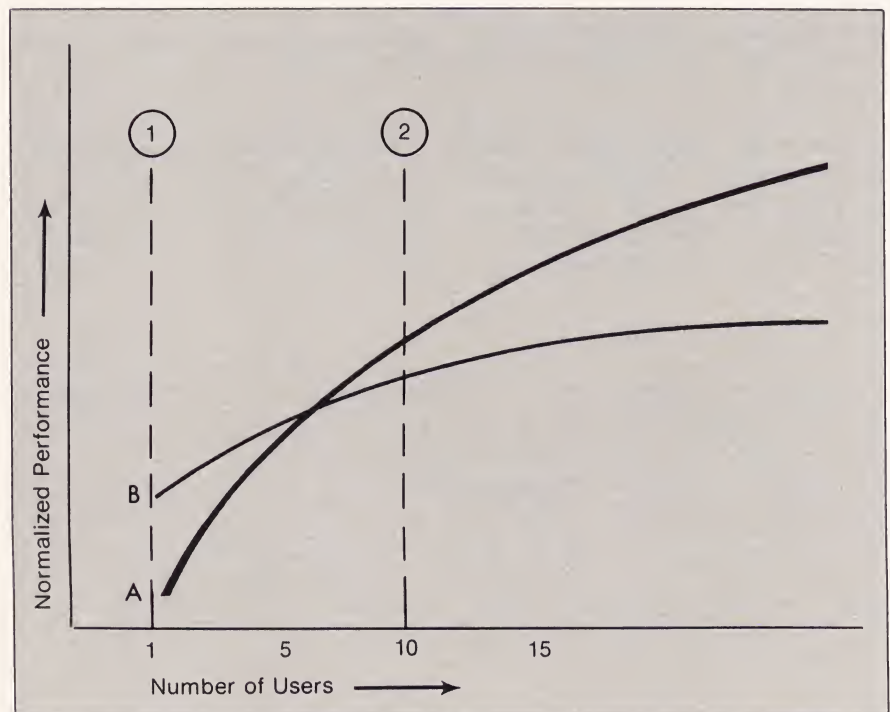


Figure 2 — Performance of two similar systems.



TIME IS ON YOUR SIDE

Historically, UNIX systems have simulated wall clock time by counting ticks from a 60 Hz powerline interrupt. When you run a **time** command, the tick count is first noted, and a subshell is then forked to run the command. When the command completes and the waiting shell is awakened, the new tick count is noted, and the difference (divided by 60) is reported externally. While simple and inexpensive to implement, these "tick" clocks are reliable only over long intervals of time. In many places, UNIX device drivers and other functions mask timer interrupts and thereby lose (or at least delay) occasional "clock" ticks. One UNIX kernel I've seen had the following comment line:

```
/* here we add 470 microseconds
to correct the clock because this
we empirically measured to be the
context switching overhead last
week */
```

Incidentally, **fork** time, shell time, and **exec** time are included in **time** reports. All this being considered, **time** commands require upwards of one second of

CPU time to provide fairly repeatable data. If your system has it, also consider using **iostat** for a second opinion. Several vendors have ported their UNIX to run on 50 and 100 Hz clock interrupts, to the dismay and confusion of benchmarkers everywhere. It is not always easy to be sure of what runs inside a given UNIX system, so we offer the HZ command below:

```
/* program to report HZ resolution */
long tb[4],times(),t1,ticks;
main()
{
    t1 = times(tb);
    sleep( 10 );
    ticks = (times(tb)-t1) / 10;
    ticks = (ticks+5)/10 * 10;
    printf("%ld\n",ticks);
}
efd
cc HZ.c;
a.out
rm HZ.c a.out;
```

HZ will deduce and print the clock resolution (50, 60, etc.) if run on System III or V. It will print 0 if run on Version 7 or Berkeley UNIX. G.D.

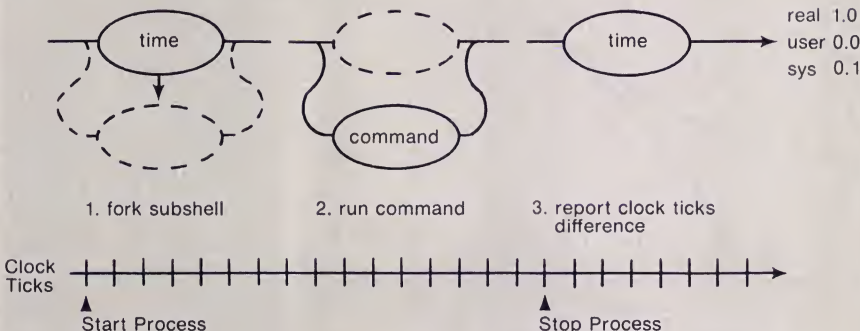
"components" in a manner analogous to hardware "connections". As a result, overall UNIX performance is caused (and limited, too) by the combination of underlying hardware and software components. Generally speaking, performance is *caused* by components, but *limited* by their connections.

Using single-task, self-measuring benchmarks, you can determine single-task component speeds—but never multitasking capacities. Not enough information is considered by such benchmarks. System topology must also be included to determine overall capacity. This is why model-free performance measurements are of limited use.

REFERENCES

- [1] CMG XV Conference Proceedings, Computer Measurement Group, P.O. Box 26063, Phoenix, AZ 85068.
- [2] "Relating Benchmarks to Performance Projections", E. F. Dronek, Summer 1984 Conference Proceedings, Usenix Association, P.O. Box 7, El Cerrito, CA 94350.
- [3] "Monitoring System and Process Performance", W. J. Meyers. "Interpreting UNIX Benchmarks", J. Saxer, Winter 1985, Conference Proceedings, Usenix Association.
- [4] "Dhrystone: A Synthetic Systems Programming Benchmark", R. P. Weicker, Communications of the ACM, October 1984, Vol 27, Number 10.

Gene Dronek, author of the AIM Benchmark standard performance suites, is Director of Software at Aim Technology. He worked in User Services at the UC Berkeley Computer Center during the early UNIX development years and has explored computer performance since 1969. Mr. Dronek now consults widely on system tuning. ■



The **time** sequence.

UNIX & CTM

HANDS-ON SEMINARS

A Complete Curriculum for: End Users • Management • Applications Staff • Technical Support

COURSES	LONDON	NEW YORK	BOSTON	WASHINGTON, D.C.	SAN FRANCISCO	CHICAGO	DALLAS	LOS ANGELES	TUITION	SEQUENCE TUITION†
UNIX Overview	Apr 9 June 11 Aug 6 Oct 8 Dec 3	Jan 15 May 28 Oct 1	Mar 19 July 30 Dec 17	Jan 15 Mar 19 May 28 July 30 Oct 1 Dec 17	Feb 19 June 18 Oct 15	Mar 12 Aug 20	Feb 19 June 18 Oct 15	Mar 12 Aug 20	\$225	\$860
UNIX Fundamentals for Non-Programmers*	Apr 10-12 June 12-14 Aug 7-9 Oct 9-11 Dec 4-6	Jan 16-18 May 29-31 Oct 2-4	Mar 20-22 July 31- Aug 2 Dec 18-20	Jan 16-18 Mar 20-22 May 29-31 July 31- Aug 2 Oct 2-4 Dec 18-20	Feb 20-22 June 19-21 Oct 16-18	Mar 13-15 Aug 21-23	Feb 20-22 June 19-21 Oct 16-18	Mar 13-15 Aug 21-23	\$735	
UNIX Fundamentals for Programmers*	Apr 15-17 June 17-19 Aug 12-14 Oct 14-16 Dec 9-11	Jan 21-23 June 3-5 Oct 7-9	Mar 25-27 Aug 5-7	Jan 21-23 Mar 25-27 June 3-5 Aug 5-7 Oct 7-9	Feb 25-27 June 24-26 Oct 28-30	Mar 18-20 Aug 26-28	Jan 14-16 Feb 25-27 June 24-26 Oct 28-30	Mar 18-20 Aug 26-28	\$735	\$1125
Shell as a Command Language*	Apr 18-19 June 20-21 Aug 15-16 Oct 17-18 Dec 12-13	Jan 24-25 June 6-7 Oct 10-11	Mar 28-29 Aug 8-9	Jan 24-25 Mar 28-29 June 6-7 Aug 8-9 Oct 10-11	Feb 28- Mar 1 June 27-28 Oct 31- Nov 1	Mar 21-22 Aug 29-30	Jan 17-18 Feb 28- Mar 1 June 27-28 Oct 31- Nov 1	Mar 21-22 Aug 29-30	\$490	
'C' Language Programming*	Apr 22-26 June 24-28 Aug 19-23 Oct 21-25 Dec 16-20	Jan 28- Feb 1 June 10-14 Oct 21-25	Apr 15-19 Aug 12-16	Jan 28- Feb 1 Apr 15-19 June 10-14 Aug 12-16 Oct 21-25	Mar 4-8 July 8-12 Nov 11-15	Apr 29- May 3 Sep 9-13	Jan 21-25 Mar 4-8 July 8-12 Nov 11-15	Apr 29- May 3 Sep 9-13	\$1225	
Shell Programming*	Apr 29-30 July 1-2 Sept 2-3 Oct 28-29	Feb 4-5 June 17-18 Oct 28-29	Apr 22-23 Aug 19-20	Feb 4-5 Apr 22-23 June 17-18 Aug 19-20 Oct 28-29	Mar 11-12 July 15-16 Nov 18-19	May 6-7 Sep 16-17	Jan 28-29 Mar 11-12 July 15-16 Nov 18-19	May 6-7 Sep 16-17	\$490	\$1125
Using Advanced UNIX Commands*	May 1-3 July 3-5 Sept 4-6 Oct 30-Nov 1	Feb 6-8 June 19-21 Oct 30- Nov 1	Apr 24-26 Aug 21-23	Feb 6-8 Apr 24-26 June 19-21 Aug 21-23 Oct 30-Nov 1	Mar 13-15 July 17-19 Nov 20-22	May 8-10 Sep 18-20	Jan 30- Feb 1 Mar 13-15 July 17-19 Nov 20-22	May 8-10 Sep 18-20	\$735	
UNIX Internals	May 7-10 July 8-12 Sept 4-6 Oct 30-Nov 1	Feb 11-15 June 24-28 Nov 11-15	Apr 29- May 3 Aug 26-30	Feb 11-15 Apr 29-May 3 June 24-28 Aug 26-30 Nov 11-15	Mar 18-22 July 22-26 Dec 2-6	May 13-17 Sep 23-27	Feb 4-8 Mar 18-22 July 22-26 Dec 2-6	May 13-17 Sep 23-27	\$1375	
UNIX Administration*	May 15-17 July 15-17 Sept 18-20 Nov 11-13	Feb 19-21 July 9-11 Nov 19-21	May 7-9 Sep 10-12	Feb 19-21 May 7-9 July 9-11 Sep 10-12 Nov 19-21	Mar 26-28 July 30- Aug 1 Dec 10-12	May 21-23 Oct 1-3	Feb 12-14 Mar 26-28 July 30- Aug 1 Dec 10-12	May 21-23 Oct 1-3	\$735	
Advanced 'C' Programming Workshop*	May 20-21 July 22-23 Sept 23-24 Nov 18-19	Feb 25-26 July 15-16 Dec 2-3	May 13-14 Sep 16-17	Feb 25-26 May 13-14 July 15-16 Sep 16-17 Dec 2-3	Apr 15-16 Aug 5-6 Dec 16-17	Jan 14-15 June 3-4 Oct 7-8	Apr 15-16 Aug 5-6 Dec 16-17	Jan 14-15 June 3-4 Oct 7-8	\$490	\$1125
Advanced 'C' Programming Under UNIX*	May 22-24 July 24-26 Sept 25-27 Nov 20-22	Feb 27- Mar 1 July 17-19 Dec 4-6	May 15-17 Sep 18-20	Feb 27-Mar 1 May 15-17 July 17-19 Sep 18-20 Dec 4-6	Apr 17-19 Aug 7-9 Dec 18-20	Jan 16-18 June 5-7 Oct 9-11	Apr 17-19 Aug 7-9 Dec 18-20	Jan 16-18 June 5-7 Oct 9-11	\$735	
Berkeley Fundamentals and 'csh' Shell*	June 3-7 July 29-Aug 2 Sept 30-Oct 4 Nov 25-29	Mar 4-8 July 22-26 Dec 9-13	May 20-24 Sep 23-27	Mar 4-8 May 20-24 July 22-26 Sep 23-27 Dec 9-13	Apr 22-26 Aug 12-16	Jan 28- Feb 1 June 10-14 Oct 21-25	Apr 22-26 Aug 12-16	Jan 28- Feb 1 June 10-14 Oct 21-25	\$1225	

*Including hands-on training workshops

TMUNIX is a trademark of Bell Laboratories

†Savings for consecutive seminar dates

CALL FOR DETAILS ON: ON-SITE SEMINARS • VIDEO-BASED TRAINING • INTERACTIVE VIDEODISC TRAINING
To reserve your seminar space now or for additional information, call: (800) 323-UNIX or in Illinois (312) 987-4082

MANY UNIX-BASED SYSTEMS—ONE UNIX TRAINING COMPANY

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 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.

COMPUTER TECHNOLOGY GROUP

Telemedia, Inc.

310 S. Michigan Ave., Chicago, IL 60604

Is your UNIX performance the best you can get? Though daunting at first glance, it's a question that *can* be answered. This article will help you assess what your system currently offers and will suggest steps for netting improvements. It contains some general considerations followed by case studies that illustrate problems, the way UNIX works, the logic behind UNIX troubleshooting, and some of the solutions that already have been obtained. What this article cannot do is analyze *your* need for performance tuning. This assessment is every bit as important as the answers to the questions that follow.

ANALYZE YOUR APPLICATION. First, analyze your application-dependent program. You have a lot of control over the program—you may even have written it! Exercise this control if the program runs slowly, or you will necessarily limit the value of any other tuning.

Some programs run slowly because of a number of small-scale errors in programming judgment. Others, though, are bogged down by major performance-consuming errors. Typically, these fundamental mistakes are based on failures to understand an application, a program that implements the application, or a combination of the two.

Fundamental errors of logic defy the benefits of better hard-

ware. An infamous quote reminds us that "... a CRAY 1 can execute an infinite loop in six seconds while a VAX will take six days." Though this is a joke, the point is simple: an infinite loop is wasteful of resources. Make sure your program does not waste *your* resources.

CHOOSE THE BEST FEATURES UNIX HAS TO OFFER. Second, make sure your program employs the best alternative UNIX provides. For example, a pipe is not the only means of passing information between two processes in UNIX. Piping is convenient and can pass unlimited amounts of information, but it is also too slow to be an efficient technique in real time. Shared memory is thus a better option for real time applications.

CONSIDER THE HARDWARE. Third, buy the hardware that best serves your needs. If you are using floating point calculations, for example, software floating point performance is never going to be as fast as hardware floating point performance. If performance is your goal, and performance demands a specific hardware feature, buy it!

ANALYZE THE OPERATING SYSTEM. Fourth, analyze the performance of the operating system to learn where time is being spent. The measurement programs that UNIX provides are usually known as **ps(1)** (process

status), **pstat(1)** (process statistics), **iostat(1)** (input/output statistics), **vmstat(1)** (virtual memory statistics), and **netstat(1)** (network statistics). Each provides statistical information you'll need for your investigation of the operating system. Of course, different implementations of UNIX may have different names for these or similar programs.

From **ps(1)**, read:

- How many processes are running at the same time?
- How many processes are waiting?
- How much memory do the programs consume?

From **vmstat(1)**, read:

- How much memory is in use by the system?
- How many page faults are occurring?
- How many interrupts are being sent to the CPU?
- How much low-speed terminal I/O is happening?
- How many processes are waiting?
- How many processes are active?

From **iostat(1)**, read:

- How many interrupts are there per process?
- What resources are used by each process?

ZEROING IN ON PROBLEMS

The first phase of system tuning

by Clement T. Cole and Ed Breslin



H.KIM 03



- What is in the file table?
- What is in the inode table?

From **netstat(1)**, read:

- How many network interrupts?
- How many connections are in use?
- How much memory is being used by the network?
- How many packets are being lost?
- How many packets are bad due to collisions?

Always check the number of interrupts the CPU is processing, the amount of memory different programs or parts of the system are using, the amount of memory generally available for user programs, and the amount of memory available for I/O buffers, terminal buffers, and swapping. Your analysis should tell you where the performance-limiting gates are in your system.

TUNING PARAMETERS

With this information, you can then tune the system's parameters to allow you to push through impediments. One of the most important tunable parameters in UNIX is the buffer structure. How many I/O buffers do you need? I/O buffers are typically either 512, 1024, 2048, or 4096 bytes in size, depending on the I/O subsystem. Some of the most difficult I/O challenges are offered by the 16-bit PDP-11, in which the problem of a very small 64K byte memory space is exacerbated by the need to use memory to address each I/O buffer.

The optimum solution is to restrict the buffers to the smallest number possible without degrading the performance of the machine. To determine the number, look at **ps(1)**, **vmstat(1)**, and **io-stat(1)**. How much time does a process spend waiting for the

disk? How many seeks (read/write head movements) are necessary? If there are many processes waiting and very few seeks occurring, you'll know that you should reconsider the number of I/O buffers.

A classic case that we encountered here at Masscomp involved

Fundamental errors of logic defy the benefits of better hardware.

a system configured with 3 MB of memory (much of it free), a fast SMD-style disk controller, and a fast disk. Despite all that hardware, though, the seek activity of the read/write head was surprisingly minimal. Something was obviously limiting the flow of data to disk: the machine was I/O-bound. Common sense suggested checking the I/O buffers.

By doubling the buffer space to ten 4096-byte I/O buffers, the amount of data queued up and available for the disk controllers increased to over 40K bytes. That would be too large on a PDP-11, but fine for the MC-500 or the VAX. In this case, it succeeded in giving the disk scheduler the necessary resources to minimize the read/write head's seek movement. For disk seek algorithms to improve the efficiency and order of head motion, I/O buffers must provide a flow of data large enough to allow schedules to be made for future head movements. Limited I/O buffers restrict seek scheduling to one read/write at a time.

By increasing the number of I/O buffers, we bought I/O speed at the cost of memory available for programs. Thus, a plentiful re-

source was converted to supplement a scarce resource.

Two other important tunable parameters are open file limits and caches for keeping frequently used file addresses in memory. To determine the fixed number of open files available for the system to use, review the parameters known as *files* and *inodes*. *Files* represents the systemwide number of simultaneously open files. *Inodes* are the systemwide resource that describe each file currently being manipulated in the system. Each of these data structures is a table. Every time a UNIX application performs an **open** system call, a spot is carved out in the **open** file table and a pointer is made to an inode, indicating that I/O should be performed on that file.

More than one process can have the same file open. Every process has separate file table entries—like multiple carbon copies. UNIX also has a table entry for each separate file that is open (or named via system calls, like the "get file" statistics call, **stat(2)**). The table known as the inode table remains open whether or not it is currently in use by one or more processes.

If you make the *files* or *inode* parameter too large, it will allocate a large amount of memory to create each table and will define a large number of addresses in which to place files. The larger the number of table entries, the slower each search of the table becomes. Program memory address space, moreover, is consumed. If you try to improve performance by making the parameter very small, though, you run the risk of generating errors as the system tries to work with limited resources.

The C-list is another important tunable parameter. As a data structure that represents the number of buffers currently avail-

able in the terminal character I/O pool, each C-list represents approximately 16 bytes of data. UNIX does not have dedicated I/O queues for each terminal line. But it does have a common memory pool for immediate allocation. If there is a great deal of terminal traffic, you will end up using a lot of C-lists. If there isn't much terminal traffic, you will use very few. The tradeoff is speed vs. memory space.

IF ANALYSIS POINTS OUT A PROBLEM. FIX IT! Once you have done an analysis that has uncovered the limiting gates to your system's performance, change them. Many times the tuning of parameters will net you a satisfactory solution. On some other occasions, though, it becomes clear that nothing short of a hardware upgrade will do. Once you've drawn this conclusion, you must act on it.

By all means, add memory if memory is in short supply for active processes. Add fast disk controllers and disks if your system shows itself to be disk-bound or I/O-bound. But if your system is compute-bound, you may wish to stay with a slower, less expensive disk controller and instead spend your money on something that helps the CPU, such as a floating point accelerator.

CASE HISTORIES

The following case histories illuminate the problems of performance tuning. They are based on difficulties already encountered in the UNIX community and reflect problems that performance tuners may do battle with again.

Case History #1: *The case of the terminal connection interrupt.*

Symptoms: After an office reorganization, a system suddenly grew sluggish in a multiuser environment. When **ps(1)** was run, it

showed a rapid increase in process numbers.

Discussion: As it turns out, there was one RS-232 line that had been connected to a machine but had not yet been connected to a terminal. When hooking a terminal up, it's best to connect all the

If performance is your goal, and performance demands a specific hardware feature, buy it!

flow control (modem) lines. Connecting a cable to the system but not to a terminal at the other end creates an antenna that may pick up noise on the RS-232 modem control lines.

Most UNIX terminal controllers, like the DEC DH-11, are set up to give an interrupt whenever a transition, such as *carrier detect*, occurs on one of the RS-232 lines. If the *carrier detect* line is asserted, the host is interrupted and responds by creating a process and sending out a login prompt.

If the line is deasserted, the host is sent another interrupt that it responds to by deleting the active process. Noise on the line can appear as assertions and deassertions to the system. UNIX unsurprisingly responds by creating and deleting processes. That is why process ID numbers grew at such a fast rate in this case when the cable was not terminated.

Solution: Find out where the bad RS-232 line is. Tell UNIX to ignore the incoming interrupts from the

line—or better yet, put a terminal on the end of the line to stop the floating connections.

Case History #2: *The case of the **sendmail** blahs.*

Symptoms: A system that had been running acceptably suddenly suffered a performance drop. Just as suddenly, normal performance resumed later.

Discussion: This sort of problem can present a confusing time lag since no difficulties are immediately evident in the programs in process. Nevertheless, in this case, as soon as the **sendmail** process was started, a check of **vmstat(1)** and the number of system calls revealed that system calls were suddenly incrementing at a very rapid rate. That in itself raised suspicions since system calls are expensive in terms of computer resources and represent a threat to a sustained standard of performance. It was discovered that a subroutine of the **sendmail** daemon process was writing a log of system events whenever system mail was delivered. On investigation, it was found that it was opening a system log, doing a short write, and then closing the log every time any type of I/O was done on normal **sendmail** I/O channels.

Solution: The log portion of **sendmail** was changed so it would open the system logfile only once during the life of a process, use only buffered I/O for the writes, and close the log on process termination. After the fix, the system call overhead was reduced by a factor of 10.

Note that the main program itself was not the cause of the problem. Rather, it was the subroutine implementing the system logging function that needed change. Total system performance was



bound in this instance by a single program. This should serve as evidence that you need to tune an entire program—not just its parts.

Case History #3: *The case of the whole being different from the parts.*

Symptoms: A machine showed an unexpected increase in performance just after the installation of a rewritten C compiler. Previously, the compiler had run as separate programs connected by UNIX pipes. As rewritten, two of the separate programs were combined.

Discussion: The portable C compiler from AT&T that originally was used had been written to run on a PDP-11 with only 64K bytes of memory address space. To keep within these constraints, it had run as a set of smaller programs to avoid overflowing memory space. It became possible to ignore some of these memory size restrictions when new computers with increased memory address space arrived. So two of the compiler's programs, the Semantic Analyzer and the Code Generator, were combined and the pipe between them was eliminated.

The first pass of the compiler was a C pre-processor that could be called separately. The second pass was a Semantic Analyzer that could take syntactically correct C and produce syntax trees, a data structure that described the program symbolically. Then the Code Generator produced code that could be assembled by the UNIX assembler and linked to a working program. It is important to note that syntax trees were originally passed between the Semantic Analyzer and Code Generator by using writes and reads across a pipe containing binary information. By rewriting these

portions of the C compiler, information could be passed in memory instead.

Solution: Passing information in memory, as a result of combining the Semantic Analyzer and Code Generator, eliminated a pipe and substituted a shared memory data transfer that was much faster. Note that the elimination of a single pipe led to a substantial performance increase. The pipes.

Some programs run slowly because of a number of small-scale errors in programming judgment.

which had been used to get around the problem of the PDP-11's small address space, had limited the program's performance.

Case History #4: *The case of the unbalanced memory.*

Symptoms: On investigating the ailments of a limping PDP-11/70, it was discovered that lots of processes were being swapped out while they waited for memory to become available.

Discussion: At first glance, it looked like a swap-bound system that could be cured by the simple addition of memory. But there is a point at which the addition of memory fails to provide a cure for the swap-bound malaise.

In a non-virtual, swapping system such as the PDP-11, swapping can occur even when there is available memory. This is particu-

larly true when memory size exceeds 2.5 MB.

In this particular instance, the addition of memory actually made the performance problem worse. Instead of speeding up, the 11/70 slowed and the time for in-core process expansion, the memory-to-memory copy procedure, increased dramatically. The machine had moved from a swap-bound condition to a memory-bound one. Why? Because the CPU had to do expansion by copying programs from one part of memory to another. Previously, the disk controller had been used to copy processes by forcing them to be swapped in and out of memory.

