### CONTENTS

**The USENIX Association Toronto Meeting** .................................................................................................................................................................................. 3
**Call for Exhibitors at Toronto** ......................................................................................................................................................................................... 3
**What UNIX Documents Would You Like USENIX to Offer?** ........................................................................................................................................... 3
**Call for Papers for the Software Tools Users Group Toronto Meeting** ....................................................................................................................... 4
**Assistance Needed** ............................................................................................................................................................................................... 4
**USENIX Association Office Report** ................................................................................................................................................................. 4
**UNICOM Meeting Notes** ...................................................................................................................................................................................... 5
  - W1 — Welcoming Remarks and Keynote Address .............................................................................................................................................. 5
  - W2 — News from . . .................................................................................................................................................................................. 7
  - W3A — Performance Enhancement and Measurement ................................................................................................................................................ 10
  - W3B — UNIX System V ........................................................................................................................................................................... 13
  - W4A — Networks .................................................................................................................................................................................... 16
  - W4B — Development Tools ................................................................................................................................................................. 17
  - T1A, T2A — Bell Labs Reports .......................................................................................................................................................... 20
  - T1B — UNIX in the Office Environment ................................................................................................................................................... 23
  - T2B — New UNIX Implementations I .......................................................................................................................................................... 25
  - T3A — Programming Languages and Environments ................................................................................................................................ 27
  - T3B — Novel Applications of UNIX ....................................................................................................................................................... 29
  - T4A — Technical Reports ................................................................................................................................................................. 32
  - T4B — Database Systems ................................................................................................................................................................. 34
  - F1A — Graphics ..................................................................................................................................................................................... 37
  - F1B — New UNIX Implementations II ........................................................................................................................................................... 40
  - F2A — New UNIX Implementations III ..................................................................................................................................................... 44
  - F2B — Marketing and Venture Capital ................................................................................................................................................... 46
  - F3A — Portability ................................................................................................................................................................................ 49
  - F3B — Languages and Programming Environments .................................................................................................................................. 50
  - F4 — Standardization.............................................................................................................................................................................. 52
**Summary of USENIX Board Meetings at San Diego** .................................................................................................................................................. 55
  - Monday January 24th, 1982 ................................................................................................................................................................. 55
  - Meeting of the Board with the Membership Wednesday January 25th, 1983, at 6 pm ........................................................................ 57
  - Meeting of the USENIX Board during UNICOM ........................................................................................................................................ 57
  - USENIX Board meeting ............................................................................................................................................................................ 58

---

The deadline for submissions for the June issue of ;login: is May 27
NOTICE

;login: is the official newsletter of the USENIX Association, and is sent free of charge to Individual, Public and Institutional members of the Association.

The USENIX Association is an organization of AT&T licensees, sub-licensees, and other persons formed for the purpose of exchanging information and ideas about UNIX® and UNIX—like operating systems and the C programming language. It is a not-for-profit corporation incorporated under the laws of the State of Delaware. The officers of the Association are:

President     Lou Katz    Directors     Bruce S. Borden
Vice-President John L. Donnelly    Alan G. Nemeth
Secretary     Lewis Law    Deborah K. Scherrr
Treasurer     Thomas Ferrin    Waldo M. Wedel

The Executive Director of the Association and editor of ;login: is Tom Strong.

Membership information can be obtained from the Association office:

USENIX Association
P.O. Box 7
El Cerrito, CA 94530
(415) 528-UNIX

Members of the UNIX community are heartily encouraged to contribute articles and suggestions for ;login:. Your contributions may be sent to the editor electronically at

ucbvax!g:usenix

or through the US mail to the Association office at the address above. The USENIX Association reserves the right to edit submitted material.

;login: is produced on UNIX using troff and a variation of the —me macros. We appreciate receiving your contributions in n/troff input format, using any macro package. If you contribute hardcopy articles please leave left and right margins of 1" and a top margin of 1½" and a bottom margin of 1¼". Hardcopy output from a line printer or most dot-matrix printers is not reproducible.

This newsletter may contain information covered by one or more licenses, copyrights, and/or non-disclosure agreements. Permission to copy without fee all or part of this newsletter is granted only to Institutional members of the USENIX Association, provided that copies are made only for internal use at the member campus or plant site. No other reproduction of this newsletter in its entirety or in part may be made without written permission of the Association.