The increase in CPU activity led to an examination of the disk controller. Remember, before adding memory, it had been the disk controller rather than the CPU that had moved data in memory. The memory shortage that had triggered the action of the controller had also offloaded the processing burden from the CPU. That sped up the system's throughput. But when memory was added to the system, the disk controller no longer went to the aid of the CPU, forcing the CPU to do the copy itself! The moral of this story? More memory may not necessarily mean more performance.

In virtual systems, such as the Masscomp MC-500 or the DEC VAX, processes can grow rapidly on machines that manage 2 to 8 MB of memory. Despite the luxury of having plenty of available memory, such systems can thus exhibit inexplicably sluggish performance. On single controllers, moreover, interrupts can show sudden, dramatic increases.

A check of **ps(1)** and **vmstat(1)** on such a system will probably indicate that the wait is for disk I/O. If so, the throughput of the disk

Continued to Page 96

IBM's[®] Unix-based[™] system of the future is the CIES system of today.



IBM knew that someday even small businesses would catch on to the fact that PCs just aren't enough. Too little power. Too little memory.

IBM's solution: a UNIX-based multi-user system. Someday.

It's the same solution CIE Systems came up with over two years ago. And it's a solution we've been perfecting ever since.

The new CIES 680/100 and CIES 680/200 are not in the future. Not something you need now, but can't get.

They're here. Today.

They're here with multi-user expandability of from one to 40 users, doing different jobs simultaneously.

They're here with up to 2 Megabytes of memory.

They're here with up to 300-plus Megabytes of hard disk, along with a floppy disk drive and a streamer tape drive for backup.

They're here with all the power. All the memory. All the customer support. All the ready-to-run applications a business needs, including a complete general accounting program, word processing, electronic spreadsheet, even a complete medical practice program for physicians.

Not someday. Today. From CIE Systems, the ever-growing giant in super-micros backed by C. Itoh & Company Ltd., the fifth largest corporation in the world with over \$60 billion in annual sales.

For more information on the system you need today, not someday, just call or write.

CIE Systems, Inc., 2515 McCabe Way, Irvine, CA 92713-6579. (714) 660-1800. Call toll free 1-800-854-5959. In California, phone 1-800-432-3687.

[™] CIES 680 is a Trademark of CIE Systems, Inc.

UNIX is a Trademark of Bell Laboratories

[®] IBM is a Registered Trademark of International Business Machines Corp.

© 1984 CIE Systems, Inc.



A C. ITOH ELECTRONICS COMPANY

CIES SYSTEMS SALES OFFICES

Southwest
2515 McCabe Way
Irvine, CA 92713
(714) 660-1800

Northeast
Executive Mews, M64,
1930 East Marlton Pike
Cherry Hill, NJ 08003
(609) 424-6925

North Central
1 Cross Roads of Commerce
Suite 305
Rolling Meadows, IL 60008
(312) 392-1331

South Central
17311 Dallas Parkway
Suite 230
Dallas, TX 75248
(214) 248-8355

Southeast
4501 Circle 75 Parkway
Suite 1190 A
Atlanta, GA 30339
(404) 953-1876

New England
400 Amherst St.
Space #41
Nashua, NH 03063
(603) 881-7031

Northwest
2700 Augustine Dr.
Suite 238
Santa Clara, CA 92504
(408) 748-0452

England
Beacon House
26/28 Worple Road
Wimbledon, SW19, 4EE
01-946-4960

CIE Systems
a giant step ahead..



Most of the computer performance enhancements of the last 30 years have been directed at improving system throughput (measured in units per time interval, such as: *jobs per hour*). Thus, people have come to perceive "performance" as simply a synonym for throughput in batch systems. Another measure of performance that has since arisen is "load capacity", which is just another assessment of throughput for multitasking computer systems.

Enter the interactive timesharing systems of the 1980s—dedicated microcomputer systems, office automation software, and other interactive applications like CAD/CAM graphics. With them has come the need for a new criteria of performance—one that can be related to the end user's personal experiences.

Enter the era of response time measurement. The response time measure of performance is non-uniformly applied to such aspects as keystroke echo times, command completion times, command start-up times, and any other activities that force the user to wait in front of a CRT.

The perception of "slowness" in the user's mind often stems from erratic response times. This is manifested in several ways, the most common of which is keystroke response. If "normal" keystroke responses are sufficiently fast to be subjectively instantaneous, then other system activities that produce noticeably long responses—such as swapping and paging—seem abnormally slow. Note that the

unqualified claims that system A is great and operates *X* times faster than system B. To get to the bottom of such claims, you must have an understanding of the claimant's perspective. Be especially nervous when someone claims that something is faster or better; ask them to define their terms and give examples. If you get only a glowing one-sided report—ask someone else.

The basic truth is that not all computers are alike. Neither are they equally suitable for every job. Nor does everyone judge them in the same way. Furthermore, since requirements change, we need to view performance evaluation in terms of a system life cycle, rather than from a short term perspective.

This evaluation is broken into three major phases:

- 1) Estimating your requirements and choosing a system accordingly.

- 2) Getting the most out of what you finally purchase.

- 3) Comparing what you've bought to your future expansion needs.

In evaluating systems, be careful of religious zealots who only evaluate the relative merits of a computer from a dogmatic viewpoint, ("XX is such a dog. It can't do ZZ. I don't know why anybody would ever want it.") Every system has its strong points and its weak points. There are always going to be tasks for which it is best suited, and tasks for which it should never be used.

Probably the single most misunderstood and mis-

Getting back to basics

SYSTEM TUNING TRADITION

by John Bass

user learns to *expect* immediate response as the norm. Also, the user notices and remembers slow responses even when the system may be "normally" fast. It has been observed that users of heavily loaded systems with uniformly slow response soon come to accept *that* as normal. These users also generally grow unaware of periodic faster service.

EXPERIENCE FROM BYGONE DAYS

In the old days, a system that did *X* times more jobs per day than another computer was considered a faster and better system. Today's "response time" criteria, though, is overwhelmingly subjective and not significantly demonstrable. It doesn't matter to the end user that the system can do *X*, *Y*, and *Z* ten times faster than an old system when *Q* runs slow in relation to the user's past experience or current need. As a result, the marketplace is flooded with

represented feature of the UNIX system is the swapping vs. paging issue. "Swapping"—as used in V6, V7, System III, and System V—refers to the act of moving a user's entire program and data in and out of memory to make room for some other user's program and data. This is done via large I/O chunks on most UNIX systems to avoid "swap" operations that require no more than a "few" I/O transactions. Jobs may be loaded into the memory as one contiguous chunk of memory or may be "scatter loaded" in a number of smaller segments.

Swapping is done on systems with base and displacement MMUs like those found on PDP-11s and many micros like the Tandy 16B and the Fortune 32:16. Swapping can also be used with paging MMUs on systems like the VAX or Onyx C8002, using either scatter loading or contiguous allocation. Another common MMU design is based on "buddy allocation" of the sort found on the Motorola M68000 MMU. This offers an efficient way to man-



age memory using variable sized pages or segments that increment in size by a power of 2 (i.e. 1K, 2K, 4K, 8K . . . 256K, and so forth).

“Demand paging” (often called simply “paging”) allows jobs to be broken up into small chunks known as “pages”. Paging is found in 4.1BSD and 4.2BSD-based systems, and soon will be found in certain new AT&T releases. Though page sizes are often fixed within a single system, their lengths can

In the old days, a system that did X times more jobs per day than another computer was considered a faster and better system.

vary from 512 bytes to 4K bytes. Many 4.XBSD-derived systems use a 1K page size. With paging, jobs live on the disk paging file and then are brought into memory on “demand”, a page at a time, as the memory is accessed. When a page hasn’t been used for “awhile” and some other process needs the memory, the page is written back to disk. Thus, demand paging may allow more efficient memory usage simply by requiring the active parts of a program to reside in memory (the working set) only during program execution. This also allows a demand-paged system to run programs that are much larger than the real memory of the machine.

Depending on a system’s configuration and application: 1) paging may be the only choice; 2) the differences may be insignificant; 3) paging may be much better; or 4) swapping may be much better. This depends on how much memory the system has, how big the program is, what the rest of the system job mix is, and the relative cost of paging a program in and out vs. swapping it in and out. Let’s look at some examples.

Configuration One—Paging is the only choice: This system probably runs certain CAD/CAM, graphics, LISP, or other programs that need a larger virtual address space (500K byte, 1 MB, or more) than is available in real memory. Back in 1978-79, applications like these for ARPA could be run on machines of the day that were cheap and available. That led to ARPA’s funding of a Berkeley effort in 1979-80 to complete the UNIX 32V paging experiments for the 4.1BSD product. It was tuned and

optimized to support the ARPA VISION and LISP projects. Additional funds were made available so that 4.2BSD could be specifically enhanced for those applications.

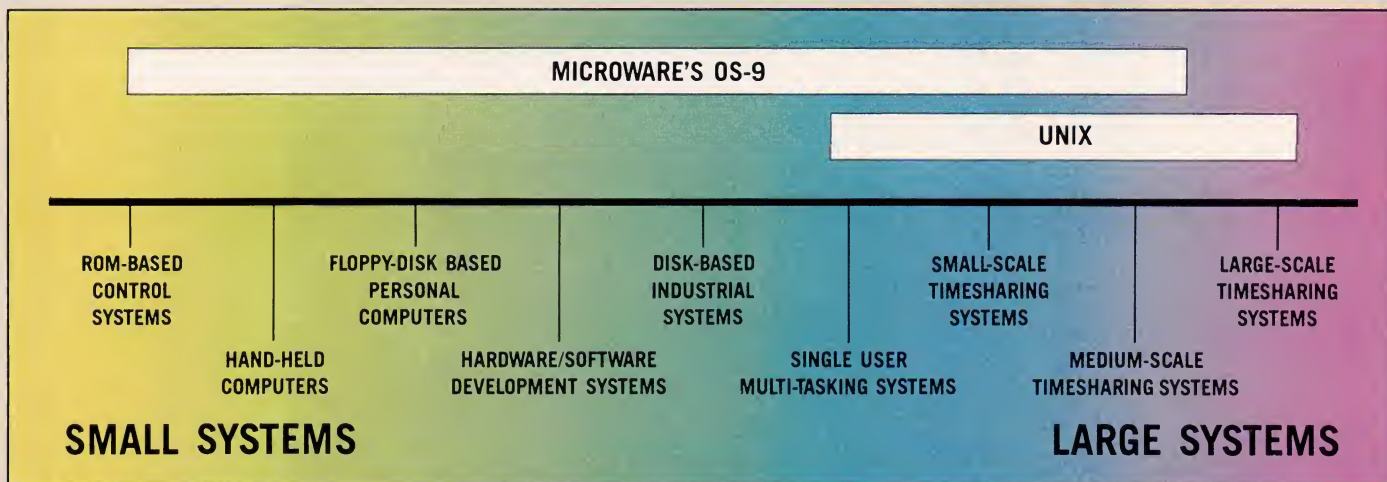
Configuration Two—The differences are insignificant: This system has memory available that is roughly equal in size to all of the jobs needing to run concurrently. Job mixes on such a machine can include a few 100K byte to 500K byte programs and a larger number of 10K byte to 50K byte jobs. On the small end of the scale, one might find such standard UNIX utilities as **vi**, **sed**, and **make**. The disk channel on such a machine has a burst rate fast enough to swap the largest job in a few hundred milliseconds or so. The disk device has a very fast seek rate (less than 30 ms) and the disk queue is always short enough to provide service in a few hundred milliseconds. Such a configuration allows both demand paging and swapping to provide reasonable response times.

Configuration Three—Paging is much better: This system doesn’t have enough memory to run several large interactive jobs at once. The swap in/out times of larger programs will make overall interactive performance on such a system irregular with response times fluctuating in the one to three second range. This was a problem that occurred when several users attempted to run **vi** on PDP-11/45s, but was overcome when machines like the 11/44 and the 11/70 offering memory sizes larger than 256K bytes were released. Irregular response time problems can also occur on 256/512K byte micros that attempt to run several large spreadsheet programs concurrently. Since the same micros often have very slow disks, a memory size of 768K byte/1 MB or more is often recommended—regardless of paging or swapping. The most notable applications of this class are found in research and engineering environments that run program development concurrent with numerous large jobs (ranging from 300K byte to 1 MB on machines offering two to four MB of RAM).

Configuration Four—Swapping is much better: This system mainly handles processes in the 10K byte to 150K byte range, along with a few larger programs—generally leaving enough real memory for large jobs and small jobs to run concurrently. Disk channels in machines of this sort are always under heavy demand (i.e. file system throughput becomes a significant factor in response time), and queue service times are usually relatively long (i.e. long disk queues and/or very slow seek times on the disk device will result). This system will also be cost sensitive. Small RAM sizes are required as well as

Continued to Page 100

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.

MAXIMUM UTILIZATION



Shirley Sereque/K&S Services

An interview with George Goble

If anyone has ever made a career of improving computer system performance, it's George Goble. As the Systems Engineer of Purdue University's Electrical Engineering Department, he has long wrestled with limited resources and seemingly unlimited demands. Like other systems engineers in academic environments, Goble has learned how to stretch meager funds to obtain larger, faster machines. But more importantly, he has also displayed an uncanny knack for getting the most out of what he already has.

In tandem with Mike Marsh,

George Goble became the first to tie two VAXen together. DEC followed suit six months later with its dual processor 782. Meanwhile, stories about the phenomenally heavy use of PDP-11s and VAXen streamlined by Goble have become legendary.

To probe the mind of this resourceful man, UNIX REVIEW asked Dick Karpinski, the manager of UNIX Services at the University of California at San Francisco, to pose some open-ended questions. Goble's responses follow.



REVIEW: *What does performance mean?*

GOBLE: That's like defining the meaning of life.

REVIEW: *One description suggests that performance means getting the most out of what you've got.*

GOBLE: That's as good as any. The time you're going to spend has to be considered also. If you can spend two days speeding something up and get a 50 percent improvement for your efforts, it's worth it. But you can't spend six months getting a one percent improvement out of a machine. You have to draw the line somewhere, but different people have different thresholds. Berkeley takes it right down to the final degree. When Bill Joy was there he could just look at code and benchmark it optically. That's why 4.2BSD is so fast; he went through and spent four or five months (which would have been man-years of effort by anybody else). Berkeley was just incredible at this stuff and it still is. Mike Karels and Kirk McKusick and the people there do good work. I don't even profile the kernel anymore, I just call up Berkeley and talk to the guys there a couple times a week because I figure they've probably already profiled it.

REVIEW: *So your view of how to identify problems in UNIX code is to call up Berkeley and talk to the folks who have already profiled it?*

GOBLE: I just chat with them every now and then and these things come up. They keep digging up these little five and 10 percent boosts so it's not even worth it for me to mess with it.

We've got a dual-processor Gould machine installed here. It's best for me to spend my time working on that. Some people running single-user processes have found that each CPU runs 10 times faster than a VAX 780. Why should I spend my time working to get five percent performance improvements on the VAX?

Gould gave us this thing for free and it still has some rough edges. We are the first 4.2 site for it. Gould worked a year porting it, but we're essentially back to where 4.2 was when it first came out. We've got our work cut out for us.

REVIEW: *At Purdue, you started out with PDP-11s, didn't you?*

GOBLE: Right, 11/45s and 11/70s—then VAXen, dual VAXen and now the Gould.

REVIEW: *And the dual VAX was your own invention?*

GOBLE: Right, Mike Marsh and I put them together and made it work.

REVIEW: *Was that before DEC did it?*

GOBLE: Yeah, we got ours out about six or eight months before DEC had the 782 completed.

REVIEW: *Did they come around to learn from you?*

GOBLE: I guess the 782 was under development, but the 4.1 UNIX that they sold for the 782 was based on our dual UNIX. I actually went to DEC and brought it up in a couple of days. So even though that hardware configuration is different, it was close enough that I could run the same kernel. All we had to do was set up the shared memory right. You just take the terminator off the SBI (synchronous backplane interconnect) and stuff another processor on it. Memory is the limiting factor anyway, so whether you have two CPUs on the same bus grabbing memory or dual port all your memory and put a CPU on each end, you're just doing the arbitration at a different point. If you have an 8 MB single VAX you just take the terminator off, put another processor on at about 50-60 grand and you're in business. It's actually not quite that easy.

DEC had shared memory and the MA-780 controller, in which you can only put 2 MB instead of four. So, to get your eight megs you had to have four memory controllers, which were around 28 grand

apiece, not counting the memory. Then to make their diagnostics work, they had to have a separate non-shared memory controller on each CPU that did nothing but diagnostics.

REVIEW: *Sounds like a more expensive way to go.*

GOBLE: Right, and ours did basically the same thing. But if you had an existing machine our way meant you'd only need two memory controllers (just the standard cheap MS-780s) and you could get nearly twice as much memory. DEC knew it would work our way, too. There were a lot of people at DEC who wanted to implement the Dual VAX the way we did it. But they were told not to.

REVIEW: *Almost sounds as though DEC was trying to sell hardware?*

GOBLE: Yeah. The rumor I heard was that Marketing figured out that a single VAX at the time was probably worth about 500 grand if you bought it from DEC with all the DEC memory and DEC peripherals. Remember, this was back in the 40 grand/megabyte of memory days. Therefore, if a machine came along offering 1.8 or 1.9 times the VAX's power, it would have to sell for about 800 grand, according to Marketing figures. A fully loaded 782 looked like a "freight train" to quote Mike O'Dell. It wouldn't fit in a room diagonally. It contained 12 or 13 bays. Ours was just added on a single, standard VAX bay, so it was only three feet longer than a single VAX. That was a lot of bang for the buck back then.

REVIEW: *Are there any standards for system performance?*

GOBLE: If you're only running in single-user mode, you really only need to take a look at the CPU because that's the only thing you're limited by. On that sort of machine, there's really not much that needs to be done. Where you really start getting hurt is when you run 90 or 110 users on dual VAXen. The biggest problem



we've seen comes when a programmer inadvertently does lots of syscalls like unbuffered writes, and bogs down the system with all sorts of one-character writes to terminal.

If you have your own machine with nobody else on it and you call somebody up at 1200 baud with **tip**, you might be eating 30 or 40 percent of the machine taking in that input because the system has to wake up on every character arriving in raw mode. It works, but if you've got 20 people doing that on a system with 115 users logged in, the system's going to come apart.

REVIEW: *How do you avoid those sorts of situations?*

GOBLE: You monitor the system and find the people that are inadvertently forking commands where they don't have to, or are leaving processes running in an infinite loop, or are reading files but not checking them so that they're just buzzing syscalls out at full speed. Once you find these people and educate them, it will go a long way. It can almost double the throughput of a system.

REVIEW: *How do you deal with high speed input into serial ports?*

GOBLE: Hardware can handle 19.2 kilobaud input and if you soup it up, it'll do 38.4 depending on whose DH (a serial line controller board) you have and what UARTs (Universal Asynchronous Receiver/Transmitters) you're using. The hardware can do it, but if you send data faster than a person can type it, most terminal drivers will fail unless you've made special arrangements.

REVIEW: *That's the Silo problem we've been dealing with for years, isn't it?*

GOBLE: Right, you get the Silo problem not because of the hardware but because the UNIX inter-

rupt routine is so inefficient. For every character, you have to search the process table, wake up a process, shovel it through the C-list, and do whatever other garbage is necessary. In doing that,



You get the Silo problem not because of the hardware but because the UNIX interrupt routine is so inefficient.

you're actually putting the CPU so far behind that it can't get back quick enough to get the next character out. The hardware ends up overrunning because the CPU can't service it.

REVIEW: *So, it comes down to identifying the places that take a few microseconds and figuring out how to use them less often or use them more effectively?*

GOBLE: Yes. Back in the Version 6 days, we built a line discipline called *upload* to handle the serial line problem. At interrupt time, a character would come in and get put into a circular buffer in user space. It was ugly and machine-dependent and it messed with mapping registers. But it worked and you could take 38.4 kilobaud input on a serial port with 70 people logged into an 11/70 and not notice it. You wouldn't get overruns or anything.

REVIEW: *That's marvelous.*

GOBLE: Now, we've rewritten this to be a new line discipline. It's a little slower - but not much - and it's portable. So it's not ugly anymore. Now it copies the data so you can read it out with a normal read/call.

The old discipline used non-blocking I/O, whether you wanted it or not. You couldn't **fork**, you couldn't grow your stack or do anything that might cause a **move** in core or you'd crash. It had lots of restrictions.

The new version is just a plain old line discipline. It's machine independent, you can drop it in and run it. I sent it to Berkeley on a tape a couple of months ago. They're aware of it.

It only takes an **ioctl SETD** to set it and you can even talk to the shell with it if you want. It's like raw mode. It tries to be smart: in a terminal emulator like **tip** it wakes up fairly often so you don't get a mushy keyboard effect. When you transfer a file, the stuff comes blasting in and starts waking the process up once a second instead of on every character. It can take 19.2 kilobaud input, no sweat.

REVIEW: *You've just got to have a big enough buffer?*

GOBLE: Right, it rips off an 8K disk buffer. You have four seconds before you have to do anything. That makes it easy.

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

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

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

Just think how much easier that'd make it for you to prepare documents and graphics, do computer aided engineering or write and debug programs. Imagine, for

example, while a program is compiling in one window, you edit the source code in a second window, check output in a third window, and send and receive mail concurrently in a fourth window.

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

As good as the 5620 makes Teletype Corporation look, it can make you look even better. To find out how, write Teletype Corporation, 5555 Touhy Ave., Dept. 3223-00, Skokie, IL 60077. Or call 1 800 323-1229, ext. 615.

TELETYPE: VALUE SETS US APART.

"Teletype" is a registered trademark and service mark of Teletype Corporation.
*UNIX is a trademark of Bell Telephone Laboratories, Inc.





REVIEW: Are you less concerned now with memory use than you used to be?

GOBLE: Yes, with the VAX it's easy. You have address space.

REVIEW: What about networking performance?

GOBLE: Bill Joy sped up TCP/IP by an order of magnitude. But even so, it runs slower by a factor of two in the Purdue Net because it checksums things and has to copy data. But Purdue Net was designed specifically for PDP-11s and VAXen, so we're moving to TCP/IP.

We may be taking a performance loss, but it really doesn't matter since people tend to use networks to do terminal sessions, but continue to send files to each other via **mail**. Using **sendmail** as a copy mechanism is 200 times slower than just **cp** or going through a FTP (file transfer protocol) on a network. So why should we be concerned about the two-to-one difference between Purdue Net and TCP/IP when people are accepting a 200-fold performance loss on file transfer? It doesn't make a hill of beans of difference whether we use TCP/IP or Purdue Net because we're being killed by **sendmail**, not by the network.

REVIEW: Are you going to change **sendmail**?

GOBLE: We're thinking about it. We'll be educating users to avoid **sendmail** as network file systems become available. I had written a network file system four or five years ago but I never typed it in. I figured it would swamp things.

REVIEW: Well, Sun, among others, is proposing to put one out now.

GOBLE: Several manufacturers have picked that software up. It won't offer the best efficiency but, it'll sure beat **mail**, so that's an improvement if you look at the total picture.

REVIEW: The idea is to make the network as easy for people to use as **mail**. Then maybe they'll use it.

GOBLE: Right. We have several small VAXen that want better networking than **uucp**. Kirk



It doesn't make a hill of beans of difference whether we use TCP/IP or Purdue Net because we're being killed by **sendmail**, not by the network.

Smith and I implemented a serial line TCP/IP called *slip*. I guess a couple of others have written something like this before. We picked up one of those and found it was horribly inefficient. It did single character puts and every time you took an input character,

it called a whole bunch of subroutines—which is expensive on a VAX. Kirk Smith did most of the work on cleaning up the protocol on *slip* and getting it to work. Then I went into it for efficiency and pointed out to him what tricks to use. For example, I showed him the Berknet input trick of taking the character interrupt, stuffing the character in the buffer, and checking to see if it's at the end. Then you return right away without doing any subroutine calls and expand it in a macro so it looks like a subroutine call.

That speeds up the network by a factor of 30 or 40. So we can run this thing at 19.2 kilobaud, which is a third of the speed you get with Arpanet IMP. Hook up any old plain serial line and then run it where you have to, and put line drivers on it. You can be up in a day and run **rlogin** through it. That stuff's in place and working during terminal sessions. You won't even notice it slowing down.

REVIEW: Is this a reasonable way to tie Ethernets together for moderate capacity?

GOBLE: Not really. It's nice for a small VAX 750 with three or four people using it, if it's too far from an Ethernet. We just use machines to gateway Ethernets now. But *slip* (at 19.2 kilobaud) is just too slow to gateway between Ethernets. It'll take three or four minutes to transfer a 400 KB file like **vmunix**. The thing actually runs right at 19 Kbaud and you get an effective 17 kilobaud throughput. When you count the overhead for TCP/IP, it takes 10 percent of your machine so *slip* is a pretty good deal. It might be on 4.3 too. I gave all that to Berkeley. I think Kirk Smith did 90 percent of the work, though.

REVIEW: With performance in mind, do you just look around for problems?

Only Sperry can make the following four statements.

Our PC runs the XENIX™ system, as well as MS-DOS™.

Our 4 new microcomputers run the UNIX system.

Our new minicomputer runs the UNIX system.

Our Series 1100 mainframes run the UNIX system.

All of which means there is a great deal we can do for you.

For instance, our family of computers based on UNIX systems has incredible transportability for all your software.

And being able to accommodate from two to hundreds of users, it's impossible to outgrow our hardware.

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

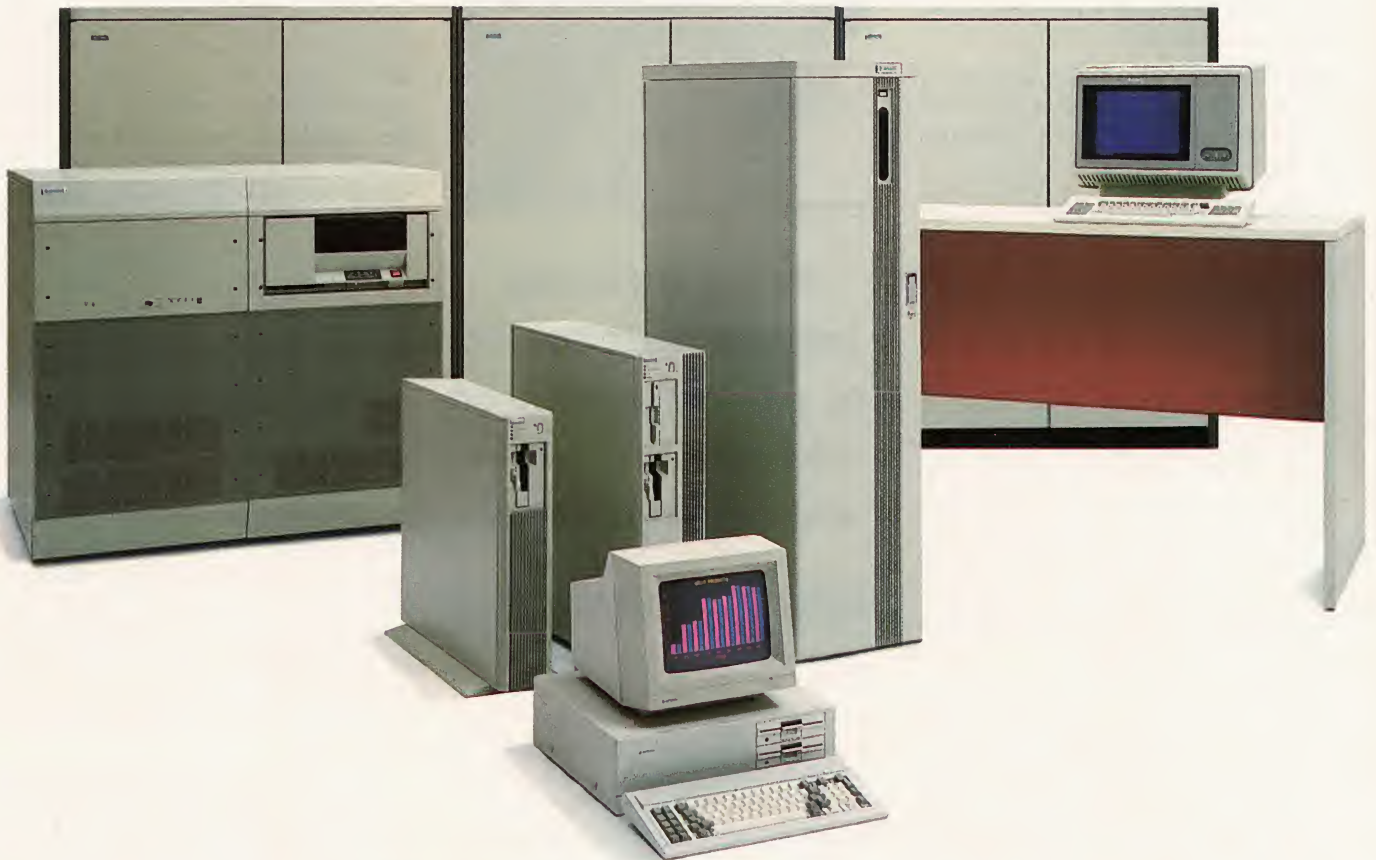
And a fast way to find out

more is to get a copy of our Sperry Information kit. For yours, or to arrange a demonstration at one of our Productivity Centers, call **1-800-547-8362.**

*UNIX is a trademark of AT&T Bell Laboratories
XENIX and MS-DOS are trademarks of Microsoft Corporation
© Sperry Corporation 1985.



Introducing an idea that makes obsolescence obsolete.



The UNIX* operating system from PC to mainframe.



GOBLE: We know how many users we can get and we watch the load average (the number of things in the run queue) and we watch systime, too. If somebody screws up the percent of systime to usertime, it gets really bad. You have to watch that.

REVIEW: *These are things that are available to any user?*

GOBLE: Right. The **vmstat** utility coughs the numbers up. We also have a different way of getting it out here. We actually have a network-wide display like **vmstat** that brings in information from every machine, updated every five minutes. The data relates to the number of users, load average, expenditure of user time, and systime accounting.

REVIEW: *Do you have anything in there to point out exceptional systime expenditures relative to user time?*

GOBLE: We do because it really wrecks our dual CPU if you have some processes doing nothing but syscalls. That makes the slave processor sit idle. This isn't the way Bell did it. Bell actually went through and semaphored everything. They run multiple CPUs in kernel mode. It's a lot of work to do that. And I'm not sure if you can ever get a solid kernel with their approach. But I only spent two weeks writing code for the dual processor and it gets the job done.

REVIEW: *You didn't change the kernel very much?*

GOBLE: Just the master/slave stuff. Slave just runs user processes and we feed all the syscalls through the master. If you keep systime below 50 percent, then when you go dual processor that 50 will almost turn into 100. As long as a master is doing all the syscalls and still keeping up, it's going to be just as good as a machine where you can run two kernels at once. My scheme only

works for two CPUs but it's quick and it works. I haven't got 10 people to spend a year implementing something that will be obsolete as soon as it's done. Instead, we watch syscalls very carefully.



You can generally improve performance by a factor of three or four by just going in and cleaning stuff up.

REVIEW: *In principle, when you're trying to identify a problem in UNIX, you end up using a profile. But you've said you'd rather use Bill Joy's eyeballs.*

GOBLE: Yeah, but if you run a process and it ends up using a lot more systime than you think it should, you know it needs looking at. Then you can profile that user process to find out where the problems might be.

One of the things we have done that nobody else seems to have picked up is that we've found some scheduler bugs in 4.1. Bill Joy has actually worked on rescheduling some because interactive things like **vi** could actually hang up a second or so even if you had a light load. Berkeley ended up changing the rescheduling so that it now occurs every tenth of a second. The problem's still there but now it's just not noticeable to a human anymore.

REVIEW: *Anything under a 100 milliseconds doesn't matter?*

GOBLE: Right. We had a PDP-11/44 that would run on the Unibus. It would monitor VAX main memory and it looked like a DMA device so it wouldn't impact the VAX. We could get displays like a process table updated 10 times per second. You could see processes flipping in and out and you could see forking, process switching, and find out which CPU was running on which process.

We used that system to develop the dual CPU kernel. Over a one-second interval you could watch the run queue go from maybe three to 10 or 15 before dropping down to one or two again. You could see screen editors waiting for service and they would sit idle for a whole second before processing a single character. You could never see this stuff with a **ps** command. But with that monitor, a peripheral processor just generated the display and let you look through core and processes and actually see the thing running in real time.

We made a movie and showed it at Usenix. You could actually see the system going multiuser and see 100 **inits** just appear out of nowhere at once. We've also added a "research" bit. After you use 20 seconds of CPU time it gives you a **nice + 10** if you're not interactive. Those jobs get maybe 10

percent of the CPU, keeping the rest of the CPU free for the interactive jobs.

REVIEW: *Sounds like a policy decision.*

GOBLE: It's worked out well here. Interactive work really doesn't consume that much of the CPU, so this gives good response and the background things probably run 90 percent as fast as they would have otherwise. Who cares whether a job takes six hours or six and a half hours? If it means good interactive response for the whole system, I think it's worth it. Even on the Gould machine, running eight or 10 times faster than a VAX, five batch jobs in the run queue have made **vi** run slowly. Everybody grumbles about it. So I've had to **renice** everything to **+10** by hand, to simulate the research bit. We've just had the Gould up for less than a week, so it's a little premature to talk about the effects of that approach.

REVIEW: *What do you tell application programmers about system performance?*

GOBLE: Just to try to minimize system calls. Some grad student will write a program that dogs a lot of system calls. He'll run it in the middle of the night, and it'll run fine because he had the whole VAX to himself. Then his professor will say "Gee, this is neat", and will pass it out to 600 students. They'll run it in the middle of the afternoon and we'll have a disaster on our hands.

REVIEW: *So, in fact you don't design for efficiency, but rather work to find inefficiency? Then you profile it and fix the slow parts?*

GOBLE: Right, but only if the program gets used heavily. Then you do a panic scramble. For instance, people might be deleting 20 files by doing a system **rm**. That will mean 40 process forks and 20

loads of the shell. It could just as easily be done with a **for** loop including an **unlink** call. That would be a thousand times faster. When you have these sorts of things, what difference does it make if your kernel runs five percent faster?

At Bell Labs I've seen stuff running 80, 90 percent systime loads all the time. Ritchie and Thompson aren't supposed to clean that stuff up because they're exploring new ideas there. But here we must have 10,000 active accounts on our network. When 600 of those users run the same inefficient program, you find out about it—immediately.

REVIEW: *Have you developed any specific guidelines for programs at Purdue?*

GOBLE: Well, you try, but for every 1000 programs written, probably only three or four of them actually achieve wide use. It's just easier to let problems happen once and then find them. We penalize excessive system calls by lowering the priority of the programs that contain them, so programmers responsible for bogging the machine down with systime will get no response. When they come and complain we'll find out what the problem is and educate them about how to avoid it in the future.

REVIEW: *How do benchmarks tend to mislead?*

GOBLE: My opinion on benchmarks in general is that you can make them show anything you want; you just have to know the answer before you start. You can sure shift results by factors of three and four with no problem.

It used to be when you had a simple Von Neumann machine with maybe a cache and flat-out register increment speed, you could compare machines. You could say this machine is five times faster than that one and it

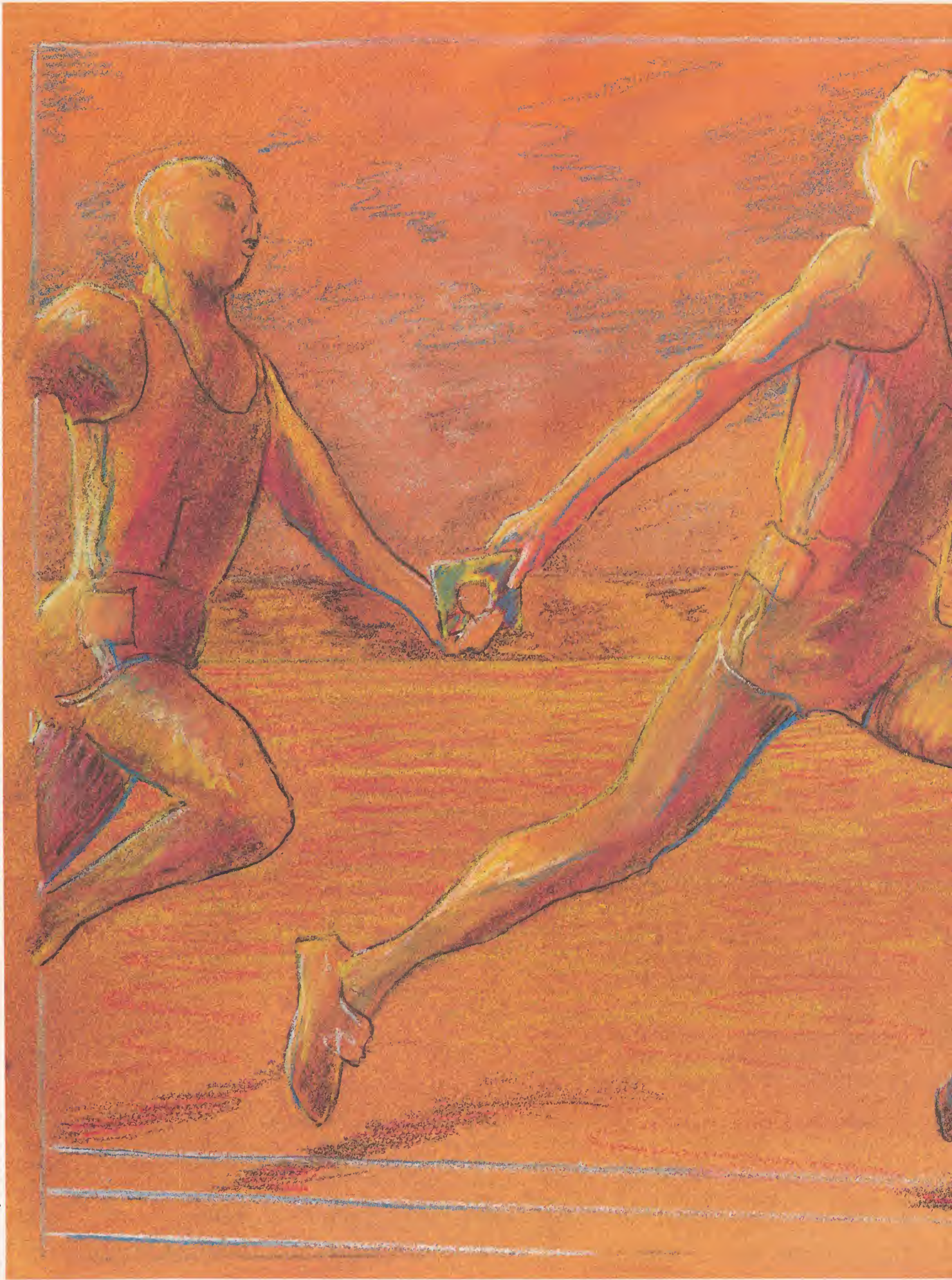
would really mean something. But I just did a register spin on our Gould machine we have here and it showed one processor ran 3.5 times faster than a VAX 11/780. Yet a C program that just ran on the machine ran 10 times as fast as it would have on a VAX. Another might run 15 times as fast as on a VAX. A compile might run four times faster. It really depends. You also have to take disks into account. You can get a rough number from benchmarks but you might as well run a bunch of your own programs on the machines you're trying to compare. That's probably as good a test as you can get. Differences of 10 percent aren't even noticeable. But when you get differences that you can measure by factors of three and four, they'll start showing up.

REVIEW: *What would you do with a machine that seemed to be overloaded if you didn't have kernel hackers around?*

GOBLE: Find a UNIX consultant and hire him. You can generally improve performance by a factor of three or four by just going in and cleaning stuff up. So hire a UNIX consultant for a couple of months to just skim over the obvious stuff.

REVIEW: *Thus saving the purchase of three or four more machines?*

GOBLE: Right. Read our Dual CPU report and you'll see that if we didn't monitor load, we'd probably only get a fourth as much work done. That's a lot more effective than tweaking another three percent out of the kernel. Still, Berkeley will get 10 percent improvement and propagate it on 10 other VAXen. That's almost like getting a VAX for free. The message in that, I guess, is that before you go out and buy another machine, make sure you're running lean and clean on what you already have. ■





PERFORMANCE TRADEOFFS

Optimizing software for the UNIX environment

by Roger J. Sippl

Does the UNIX operating system architecture offer enough performance for a commercial data processing environment? Now that UNIX is in use on over 100,000 commercial data processing machines, and seems destined to appear on millions more, that may be a silly question. But in the early days of UNIX "commercialization", it was one that was often asked.

UNIX has an elegant internal architecture. But it must be *understood* by application builders if it is to be optimized to the benefit of end users. Fortunately, the internal concepts that applications builders must master are fairly simple and logical.

Important design issues built around the internal constructs of UNIX will be reviewed in this article from the point of view of three of the most common kinds of business application software—the spreadsheet, the word processor, and the database management system.

Custom software built by commercial data processing professionals generally obey the same rules and benefit from the same design techniques that apply to these three typical office automation packages. Common custom-built applications, (especially those written in a language, like COBOL, that uses an ISAM indexed file facility) are built on top of database systems. However, some applications—such as calendaring, statistics, scheduling,

communications, and budgeting—use machines in a way more closely resembling spreadsheets or word processors than databases.

THREE MAIN RESOURCES: MAIN MEMORY, DISK, AND CPU

The main architectural issues in software design revolve around the use of the computer's major resources—memory, disk, and CPU. If any or all of them are overburdened, the application's performance will suffer—and so will the user.

"Poor performance" almost always translates simply into "insufficient speed". Wrong answers are another problem altogether, typically resulting from program bugs or erroneous code. Slow or irregular execution—or, on occasion, some other behavior that is "temporarily annoying" to the user—are the conditions described by "poor performance".

UNIX is designed so that memory, disk, and the CPU are intertwined. Often, this can lead to a weakness in one area being hidden from the user by an abundance in another. The most obvious example is virtual memory. With all UNIX systems, more programs can be run than will actually fit into main memory. UNIX will "swap out" one or more of the programs currently in use to make room for a program of more immediate interest.

When UNIX looks to swap a



process out, it tries to find a program that hasn't been used much lately. In theory, the "least recently used" (LRU) program should be the best candidate for removal. Of course, when the user of the swapped program finally does provide some fresh input, UNIX must pull the program back into main memory, and possibly swap somebody else's process out.

When UNIX swaps a program out, it temporarily writes the program to a "swapping disk", where the process remains until fresh input causes UNIX to put it back into memory. On most UNIX machines, the "swapping disk" is simply a partition on one of the drives.

By using the "virtual memory" principle, UNIX can accommodate more users and/or more programs in main memory than the

actual amount of main memory would otherwise allow. When real memory is exceeded, the computer's disk I/O system labors to make up for the lack of sufficient memory. UNIX will lean on a disk drive to handle the overflow if it has to. However, swapping is typically slow—enough so to be noticed.

If a machine must swap a great deal, it's typically said to be "thrashing". A thrashing machine spends most of its CPU and disk input/output time just trying to keep up with virtual memory housekeeping—at the expense of getting much useful work done.

Causing the machine—especially a small, multiuser UNIX machine—to swap is probably the most important thing to avoid. It may sound obvious, but the best way for an application builder to

avoid heavy swapping is to write small applications.

If that is not possible, it is sometimes a good idea to break large programs into two pieces that can "pipe" data back and forth. Then, if swapping is necessary, at least the system will be able to swap something smaller. It is usually easier for UNIX to find a "hole" for a small program when it swaps back into main memory. (See Figure 1.) Breaking programs is also good if one part is used infrequently. There's certainly no reason for inert chunks of code to always be resident in main memory, taking up valuable space.

In addition to considering space efficiency, software engineers must write code designed for speed. This also seems obvious, but taken together with the "write small programs" rule, it means: write in C or some other compiled language that generates machine instructions optimized for both fast execution and minimum code size.

Another reason to write in C is to produce programs that are "re-entrant". This is important because UNIX allows a program's code to be shared among multiple users. Figure 2 shows a version of a UNIX "memory map" with four users running programs at the same time. Notice that there are two different types of word processors in use, as well as a third program doing spreadsheet work. Also note, though, that there are only three user code areas available.

John and Susan have their own code areas since they're the only ones running their respective applications. But Steve and Mary are able to share the same code segment. This sort of efficiency is possible only when programs are written in a language such as C that supports re-entrant execution of code. Interpreted languages do not support re-

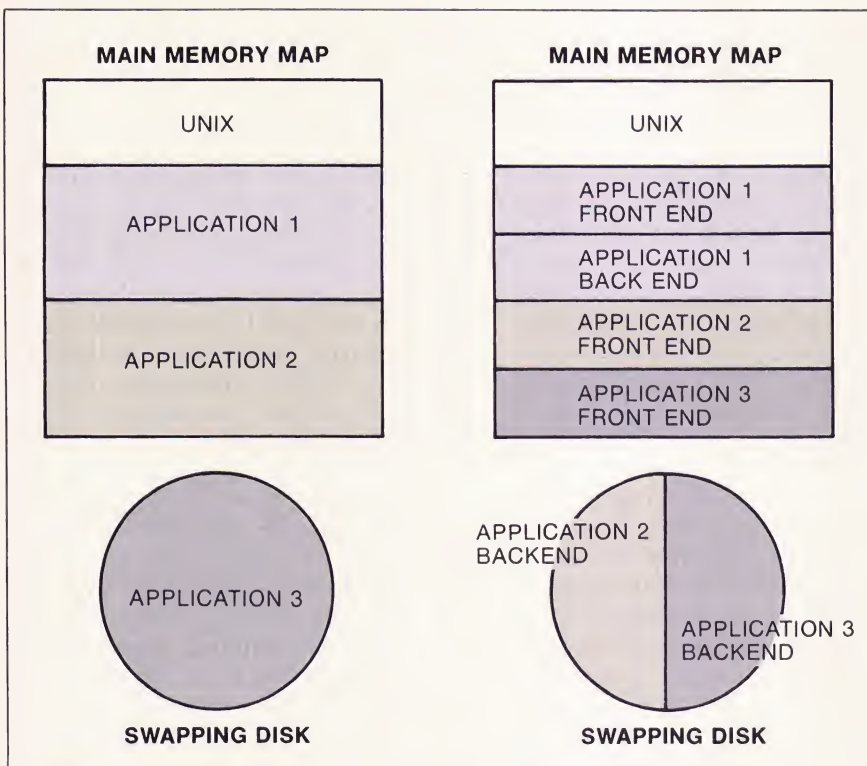


Figure 1 — In the first representation, only three applications are in use. But in the second representation, all three programs can be used simultaneously. The difference? Only the size of the code portions being swapped has been changed.

entrant execution because code being interpreted is actually regarded as data—from the UNIX point of view. (However, the interpreter itself, if written in C, will have a re-entrant code segment.)

Another example of how re-entrant code can help overall performance is shown in Figure 3. This figure shows two alternative process architectures of the same database-oriented program. One typical program that might be found in a database environment would be an interactive query program that shares a great deal of code with a report writer or a custom database program.

By putting the common code into a separate process, the total amount of space needed to run them all simultaneously is decreased.

To gain a little extra speed, the

application could be “profiled” using the UNIX **profile** utility, and certain very tight “inner loops” could be re-written in assembler as well as a high level language. If the assembler is convenient on the target machine, this inner loop optimization can be very useful in conserving CPU resources, in addition to providing the user with greater speed (which may or may not be noticeable, depending on the location of the loop in the program).

The principles for writing code that is small, fast, compiler-optimized, and re-entrant hold true for just about any software. Within each application category, moreover, are other specific areas of concern. For example, word processors have to be particularly careful not to waste cursor motion when refreshing the screen of a

dumb terminal. The **courses** library available on most UNIX machines is a good tool for helping with that, although it tends to be big.

Thus, if the machine is getting to the point where it needs to swap processes, the size of a program can literally slow it down more than the slow feature the code was designed to improve. This is a classic time/space tradeoff that has impact in many areas.

Looking at the tradeoff in another way, the swapping requirements of UNIX sometimes mean that the best way to get a computer to go fast is to add memory. This will become even more true as memory becomes cheaper and software designs become increasingly memory intensive. It’s particularly true for database style applications.