*UNIX is a trademark of Bell Laboratories.
The USENIX Association Toronto Meeting

The Summer, 1983, technical meetings of the USENIX Association and the Software Tools Users Group will be held July 12—15 in Toronto, Ontario, Canada, at the Harbour Castle Hilton Hotel. The host of the meeting will be Human Computing Resources Corporation. The local arrangements chairperson is Suzanne MacNary. Local arrangements are being handled by Rogal Boston. The technical program chairperson for USENIX is Michael Tilson of HCR.

A meeting announcement, call for papers, and other information was mailed in early April. A registration packet will be mailed in early May. Further information on the meeting may be obtained from:

Rogal Boston  
72 Langley Road  
Newton Centre, MA 02159  
(617) 965-1000

The Toronto meeting will feature both a conference and an exhibition devoted to the topics of interest to the UNIX community. Presentations will be made in the following areas:

UNIX Applications: graphics, data base systems, office automation, word processing, etc.
UNIX Systems: the implementation of network and distributed systems for porting of UNIX to new computer systems and system management and performance.
UNIX Programming Tools: utility programs, programming environments and new programming languages and implementation.
The UNIX World: standards, analysis of commercial trends, and other related topics.

The program committee also intends to schedule other kinds of presentations, including panel discussions and invited speakers. They are open to suggestions from both the academic and commercial communities. Suggestions should be submitted as soon as possible to Michael Tilson at (416) 922-1937 or decvax!hcr!hcrvax!mike.

Call for Exhibitors at Toronto

All companies interested in exhibiting at the Toronto Conference are invited to send for an exhibitors kit. Please address your request to Paula at Rogal Boston at the address above.

Exhibitors kits were mailed in March to companies that exhibited at previous meetings.

What UNIX Documents Would You Like USENIX to Offer?

The Association is investigating, subject to obtaining the appropriate copyright releases, printing UNIX documents and manuals needed by its members. The manuals would probably be offered in the 6"x9" format. We would like to know if you would like this service to be undertaken, and exactly what documents and how many of each you will be needing during 1983. Please mail a statement of your projected needs for 1983 to the Association office.

If you know of sources for UNIX manuals that sell to the general public please let the office know.

If the Association offers manuals in the future they will be priced at a discount to members.
Call for Papers for the Software Tools Users Group Toronto Meeting

Presentations are invited for the Software Tools Users Group meeting in Toronto. Talks may include descriptions of projects using ratfor and/or the tools, newly-created or enhanced tools, thoughts about future directions for the tools, software portability and programming environments, or other areas of interest to the tools community.

Abstracts for proposed talks should be submitted to the Software Tools program chairman:

Neil Groundwater
Analytic Disciplines, Inc.
8320 Old Courthouse Road, #300
Vienna, VA 22180
703-893-6140

They must contain the following information:

Title
Name of author
Institution or company
Mailing address
Phone number (and network address, if available)
Audio-visual requirements

Suggestions for other types of presentations are also solicited.

Assistance Needed

The following request for information has been received at the Association Office. If you can help please contact the person listed and send the information to the Office.

A BASIC compiler/interpreter that will work on a VAX running 4.1BSD.

Dennie Van Tassel
U.C. Santa Cruz
Computer Center
Santa Cruz, CA 95064
(408) 429-2434

USENIX Association Office Report

All 1983 members have been sent the first 1983 issue of ;login:.

There have been a couple of more additions to the 1983 Software Distribution tape, including nroff drivers for the Diablo 630 and for the Concept C108 and AVT terminals with the diablo/nroff graphics character set. Other contributions for the tape were discussed on page 11 of the last issue of ;login:.

The office has mailed all 1982 distribution tapes for which we have received release forms and been able to confirm licenses. All 1982 Institutional members who have not been sent their tapes have been sent at least one letter telling why and requesting the needed information.
UNICOM Meeting Notes

The following are notes on the presentations at the UNICOM meeting at the Town and Country Hotel in San Diego on January 26—28, 1983. There were concurrent sessions each day except for the first morning. The notes have been prepared by a variety of people; the reporter for each session is noted at the beginning of the session.