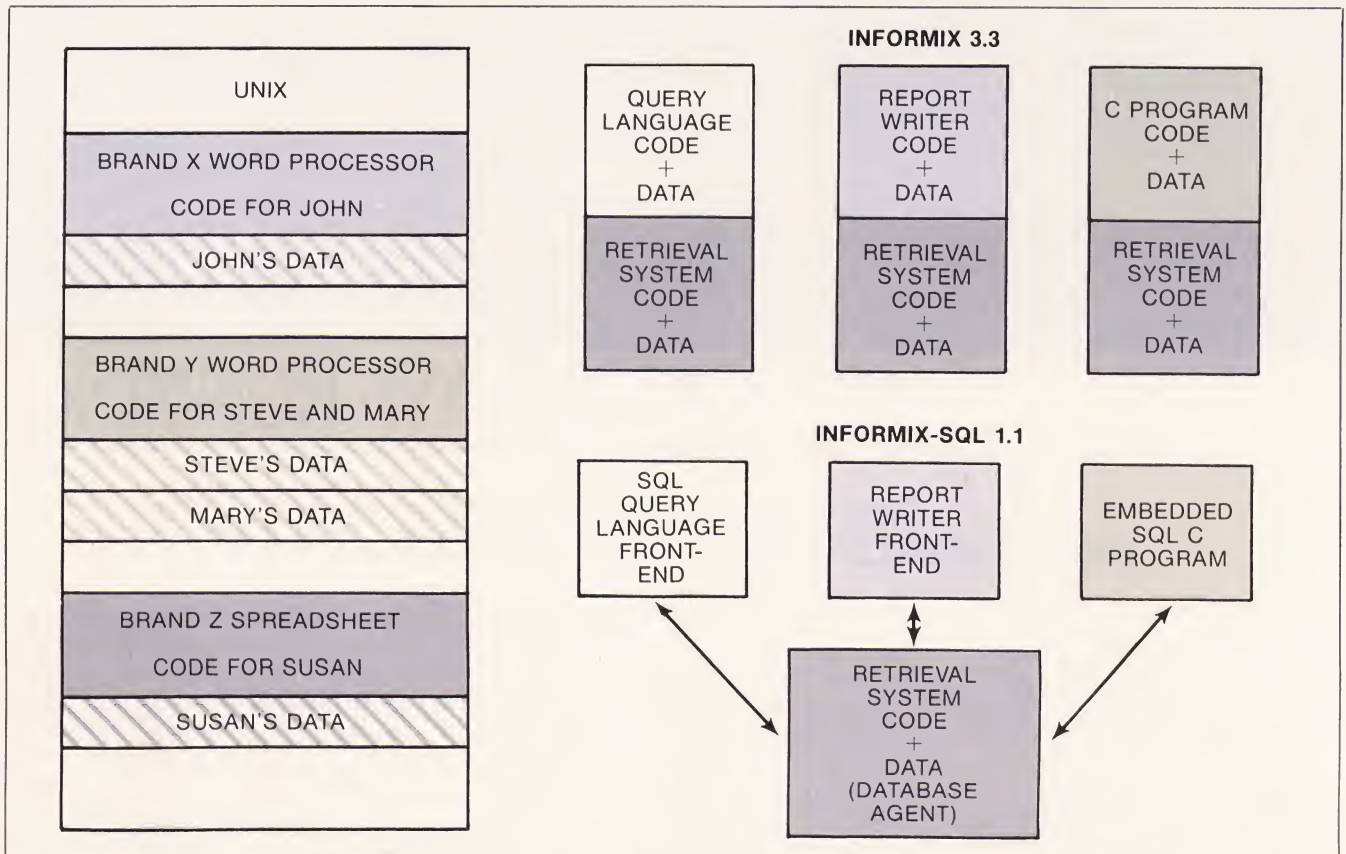


Figure 2 — A typical UNIX main memory map.

Figure 3 — Two similar database products using different architectures.



ARE YOU ALWAYS RUNNING UP

UNIX FOR ANY ENVIRONMENT— DATA GENERAL'S HOSTED MV/UX™ AND NATIVE DG/UX.™

Now you can get around just about any programming obstacle. Because whatever environment you're working with, Data General's MV/UX and DG/UX give you more UNIX solutions than anyone else.

THE CHOICE IS YOURS

A hosted UNIX environment, MV/UX is fully integrated with Data General's Advanced Operating System/Virtual Storage (AOS/VS). Which gives you access to our CEO® Comprehensive Electronic Office software. Along with a full

range of commercial and technical applications, and the Ada® Development Environment.

A native UNIX operating system, DG/UX is compatible with BSD UNIX and AT&T's UNIX System V Release 2, offering performance enhancements and network support.

FULLY COMPATIBLE

Both MV/UX and DG/UX have a lot in common. They operate on our entire ECLIPSE® MV/Family of computers. The industry leaders in price performance

from superminis to single-user workstations. Each works with the industry standard communications network, too. Both come with high-quality, comprehensive documentation and support services.


All of which gives you quick solutions to meet your development, communications and production needs.

CALL NOW

For more information on UNIX systems that are a generation ahead, call: 1-800-DATAGEN or write Data General, 4400 Computer Drive, Westboro, MA 01580.

AGAINST UNIX™ LIMITATIONS?



 **Data General**
a Generation ahead.

CEO and ECLIPSE are U.S. registered trademarks. MV/UX and DG/UX are trademarks of Data General Corporation. UNIX is a trademark of Bell Laboratories. Ada is a registered trademark of the U.S. Dept. of Defense (OUSDRE-AIPO). © 1985, Data General Corporation, Westboro, MA

Circle No. 284 on Inquiry Card



DISK ACCESS TIME SAVINGS THROUGH DATA BUFFERING

Database applications hinge on many of the same performance issues that word processors and spreadsheets do. Most database products have screen packages, and thus—like word processors—must worry about quickly refreshing the screen. Most database systems also have report writers, and thus—like spreadsheet programs—are concerned with the speed of calculation.

However, spreadsheets and word processors do not constantly store and retrieve data from disk. Typically, a spreadsheet or word

processor user reads in work from the disk at the beginning of a session or starts with no data at all. Often the data is only written out to disk when the session is complete. This makes for minimal disk activity, and the amount of data written to disk is usually quite small.

Database programs, on the other hand, are constantly reading and writing to disk. The amount of data requested from disk is usually large—sometimes enormous.

Compared to the speed of the processor, even the fastest disk drives are very, very slow. The key to performance in such "disk intensive" applications as database programs, then, is to avoid the disk as much as possible.

Here the interaction between main memory, CPU, and disk comes into play. Any one of the three can make up for a weakness in one of the others. So, let's say the disk drive is slow. No problem—just keep as much of the data as possible in main memory so as to minimize reads to disk. Of course, it has to be read in at least once but, after that, it should be kept around as long as possible in case it's needed again. (See Figure 4.)

SHARED MEMORY—TO SAVE BOTH TIME AND SPACE

System V and some other versions of UNIX provide a feature called "shared memory" to allow different programs executing in main memory to share some of their data, much like re-entrant programs allow their code to be shared.

There are some differences, however. UNIX sets up sharing of re-entrant code automatically for C programs. In order for two programs to share data, they must do it intentionally. In fact, it is unusual for two programs to share the same data in main memory,

and when it does occur, it must be controlled very carefully. However, the time and size savings gained by using shared memory in database applications is so beneficial that its use is likely to become widespread for database type applications. This is one of those rare instances when both time and space can be gained through the use of an elegant technique.

To explain how shared memory saves both time and space, let's examine two different database architectures. Figure 5 shows three different users of a database program that doesn't use shared memory. Figure 6 shows the same memory map of a computer where the database software utilizes shared memory to hold the bulk of the data in the user's data space.

Notice that there is one code

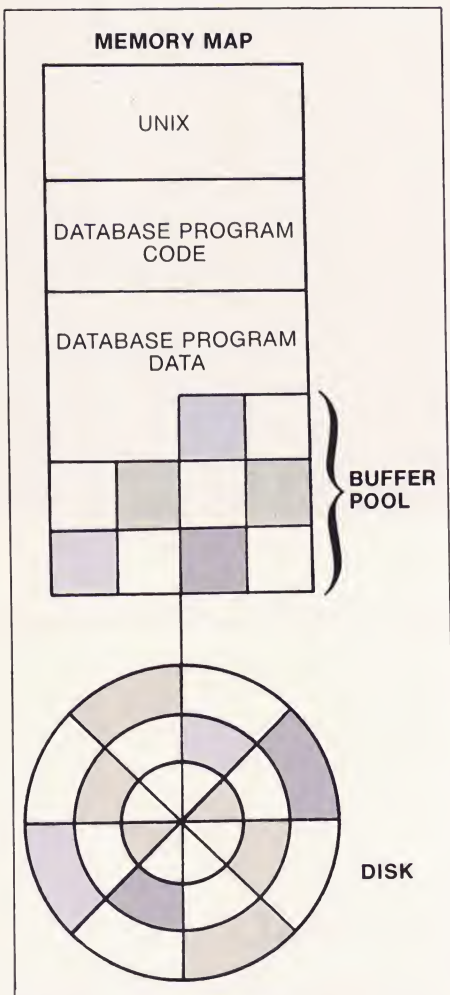


Figure 4 — Main memory buffer pooling of a database.

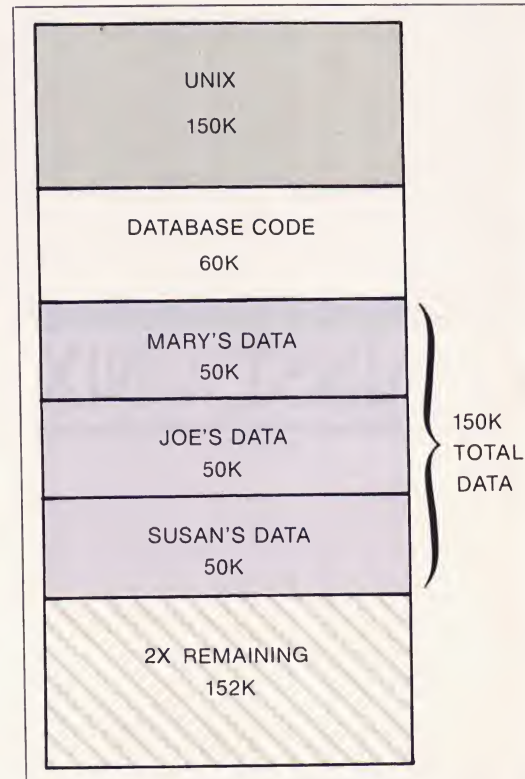


Figure 5 — Three users of a database program that doesn't use shared memory.

space for all three of the users. This is because of the re-entrant nature of most programs running under UNIX. Also, there is a "non-shared" data area for each of the three users. However, the bulk of the data in the three users' data areas is consolidated into a single shared data area, and the total amount of data used by those individuals in Figure 6 is thus substantially reduced.

Not only is the use of space more efficient under this scheme, allowing other programs to run without swapping, but the fact that the three users do not *each* have a separate copy of data is critically important for good multi-user performance.

In the situation shown in Figure 5, there would be three copies of the same data in main memory if all three users were accessing

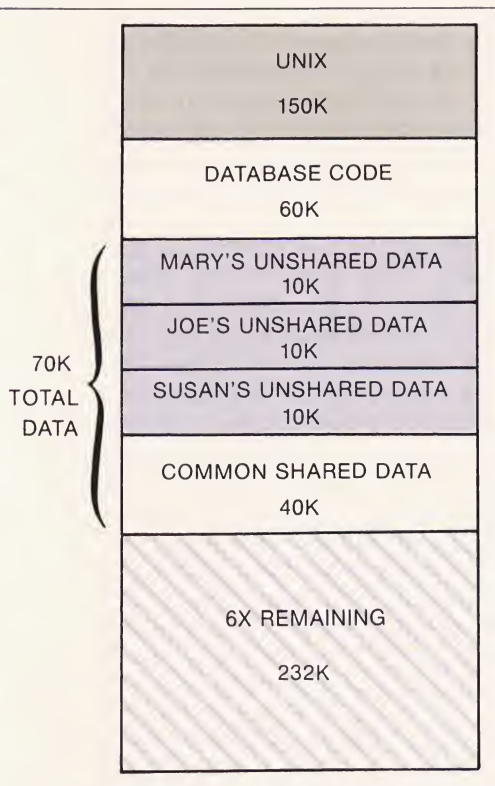
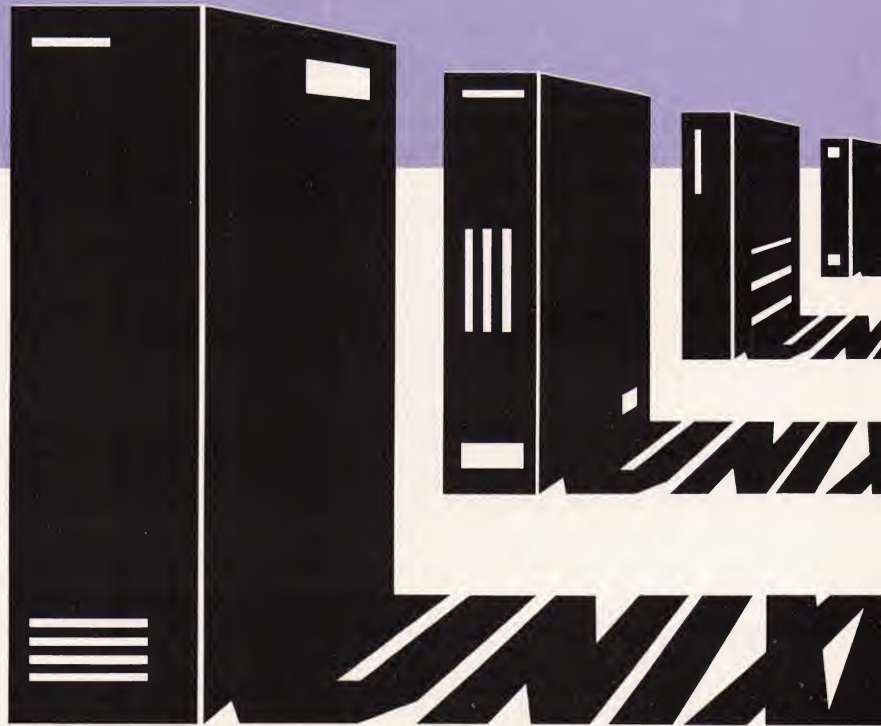


Figure 6 — Three users of a database program that uses shared memory.

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



some of the same data. This is obviously wasteful, but when you consider that this data is also subject to change, another, much larger problem comes into play.

The data being buffered from the disk in these main memory data areas is not only the actual database data, but is also the structural data, such as portions of search indexes. When a record is added, updated, or deleted, several pieces of data can be changed. Since the buffer pools are separate, the other users "don't know the data has been changed" until the user changing it writes it out and the others read it in again.

It is up to the database program to decide when to write it out and read it back in (and remember, the goal is to do it as little as possible). But if one of the user's buffers are changed, the database program has to "refresh" the other user's buffers with the latest version. Since almost all database applications "localize" the update activity among users to the most recently added records, this "refreshing" can be quite expensive. If update activity is heavy, it can defeat most of the advantage of buffering.

However, shared memory allows only one copy of the buffer pool (as shown in Figure 6). Therefore, if one user changes some of the buffers, there is no other buffer pool that needs to be refreshed. All of the users, after all, use the same set of buffered data.

This saving of space by using a common buffer pool actually allows the database program to buffer more data. Therefore, even if there isn't anything else for the machine to do with the empty space, an application written to use space "dynamically", as activity dictates, will have a bigger common buffer pool, and will run faster than it would otherwise.

The bigger buffer pool will eliminate many disk accesses as the buffer pool becomes filled with the most often used pieces of the database. Because there is only one buffer pool, refreshing is not necessary. This is particularly important on computers with many users and high update activity.

A LOOK TO THE FUTURE— NETWORKING TRADEOFFS

Information users will benefit greatly from having their computers in a network. Users of small personal computers will soon be

**Starting with simple
benchmarks and
advancing to complex
terminal emulators,
systems programmers
have been lured into
wondering just how
well their systems are
performing.**

able to network to larger PCs, UNIX supermicros, and even large mainframe computers, and thus will be able to share the resources of those other computers.

The shared resources could be large disk drives, fast printers, laser printers, or more importantly, information centrally located and controlled by a "file server" in the network. Most of the architecture issues discussed thusfar can be viewed from a new perspective when a network of computers is involved.

The tradeoffs for buffer pooling

are slightly different in a network environment. Consider the simple case of a network of PCs or supermicros. The user of one of the computers in a network can specify that files in one of the other computers be as accessible as the disk drives of the local computer.

In this style of networking, the "request for data" message sent by the operating system of the local computer will be "send me block *x* from the disk drive", and the server computer will respond by sending the appropriate block of data (usually 512 bytes).

If the remote computer user is going to change that data and write it back, the change may overwrite a modification made by the user at the computer containing the data. To avoid clashes of this sort between multiple users, "concurrency control" is handled with "record locking" solutions. For the most part, the problem is solved on UNIX systems, but it is still something to watch for when designing software for networks.

Another problem with networks is that memory on two machines cannot be pooled for efficiency in the way that it can on a single UNIX system with shared memory. However, on the good side, networks can be designed so that the computer with the data has a great deal of main memory. Requesting computers, then, can phrase their requests in "higher level terms", such as: "Get me the average salary of all the employees in the United States."

In this case, where a high level request is sent to the server, very little data is moved on the network. The request is a small amount of data, and the answer is only a single number (in this case). If this request were to be satisfied by sending the operating system "read me a block" type requests, the majority of the file

Continued to Page 93

NAME THE MOST WIDELY USED INTEGRATED OFFICE AUTOMATION SOFTWARE FOR UNIX™ SYSTEMS.

“UNIPLEX II”™

YOU'VE GOT IT!

User satisfaction is the primary reason no other product can make this claim. Already in its second generation, UNIPLEX II offers features designed to meet the requirements of the most demanding user.

The beauty of UNIPLEX II is its simplicity. One personality and one command structure throughout the program provide an ease of use never before experienced with UNIX application software.

UNIPLEX II integrates sophisticated word processing, spreadsheet, and relational database applications into a powerful one-product solution.

UNIPLEX II uses termcap, so it can run on virtually any computer terminal. “Softkeys” allow the user to define function keys which are displayed on the 25th line of most terminals to provide versatility and ease of use.

All this at a price you'd normally pay for a single application software package.

UNIPLEX II is available immediately from UniPress Software, the company that's been at the forefront of quality UNIX software products longer than anyone else.

OEM terms available. Mastercard and Visa accepted.

Call Today! Once you've got it, you'll see why UNIPLEX II is the most widely used integrated office automation software for UNIX-based systems.

Write to: UniPress Software, 2025 Lincoln Hwy., Edison, NJ 08817 or call: 1-800-222-0550 (outside NJ) or 201-985-8000 (in NJ); Telex: 709418. Japanese Distributor: Softec, Telephone: 0480 (85) 6565. Swiss Distributor: Modulator SA, Telephone: (031) 59 22 22.

UniPress Software
Your Leading Source for UNIX™ Software

INDUSTRY INSIDER

What a show!

by Mark G. Sobell

UniForum '85 in Dallas drew more people and more vendors than any previous UNIX show. What's more, a lot of those people weren't just kicking tires—many attendees were corporate end users handling large procurements for Fortune 1000 companies. The vendors, meanwhile, gave a heartening show of cooperation indicating a desire to understand and fulfill the needs of end users.

The week started with an AT&T announcement declaring the need for a clearly defined and broadly accepted system standard. AT&T revealed that it would cooperate with Microsoft to bring XENIX up to the current System V release *and* that the two companies would continue to work together to see that XENIX System V and the AT&T product evolve together. This announcement means that System V will almost certainly be available on IBM PC-ATs running XENIX. Whether IBM will choose to *ship* System V is another question, but I see no reason for it to hold its ground with System III. (IBM had no comment except to say it was looking at System V.)

AT&T also announced an agreement with UniSoft, the people responsible for many of the UNIX ports available today. Under the agreement, UniSoft will develop a verification suite to ensure that systems claiming to run Sys-



tem V UNIX meet certain criteria. The first manufacturer to have its system verified? Microsoft. In fact, XENIX will be the pilot system for the verification suite, and Microsoft will work with UniSoft to get the bugs out of its system.

/usr/net

On the UniForum floor, a 2900-foot yellow Ethernet cable connected computers in 27 different manufacturers' booths. Each cooperating vendor gave the others system logins. What was the major force behind this effort? Dave Langlais and Mario Castro of the Wollongong Group spent the week before the show stringing the yellow cable between booths on two floors. The cable itself was donated by Belden while the transceivers (one for each computer on the network) were donated by Interlan.

"This network epitomizes what UNIX is about," exclaimed Langlais. "When you have 27 companies, including Gould, DEC, and Data General cooperating in this fashion, you see the environment and spirit that is UNIX." The network, dubbed /usr/net, was based on the TCP/IP standard. It connected machines ranging in size from a Gould Fire-breather to an IBM PC and transferred data at rates in excess of 50 Kbps.

What good was the network? Craig Mathias, director of marketing for NBI, told me, "Four people ported their software to our U! (pronounced 'u factorial') system using the network. It slowed the demos a little, but now we have four more products that run on our system."

TRUE INTEGRATION

Relational Database Systems (RDS) held a press conference at the Big D Ranch just outside Dallas to announce wide ranging support for its SQL "hooks" program. SQL, Structured Query Language, is the IBM standard for mainframe data access and manipulation. RDS claimed that its new program will allow applications based on its new software to be automatically integrated. Representatives from cLINE, Syntactics (Crystal Writer), Horizon, Computer Methods Limited

NEW RELEASE

UNIPRESS EMACS™

VERSION 2

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

Subject: Multi-window, full screen editor.

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

New Features:

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

Features:

- Multi-window, full screen editor for a wide range of UNIX, VMS™ and MS-DOS™ machines.
- "Shell windows" are supported, allowing command execution at anytime during an edit session.
- MLISP™ programming language offers extensibility for making custom editor commands! Keyboard and named macros, too.

- "Key bindings" give full freedom for defining keys.
- Programming aids for C, Pascal and MLISP: EMACS checks for balanced parenthesis and braces, automatically indents and reformats code as needed. C mode produces template of control flow, in three different C styles.
- Available for the VAX™ (UNIX and VMS), a wide range of 68000 machines, IBM-PC™ Rainbow™ 100+, and many more.

Price:

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

*Call for terms

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

OEM terms available.
 Mastercard/Visa accepted.

(XED), Quadratron, Language Processors Inc. (LPI), UX Software, and UniSoft—among others—chimed in their support.

RDS Embedded SQL for the C language is comprised of a pre-processor, a backend process called the "Database Agent", and a library of functions that communicate with the database agent. The backend database agent accepts SQL queries and returns the data selected by the query. This allows C programmers to embed high-level SQL statements in their code. To ensure that they do, RDS announced that it would be giving away source code for the library portion of its embedded SQL to software developers. The

advantage for developers? All applications using the embedded SQL will be able to access data through a common database agent and thus be automatically integrated at the database level.

Bob Ackerman, vice president of sales and marketing for UniSoft, said that RDS' new products and programs solved two major problems for his company. The first problem he pointed to was the demand made by the more than 85 manufacturers using UniSoft UNIX for a highly functional relational database system offering good performance and supporting SQL.

"Although these features were each available in separate data-

base systems, up to now they have not all been available in one system," said Ackerman. "The reasons our clients want performance and functionality are obvious. The reason they want SQL is because it is the emerging standard for micros and supermicros; it allows manufacturers to sell into IBM shops and to share data with IBM machines.

"Our second problem, which we fully expected to solve independently of the first, was how to get applications to talk to each other in a way that our clients could depend on. RDS Embedded SQL for the C language (Informix-ESQL/C) solves the integration problem very nicely and provides another hook into the IBM world."

David Ritter, president of cLINE, summed up the effect of the new RDS products by saying, "This 'hooks' program is an example of what can happen when software developers decide to work together. So many of us have spent time re-inventing the wheel. But with an open systems approach, this can be avoided, and we can all serve our users better."

ANOTHER LINK BETWEEN PCs AND UNIX

A promising startup, Network Innovations, introduced its product, Multiplex, at UniForum. The new package is the latest in a line of new products that target the very important link between PCs and UNIX. But Network Innovations has taken a new slant. Like other products, Multiplex allows you to maintain a database on a UNIX system and retrieve data using a PC connected to it. The innovative part of this new product is that you can retrieve the data in a format compatible with any one of several popular PC programs, including Lotus 1-2-3, dBASE-II, WordStar, Multiplan, and VisiCalc. Multiplex uses a Lotus style

WHEN SERIOUS PROGRAMMING IS YOUR BUSINESS...

The Concurrent Euclid language for systems programming provides the best in efficiency, portability, reliability, and maintainability

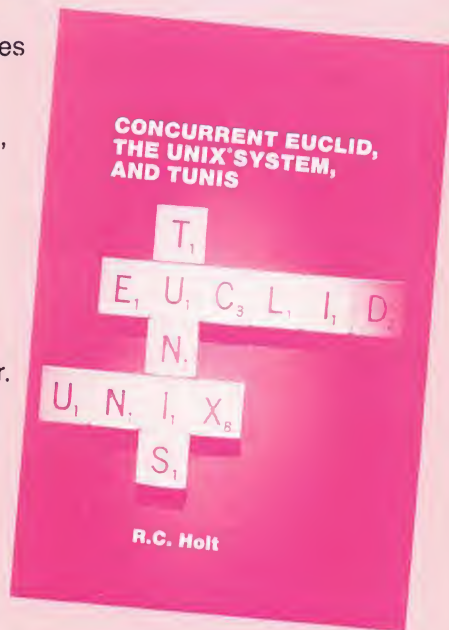
Compilers running on UNIX/VAX, UNIX/11, VMS/VAX, with code generated for MC68000, MC6809, NS32000, 8086/8088 PDP-11, and soon running on IBM-PC

CONCURRENT EUCLID

Compiler: CSRI Distribution Mgr. Sandford Fleming Bldg 2002 10 King's College Road Toronto, Canada M5S 1A4 Tel: (416) 978-6985

Book:

CONCURRENT EUCLID, THE UNIX SYSTEM AND TUNIS Available from: Addison-Wesley Publishing Company, Reading, MA. 01867 Tel: (617) 944-3700



CONCURRENT

E U C L I D

Circle No. 280 on Inquiry Card

interface and context-sensitive help. With it, you can pull data from your UNIX database and enter it directly into a Multiplan spreadsheet, for example, without ever going back to the DOS prompt. Using the program's Query Template facility, you can store a complex query in such a way that it can be performed by someone unfamiliar with the data or the program.

Although it is transparent to the user, Multiplex uses SQL queries on the UNIX system to extract information from the database. Because of this query facility, the next natural step for Multiplex would be an interface designed to tie an IBM mainframe computer database to popular PC programs.