Due to a misunderstanding on the part of the editor notes for seven of the twenty one sessions are not available. The abstracts for the talks in those sessions have been printed. Our apologies to the speakers and the readers.

W1 — Welcoming Remarks and Keynote Address

Wednesday, 8:45—10

Reporter: Tom Strong
Editor, ;login:
USENIX Association Office
P.O. Box 7
El Cerrito, CA 94530

Welcome

Tom Uter and Bill Appelbe
University of California at San Diego
La Jolla, CA 92093

These gentlemen, who were respectively the general chairperson and technical co-chairperson for UNICOM, welcomed the very large crowd. Over 100 abstracts had been submitted; 60—70 were accepted.

KEYNOTE ADDRESS

Software Army on the March — Project Strategies and Tactics

John Mashey
Bell Laboratories
Whippany, NJ 07981

Mr. Mashey developed an analogy between software development and an army campaign to explain and help validate the concept of “fast prototyping” as a project management tool. The domain under discussion was significant software and/or hardware projects in internal industrial efforts but not necessarily research or government environments.

Project planning was presented as involving three stages:

• strategies — what to do
• tactics — how to do it
• implementation — do it

The talk focused on the first of these.

No preference was expressed for any specific project lifecycle description. Decisions have to be made on which methodology to use and then how and when to allocate resources.
An abstract model was presented as a game graph. There are many ways to get from the starting point to the end, which was represented as a plane rather than a single point [like going from New York City to somewhere in California]. Routes (arcs) lead to nodes where alternate arcs can be taken, often leading to other major arcs. Some arcs, indeed some major arcs, can lead to dead ends or in unreasonable directions or require resources that are not available. The problem is to understand the attributes of the game and find a way to select the arcs to be taken. The attributes of the game:

- decisions are multi-staged, like chess
- there are many people involved — friends and opponents — and you sometimes cannot tell what others are doing
- there is often incomplete or hidden information
- the game is not completely deterministic
- the rules may vary over time
- the universe may change.

When a node is reached there are several things to consider when deciding which arc to take:

- there is a cost associated with each move
- it takes time to move
- there is risk, both in the present and the future, associated with the choice of arcs
- the capacity, both in the present and the future, required by choosing an arc
- the cost, time, risk, and capacity required to unmake a move.

Two limit-case node configurations were described:

- a single path from one node to another [like a single bridge over a river] — take care and spend a lot of time analyzing the move
- many paths from one node to another — take the easy one as you can always redo it later

An analysis of the six considerations for selecting an arc (cost, time, initial and continuing risk and capacity) will enable you to recognize the alternatives and tradeoffs so you can make informed decisions.

Decisions should be sorted by impact: identify those with large ripple effects, prefer those that preserve alternatives, and defer those that can be helped by later information. Different configurations of arcs and nodes (e.g., parallel or sequential arcs) will affect the process.

Mr. Mashey went on to present a concrete analogy between an army marching and a large software project. The major points he made are summarized below.

**Running the project:** define the project goal; do the development; maintenance; and completion.

**While running the project you need to be aware of:** budgets and schedules; the capabilities of the personnel; and requirements, both political and for the end-user.

**The types of personnel involved in a project include:** systems engineers to help the technical design; human performance engineers to help the human interface design; people who can produce reliable, maintainable production code; people, including new persons on the project, who can do the less demanding coding; fast prototypers who can, quickly and at low cost, find and try out new and/or innovative and/or risky solutions; support and tools groups to provide a usable starting base and the tools needed for development; research groups; upper management, with whom there must be two-way communication; and others, including friends and skeptics of the project and people who are threatened by it.

**Project resources required include:** existing and planned tools, including old but proven solutions (it is difficult to built a new set of tools in the middle of a project); strategic planning; and existing knowledge of the domain.
Project strategy involves:

- identifying existing components that fit into the project;
- identifying and analyzing the components that are needed and their alternatives;
- establishing a firm starting base;
- identifying the areas of highest risk or uncertainty and starting the fast prototypers on them as soon as possible; (this often involves pursuing parallel arcs); and
- involving the skeptics and end-users as early as possible.