SUMMARY

In addition to the cooperative spirit exemplified by manufacturers at the show, the most important aspect of UniForum was the appearance of people who wanted to use UNIX to solve problems, rather than simply to create more UNIX tools. As Mike Florio, vice president of marketing for The Palantir Corp. and former president of /usr/group, put it, "For the past five years, we have been selling UNIX as an end unto itself. At this show, you started to see UNIX as a means to an end—the end being to solve the user's problems. More people at this show have said, 'I used UNIX (or I need UNIX) to accomplish this end.' I

see this as a necessary and natural evolution for UNIX."

If you have an item appropriate for this column, you can contact Mr. Sobell at 333 Cobalt Way, Suite 106, Sunnyvale, CA 94086.

Mark G. Sobell is the author of "A Practical Guide to the UNIX System" (Benjamin/Cummings, 1984) and "A Practical Guide to UNIX System V" (Benjamin/Cummings, available May, 1985). He has been working with UNIX for five years and specializes in documentation consulting and troff typesetting. Mr. Sobell also writes, lectures, and gives classes in advanced shell programming and troff macro development. ■

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

Subject: C Cross Compiler for the 8086 Family.

The Lattice C Cross Compiler allows the user to write code on a VAX™ (UNIX or VMS™) or MC68000™ machine for the 8086 family. Lattice C is a timesaving tool that allows a more powerful computer to produce object code for the IBM-PC™. The compiler is regarded as the finest C compiler for the 8086 family and produces the fastest and tightest code.

Features:

- For your UNIX or VMS Computer.
- Use your VAX or other UNIX machine to create standard Intel object code for the 8086 (IBM-PC).
- Highly regarded compiler produces fastest and tightest code for the 8086 family.
- Full C language and standard library, compatible with UNIX.
- Small, medium, compact and large address models available.
- Includes compiler, linker, librarian and disassembler.
- 8087™ floating point support.
- MS-DOS™ 2.0 libraries.
- Send and Receive communication package optionally available. Price \$500.
- Optional SSI Intel Style Tools. Package includes linker, locator and assembler and creates executables for debugging on the Intel workstation or for standalone environments. Price \$8,550.

Price:

VAX (UNIX or VMS)	\$5000
MC68000	3000

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

OEM terms available.
Mastercard/Visa accepted.

**CROSS COMPILER
FOR THE 8086™ FAMILY**

**LATTICE® C
CROSS
COMPILER**

Trademarks of Lattice: Lattice, Inc.; VAX and VMS: Digital Equipment Corp.
UNIX: AT&T Bell Laboratories; IBM PC: International Business Machines.
MS-DOS: Microsoft; MC68000: Motorola; 8086/8087: Intel

UniPress Software
Your Leading Source for UNIX Software

Circle No. 279 on Inquiry Card

THE RULES OF THE GAME

The court maze

by Glenn Groenewold

As readers of this column must be aware by now, the laws that apply to the computing field are sketchy and frequently ambiguous. This is often because many of these regulations were initially intended to apply to subjects quite different from computers. For instance, it required a sizable leap to extend copyright law, originally devised to protect literary property rights, to computer programs embedded on floppy disks or microchips.

The judicial acrobatics needed to accomplish that leap were performed with precious little help from Congress and state legislatures. When the courts decide controversies, they are obliged to resolve any ambiguities—and frequently to fill any gaps—in the law. This “judge-made” law has been exceedingly important in the computer field, since it has been largely the source of the legal guidelines that presently exist. It will undoubtedly continue to be the major source of computer law in the foreseeable future since legislatures are generally unable to keep up with rapidly evolving technology.

But when we refer to “the courts” in the United States, we’re not talking about a tidy hierarchy of tribunals such as the ones that exist in many developed countries. Our unique brand of federalism operates to create a



complex maze of legal jurisdictions in which it’s easy for the unwary to get lost. Therefore, it’s a good idea to have a basic understanding of the court system so you’ll have some notion of where you’re likely to end up in the unhappy event that you find yourself a participant in a lawsuit.

STATE COURTS

The state courts are the logical place to begin a tour of the judicial system. Most of us have had some experience with these tribunals, if only to pay parking tickets. Naturally there are 50 completely separate sets of state courts. At the lowest level are the *trial courts*. In its most folksy incarnation, the trial court is represented by the good old justice of the peace. But there always will be at least one trial court, complete with juries and lawyers, resem-

bling the traditional tribunals of plays and movies. It’s here that a lawsuit will be tried. That is, it’s where the testimony of witnesses will be heard, the physical evidence received, and a decision rendered.

But as the old saying has it, there’s almost always bound to be someone disappointed by the outcome of a trial. Because of this, and since it’s recognized that trial courts do sometimes make mistakes, the system provides for an appeal to a higher court. (In relation to one another, courts are considered to be “lower” or “higher” according to whether one has authority to overturn the other’s decisions.)

These *appellate courts* bear little resemblance to what we’re accustomed to seeing in the movies. The parties wage their battle largely with paper, chiefly appellate briefs (which may be anything but brief). An appellate court does not hear testimony from witnesses and rarely receives additional documentary evidence. Instead it relies on the written record of the proceedings that took place in the trial court. There is no jury, of course, and ordinarily the only time lawyers appear in the courtroom is to make oral arguments after the briefs have been submitted. The court limits the length of these arguments, often allowing as little

as an hour for both sides to make their respective pitches.

Some states have more than one tier of appellate courts. At some point, though, a court will be reached that represents the highest level of appeal possible in the state system. In most states this is called the *supreme court*, but—to compound the confusion—not in every state. Sometimes it's called the *court of appeal* or some similar thing. New York is perhaps the most confusing of all, calling its *trial court* the "supreme court". The point to remember is: don't jump to conclusions about the importance of some state court decision if you're not familiar with the court nomenclature for that state.

Sometimes the losing party will find that a state's highest court may not be the end of the road. The rules that govern whether a further appeal can be made to the federal courts are rather complex. In general, the litigant must show that a state court has deprived him or her of some right guaranteed by the federal government.

FEDERAL COURTS

The federal court system parallels that of the state courts. There's one refreshing difference—at least we know the names of the courts.

The trial court in the federal system is called the *federal district court*. There's at least one in

every state, and large states may have several, each serving a defined geographical district.

Above this, there are two appellate court levels. The first tier is comprised of the *courts of appeal*. Most of these courts supervise the district courts within defined areas of the country called *circuits*. For this reason, they are often referred to as *circuit courts*.

One circuit court important to the computer industry is not limited to any geographical area. This is the Court of Appeals for the Federal Circuit. It was created in 1982 by combining the former Court of Customs and Patent Appeals with the Court of Claims. This is the court that considers inventors' appeals from denials of

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

Subject: Extraordinarily powerful spreadsheet with extensive math and logic facilities.

Powerful spreadsheet specifically designed to take advantage of the UNIX operating system. Q-Calc uses termcap to support any terminal. Interactive prompts and help text make it very easy to use.

Features:

- Extensive math and logic facilities.
- Large model size.
- Allows sorting and searching.
- Interfaces with the UNIX environment and user programs via pipes, filters and subprocesses. Spreadsheet data can be processed interactively by UNIX programs, with output placed into the spreadsheet.
- Q-Calc command scripts supported.
- Uses termcap.
- Optional graphics for bar and pie charts. Several device drivers are included to support graphics terminals.
- Available for the VAX™, Sun™, Masscomp™, AT&T 3B Series™, Cyb™, Apple Lisa™, Perkin Elmer™, Plexus™, Gould™, Cadmus™, Integrated Solutions™, Pyramid™, Silicon Graphics™, Callan™, and many more.

Price:

	Binary
VAX, Perkin Elmer, Pyramid, AT&T 3B/20	\$2,500
(with graphics)	3,500
MC68000™	750
(with graphics)	995

Source Code available.

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

OEM terms available.
Mastercard/Visa accepted.

SPREADSHEET

Q-CALC

patent applications.

At the highest federal appellate level is the *Supreme Court of the United States*. Since in most instances the Supreme Court doesn't have to entertain an appeal unless it wishes to, cases in the federal court system generally never get further than the circuit court. This means that decisions of these courts carry a great deal of legal weight.

WHICH COURT HEARS YOUR CASE?

Some of the laws affecting the field of computing—copyright laws for instance—are entirely federal in origin. Lawsuits regarding copyrights therefore must be

When lawsuits involve important principles of computer law, it's often desirable to have them decided by the federal courts.

brought in the federal district court.

You might suppose, then, that a lawsuit over a matter involving

only state laws—interpretation of a contract, say, or protection of a trade secret—would necessarily have to be filed in a state court. But you'd be wrong. Disputes which arise under state law frequently end up in federal courts. The most common are lawsuits in which the parties reside in different states. (Remember that in the eyes of the law, a corporation is a person and is considered to have an identifiable residence.)

Odd as it may seem, the federal courts will apply and interpret *state* laws in these situations. You may be excused for asking, the laws of *which* state? While there's a complicated body of law designed to answer that question, many contracts don't leave it to chance. This is one of the reasons why, if you examine a software licensing agreement, you're likely to find that it says that its provisions are to be construed according to the laws of some specified state—New York, for instance (AT&T), or perhaps California (UC Berkeley).

WHAT APPELLATE COURTS DO—AND DON'T DO

When lawsuits involve important principles of computer law, it's often desirable to have them decided by the federal courts. For one thing, suits usually get decided sooner in the federal system. But more important, an appeal decided by a circuit court will be binding over a wider geographical area and will usually have more influence than a decision of even the highest state court.

In general, decisions of state trial courts are not published. Decisions of federal district courts are, and so are the decisions of both state and federal appellate courts. These published decisions, filling shelf after shelf in the stacks of law libraries, are the source to which lawyers look for answers to legal questions not

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.

CENTURY
SOFTWARE
We make it easy for you.

CALL OR WRITE FOR FREE PRODUCT CATALOG

9558 South Pinedale Circle
Sandy, Utah 84092
(801) 943-8386

CP/M is a registered TM of Digital Research

Circle No. 277 on Inquiry Card

specifically covered by federal and state laws.

The actual determination in a case takes up only a small portion of a typical published decision. The rest consists of an analysis of the facts of the lawsuit and, most importantly, of the legal principles that the court feels should be applied. It's in this portion—the *opinion* of the court—that we find its interpretation of existing laws and, sometimes, the creation of new law to fill a void.

In the process, the appellate court often concludes that the trial judge tried the case incorrectly. Does this mean that the appellate court then goes ahead and tries the case itself?

No indeed. The appellate court

merely sets aside the original decision and bounces the case back to the trial court, telling it to make a new decision following the principles just laid down by the appellate justices—and, if necessary, to try the case over again.

This is exactly what the circuit court did in *Apple Computer, Inc. vs. Franklin Computer Corporation*. Apple in effect had won a battle, and an important one, but it had not won the war. *Apple vs. Franklin* demonstrates why parties often agree to out-of-court settlements of lawsuits following an appellate decision. No one can be sure how the case will turn out once it's been retried. There have been many celebrated cases in which the party that won appel-

late skirmishes ultimately got dumped on retrial.

However the final decision turns out, it has no effect on the appellate court's published opinion. It remains as a pronouncement of legal principles to be applied in the future. *Apple vs. Franklin*, for example, made it clear that computer programs can be copyrighted . . . at least until Congress or the Supreme Court decides to get into the act.

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. ■

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

Subject: Full multi-user UNIX for the Lisa.

Apple Lisa UNIX, the UniPlus + Bell Labs UNIX System V, transforms your Apple Lisa into a low-cost, high performance multi-user desktop workstation.

Features:

- The full multi-user system includes powerful UNIX utilities, C compiler and development tools, text processing tools, along with vi, csh and termcap. Full system is priced at \$1495.
- Supports Apple 5 and 10-Mbyte drives. Increased disk space is available with hard drives which range from 16 to 92 Mbytes.

Optional UNIX Applications Available:

Unify® Multi-user relational database.
Lex™ Word Processing.
Q-Calc Spreadsheet.
UniPress EMACS™ multi-window text editor system.

Programming Languages Available:

SVS Fortran
SVS Pascal
SVS Basic +
SMC Basic 4
RM Cobol
Irvine ADA

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

Dealer terms and demonstration systems are available. Mastercard/Visa accepted.

MULTI-USER OPERATING SYSTEM

APPLE LISA™ UNIX

Lisa is a trademark of Apple Computer. UNIX is a trademark of AT&T Bell Laboratories. UniPress EMACS is a trademark of UniPress Software, Inc. Unify is a trademark of Unify Corp. Lex is a trademark of ACE Microsystems.

C ADVISOR

To goto or not to goto

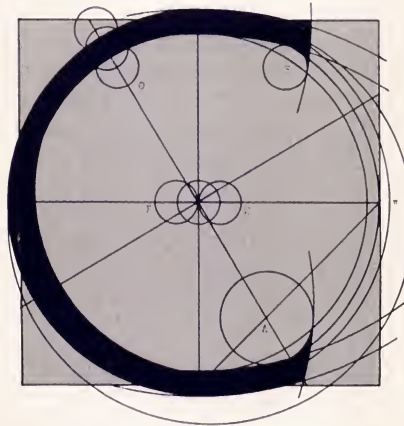
by Bill Tuthill

The 4.2BSD kernel has 1008 **goto** statements, for an average of one every 74 lines. The System V.2 VAX kernel has 176 **goto** statements, for an average of one every 154 lines. Why is this? Instructors of programming at the college level generally frown on **goto** statements as bad coding style; then again, few instructors write large working programs. Eminent computer scientists hold that **goto** statements are contrary to the tenets of structured programming; but again, their supporting examples are usually too short to be convincing.

The seminal article on this subject was Edsger Dijkstra's letter to the editor of *Communications of the ACM*, published in March, 1968, entitled "Go To Statement Considered Harmful". In this letter, Dijkstra wrote that "the **go to** statement should be abolished from all higher-level programming languages . . . everything [except] machine code . . . it is just too much an invitation to make a mess of one's program." This short letter inspired a voluminous outpouring of literature, with some writers arguing for, and others against, the construct.

Donald Knuth finally brought the controversy to an end with an article in the December, 1974 *Computing Surveys*, cleverly entitled "Structured Programming with GO TO Statements". Knuth showed examples of useful **goto** constructs, but concluded by saying "we should indeed abolish **go to** from the high-level language, [as soon as better] new language features . . . [become] available".

Some of the new language features Knuth talked about were available in Algol-68, and some were later incorporated into Bliss, the epitome of a language designed for optimizing compilers. Because



goto statements are not allowed, all Bliss procedures are logically reducible to flow graphs. For **goto**, Bliss substitutes the **leave** statement, which is similar in that it jumps to a label, but is restricted to the local control block. Like the Bourne shell, some dialects of Algol-68 provide **break-n** and **continue-n** statements to get out of nested loops.

If C provided multi-level **break** and **continue** statements, **goto** statements would not be necessary, except for handling error conditions. For example, the following code fragment:

could be replaced by this code:

```
i = 0;
top:
for ( ; i < 100; i++) {
    for (j = 0; j < 100; j++) {
        if (matrix[i][j] == -1)
            goto top;
    }
}
```

could be replaced by this code:

```
for (i = 0; i < 100; i++) {
    for (j = 0; j < 100; j++) {
        if (matrix[i][j] == -1)
            continue(2);
    }
}
```

Note that the **continue(2)** could be replaced by **break** and the program would behave the same way. Also, the following code fragment:

State of the art software and operating systems.

Tailor made consultant services.

Lachman Associates, Inc. is a custom systems software consulting firm... professionals, representing two hundred man-years of UNIX experience. LAI is a system in itself; each field consultant is linked to account and project managers within the organization.

The complete UNIX resource.

We provide complete UNIX development services for hire. Our expertise also spans all of the common PDP-11, VAX and IBM/370 operating systems.

LAI has been instrumental in a number of major UNIX ports, as well as numerous projects involving:

- high-reliability operating systems development
- a distributed transaction processing system
- development of a new UNIX-like operating system
- local area networks
- distributed file system research
- a UNIX front-end processor
- device drivers

- relational database systems
- compilers
- advanced debuggers
- a multi-processor implementation of UNIX
- data communications
- real-time systems
- processor architecture evaluation
- system performance measurements



- development of a heterogeneous networking system
- advanced graphics software
- product analysis

Our related services include market studies, customized technical training, technical documentation and software research and development.

Human engineering.

We tailor our skills and experience into an understanding of client requirements. With this knowledge, we tailor projects completely from providing contract programmers to supplement your existing staff to total project responsibility, including system specification, design, implementation, acceptance testing and training.



L A C H M A N

Lachman Associates, Inc.

Corporate Offices

645 Blackhawk Drive • Westmont, IL 60559

312 986 8840

Outside Illinois: 1-800-L•A•I•U•N•I•X

Chicago Denver New Jersey

*the UNIX people*TM

UNIX is a trademark of AT&T Bell Laboratories. PDP-11 and VAX are trademarks of Digital Equipment Corporation

SHACC UP WITH CONCENTRIC ASSOCIATES, INC.

Concentric Associates, Inc. announces a new software product:

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

Shacc'd code:

- Is faster than interpreted shell code.
- Is more resistant to piracy.
- Is more space and time efficient—you can make good use of shared text and the sticky bit.
- Will work properly in setuid files.

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

If your shell programming's not up to speed, call Concentric Associates. Also ask about our UNIX™ teaching and consulting services.

shacc

By Paul Ruel

Concentric Associates, Inc.

For further details, call or write:

Jim Butz/Concentric Associates, Inc./One Harmon Plaza/Secaucus, NJ 07094/201•866-2880

UNIX™ is a trademark of AT&T Bell Laboratories

The Profitable Way To Program In UNIX.™

UX-Basic+

Program in UX-Basic+ . . .

Before UX-Basic+ programming in UNIX meant programming in C. C gave you power and portability. Now . . . go beyond C to add productivity . . . and profit to UNIX.

Program in UX-Basic+ . . . to be powerful

UX-Basic+ provides high level functions, all with error checking syntax that access UNIX system calls and libraries including UNIX graphics and C-ISAM.™ This gives UX-Basic+ programmers the ability to tap the power of UNIX.

Program in UX-Basic+ . . . to be portable

Programs written in UX-Basic+ will execute unchanged on any UNIX machine and are completely device

independent. UX-Basic+ has been ported to all popular UNIX machines.

Program in UX-Basic+ . . . to be productive

The UX-Basic+ Development System provides an interpreter with full featured editor, development and debugging tools, a pseudo-code compiler and runtime module. UX-Basic+ includes C-ISAM, BCD arithmetic and the easily readable structured, modular code is compatible with OASIS™ Basic.

Program in UX-Basic+ . . . to be profitable

UX-Basic+ is the choice of leading UNIX software developers. Now . . . the profitable way is yours.

UX-Basic+ is available through licensed hardware manufacturers and selected distributors, or call:

UX Software, Inc.

10 St. Mary Street, Toronto, Canada M4Y 1P9
TWX: 610 491 2138 (416) 964-6909 TLX: 06-22492

Circle No. 273 on Inquiry Card

```

for (i = 0; i < 100; i++) {
    for (j = 0; j < 100; j++) {
        if (matrix[i][j] == -1)
            goto bottom;
    }
}
bottom:

```

could be replaced by this code:

```

for (i = 0; i < 100; i++) {
    for (j = 0; j < 100; j++) {
        if (matrix[i][j] == -1)
            break(2);
    }
}

```

Whether or not it is easier to read the **goto** versions or the **break** and **continue** versions is a matter of opinion. Proponents of **goto** maintain that it is hard to find the appropriate loop, while proponents of **break** and **continue** maintain that it is hard to spot the label. In fact, the **break** and **continue** versions are easier to optimize automatically, even if the incidence of multi-level loop escapes is low. Whether

or not most compilers would actually optimize such code is a moot point.

Error handling is one area where multi-level **break** and **continue** statements could not replace the **goto** statement. Roughly half of the **gotos** in UNIX source code have been inserted to clean up errors. Kernighan and Ritchie state that error handling is one good use for **goto** statements. This is usually coded as follows:

```

if (ferror(tfp)) {
    perror(tempfile);
    goto cleanup;
}
...
cleanup:
    unlink(tempfile);
    kill(child, 9);
    exit(status);
}

```

There are two alternate methods for dealing with this situation. The first is to create a **cleanup()** routine to perform the appropriate operations. However, this method involves subroutine call overhead, thus incurring both a speed and size penalty. A second method is to have an exception handler, where the routine sends itself a signal upon finding an error and then handles the cleanup without any outside help. Unfortunately, the UNIX signal mechanism is not robust enough to work reliably in exception handlers.

It is generally assumed that the best C programmers progress from applications programming to systems programming—that is, UNIX kernel hacking. Despite that opinion, UNIX applications software has a somewhat lower density of **goto** statements than does the 4.2BSD kernel. System V.2 utilities, for example, contain 1052 **goto** statements, for an average of one every 153 lines (compared with the BSD kernel's one in every 74 ratio).

There are two reasons for the use of this so-called *unstructured* construct in what is considered to be the best C software. First, the **goto** statement, if intelligently used, can result in programs that are faster and more compact than would be otherwise possible. Second, there are situations when **goto** statements are easier to read and understand than functionally equivalent structured code. Sam Lefler, one of the designers of 4.2BSD, has said that C just isn't expressive enough to avoid **gotos**.

Consider the sample program in Figure 1, which contains five **goto** statements. The labels are close to the **gotos**, and have mnemonic names, making the code fairly easy to read. If C had multi-level **break** and **continue** statements, all that would be necessary to transform the program into structured

CEEGEN-GKS GRAPHICS SOFTWARE in C for Unix

- Full Implementation of Level 2B GKS
- Outputs, Inputs, Segments, Metafile
- Full Simulation for Linetypes, Linewidths, Fill Areas, Hatching
- Circles and Arcs, Ellipses and Elliptic Arcs, Bezier Curves
- Ports for Version 7, System III, System V
- Terminal, Plotter, Graphics Printer, Digitizer Drivers Available
- End-User, OEM, Distributor Discounts Available

CeeGen Corporation
20 S. Santa Cruz Ave., Suite 102
Los Gatos, CA 95030

(408) 354-8841

TWX: 9103506750 CEEGEN LG

CEEGEN is a trademark of CeeGen Corporation
Unix is a trademark of Bell Laboratories

Circle No. 272 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

code would be the addition of a Boolean variable to keep track of whether the questions had been answered correctly. This would require an extra word of data space and at least two extra comparison operations. But as C exists today, transforming this into a program without **gotos** requires two extra Boolean variables, at least four extra initializations, and at least four extra comparisons.

Figure 2 shows the sample program after transformation into so-called *structured* code. Although it contains no **goto** statements, the second program is 14 lines longer, and is more difficult to read. A C compiler for the M68000 generates 211 lines of assembler for the program in Figure 2, as opposed to 187 lines for the program in Figure 1.

Since we don't have deeply nested loops in this example, the first program is not measurably faster than the second. But inside tight inner loops, a **goto** statement can save both space and time by avoiding the initialization and comparison of extraneous Boolean variables. Perhaps in an ideal language, **goto** statements could be abolished, but in C, they can make programs both more efficient and more legible. That's why there are so many of them in UNIX source code.

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

```
#include <stdio.h>
#include <ctype.h>
#define isokname(c) ((c) == ' ' || (c) == '-' || (c) == ',' || (c) == '.')

main()          /* getname - validate user's name and phone number */
{
    char s1[BUFSIZ/2], s2[BUFSIZ/2], *c;
    int length;

    name:
    fputs("Your name please: ", stdout);
    if (!fgets(s1, sizeof(s1), stdin))
        exit(1);
    if (*s1 == '\n')
        goto name;
    for (c = s1; *c != '\n'; c++) {
        if (!isalpha(*c) && !isokname(*c)) {
            fputs(" invalid characters\n", stdout);
            goto name;
        }
    }
    phone:
    fputs("Your phone number: ", stdout);
    if (!fgets(s2, sizeof(s2), stdin))
        exit(1);
    if (*s2 == '\n')
        goto phone;
    for (c = s2, length = 0; *c != '\n'; c++) {
        if (isdigit(*c))
            length++;
        if (!isdigit(*c) && *c != '-') {
            fputs(" invalid number (digits and -)\n", stdout);
            goto phone;
        }
    }
    if (length != 7 && length != 10) {
        fputs(" incorrect length (7 or 10)\n", stdout);
        goto phone;
    }
    printf("\nName:\t%s", s1);
    printf("Phone:\t%s\n", s2);
    exit(0);
}
```

Figure 1 — Sample code including **goto** statements.

```

#include <stdio.h>
#include <ctype.h>
#define isokname(c) ((c) == ' ' || (c) == '-' || (c) == ',' || (c) == '.')

main()          /* getname - validate user's name and phone number */
{
    char s1[BUFSIZ/2], s2[BUFSIZ/2], *c;
    int length, answered, invalid;

    answered = 0;
    while (!answered) {
        invalid = 0;
        fputs("Your name please: ", stdout);
        if (!fgets(s1, sizeof(s1), stdin))
            exit(1);
        if (*s1 == '\n')
            continue;
        for (c = s1; *c != '\n'; c++) {
            if (!isalpha(*c) && !isokname(*c)) {
                fputs(" invalid characters\n", stdout);
                invalid++;
                break;
            }
        }
        if (invalid)
            continue;
        answered++;
    }
    answered = 0;
    while (!answered) {
        invalid = 0;
        fputs("Your phone number: ", stdout);
        if (!fgets(s2, sizeof(s2), stdin))
            exit(1);
        if (*s2 == '\n')
            continue;
        for (c = s2, length = 0; *c != '\n'; c++) {
            if (isdigit(*c))
                length++;
            if (!isdigit(*c) && *c != '-') {
                fputs(" invalid (digits and -)\n", stdout);
                invalid++;
                break;
            }
        }
        if (invalid)
            continue;
        if (length != 7 && length != 10) {
            fputs(" incorrect length (7 or 10)\n", stdout);
            continue;
        }
        answered++;
    }
    printf("\nName:\t%s", s1);
    printf("Phone:\t%s\n", s2);
    exit(0);
}

```

Figure 2 — Sample code excluding *goto* statements.

THE ONE &

UNIX™ SYSTEM V. FROM AT&T. THE

There's only one UNIX System V. The one created and supported by AT&T. The one fast becoming a universal standard.

It's another reason why good business decisions are based on UNIX System V.

In more ways than one, it pays to make certain that the products you offer are based on UNIX System V. It is the stable foundation on which all of our future enhancements will be developed. It's here to stay because we pledge that our future software enhancements will be compatible with it.

Configure your systems now with products based on UNIX System V and you'll be at the forefront of its establishment as a worldwide standard. AT&T has expanded its role in markets in Europe, Japan, and the Far East to create opportunities for machine-independent, portable solutions to operating system needs. Around the world you can profit from our initiatives when you invest in UNIX System V solutions.

On the inside track

If you develop products based on UNIX System V, you'll be in

on the latest advances from the custodians of the UNIX System standard. A growing number of microprocessor ports to choose from. Expanded networking capabilities. A growing number of applications packages from software vendors. And, the power of the system that spans generations of machines as well as models and sizes—from micros to mainframes.

Service from the source

Along with its various technical and commercial advantages, UNIX System V offers something the other systems can't match—

SYSTEM V

ONLY

STANDARD BEARER.

the full backing of AT&T.

That includes worldwide support and training programs. Our commitment to work with computer manufacturers and resellers as well as user groups to develop standard products for the future. And for software vendors writing applications programs, a complete service and support program from AT&T.

The one and only from AT&T is UNIX System V—backed by AT&T service and support—designed to make UNIX System V the one for one and all.

For more information on UNIX System V from AT&T,

send in the coupon.

**UNIX System V. From AT&T.
From now on, consider it standard.**



Please send me "UNIX Software."

Mail to: AT&T, Box 967, Madison Square Station
New York, New York 10159

Name _____

Title _____

Department _____

Company _____

Address _____

City _____ State _____ Zip _____

Phone _____

UNIX™ System Licensee Yes No
 Don't Know



AT&T

Circle No. 270 on Inquiry Card

DEVIL'S ADVOCATE

Satire, spelling . . . more!

by Stan Kelly-Bootle

"May your Kids shun Satire," warned the delinquent Juvenal during Superbowl minus MXCC (the result of which, if memory serves, hinged on a last minute penalty served on the Christians for illegal motion and having only one player on the field). Today, Juvenal would be more adamant—urging you to avoid the Satiric Trade, the company of Satirists, and all who might be remotely suspected of reading Satire. Indeed, a Satirist's life these days is far from a happy one, especially for those seeking to peek 'n' poke a little fun at the computer whirligig.

The grotesque sequence of improbable events that we begrudgingly call "Reality" is unfailingly ahead of our wildest scenarios by several megaparsecs. A serious Wall Street Journal hightech editorial can now out-belly-laugh the most bizarre of Douglas Adams' galactic creations. And each Silicon Valley Wunderkind biography seems to come straight out of *Son of Monty Python*.

It's expected that we'll weep at the picture of two guys inventing things in a *garage!* What we're not told, though, is that the inventors first had to park both Bentleys in the covered driveway and then push the Bugatti Royale into a corner. Come on! Archimedes used to *dream* about owning a



car, and Edison's best work was done in a dark, damp kiosk.

On the flip side of the floppy, readers nowadays have been rendered incapable of distinguishing fact from fiction (witness the sales of nobby home computers, not to mention books on AI for a 2K Timex-Sinclair!).

One micro-marketing journal has found it necessary to head up its regular jokey column with the type of caveat they flash by you at the cinema—"Any resemblance to creatures or packages alive or dead, great or small, is entirely coincidental. The Adam Osborne mentioned in this piece is *not* the famous entrepreneur; IBM does *not* really stand for Irish Business Machines, and, notwithstanding our statement to the contrary, Chopin *never* composed in the Polish Notation."

I have encountered this blurring of Truth's coastline on several occasions. A piece I wrote in the 1960s on Shakespeare's computer, MUSE (Most Unusual Shakespearean Engine), prompted many serious requests for price and delivery information. One scholar confirmed my tongue-in-cheek hypothesis that Richard Burbage (Bill the Quill's leading actor-manager) was an ancestor of Charles Babbage: "... the most cursory inspection of Elizabethan documents proves that MUSE's *Spelchek* was grossly inadequate!"

The subject of spelling reminds me that, in spite of the miracles of modern WP and WWB, the general literary standard of our industry is forever declining. Orthographic solecisms on the printed page are too frequent and invoke but a brief howl of despair (my recent favorite is "rudimentary", simple because it has a delicious Joycean rudimentary recursiveness—also it occurred thrice in a publisher's handout), but when spelling errors jump on your screen from the bowels of the operating system, your confidence in the kernel-grinders is severely diluted. I suffered a "Non-existent File" message under AMOS for many months until Bob Toxen told me of two ways to avoid the agony. One method was never to

T A N G O TM

Use Tango to:

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

Buy Tango for:

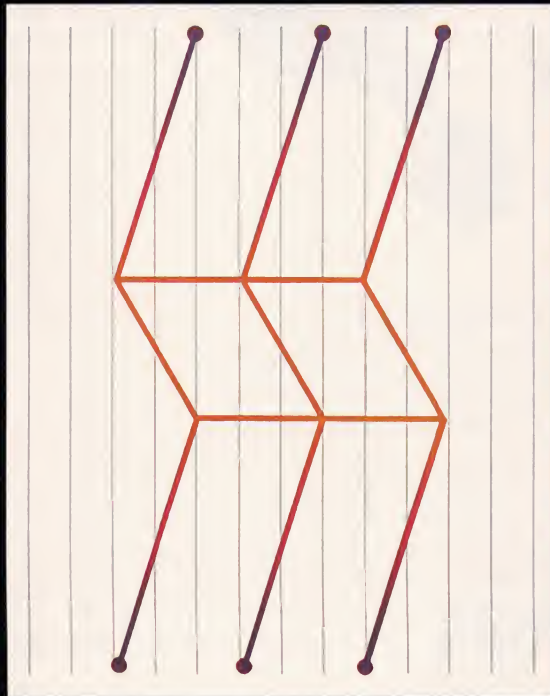
- Execution of DOS programs on the PC under UNIX control.
- Simple elegant file transfer under error correcting protocol.
- DEC, IBM, and Tektronix (graphics) terminal emulation.

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

COSI

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

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



The PC-to-UNIXTM Connection

invite the message by using only "existant" filenames; the other, in the absence of OS source, was to locate, dump, and DDT the offending block.

Of course, one could argue that the skills of programming and spelling have little in common. The latter accomplishment merely indicates a dull, slavish devotion to arbitrary rules, whereas programming . . . *gotcha!* What really counts in programming is consistency (or even "consistency?")—if my variable is declared as **amount recievable**, then that is exactly what must be added to get the total "recievable".

Returning to the subject of satire vs. reality, a friend of mine,

**When spelling errors
jump on your screen
from the bowels of the
operating system, your
confidence in the
kernel-grinders is
severely diluted.**

toying with the old probabilistic fantasy of monkeys typing Ham-

let, pondered aloud at a party on how many drunks at how many CRTs could produce UNIX System VI ahead of schedule. A nearby AT&T employee pulled him aside and warned him not to discuss company secrets in public.

Shortly after publishing a spoof software specification for IMP (Integrated Morticians Package), which offered Pre and Post-Need Casket Accounting, Armband Inventory Control, Bitmapped Epigraphics, and Nondestructive Crematorial Temperature Gradients, I received a lively call from the Colma Garden of Infinite Repose (an Equal Opportunity Corporation) asking for a demo. Yesterday I read that AlphaMicro of Irvine, CA is now offering a package for Mortuaries. Aha, but can it link into the Hertz Rent-a-Hearse Network like my fictitious IMF could? Does it have Online Floral Tributation? Does it offer **more?**

My IMP list definitely offered . . . **more!** A list, however long and comprehensive, that does not terminate with **more** is clearly suspect, both semantically and syntactically. A null list, of course, contains just the one element, **more!**

Consider the many futile attempts to improve on the King James' Version (KJV) of the Bible. All that is needed, I claim, is a dash modern list-processing:

"In the beginning God created the Heaven and the Earth, . . . more."

"And Seth begat Enosh, . . . more."

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. ■

**LOOKING
FOR
THE
PROMISE
OF
UNIX™**



YOU JUST FOUND IT.

UNIX promises to let all sorts of people do all kinds of things quickly, flexibly, and efficiently. For three years BASIS has been delivering what Unix promises to individuals, small businesses, large corporations, government agencies, universities, and resellers all over the nation. We sell:

- complete Unix computer systems
- carefully selected software
- expert consulting
- a full range of timesharing services

Our reputation for providing outstanding customer support is unparalleled. Call or write for more information.

BASIS

SPECIALISTS IN UNIX COMPUTING

1700 Shattuck Avenue Berkeley, California 94709 415 841 1800

UNIX is a trademark of AT&T Bell Laboratories

Circle No. 268 on Inquiry Card

SERIX

ANNOUNCING
SYSTEM V
UNIX™

...puts your IBM Series/1® ahead of the pack!

SERIX is the high performance CMI version of AT&T's UNIX™ System V operating system with Berkeley 4.1 enhancements ported to the IBM Series/1 minicomputer.

SERIX transforms your Series/1 into an even more powerful, flexible, and convenient processor for general data processing, office automation, communications, and process control. Its advantages are outstanding:

Reduced software costs

Long term growth path

- Software is highly portable
- Provides access to a large, growing software base

More power from the Series/1

- Optimizing C compiler uses native code features
- All code reentrant
- Dynamic memory allocation without fixed partitions

Increased programmer productivity

- Large set of utilities
- Hierarchical file structure
- Pipes, forks, semaphores, and shared data segments

Other CMI Series/1 software

- RM/COBOL™
- UNIFY™ database management system
- ViewComp™ spreadsheet
- vi visual editor
- EDX™- to -SERIX™ conversion kit

CMI Corporation is a Master Value-added Remarketer of IBM Series/1 equipment. Leasing and other financial arrangements are available. Contact us for further information.

Photographer - Michael Zagaris • UNIX is a trademark of Bell Laboratories
• SERIX is a trademark of CMI Corporation • SERIX was developed exclusively for CMI by COSI. • IBM, Series/1, and EDX are trademarks of International Business Machines Corporation • UNIFY is a trademark of North American Technology, Inc. • RM/COBOL is a trademark of Ryan-McFarland Corporation
• ViewComp is a trademark of Unicorp Software, Inc.

CMI 

A Torchmark Company

CMI Corporation
SERIX Marketing
2600 Telegraph
Bloomfield Hills, MI 48303-2026
(313) 456-0000

TWX: 810-232-1667
Telex: 499-4100 ANS: CMI CORP. BDHS

Member CDLA Member ASCD



HEWLETT PACKARD Integral Personal Computer



Reset Break

Stop

Menu

Help

System

Almost priceless.

Hewlett-Packard presents
the first 32-bit UNIX™ system
under \$5,000.

Introducing the Integral Personal Computer.

It's the newest member of the HP 9000 family of HP-UX workstations. And it delivers the kind of power and flexibility the name implies.

With a 16/32-bit MC68000. A graphics co-processor. And 800KB of standard memory, expandable to 5.8MB.

Its UNIX kernel (HP-UX/RO) is built into ROM. Which means the Integral PC can run most applications *without* an expensive hard disc. Which means more

people can afford to run your UNIX software than ever before. But not only can they afford to run it. They can learn to *use* it in

minutes. Because the Integral PC is one UNIX system that's easy to use. Its see-and-select user interface, built-in window manager and optional mouse mean even novices can tap the power of your software. No one has to learn a bunch of cryptic commands or go through a lot of expensive training. There's even an entry-level, self-paced tutor disc with every machine.

What's more, the Integral PC goes where you need it. The complete system — including our revolutionary ThinkJet printer — packs into a single 25-pound package that takes up less than a cubic foot.

So get started porting your software now. HP-UX is an enhanced version of UNIX System III, including Berkeley 4.2 BSD features.

For the name of the authorized HP dealer or HP sales office nearest you, call toll-free 1-800-FOR-HPPC.

The value of your investment in UNIX software just went up. Because the price of a complete UNIX system just came down.

The Integral Personal Computer. Just \$4,995*



Circle No. 266 on Inquiry Card

THE UNIX GLOSSARY

Performance parlance

by Steve Rosenthal

Note: only those aspects of the terms concerning system performance are included in this listing.

access time — the time it takes to get data after a request has been made. For internal memory, this is usually a small fraction of a second (perhaps a millionth), while for external memories like tape drives, it can be many full seconds. For semiconductor memory, "access time" technically defines the span between when a valid address is sent out over the bus to the memory chips and when valid data is sent back.

availability — the proportion of time a system or specific piece of equipment is ready and working, compared to the total time it is expected to be. It is reasonable to expect modern LSI-based and Winchester disk-based desktop computers to have availabilities in excess of 99 percent.

benchmark — 1) a standard or reference test program or problem used to compare various systems, programs, services, or machines. 2) as a verb, "benchmark" refers to efforts to establish ratings of one or more systems, programs, or services using a standard input problem.

Deciding which benchmark to use is a matter of judgement, and competing firms often see merit in



specific measurement tools that unsurprisingly give them an edge.

black box testing — tests used to check a program or system based solely on specified inputs and outputs, without taking into account any knowledge of the inner workings of the part being tested. Black box testing can avoid possible omissions in testing that might be made because of a reliance on expected behavior, but of necessity it is never complete (except in trivial cases) due to the huge number of possible input data combinations.

cache — a faster memory where parts of the information in main (slower) memory or disk are copied. Information more likely to be read or changed is placed in the cache, where it can be accessed more quickly. Using a cache can

significantly speed processing or disk access.

Operations that keep the right information in the cache can be performed either by hardware or software. Typically, to simplify operations, the memory or disk space is divided into blocks that are swapped in and out of cache as needed. When the cache is filled, the most common strategy is to select the least-recently used (LRU) block, write it back to its original location, and use the cache space it occupied for new data.

context switch — the act of switching from executing one process to executing another. On most machines, this requires saving state information (the various registers, pointers, and flags used by the process), flushing out certain buffers, and loading in the information needed by the new process. Usually, the data for several processes is in memory at the same time and the CPU is switched rapidly among them. Fast context switching is particularly important in multitasking systems such as UNIX.

data flow — a method of organizing how a computer reads and processes data. In a "data flow" computer architecture, the flow of work is organized around the data, instead of the traditional control-flow method where flow

SIR/DBMS bridges the UNIX portability gap

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

SPECIAL FEATURES

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

UNMATCHED PORTABILITY

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

SIR, Inc.

SIR, Inc. has pioneered the development of database software for INFORMATION ANALYSTS since 1976.

SIR/DBMS has been used to develop thousands of applications at R&D divisions of Fortune 500 companies, research institutions, government agencies, and universities across 25 countries. A full range of technical support and training services is available to you.

Call or write:
Director of Sales, SIR, Inc.
5215 Old Orchard Road
Skokie, IL 60077

312-470-9770

SIR/DBMS
The Intelligent Relational System™



depends on instructions. In the data flow method, data is split into units needed for processing, and, when possible, the processing of each unit takes place independently in parallel. Each step is

done as soon as its input data is available instead of being synchronized with a master program counter.

There have been attempts to combine data flow methods with-

in an overall UNIX or UNIX-like framework, but none have caught on so far.

FLOPS — an abbreviation for “floating point operations per second”, a measure of the speed of a machine performing arithmetic calculations. However, since mathematical calculations are also used extensively for computing indexes and for probabilistic functions used in Artificial Intelligence and text parsing, this measure may also be somewhat indicative of machine performance on non-arithmetic processes as well. A typical value for a UNIX-class machine would likely be in the kiloflops or megaflops range.

Gantt chart — a diagram showing when various parts of the system are active. It is usually graphed with time on the horizontal axis describing a set of horizontal line segments stacked one on top of each other that represent each of the activities of the system.

hardware monitor — an external electronic test device that’s hooked up to a computer system to make timing measurements or performance checks. Hardware monitors are usually more expensive and more difficult to connect to computer systems than are software versions, but they don’t slow or change the operation of the system being monitored.

histogram — a chart that shows the time (in total intervals or percentages) that the system spends in specified modes or program sections. Histograms are based on measurements made by hardware or software monitors for larger systems, and on development systems or logic analyzer measurements for smaller machines.

hit ratio — the percentage of data accesses for which the needed item is found already to be in cache, thus avoiding a reference

EASIER THAN 1-2-3...

BUT DESIGNED FOR LARGER SYSTEMS



It’s simple, C-CALC from DSD Corporation is more flexible, has more functions, and is easier to use than the best selling spreadsheet. We made it that way for a very simple reason, you’ll get more work done and make better decisions in less time. That’s what makes you successful whether you are planning for the future, forecasting trends, or analyzing profits.

The most popular spreadsheets require a great deal of time to get up and running. When we created C-CALC we kept in mind that time is your most important resource. Our On-Line Help facilities, prompts and menus allow even someone with minimal experience to see meaningful results in very little time. Our built-in training procedures let you pace your own learning with tutorial topics that range from basic to advanced. As you become more experienced, C-CALC allows you to bypass prompts and menus to save even more time.

So call DSD Corporation at (206) 822-2252. C-CALC is currently available for: UNIX, VMS, RSTS, RSX, IAS, P/OS, AOS, AOS/VS (Data General), IBM CSOS.

C-CALC is a registered trademark of DSD Corporation. UNIX is a registered trademark of Bell Labs. P/OS, RSTS and RSX are registered trademarks of Digital Equipment Corporation. AOS and AOS/VS are registered trademarks of Data General Corporation.




P.O. BOX 2669
KIRKLAND, WA 98033-0712

 EFFECTIVE SOFTWARE FOR BUSINESS

Circle No. 264 on Inquiry Card

The Sperry Challenge in Salt Lake City



Delivering innovative systems in an exciting and competitive marketplace is a formidable challenge—one that we accept.

At work within the Micro Products Division and Sperry Network Systems groups, we'll surprise you with uncommon opportunities for career growth in state-of-the-art microprocessor and network communications technology.

We are seeking individuals with expertise in the following areas: UNIX Software programming, Systems design engineering, systems programming, SNA, DCA, OSI communications architectures, LAN technology communication software engineering, and many others.

At Sperry, you'll work in a business environment which encourages your participation as you share responsibility with other team members, for developing new generations of micro-based products and data communications systems.

This is the Sperry challenge and we invite your participation.

And, nowhere in America will you find better skiing than in Utah—powder snow, blue skies and sunshine less than 40 minutes away from work.

If you are interested in the Sperry challenge, please send your resume to:

SPERRY, Employment Dept., Attn: Dept. 1J
322 N. Sperry Way, Salt Lake City, Utah 84116



to main memory or disk. Hit ratios measure the effectiveness of cache. They can be improved with bigger caches or better algorithms for placing items into cache.

interleave — 1) as applied to memory, files, or other system resources, "interleave" describes a way of increasing concurrent use of a resource by splitting it into sub-units unlikely to be accessed simultaneously by multiple processes. This splitting is usually done at a fine level, much like dealing out a deck of cards. 2) on a disk drive, instead of recording data on adjacent sectors, "interleaving" the data means writing it with a specified number of sectors inserted between each chunk. This procedure, also called

"skewing", gives the system extra time to process each sector. Interleaving can be done at the physical level (in the disk controller), or at the logical level (in the file system handler), or both.

load — refers to the total volume of work the system is asked to perform at any given time. Statistics programs will often print a "load average".

memory bandwidth — the rate at which data can be moved in and out of memory. It is calculated by multiplying the data path width by the number of memory cycles possible per second. Typically, values are expressed in megabits or megabytes per second.

MIPS — an abbreviation for "million instructions per sec-

ond", a value representing how many million machine operations a system can process in a second. Because this value changes with the instruction mix, it is best compared using a specified benchmark or load. Because the instruction sets of different machine architectures provide varying amounts of processing per instruction, MIPS can reasonably be compared only among machines of the same architecture.

overlapped operations — two or more operations that take place at the same time. Many systems have various co-processors and I/O systems that can operate simultaneously.

parallel processing — organizing a computer system to allow



Introducing the **L5** supermicro

A breakthrough in Price, Performance and Packaging

The new L5 proves that good things come in small packages! Measuring a compact 3.75"H x 17.5"W x 21"D, the L5 is small enough to fit on a desk-top or a laboratory workbench. Yet it's large enough to handle up to 32 users and, in some applications, outperform a VAX system. The L5 is small in price, too. A mid-range system can cost less than \$1,000 per user!

Find out how the new L5 offers the unique solution to price, performance & placement challenges. Call General Communications today!

L5 Features

- KD111 Processor, floating point, 8K cache
- .5Mb to 32Mb memory
- 4 to 32 users
- 20Mb to 2.5Gb external storage
- UNIX System V Fast Kernel or Real Time Kernel
- Runs RT11, RSX, and TSX
- Includes several utilities packages, necessary cabling, complete documentation and tutorials

Aggressive Dealer/OEM discounts available



General Communications Corporation

"Where lucid communications, technical excellence and common sense meet."

1 Main Street, Suite 502, Eatontown, NJ 07724 (201) 542-6560

*UNIX is a trademark of AT&T-Bell Laboratories
VAX, RT11, & RSX are trademarks of Digital Equipment Corporation
TSX is a trademark of S&H Computer

Circle No. 263 on Inquiry Card

HOW TO GET THE 'MOST' OUT OF UNIX* SYSTEMS



UNIX* SYSTEMS EXPO/85

You'll find out how to take full advantage of emerging computer technologies for your business, professional, and engineering/scientific applications at the **UNIX Systems Expo/85-Spring**. Here, you'll find the hardware, software and services you need to:

- **Link PCs to mainframes.**
- **Set up multi-user, multi-tasking systems.**
- **Create networking systems.**

This tightly-focused computer event is also especially appealing to **value-added resellers of UNIX-based systems and software developers**. It provides a maximum information exchange between sellers and buyers of UNIX systems.

Aisle upon aisle of the leading and most innovative computer hardware and software companies will be on hand to display and demonstrate state-of-the-art UNIX products.

Rounding out this timely UNIX Systems Expo will be a comprehensive conference program of more than 40 user/marketing oriented sessions. Authorities from the industry as well as corporate users of UNIX systems will lead these sessions.

Attend UNIX Systems Expo/85! You'll get the inside story from computer professionals who are building, installing, distributing and marketing UNIX systems.

You'll find it all at:
UNIX Systems Expo/85
San Francisco
Moscone Center
April 24-26, 1985

Call or write for preregistration material prior to April 1st

Exclusive production of
Computer Faire, Inc./
a Prentice-Hall Company

181 Wells Avenue
Newton, MA 02159
617/965-8350

611 Veterans Boulevard
Redwood City, CA 94063
415/364-4294

*UNIX is a trademark of AT&T Bell Laboratories

UNIX*
SYSTEMS
EXPO/85
Spring

Please send me information on:

Exhibiting Attending

Name _____

Title _____

Company _____

Address _____

City/State/Zip _____

Phone _____

(UR)

Please send to: Computer Faire, Inc. 181 Wells Avenue Newton, MA 02159

two or more processes to execute simultaneously using different resources (as opposed to the more traditional sequential multitasking system, which only *seems* to execute processes simultaneously when it switches quickly from one to the other).

prefetch — to move the next sequential instructions or data into a queue or cache from main memory before they are needed. Because in most cases the next requested instruction or data item immediately follows the last, this arrangement speeds execution. Systems that use this method to store up several items before use are said to be "pipelined". Prefetching and pipelining can result in slower execution if references are scattered. See *cache*.

reliability — the measure of probability that a system will be working when expected, expressed as a percentage, decimal fraction, or mean time between failure (MTBF). Reliability does not take into account scheduled maintenance, so actual availability is often lower than reliability figures would suggest.


response time — the interval between when a user completes an instruction or command (most often by pressing the RETURN key) and when the system finishes outputting the result. Response time depends both on loading and the type of command being executed.

rotational latency — on a disk drive (or other rotating memory

such as a drum), "rotational latency" refers to the time between when data is available for writing or for reading and when the right spot on the disk comes around. On the average, this takes half a disk rotation, so the faster the disk turns, the shorter the average rotational latency will be. Worst case rotational latency, of course, is the period of one complete revolution.

thrashing — describes a condition in which a system spends most of its time either switching contexts or swapping processes in and out from memory to disk instead of doing useful work. This is caused by poorly designed scheduling algorithms that become unstable when too much load is applied to the system.