A project can fail in spite of the best planning. It may have been the wrong solution to the problem or hardware or software limitations or that someone came up with a better solution. When that happens management must know how to recognize it and be prepared to cut their losses.

W2 — News from . . .

Wednesday, 10:30—12
Armando Stettner, chair

Bob Marsh, /usr/group

Mr. Marsh welcomed the audience and then presented seven items.
1. /usr/group plans to change its name to uniFORUM to better reflect the purposes of the group.
2. The terms of several directors are up in June. A nominating committee has been formed. Mr. Marsh will not be running for another term as president.
3. The draft standard for UNIX is seen as the most important, highest priority item for the group. [See the previous issue of ;login: for more information on the draft standard...Ed.]
4. A local chapter startup kit is available from the group.
5. The newsletter, commUNIXcations, will be featuring issues devoted to specific topics such as word processing on UNIX.
6. Corporations may join the group as sponsors. (Normal membership is available to individuals only.)
7. The /usr/group Board decided in October to put on its own conference annually. The Board felt that they had different goals and objectives that could not be met at a joint conference with USENIX. However, said Mr. Marsh, this may not be the best thing and the decision was being reconsidered. A decision was promised by Friday of UNICOM. [See the announcement in the last issue of ;login:...Ed.]
Lou Katz, USENIX

Mr. Katz stated the purposes of the USENIX Association and told of the efforts of the new Association office to support those purposes.

The future of joint conferences was under discussion within the USENIX Board and with members of the /usr/group Board. The main question to be resolved was what was best for the UNIX community. An answer to the question of future joint conferences with /usr/group was to be announced by Friday.

There was to be a Board meeting with the membership on Wednesday. The Board was reviewing how to make the tutorials more available, as there had been tremendous oversubscription of some of them.

Mark Horton

Mr. Horton made the distinction between usenet and the uuep network: usenet is that collection of sites who receive the newsgroup called “net.general.” There was to be a Birds—of—a—Feather that night to provide information on how to join usenet.

Bob Guffy, AT&T

Mr. Guffy told us AT&T was going to announce System V. An information packet on System V was to be available on the vendor tables. He reported that UNICOM was the largest gathering of people with an interest in UNIX.

Heinz Lycklama, /usr/group Standards Committee

Mr. Lycklama introduced the draft UNIX system interface standard that is being developed by the /usr/group Standards Committee. It is not a UNIX standard; rather it is a system interface standard for end users and applications writers. It is based on System III. [The draft standard was discussed in the last issue of ;login:; it may be obtained for $50 from /usr/group, P.O. Box 8570, Stanford, CA 94305...Ed.]

Robert Fabry, UC Berkeley

Dr. Fabry announced that 4.2BSD is “almost” ready. 4.2BSD is the latest version of UC Berkeley’s paging UNIX for the VAX. In 4.2BSD, neither the system nor applications are limited by address space. It will be available to all UNIX/32V and System III licensees. Binary licenses will be available from several sources (not UCB). It was developed with financial support from ARPA, manufacturers, and industry. The software is in the public domain except for that covered by AT&T’s proprietary interests.

Features of 4.2BSD are: Ethernet * support with 1.2 Mbit per second end-to-end throughput on an 11/750; Arpanet support; a file system that is 4–10 times faster than that on 4.1BSD; new methods of interprocess communication; new symbolic debugger, new sendmail; faster /77; a worm shell (for local networks); file mapping in address space; support for the VAX 11/730 and new peripherals; and isolation of VAX specifics.

4.2BSD is about 53,500 lines of code or about 46% bigger than 4.1BSD. Of this about 4800 lines is VAX-specific C code and about 1100 lines is VAX-specific assembler code. Large multi-user systems with lots of devices and networks use about .5mb main memory for resident code and buffers. Such

---

1 VAX is a trademark of Digital Equipment Corporation.

2 Ethernet is a trademark of Xerox Corporation.
4.2BSD is expected to be available in May. Current 4BSD licensees will receive information as the time approaches. For more information about 4.2BSD write to:

Pauline Schwartz
Computer Systems Research Group
Computer Science Division, EECS
University of California
Berkeley, CA 94720

The paper “Hints on Configuring a VAX” has been revised and is available from the same address.

Bill Joy, Sun Microsystems

Mr. Joy was one of the primary people responsible for the Berkeley Software Distributions. He has recently begun working at Sun Microsystems while helping to finish 4.2BSD.

Mr. Joy sees two de facto standards for UNIX: 4.2BSD and System V. These, he suggested, support two different user communities: those whose needs are similar to those of the ARPA community (very large programs, virtual memory requirements, extensive networking, etc.), and those with more traditional timesharing applications who need to support more users and business and computer center operations.

He suggested that a common base system interface standard be developed so applications software could be run on either system.

Bill Munson, DEC

Mr. Munson announced that “DEC supports UNIX” across its full product line. DEC strongly supports the /usr/group system interface standard and will implement it in VMS/VNX for “commercial use.”

DEC has announced a C compiler for their Professional computer and is considering offering V7 for it. They will be offering binary licenses for UNIX V7 on PDP machines with their V7M-11 product during the summer of 1983. 4.1BSD will be offered for “technical use” late in 1983. On their 36-bit machines they offer the University of Utah Version 7 port that runs under TOPS-10.
UNIX System III and 4.1BSD; a Practical Comparison

John Chambers
Office of Academic Computing & Biostatistics
449 Administration Building
University of Texas Medical Branch
Galveston, TX 77550
(512) 471-3241
lee@utexas-ll

John Quarterman
Computation Center
University of Texas at Austin
Austin, TX 78712
jsq@utexas-ll

A practical comparison of System III and 4.1BSD stemming from production experience with both systems on the same hardware (a VAX-11/780) may be useful to those deciding which to use, especially as the educational license for System III has only recently been made available, and 4.2BSD may not be suitable for many installations. Previous discussions of System III have emphasized its origins, rather than directly comparing it with 4.1BSD.

This paper compares the two systems in several areas, including: initial installation, booting, and configuration; languages, shells, typesetting, graphics, source code control, and data base; terminal handler (fcntl, ioctl, KMC-11 support) and other device drivers; operations, maintenance, and robustness; and games.

Contiguous Load Modules for UNIX

Steve Zucker
Interactive Systems Corporation
1212 Seventh Street
Santa Monica, CA 90401
(213) 450-8363

Most UNIX file input/output is performed a block at a time, the only exceptions being swapping and “raw” disk I/O for which access to an entire logical disk is required. The normal handling of disk allocation and access is quite reasonable for most uses, especially on a multiuser, multiprocess timesharing system in which the independent execution streams tend to randomize accesses to the disk, and in which programs request I/O in block-sized units. However, the loading of programs in response to the exec system call is an obvious candidate for multi-block, contiguous transfers. This is all the more important since program loading is critical to user-perceived response time; there is “dead time” from the issuing of a command until the requested program is loaded. While any UNIX system would benefit from contiguous load modules, they are practically a necessity for systems based on slow-access, high-rotational-latency disks such as today’s floppies.
Two components are required to support contiguous load modules. First, there must be a utility to reorganizes a disk so that the blocks of load modules are stored contiguously. Second, the operating system must be modified to recognize that a program to be loaded is a contiguous file and to process the load module accordingly. A relatively simple reorganization of the dump and restor utilities accomplishes the disk reorganization. Because the memory into which a program is to be loaded is already locked down in exec, it is also a relatively simple matter to have the system read a contiguous load module in a single raw disk operation, once it recognizes that the load module is contiguous.

There are three ways that contiguous load module might be recognized: an indication in the file header, an indication in the file mode word, and dynamic recognition. An indicator in the file header would be “forgable” and has serious security implications. Changing the file mode word or other part of the inode would introduce a serious incompatibility with existing programs. Dynamic recognition, in which the information is gleaned from the block pointers in the inode and indirect blocks themselves is certainly the method of choice. The number of additional disk accesses for reading indirect blocks is on the order of one percent of the number of blocks in the file. Furthermore, files that are not contiguous would normally show up as such within the inode itself, so the overhead involved in the check is negligible for non-contiguous files. In addition, it would probably be worthwhile to take advantage of files that were “almost” contiguous or even those that had only a few contiguous blocks by scheduling input of in-sequence blocks as multi-block transfers, but allowing out-of-sequence blocks by breaking up the input appropriately.