FOR EVERYTHING UNIX*



is the only name you need to know.

The multi-disciplined staff at Daystrom — with backgrounds in academics, industry, research, finance, accounting and information management — can help with virtually any aspect of your computer-related requirements.


For example, you can take advantage of their consulting expertise to do a thorough **needs evaluation** of your overall operation. . . or any segment of it.

In **software**, you have a variety of standard programs available such as system administration, application generating, relational data base management and accounting systems, or, if necessary, have us write one to handle your special need.

And, when it comes to **hardware**, there's the "M-Family" — with the best power/performance ratio at least cost in the industry. Choose from a variety of components to custom configure a system to meet your exact specifications.

Contact Daystrom Systems today and find out for yourself how much help they can really be.

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



SYSTEMS INC.

11206 Euclid Avenue • Cleveland, Ohio 44106 • 216-421-8737

Making computers work the way you always thought they should.™

Daystrom is a trademark of Daystrom Systems, Inc

Q-CALC

A superior spreadsheet on UNIX*

As powerful as Lotus 1-2-3*

- large spreadsheet
- many business functions
- complete GRAPHICS package
- translates 1-2-3 models into Q-CALC
- already ported to: VAX, Callan, Fortune, Nixdorf, Cyb, Plexus, Codata, Cadmus, Masscomp, SUN, etc.

Available since Jan. '84

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 Development Corp. UNIX is a trademark of Bell Labs.

Circle No. 261 on Inquiry Card

Circle No. 259 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



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

Take the **RELIABLE**
Approach!

MCBA™ Software *Now* Available
on Microcomputers

- NCR Tower 1632
- Codata 3300
- Plexus P/(all)
- General Ledger
- Payroll
- Receivables
- Payables
- Order Processing
- Purchase Orders
- Inventory
- Bill of Materials

For Business Software, call us

 **RELIABLE**
DATA SYSTEMS

Full Function Full Feature
Fully Integrated

- RM Cobol
- Source Code Available to Dealers
- Multi-User and Shared Files
- Multi-Company Data Bases
- Many run standalone
- Add others as needed
- Field Proven
- Full Documentation

900 San Antonio Road
Los Altos, CA 94022 (415) 949-3600
Dealer Inquiries Welcome

MCBA is a registered trademark of
Mini Computer Business Applications, Inc.

Circle No. 258 on Inquiry Card

POSITIONS FOR

UNIX™ & 'C'
INSTRUCTORS

UNIX is a trademark of Bell Laboratories

Full and part-time positions available throughout the U.S. Must have taught and worked with UNIX. Communications skills critical. Willingness to travel necessary; however, relocation is not required. Please send resume, including salary history, to:

Carrel Schulke

COMPUTER TECHNOLOGY GROUP

The UNIX Training Company
310 S. Michigan Ave., Chicago, IL 60604
(312) 987-4084

**COMPUTER
TECHNOLOGY
GROUP**

Telemedia, Inc.

Circle No. 257 on Inquiry Card

transfer rate — the rate at which information is passed between parts of a system. Rates are usually given in bits per second or bytes per second. Typical rates are anything from 10 cps (characters per second) for a slow printer to 20 Mbs (20 million bits per second) for a hard disk drive.

tune — the verb describing optimizations of the operation of a UNIX system or a program on a particular type of machine or configuration. Tuning may involve changing register-assignment strategy, changing the way space is allocated by memory management routines, altering block sizes for I/O operations, and so forth. Because tuning involves heuristics more than guaranteed optimal procedures, it is more an art than a science.

utilization — the amount (or ratio) of time that a given system resource is in use. Ideally, if resources exactly match a given load, the processor, I/O, and memory will all have an equally high utilization factor.

Von Neumann architecture — the standard way present-day computers are designed, with instructions and data both in a single main memory, and a flow of control from one instruction to the next. UNIX processing is implicitly based on a Von Neumann model.

Whetstones — a standard benchmark program that solves equations for a Whetstone bridge, involving a number of floating point additions, multiplications, and divisions. This benchmark is particularly useful for scientific machines and performance is often expressed in "Whetstones per second".

white box testing — tests for checking a program or system based on knowledge of the internal structure and action. White box testing can be more efficient

than black box methods (where nothing is assumed about the internals of a system), but it can miss whole classes of mistakes if the testing procedure makes erroneous assumptions about the functioning of the system.

working set — on a virtual memory computer system (one which organizes its address space in pages brought in from disk to main memory as needed), the "working set" refers to the collection of pages needed to execute a process at any instant. If the full working set can be kept in main memory, the process will run faster than when pages must be swapped. Paging schedulers are complicated by the fact that a program's working set may change as the program executes.

writeback cache — a faster memory used to store data on its way to or from main memory that does not write data out to main memory until the cache is full or ordered to perform an update. Writeback cache can be faster than the alternative writethrough method, but requires extra steps to make sure that the main memory always receives all updates made to the cache.

writethrough cache — a fast memory that retains an extra copy of any data recently read from or written to main memory. Although the data is available more quickly from cache memory, main memory is updated fractions of a second afterwards, ensuring that the main storage copy of data is current to within a few memory cycles of the cache.

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

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

PERFORMANCE TRADEOFFS

Continued from Page 58

would have to be moved over the network (at least in the case of this example).

Sending high level requests over the network allows the server process to maximize local buffer pooling, and eliminates much of the slow access to the disk. There is a tradeoff here, however, as the CPU on the server could become "CPU-bound" instead of disk-bound, and the speed of the processor in the server could become the limiting factor in overall network performance.

CONCLUSION

UNIX began its commercial life with a very elegant architecture. With the addition of record locking, shared memory, and perhaps someday, shared libraries, the opportunities for application builders to optimize their programs will increase. The multitasking architecture of UNIX allows application builders to achieve an elegant economy, but discourages or disallows single-user style techniques, such as overlays.

Since database architectures are the ones most intimately involved with operating system internals, and are the most often used and most subtle of all the application code modules, it would seem that application builders would stand to benefit if UNIX hardware manufacturers were more aggressive with their database software offerings.

The application builders need "built-in" or readily available database tools that have most or all of these tradeoffs pre-analyzed and set up for them with as high a level of interface as can be provided. When multiuser and networked database software is considered as much a part of UNIX as **grep**, UNIX will be assured of its place as the standard operating system for multiuser data processing equipment.

Roger J. Sippl is the President and co-founder of Relational Database Systems, Inc., located in Palo Alto, CA. Formed five years ago,

RDS is a leading supplier of database software for UNIX. Mr. Sippl has served as a board member of /usr/group and is considered an authority on the UNIX marketplace. ■

YOU can print your
TROFF 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

424 Palmetto, Sunnyvale CA 94086 (408) 746-3754

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

Circle No. 80 on Inquiry Card

UNICOMP Technical Type

Typesetting Service

- UNIX*, Troff, Wizard
- Compugraphic 8400 Typesetter
- Quality appearance of books, proceedings, newsletters.
- Specializing in technical documents.
- Documents accepted via magnetic tape, phone lines, or paper.
- For information, samples, or estimates, call

505/662-EDIT (3348)

1580 Camino Redondo
Los Alamos, NM 87544

*Unix is a trademark of Bell Laboratories

Circle No. 256 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. 255 on Inquiry Card



OASIS™ and UX-BASIC™ PROGRAMMERS

See clearly and concisely the proper order for every valid choice within each BASIC command and statement using our

SYNTAX CHARTS

Fifty charts show the precise syntax for every major language element.

Only \$15 postpaid. Satisfaction guaranteed. CA residents must add tax. Send check or MO to:

BEAR SYSTEMS GROUP
55 Sutter St. Suite 279
San Francisco, CA 94104

415 = 824-3332

™ OASIS is a Trademark of Phase One Systems, Inc.
™ UX-BASIC is a Trademark of UX Software, Inc.

Circle No. 254 on Inquiry Card

GO FORTH, UNIX! ... with u4th UNIX/XENIX/FORTH DEVELOPERS

Are you interested in improving your software productivity? Do you have a **UNIX** system or engineering work station? Now you can realize the exceptional capability of moving your Forth applications to the world of **UNIX** and **XENIX**. Experience the productivity enhancement of an interactive programming environment, and still code in C when necessary.

UBIQUITOUS SYSTEMS announces **u4th**, the first Forth completely tailored for **UNIX**. **u4th** is a fast direct-threaded Forth written in portable C, yet capable of execution speeds comparable to many assembler Forths. Great for **AI** research and delivery.

Some of its features are:

- Access to **UNIX** utilities and calls
- Ability to incorporate C primitives
- Object-Oriented Forth included!

Binary license: Xenix \$395.00

Plexus \$895.00. OEM's: Special terms

UBIQUITOUS SYSTEMS

13333 Bel-Red Road N.E. Bellevue, WA 98005

(206) 641-8030

9 00 a.m. - noon weekdays

UNIX™, AT&T

XENIX™, MICROSOFT

Circle No. 253 on Inquiry Card

Continued from Page 24

ing various *benchmark* programs that measure some of the more specific behaviors or actions of the system in isolation, often using the computer's own timing tools (in the case of UNIX systems, the **time** command) to record the results. These tests record the time taken to, say, compute some prime numbers, read a disk file, or run some number of copies of a program.

The task then becomes one of trying to correlate test results with expected performance goals (that is, to complete a pattern of application-specific tasks within a specified time while respecting response time constraints). The difficulty comes in estimating the degree to which information is lost by not modeling the real-time interactions of various tasks—not only the concurrency, but the staggered sequences of initiations typical of human users. It's as if runners were tested by sending them down a narrow track one at a time—or all at the same time. Neither situation is an accurate reproduction of race conditions.

Conventional benchmarks may make some systems look too good, while making some other systems look too bad. If one is trying to obtain the best cost/performance ratio, either direction creates problems.

But of all the numerical information available on which to base the purchase of a general-purpose system, possibly the worst is some overall abstract unit of action such as MIPS (millions of instructions per second). Consider two similar runners in a 42 km marathon: the one who takes more steps per minute (an action) may or may not win the race (the objective). It depends on how far each step carries the runner toward the finish line. In a like vein, computer "A" might execute twice as many instructions per

second as computer "B", but each instruction of "B" might be more than twice as effective in achieving the desired result (such as getting this article formatted for typesetting on time!). Measures such as MIPS are valid only when comparing models of machines within a single computer architecture, such as the DEC VAX series, with a test workload that accurately reflects the mix and usage of instructions in the target applications environment.

Even then, "MIPS" is only a measure of raw CPU speed. Disk access time is often far more important.

CLOSING CEREMONIES

We have looked briefly at a definition of performance and some of the approaches taken to measure it. As usual, the more closely one is willing (or able) to look at the issue, the more likely a satisfactory outcome will result. Remember that when you're buying machines, you have to live with the winner, so it's best to screen the vendors' claims and use your own needs as your guide. Not every user will have the resources to construct a complete and accurate model of his or her computing environment, but even a rough "scorecard" of requirements (especially if written down *before* the competition begins) can help one judge which benchmark numbers will be helpful and which ones may simply be misleading.

Rob Warnock is an independent computer architecture consultant with nearly 20 years of experience in data communications hardware and real-time and timesharing operating systems. He has worked for Digital Communications Associates, Xerox XTEN, and was the system architect behind the Fortune Systems 32:16. Mr. Warnock currently resides in Foster City, CA. ■

Circle No. 252 on Inquiry Card ▶



PROBLEM DIAGNOSIS

Continued from Page 36

system needs to be reorganized.

Solution: Check for balanced disk usage. If two disks run off the same controller and you are only

swapping on one, typically with a lot of memory available, balance the swapping between the two disks.

Even better, use two smart controllers. That will make it easier

to achieve a balance between the two drives and will minimize movement of the disk arm. Arrange it so that swaps are handled by one and I/O is handled by the other.

One of the most interesting improvements you can make—after looking at what programs are used most—involves moving heavily used programs to different disks. One way to do this is to rearrange the partitions that divide logical areas on the disk. For example, since the C compiler, **nroff(1)**, and **troff(1)** make heavy use of `/tmp`, the UNIX temporary storage directory, you can obtain dramatic speed improvements by placing `/tmp` on a small but quick disk.

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, 424 Palmetto, Sunnyvale CA 94086
(408) 746-3754**

This ad was set using Xroff on a Xerox 2700 laserprinter **Circle No. 80 on Inquiry Card**

Case History #5: The case of the swamp disk.

Symptoms: Performance on a multidisk system slowed radically. A bit of investigation showed that high seek and interrupt counts were registering at one of the disks. All of the system's controllers were trying to access that single disk, leaving the other disks on the system idle.

Discussion: UNIX normally tries to keep all disk spindles at its disposal busy and, using its own disk scheduling algorithms, to balance the load on all drives. It can be outsmarted, however, if a system administrator who is not attentive enough to throughput fails to organize the load on each disk so as to maximize the balanced use of each disk. Without this sort of effort, read/write head seek movement cannot be minimized.

Version 7 UNIX and Berkeley UNIX systems make use of disk partitions to lay out the disk I/O system. Partitions divide the disk into logical concentric rings that are treated as logically independent disks. Unless the partitions

READY TO INVEST IN PROJECT MANAGEMENT SOFTWARE?

LOOK AT THESE NUMBERS.

1-801-531-8550

\$2.53

AT&T charge for 5-minute phone call, New York to Salt Lake City

SOFTTRAK

Application Software for Project Management

1977 West North Temple, P.O. Box 22156 AMF, Salt Lake City, Utah 84122

Circle No. 251 on Inquiry Card

are laid out carefully, disk heads will never end up where they should for fast access and maximized performance because different processes will force the heads to move to different parts of the disk. Redrawing the partitions may be necessary to balance the load.

Solution: If the problem persists when the controllers, disks, and partitions are balanced, the next step is to check the load on each controller. The problem might be that each controller is bearing a bigger burden than it can manage. To prevent waits for the channel to memory, put each drive on a separate controller.

As a last effort to cope with the problem of balance, check to see if kernel I/O buffers are in short

supply, particularly in association with an abundance of unused memory. If so, increase the number of kernel buffers by making use of the unused main memory.

SUMMARY

Performance tuning involves more than keeping your fingers busy on the keyboard. It is a complete operation that entails assessing what you want *before* you do anything. Above all, it's knowing what you want to accomplish. Only by knowing your object is it possible to intelligently convert wasted system resources to *performance* resources. You have to know what parts of the computer you need and what parts you can do without if you are to improve a specific kind of performance.

Plan your strategy first, then measure the results.

Clement T. Cole is an Engineering Supervisor at Masscomp. Previously, he consulted nationally on UNIX-related issues and worked for various concerns, including Tektronix, Inc. and the Mellon Institute of Science at Carnegie-Mellon University. Mr. Cole holds a BS in Electrical Engineering and Mathematics from CMU, as well as an MS in Computer Science from UC Berkeley.

Ed Breslin is Masscomp's Technical Marketing Promotional Writer. Formerly with DEC and Ztel, he holds a Society for Technical Communication Award for Excellence (Boston Chapter, 1982). Mr. Breslin graduated from Swarthmore College and Washington University (St. Louis). ■

ACUTY® 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.



225 West 30th Street, National City, California 92050

Circle No. 250 on Inquiry Card

dyalog APL™

Version 3.0 Available Now!

The Reliable High Performance APL for UNIX* Systems

Dyalog APL is fast!

Version 3.0 is up to 10 times faster than previous versions!

Dyalog APL is functional!

- Nested Arrays
- Full Screen Editor
- Full Screen Data Manager
- Event Trapping
- Interface to all UNIX* Facilities
- Optional Graphics

Dyalog APL is reliable!

Dyalog APL has been in commercial use for over two years and is available NOW for most UNIX* Systems so call or write today.

MIPS Software Development, INC.

31555 W. 14 Mile Rd. #104
Farmington Hills, MI 48018
(313) 855-3552

*Improvements are a function of system and usage
*UNIX is a trademark of AT&T Bell Laboratories.

Circle No. 249 on Inquiry Card

A NEW DIMENSION IN DATABASE MANAGEMENT



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

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

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

RHODNIUS Incorporated

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

(416) 922-1743

Telex: 06-986766 TOR.

The multi-user data base management system that puts mainframe power in your micro.

ZIM is a fully integrated fourth generation application development language that is loaded with these features and more.

- **POST RELATIONAL**
 - Entity Relationship Model
 - Powerful extension of Relational Model
- **REPORT WRITER**
 - Unlimited break levels, summary/detail reports, output to disk, printer, terminals
- **FORMS PAINTER AND MANAGER**
 - Menus, data entry, data display
 - Box fields
 - Old field value recall
- **DATA DICTIONARY**
- **COMPILER • PROGRAMMING LANGUAGE**
- **APPLICATION COMPLEXITY SUBJECT ONLY TO HARDWARE LIMITATIONS**
- **UNLIMITED FILE RELATIONS**
 - One to one
 - One to many
 - Many to many
 - Unrelated



- **RETRIEVAL STRATEGY OPTIMIZER**
 - Automatic use of 8087 chip (if available)
- **APPLICATIONS PORTABILITY**
 - **MULTI-USER**
- Full transaction processing control
- **C LANGUAGE INTERFACE**
- **QUALITY PRODUCT SUPPORT**

Zim is a mainframe data base management system that runs on micro-computers. If you want mainframe power, speed, flexibility and freedom from arbitrary limitations all at a micro price, talk to us about an evaluation system.

Dealer inquiries are welcome.



1785 Woodward Drive
Ottawa, Ontario K2C 0R1
(613) 727-1397

OUR FEATURE STORY

**NCR Version
Now Available**



Continued from Page 40

cheaper (thus slower) disk subsystems. Most desktop micro systems and a number of other mini/micro UNIX systems fall in this class. The two most notable applications of this class are general program development and office automation.

Given these four choices, you should determine which approach best fits your needs. Think of how the system will normally be used at peak times (10 am to 4 pm for most shops), and choose a system that performs well under that load. Discount the infrequent cases where nothing seems to work well.

SWAPPING AND PAGING TRADEOFFS

In a more technical vein, let's look at where swapping and paging work comes from and where it seems to be headed:

1) Paging schemes based on working set theory were studied at a time when all major systems programs were coded in assembler and were hand-optimized for page boundary alignment. Some compilers at the time even offered optimized allocations and were coded to minimize cross page references.

2) Most high-level structured code today produces a large number of non-local references. Thus, modern working sets are generally several times larger than those used during the days of the earlier paging studies.

3) During the 1970s, timesharing systems such as TENEX grew out of the XDS940 and Berkeley Timesharing System molds. Those systems offered integrated environments where all functional utilities were self-contained in programs. In today's UNIX environment, the *tools* approach is used to create a large number of small programs that afford constrained functionality. Many UNIX programs are only a few pages long and yet have a 100 percent working set.

4) A typical UNIX disk subsystem supports 15 to 35 transactions per second on many micros, and can handle from 35 to 55 transactions per second on larger/faster systems like the VAX 780. Swaps are generally segmented into pieces from 0.8 to two times the normal transaction time (i.e. at an average transaction rate of 35 blocks/second, a transaction period is 28 ms, thus on a 5 Mbit/second, 5¼-inch Winchester drive, a swap segment length should be about 10K bytes to 20K bytes). This allows some useful disk work to be interlaced with the swap period, thus letting memory jobs get useful work done. At this 35 transactions/second rate, a system with four concurrent disk-bound jobs will swap a 100K byte program in about 0.56 seconds (100K byte/20K byte * 28 * 4). Any program under swap segment size will be serviced in 0.11 seconds (28 ms * 4).

5) Paging on the same configuration occurs at

0.11 seconds (28 ms * 4) per page (1K byte on most systems). Thus a 100K byte program would take 11 seconds and a 20K byte program would take about 2.2 seconds. Extrapolating from this model, we can see that swapping frees and fills memory about 20 times faster than paging on this machine. This becomes a very significant issue when a system is asked to handle the standard UNIX toolbox of 10 to 50K byte programs or when it's asked to swap images for data segments of shared text programs like **vi**.

6) A 1 MB program swap on the same configuration would take 5.6 seconds. Assuming a 56K byte working set, active memory could be paged in exactly the same amount of time. Paging in this environment would free about 80 percent of the 1 MB space for other programs. This could be significant for a machine with little more than a megabyte of RAM.

7) All early studies of paging were done on systems large enough to provide a dedicated controller and spindle for the operation, thereby assuring peak paging rates and uniform paging response. Many Berkeley-based systems are run on a single controller (and often a single spindle), meaning that file system traffic greatly interferes with paging throughput and response time, which in turn causes non-linear degradation in system throughput, efficiency, and response times. Often a step reduction is visible when the workload backs up on the disk queue, resulting in scheduler/memory reaper thrashing at certain load levels.

In a nutshell, most systems should be configured with enough memory to hold active programs in memory without swapping or paging. If this cannot be done, swapping is preferred for those systems with job sizes in the 10K byte to 250K byte range, while paging is optimal for systems handling jobs in the 500K byte to 1 MB range. In fact, on the top end of the scale, paging is essential. Jobs running between 200K bytes and 650K bytes can use either method. According to the application and job mix within this range, one may be better than the other, but there are too many variables to allow for absolute guidelines.

Paging on disks with very slow seek times is certain death in most multiuser environments. Burst throughput in the disk/controller, meanwhile, is an important issue for swapping systems. (As an example, common 5¼-inch disk drive rates are 522K bytes/second for 1:1 interleave, and 261K bytes/second for 2:1 interleave. Eight and 14-inch SMD drives have rates about double these figures, depending on the brand.)

PHYSICAL DATA PLACEMENT

Another interesting but often misunderstood per-

formance issue revolves around file system metrics and workload characterization. These measures can shed light on many of the variables in response time equations. Among those variables are:

- 1) disk drive technology types and positioning curves.
- 2) the number and relative priority of processes using the file system.
- 3) partition layouts of physical drives.
- 4) data compactness and fragmentation within each file system.
- 5) driver disk seek optimization algorithms.
- 6) file system allocation strategies.
- 7) manual or automatic optimizations of file placement.
- 8) types of file access (**open**, **exec**, **read**, or **write** each requires different overheads and may or may not use **read ahead**).

A reasonable treatment of this topic would require a small book. For the purposes of this article, we'll restrict ourselves to one major theme—physical data placement. In the old batch days, careful hand-optimization of file allocations between various disk drives and within each spindle tuned job streams to minimize head seeks. This tuning often made a difference in runtimes of up to an order of magnitude by eliminating head thrashing (extra seek activity). Over the last 10 years, this common practice has become a lost art due to advances in disk drive technology and the development of large multitasking systems with dynamic file allocation. At the same time, the industry has re-introduced slow disk drives using low cost, stepping 5¼-inch technology.

UNIX and the hardware it operates are both prime examples of these trends. While I would never advocate going back to manual VTOC extend declarations, it does make sense that often-used UNIX files be hand-placed to ensure the best possible access times. It also makes sense to be certain that large, seldom-used files are moved out of the way of normal access. For UNIX systems, careful placement needs to occur at three levels:

- 1) data placement.
- 2) directory placement.
- 3) inode placement.

And for each item, two levels of placement are important:

- 1) cylinder placement.
- 2) placement within rotational interleave.

For inodes and multiblock directories, moreover, the block in which a particular item is located can become an important issue. This is because many UNIX system **exec** operations (used for loading and running programs) represent a significant percentage of total disk I/O requests, and an even larger

percentage of disk I/O time. Given the following commands:

```
$ PATH="/bin:/usr/bin:/usr/ucb:/u/joe/mybin:"; export PATH
$ someprog arg1 arg2
```

the system will attempt the following actions, some of which will be buffered in the system *buffer pool* and *inode table* (this assumes *someprog* is in *mybin*). Those not buffered will result in disk I/O:

- 1) root inode is acquired.
- 2) blocks 1-n of the root directory are read in a search for "bin"
- 3) /bin inode is acquired.
- 4) blocks 1-n of the /bin directory are read in a search for "someprog".
- 5) /usr inode is acquired.
- 6) blocks 1-n of the /usr directory are read in a search for "bin".
- 7) /usr/bin inode is acquired.
- 8) blocks 1-n of the bin directory are read in a search for "someprog".
- 9) /usr inode is acquired.
- 10) blocks 1-n of the /usr directory are read in a search for "ucb".
- 11) /usr/ucb inode is acquired.
- 12) blocks 1-n of the ucb directory are read in a search for "someprog".
- 13) /u inode is acquired.
- 14) blocks 1-n of the /u directory are read in a search for "joe".
- 15) /u/joe inode is acquired.
- 16) blocks 1-n of the joe directory are read in a search for "bin".
- 17) /u/joe/bin inode is acquired.
- 18) blocks 1-n of the bin directory are read in a search for "someprog".
- 19) /u/joe/bin/someprog inode is acquired.
- 20) the **exec** call reads the file and loads "someprog" into memory.

This search often requires 20 or more random disk accesses. Each access typically requires a seek, a wait for rotational latency, and a disk transfer. Thus the entire process will generally take between one and three seconds. By locating the inodes for search path directories in the same disk block, as many as half of the disk accesses mentioned above can be eliminated—if the requested inode block remains in the buffer pool.

TIME—THAT'S THE RUB

On many UNIX systems, the process of locating the most active inode, numbered 2 through *n* in the first block, will reduce common disk I/O by 15 to 30 percent and will make significant improvements in



open response times. Clustering frequently referenced files into several other inode blocks can attain even greater improvements in response time. As a side benefit, **sync** and **update** make less of an impact on the system because many of the inodes updated are located in the same block. This further reduces disk I/O overhead. By the way, it also cuts back on one source of buffer pool purging, thus increasing the overall hit rate!

If the above directory blocks are randomly allocated, we might be looking at between 10 and 20 file system transactions. Assuming the typical desktop microcomputer running one or two users with a 10 or 20 MB hard disk, we will probably see a typical transaction time of 1.12 seconds for 10 random I/O transactions. (This is arrived at by determining the average seek time + settle time + average latency + transfer time + interrupt latency. In this case, that's $85\text{ms} + 15\text{ms} + 9\text{ms} + 2\text{ms} + 1\text{ms} = 112\text{ms}$ per random request, and $10 * 112\text{ms} = 1.12$ seconds.)

This amounts to between one and two seconds of disk wait time to read all the directory blocks ran-

domly. If the directories were pre-allocated and extended to normal length at cold boot time, they could be searched at a much higher rate, depending on the file system interleave factor. For a typical UNIX micro, interleave time is about 6 ms. The time to do the I/O would be 129 ms for sequential disk transactions in the interleave pattern. (This is determined by adding average latency + number of sectors * interleave time. In this case, we find a range of $(9 + 10) * 6 = 69$ ms to $(9 + 20) * 6 = 129$ ms.) This represents another major reduction in command response time for small systems. But the effect on larger systems is not as great due to disk queue contention.

When this strategy is applied to the placement of other key system and applications files, some remarkable improvements can be seen in overall system response times. Reduced file system I/O makes for shorter disk service time for both file system traffic and paging/swapping traffic. This type of tuning on cold boot can increase overall performance by 15 to 50 percent. Some activities may run as much as two to 10 times faster. If done on a regular basis across the whole file system, results can often be startling.

As a closing note, it is important to segregate stale, dormant files in a remote area of the disk where they don't have to be sought over. Thus, one cold boot approach is to create all of the directories and inodes for active files, and then create a large "filler file" to pre-allocate necessary free space. Following this, all the inactive files can be loaded and the "filler file" can be removed. This will minimize seek time wasted on seldom used files.

How many manufacturers have set up their cold boot procedures for such optimization? Few, if any. On the other hand, how many claim to have significantly optimized their kernels so as to gain a five percent overall performance increase? Most, if not all.

I'd say, it's "back to basics" time. Let's get vendors and systems staffs to look deeper into the issues that really affect command response time. For the end user (particularly vertical market VARs) these layout improvements can be made with a couple of hours or days of work if a little thought is first invested in determining what files and features are—and aren't—often used. Look for some automatic file system sorters and maintainers that can automate this task for non-technical end users to hit the market soon.

John Bass has worked as a UNIX systems engineer since Version 6 was first released. His technical assistance was key to getting both Onyx Systems and Fortune Systems off the ground. ■

Great Learning Exhilarates

UNIX™ is a trademark of Bell Laboratories

And it saves you money.

CALL NOW **USER TRAINING CORPORATION**

(408) 370-9710

591 W. Hamilton Ave. • Campbell, CA 95008

See us at F.O.S.E.—Booth S 36
Circle No. 247 on Inquiry Card



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

C O D A T A S Y S T E M S C O R P O R A T I O N

CALENDAR

EVENTS

MARCH

March 30-April 2 10th West Coast Computer Faire, San Francisco. Contact: Computer Faire, Inc., 181 Wells Ave., Newton, MA 02159. 617/965-8350.

APRIL

April 14-18 Association of Computing Machinery, San Francisco: "Human Factor in Computing Systems." Contact: ACM Conference Coordinator, 11 W. 42nd St., New York, NY 10036. 212/869-7440.

April 23-26 UNIX Systems EXPO, San Francisco: "EXPO/85." Contact: Computer Faire Inc., 181 Wells Ave., Newton, MA 02159. 617/965-8350.

TRAINING

MARCH

March 2 AT&T Technologies, Sunnyvale, CA: "Overview of the UNIX System." Contact: AT&T Technologies, Corporate Education & Training, PO Box 2000, Hopewell, NJ 08525. 800/221-1647.

March 4 NCR Corp., Chicago, IL: "C Programming." Contact: NCR Customer and Support Education, 101 W. Schantz Avenue, Dayton, OH 45479. 800/845-CASE, or in OH, 800/841-CASE.

March 4 AT&T Technologies, Princeton, NJ: "UNIX System V Administration." Contact: AT&T Technologies, Corporate Education & Training, PO Box 2000, Hopewell, NJ 08525. 800/221-1647.

March 4-5 Intelligent Solution, Atlanta, GA: "UNIX Concepts." Contact: Intelligent Solution, 849 22nd St., Santa Monica, CA 90403. 213/207-5356.

March 4-8 Uniq Digital Technologies, Chicago, IL: "The UNIX Operating System." Contact: UDT, Avenue of the Stars, Suite 400, Los Angeles, CA 90067. 213/277-6288.

March 4-15 Information Technology Development Corp., Cincinnati, OH: "UNIX System Administration." Contact: ITD, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.

March 5-7 Bunker Ramo Information Systems, Trumbull, CT: "Diagnostic UNIX." Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

March 7-8 Structured Methods Inc., Chicago, IL: "Introduction to UNIX." Contact: SMI, 7 West 18th St., New York, NY 10011. 212/741-7720 or 800/221-8274.

March 8 Intelligent Solution, Atlanta, GA: "UNIX Overview." Contact: Intelligent Solution, 849 22nd St., Santa Monica, CA 90403. 213/207-5356.

March 11 AT&T Technologies, Sunnyvale, CA: "UNIX System Document Preparation Utilities." Contact: AT&T Technologies, Corporate Education & Training, PO Box 2000, Hopewell, NJ 08525. 800/221-1647.

March 11 AT&T Technologies, Princeton, NJ: "Internal UNIX System Calls and Libraries Using C Programming." Contact: AT&T Technologies, Corporate Education & Training, PO Box 2000, Hopewell, NJ 08525. 800/221-1647.

March 11-15 Structured Methods Inc., New York, NY: "UNIX System Workshop." Contact: SMI, 7 West 18th St., New York, NY 10011. 212/741-7720 or 800/221-8274.

March 13-15 Digital Equipment Corp., Orlando, FL: "Comprehensive Overview of the UNIX Operating System." Contact: Digital Educational Services, 12 Crosby Drive, Bedford, MA 01730. 617/276-4949.

March 18 AT&T Technologies, Lisle, IL: "UNIX System V Device

Drivers." Contact: AT&T Technologies, Corporate Education & Training, PO Box 2000, Hopewell, NJ 08525. 800/221-1647.

March 18 NCR Corp., Chicago, IL: "UNIX Operating System." Contact: NCR Customer and Support Education, 101 W. Schantz Avenue, Dayton, OH 45479. 800/845-CASE, or in OH, 800/841-CASE.

March 18 AT&T Technologies, Sunnyvale, CA: "UNIX System V Device Drivers." Contact: AT&T Technologies, Corporate Education & Training, PO Box 2000, Hopewell, NJ 08525. 800/221-1647.

March 18 NCR Corp., New York, NY: "UNIX System Administration." Contact: NCR Customer and Support Education, 101 W. Schantz Avenue, Dayton, OH 45479. 800/845-CASE, or in OH, 800/841-CASE.

March 18 AT&T Technologies, Dublin, OH: "UNIX System V Administration." Contact: AT&T Technologies, Corporate Education & Training, PO Box 2000, Hopewell, NJ 08525. 800/221-1647.

March 18-19 Intelligent Solution, Anaheim, CA: "UNIX Concepts." Contact: Intelligent Solution, 849 22nd St., Santa Monica, CA 90403. 213/207-5356.

March 18-20 Structured Methods Inc., Washington, DC: "UNIX System Internals." Contact: SMI, 7 West 18th St., New York, NY 10011. 212/741-7720 or 800/221-8274.

March 18-22 Bunker Ramo Information Systems, Trumbull, CT: "Introduction to UNIX." Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

March 18-29 Information Technology Development Corp.: "The C Programming Language." Contact: ITD, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968

March 18-29 Information Technology Development Corp.: "The UNIX System." Contact: ITD, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968

March 19 AT&T Technologies, Sunnyvale, CA: "Overview of the UNIX System." Contact: AT&T Technologies, Corporate Education & Training, PO Box 2000, Hopewell, NJ 08525. 800/221-1647.

March 22 Intelligent Solution, Anaheim, CA: "UNIX Overview." Contact: Intelligent Solution, 849 22nd St., Santa Monica, CA 90403. 213/207-5356.

March 25 AT&T Technologies, Princeton, NJ: "UNIX System V Device Drivers." Contact: AT&T Technologies, Corporate Education & Training, PO Box 2000, Hopewell, NJ 08525. 800/221-1647.

March 25-26 Structured Methods Inc., Dallas, TX: "Introduction to UNIX." Contact: SMI, 7 West 18th St., New York, NY 10011. 212/741-7720 or 800/221-8274.

March 25-29 Specialized Systems Consultants, Bellevue, WA: "C Programming Workshop." Contact: SSC P.O. Box 7, Northgate Station, Seattle, WA 98125. 206/367-8649.

March 27 Specialized Systems Consultants, Bellevue, WA: "UNIX for Managers." Contact: SSC, P.O. Box 7, Northgate Station, Seattle, WA 98125. 206/367-8649.

March 25-29 Bunker Ramo Information Systems, Trumbull, CT: "Advanced UNIX." Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611 203/386-2223.

APRIL

April 1-2 Intelligent Solution, Washington, D.C.: "UNIX Concepts." Contact: Intelligent Solution, 849 22nd St., Santa Monica, CA 90403. 213/207-5356 or 800/367-0948.

April 3-4 Intelligent Solution, Washington, D.C.: "Programming in C." Contact: Intelligent Solution, 849 22nd St., Santa Monica, CA 90403. 213/207-5356 or 800/367-0948.



ANN ARBOR INTRODUCES THE XL SERIES:

A DISPLAY OF EXCELLENCE IN FORM & FUNCTION

Product excellence doesn't happen overnight. At Ann Arbor we've been designing quality terminals for professionals for over 15 years. We've innovated many of the time- and eye-saving features you now take for granted. And many you don't, if you're not already an Ann Arbor user. Now we're unveiling another masterpiece: Our XL Series of ANSI-Standard Terminals.

Designed with you in mind. The real beauty of the XL Series is its focus on your comfort. Its fully programmable keyboard saves you typing time and effort. Its dynamically selectable display lets you zoom more data onto the screen when you want more context, and

less data when you just want bigger characters. With 15-inch or 17-inch portrait or landscape screens, offered in a full spectrum of phosphors, Ann Arbor is easy on your eyes.

Within the XL Series you'll find displays up to 170 characters per line and up to 66 lines per screen.

With plain English setup lines that give you a free hand in feature selection. And a whole palette of diagnostics and data-line monitoring aids to complete the picture.

There's an XL for every application.

Start with the Genie+Plus XL to meet your basic terminal needs. Move to the Ann Arbor Ambassador XL when you want a full-page display.

When words alone no longer tell the story, switch to the GXL and add vector graphics. Finally, when you need even more memory and display than any other terminal provides, move all the way to the Guru® XL.

Take a closer look.

Study the specs. View the ergonomic design. See for yourself what it is that makes the Ann Arbor XL Series the state-of-the-art display terminal you've been looking for.

For more information, write to Ann Arbor Terminals, Inc. at 6175 Jackson Road, Ann Arbor, Michigan 48103. Or call 313/663-8000.

ANN ARBOR
TERMINALS

April 5 Intelligent Solution, Washington, D.C.: "UNIX Overview." Contact: Intelligent Solution, 849 22nd St., Santa Monica, CA 90403. 213/207-5356 or 800/367-0948.

April 8-12 Structured Methods Inc., New York: "UNIX System Workshop." Contact: SMI, 7 West 18th St., New York, NY 10011. 212/741-7720 or 800/221-8274.

April 8-19 Information Technology Development Corp., Cincinnati: "The UNIX System." Contact: ITD, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.

April 9-11 Bunker Ramo Information Systems, Trumbull, CT: "Diagnostic UNIX." Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

April 10-12 Specialized Systems Consultants, Bellevue, WA: "Hands-On UNIX for Programmers." Contact: SSC, P.O. Box 7, Northgate Station, Seattle, WA 98125. 206/367-8649.

April 15 NCR Corp., Dayton, OH: "UNIX Operating System." Contact: NCR Corp., CASE-Special Orders, 101 W. Schantz Ave., Dayton, OH 45479. 800/845-2273 or 800/841-2273.

April 15-16 Bunker Ramo Information Systems, Trumbull, CT: "UNIX/C Applications." Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

April 15-19 Structured Methods Inc., Washington, D.C.: "C Language Workshop." Contact: SMI, 7 West 18th St., New York, NY 10011. 212/741-7720 or 800/221-8274.

April 17-19 Digital Equipment Corp., Chicago: "Comprehensive Overview of the UNIX Operating System." Contact: Digital Education Resources, 12 Crosby Drive, Bedford, MA 01730. 617/276-4949.

April 18-19 Structured Methods Inc., New York: "Using Lex and yacc." Contact: SMI, 7 West 18th St., New York, NY 10011. 212/741-7720 or 800/221-8274.

April 18-22 Bunker Ramo Information Systems, Trumbull, CT: "Introduction to UNIX." Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

April 22-23 Uni-Ops, San Francisco: "Post-Structural Programming." Contact: Uni-Ops, P.O. Box 27097, Concord, CA 94527. 415/945-0448.

April 22-23 Specialized Systems Consultants, Bellevue, WA: "Hands-On UNIX for Non-Technical People." Contact: SSC, P.O. Box 7, Northgate Station, Seattle, WA 98125. 206/367-8649.

April 22-May 3 Information Technology Development Corp., Cincinnati: "UNIX System Administration." Contact: ITD, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.

April 22-May 3 Information Technology Development Corp., Cincinnati: "Advanced C Programming." Contact: ITD, 9952 Pebbleknoll Dr., Cincinnati, OH 45247. 513/741-8968.

April 23-27 Plum Hall, Somers Point, NJ: "Advanced C Topics." Contact: Plum Hall Seminars, 1 Spruce St., Cardiff, NJ 08232. 609/927-3770.

April 24 Bunker Ramo Information Systems, Trumbull, CT: "UNIX Marketing." Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

April 25-29 Bunker Ramo Information Systems, Trumbull, CT: "Advanced UNIX." Contact: Bunker Ramo, Trumbull Industrial Park, Trumbull, CT 06611. 203/386-2223.

April 29-30 Structured Methods Inc., Atlanta: "Introduction to UNIX." Contact: SMI, 7 West 18th St., New York, NY 10011. 212/741-7720 or 800/221-8274.

Please send announcements about training or events of interest to: UNIX Review Calendar, 500 Howard Street, San Francisco, CA 94105. Please include the sponsor, date, and location of event, address of contact, and relevant background information.

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

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

<p>USA</p> <p>In Arizona:</p> <p>Artemis Computer 602/957-0469</p> <p>Systems Solutions, Inc. 602/224-0026</p> <p>Professional Data Systems, Inc. 602/265-6656</p> <p>In California:</p> <p>Quintec 818/889-4819</p> <p>American Computer & Communications 415/348-1956</p> <p>MCM Enterprises 415/327-8080</p> <p>American Computers & Engineers 213/477-6751</p> <p>Kierulff Electronics 213/725-0325</p> <p>Accountability Systems 714/639-4570</p> <p>Excollibur 916/972-9252</p> <p>Kierulff Electronics 714/278-2112</p> <p>Kierulff Electronics 408/971-2600</p> <p>Cromemco, Inc. 818/346-6690</p> <p>In Connecticut:</p> <p>Datocraft, Inc. 203/673-6952</p> <p>In Florida:</p> <p>Automated Computer Systems 305/594-3819</p> <p>Computer Centre 813/484-1028</p> <p>Royal Data, Inc. 305/267-1960</p> <p>In Georgia:</p> <p>Cromemco, Inc. 404/391-9433</p> <p>Systems Atlanta 404/928-0240</p> <p>Kierulff Electronics 404/417-5252</p> <p>Southern Exchange, Inc. 404/921-2662</p> <p>In Illinois:</p> <p>Commercial Data Systems 309/797-9401</p> <p>Computerland 312/967-1714</p> <p>Southern Exchange, Inc. 404/921-2662</p> <p>Alpine Computer Center, Inc. 815/229-0200</p> <p>Cromemco, Inc. 312/934-0650</p> <p>Alternate Computer Services 312/893-1992</p> <p>In Indiana:</p> <p>Memory Bank, Inc. 312/891-1064</p> <p>219/931-0203</p> <p>Harbourtown Soles 317/877-4900</p> <p>Microcomputer Specialists, Inc. 219/762-8541</p> <p>In Kansas:</p> <p>Tradewind Systems 316/624-8111</p> <p>In Louisiana:</p> <p>Muse Data Technologies 504/293-0320</p> <p>Standard Systems, Inc. 318/625-8613</p> <p>In Maryland:</p> <p>Dynamic Data Processing 301/657-1211</p> <p>In Massachusetts:</p> <p>Kierulff Electronics 617/667-8331</p> <p>Cromemco, Inc. 617/938-7010</p> <p>In Michigan:</p> <p>United Microsystems Corporation 313/668-6806</p>	<p>Jepson Group, Inc. 616/698-8700</p> <p>Automated Business Consultants 313/478-0557</p> <p>In New Jersey:</p> <p>Kierulff Electronics 201/575-6750</p> <p>In New Mexico:</p> <p>South West Computer Stores, Inc. 505/292-6568</p> <p>In New York:</p> <p>Custom Computer Specialists 516/231-1155</p> <p>C.C.S., Inc. 212/986-7520</p> <p>Trexis, Inc. 914/268-5161</p> <p>In Ohio:</p> <p>Lucas Office Equipment & Service, Inc. 513/433-8484</p> <p>Odyssey Systems, Inc. 216/526-9933</p> <p>(ISIS) Innovative Systems/Integrated Software 419/531-0220</p> <p>In Pennsylvania:</p> <p>Modular System Design 412/521-6700</p>	<p>INTERNATIONAL cont.</p> <p>In Australia cont.:</p> <p>Micro Data PTY LTD. 61-9/328-1179</p> <p>Insystems P/L 61-3/690-2899</p> <p>In Canada:</p> <p>Cro-Corp Computer Solutions 403/286-8459</p> <p>D.E. Systems 613/729-5164</p> <p>Future Electronics 610/421-3251</p> <p>In Costa Rica:</p> <p>Control Electronic 506/24-44-44</p> <p>In Denmark:</p> <p>Merkur Data Services A/S 45-2/63-01-55</p> <p>In England:</p> <p>Joragate Ltd. 44-1/671-6321</p> <p>In France:</p> <p>Penaronda Informatique 56-979618</p> <p>In Greece:</p> <p>Algorithm Ltd. 370-1/933-0551/5858</p> <p>In Hong Kong:</p> <p>Vando Computer & Equipment Co. 852-3/348702-5</p> <p>In Israel:</p> <p>Information Systems Ltd. 03-775111</p> <p>In Italy:</p> <p>CO. N.I.A. 39-51/375001</p> <p>In Japan:</p> <p>Asahi Gloss 81-3/218-5848</p> <p>In Mexico:</p> <p>Micromex, S.A. de C.V. 905/687-8886/8913 905/536-5503</p> <p>In Mid-East:</p> <p>Multi Medio Video, Inc. CA USA 408/727-1733</p> <p>National Computer System Pakistan 3156644</p> <p>Computer System Marketing Center Saudi Arabia 966/2-651-7707 966/2-653-0580</p> <p>In The Netherlands:</p> <p>Rocom B.V. 31-40/524055</p> <p>In Norway:</p> <p>Micro Systems A/S 47-2/41-69-76</p> <p>In Scotland:</p> <p>Micro Centre Complete Micro 44-31/556-7354</p> <p>In Sweden:</p> <p>Datrolsering Konsult AB 46-8/753-3090</p> <p>In West Germany:</p> <p>Cromemco GmbH 49-6196/481606</p> <p>Digitronic Computer-systeme GmbH 4103/88672/73</p> <p>Cosy-X Computer Systeme GmbH 2173/52071/72</p> <p>Comico Deutschland 49-2151/795577</p> <p>CE Computer GmbH 02151-22121</p> <p>Anitex Computer Systems GmbH 08142-13091</p>
--	---	---



Cromemco®

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

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

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

More capability and expandability

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

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

But we give you even more.

A complete solution

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

standard protocols. Mainframe interface.

And software support is here to meet your needs. We offer major programming languages, 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,
Germany.



Circle No. 244 on Inquiry Card

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

Cromemco®

ADVERTISER'S INDEX

Absolute Computers	19	IBC	Cover IV
Ann Arbor Terminals	105	Image Network	93,96
Applix, Inc.	9	Lachman Associates	69
AT&T Technologies	16,76,77	Mark Williams Co.	17
B.A.S.I.S.	80	Microware Systems Corp.	41
Bear Systems	94	MIPS Software	97
Ceegen Corp.	72	Mt. XINU	25
CCA Uniworks	95	Peripheral Systems	20,21
Century Software	66	Quality Software Products	90
CIE Systems	37	Relational Database Systems	5,6,7
CMI Corp.	81	Reliable Data Systems	92
Codata Systems Corp.	103	Rhodnius, Inc.	98
Computer Cognition	97	R Systems	Cover III
Computer Faire	89	Scientific Placement, Inc.	93
Computer Methods	11	Sir, Inc.	85
Computer Technology Group	31,92	Soft Fair	12
Concentric Associates	70	SofTrak	96
COSI	79	Sperry Corp.	47,87
Cromemco	106,107	Teletype Corp.	45
Data General Corp.	Centerspread	Ubiquitous Systems	93
Daystrom	90	Unicomp	94
DSD Corp.	86	Unify	Cover II, p.1
Emerald City, Inc.	91	Unipress Software	59,61,63,65,67
General Communications Corp.	88	Unisoft	57
Gould Electronics	73	University of Toronto	62
Handle Corp.	15	User Training Corp.	102
Hewlett Packard	82,83	UX Software	71
Human Computing Resources	13	Zanthe	99

COMING UP IN APRIL

Local Networking

- Network Protocols Defined
- UNIX Networking Hurdles
- MMU/CPU Interaction
- Communications with Non-UNIX Systems
- Network Management and Maintenance
- Security and Data Integrity

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"/>
November 1984	<input type="checkbox"/>
December 1984	<input type="checkbox"/>
January 1985	<input type="checkbox"/>
February 1985	<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 _____



“R WORD is an excellent program. It is one of the easiest, yet full-featured, word processors I have ever used.”

Bob Woodard of the University of Iowa continues in his letter to us about R WORD, “It beats the heck out of the word processing package available with our machine. R WORD is definitely the favorite of the people who have tried it.”

We’ve received many letters like Bob Woodard’s, praising the R Family of Office Automation Software. Now, we’d like **you** to meet R Family — three programs which are powerful, easy-to-use, well-documented, cost-effective, and supportable. And they’re designed for single-users, multi-users, and multi-computer users.

R Office Manager
Suggested List \$295

R Office Manager is the most powerful organizational and managerial package available. It consolidates and streamlines the basic office functions of today’s complex corporate environment. R Office Manager coordinates phone messages, calendars, calculations, things-to-do lists, messages, names and addresses, and more.

R WORD II

Suggested List \$895

R WORD II is designed for a production word processing environment which requires full-featured word processing, yet R WORD II offers amazing simplicity of use. Features include Help Menus, Cursor Controls, Editing Functions, Formatting, Three-Level File Structure, Printing Features, Mail Merge, Default Set Up, and Utility Functions.

The program also offers Global Search and Replace, Directories, Spelling Checker, Table Math, Headers and Footers, Automatic Table of Contents, Automatic Paragraph Numbering, Document Assembly, and much more.

R WORD III

Suggested List \$1295

R WORD III offers all the features of R WORD II and R Office Manager, plus complete data base, file management, and report generation features which allow you to integrate any information found in other programs on your computer.

1-(800) 527-7610

Call us toll-free for more information about R Family. We are currently ported to and shipping R Family programs on the Tandy Model 16, NCR Tower + XP, Apple Lisa, Pixel, Plexus, and Altos 68000. Additional UNIX and XENIX ports are occurring weekly. We are also shipping for all MS-DOS and PC-DOS operating systems. Manufacture, distributor, dealer, national account, and porting inquiries are invited.

When you select word processing, office management, and office automation software, consider the benefits of R Family. This one set of programs can satisfy single-user, multi-user, and multi-computer requirements. R Family offers you consistency, versatility, and supportability, as well as the benefits of transportability and connectivity.

The R Family - single and multi-user office automation software:

**DOS • UNIX • XENIX
COS • DX10 • DNOS**

Circle No. 243 on Inquiry Card



R SYSTEMS, INC. 11450 Pagemill Road, Dallas, TX 75243 (214) 343-9188 (800) 527-7610



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. 242 on Inquiry Card