**Improved Schedulers for Non-Paged UNIX Systems**

*Darwyn Peachey*

Hospital Systems Study Group
3337 8th Street East
Saskatoon, Saskatchewan
Canada S7H 4K1
(306) 374-3480
dcvax!harpo!utah-cs!sask!hssg40!peachey

The design of the process scheduler used in a timesharing system can greatly affect the responsiveness of the system, particularly in the case of a non-paged computer under heavy load. At HSSG we have replaced the “vanilla” UNIX Version 7 scheduling algorithms with a new scheduler based on the FB(n) feedback queue scheme. The new scheduler also features: (1) a greater degree of communication between the CPU scheduling and swap scheduling routines; and (2) an improved approach to the management of main memory, aimed at reducing the space lost to external fragmentation.

The presentation will explain the algorithms used in the modified scheduler. The performance effects of the modified scheduler will be illustrated using benchmark results from various PDP 11’s with UNIX Version 7 and System III.

**An Implementation of the vfork System Call for PDP—11 UNIX**

*Micahel Karels*

Department of Molecular Biology
University of California
Berkeley, CA 94720
(415) 642-7359
ucbvax!virus!mike

A vfork system call has been implemented for the Version 7 PDP-11 UNIX system. This system call duplicates the function of fork except that the parent and child processes share data and stack segments until the child process calls either exec or exit. As most forks are followed nearly immediately
by an exec, this avoids copying data which will soon be released without being modified. This method of process creation thus saves the cpu time or swap needed to generate a new copy. The implementation uses a form of scatter loading of processes so that a process in vfork can be treated like any other process. Scatter loading has the additional advantage of increasing memory utilization by reducing fragmentation.

Processes are loaded into memory in four separate sections, one each for shared text, data, user stack, and the kernel per-process data and stack. At the time of a vfork, a new proc entry and user structure are allocated and initialized in the same way as for fork. However, instead of copying the data and stack for the child, the parent’s memory is temporarily associated with the child process. The parent then waits for the child to finish with its data. When the child completes the vfork with an exec or exit, it notifies the parent that it can reclaim its resources and continue execution. The majority of the changes to the kernel are related to scatter loading; the changes for vfork itself add about 45 lines of code.

The vfork system call allows creation of a new process in a much more efficient manner than with fork, saving 60% of the overhead of a fork. It was originally implemented on the VAX because of the problems associated with the copy in fork. This modification is most useful for the shell, csh, and the editor, ex, as well as any other programs with large data segments that must fork in order to start sub-processes. Scatter loading and vfork for the PDP-11 are available as a part of the Second Berkeley Software Distribution (2BSD).

Handling Very Large Programs on a 16—Bit Super-micro

Mitch Bishop
Zilog, Inc.
1315 Dell Avenue, Building 2
Campbell, CA 95008
(408) 370-8000, ext. 4522

Many of the “super-micros” in today’s marketplace are offering more functionality at a continually lower price. As system hardware becomes more powerful and cheaper to produce, system software takes on the burden of making a computer marketable and usable. Zilog’s entry in the 16 bit “super-micro” market is the System 8000, a powerful multi-user system that backs up excellent performance with system software called ZEUS (Zilog Extended UNIX System).

The System 8000 was first shipped in August of 1981 with a full UNIX Version 7 port, along with many new hardware and software features. Its initial performance was comparable to that of the PDP-11 series running UNIX. (See Mini Micro Systems, “Challenging the Minis” September 1981.)

The ZEUS Operating System was updated in June 1982 with further extensions to support the execution of very large (greater than 128kb) programs. Utilizing the efficient segmented architecture of the Z8001 microprocessor, ZEUS offers dramatic advantages over competing UNIX systems.
System V Offering

W. R. Guffy
AT&T

The UNIX kernel has evolved to stability with System V. There have been significant performance enhancements from within and outside Bell. The current functionality is "unalterable" although there could be additions in future releases. There will be future releases of UNIX beyond System V to improve performance and/or add new features. AT&T is committed to upward compatibility with future releases beyond System V. These releases may well be unbundled since the basic system and kernel are to be unalterable.

Mr. Guffy made the point that offering supported software is a new venture for Western Electric. He stated that they are not intending to displace existing companies that are offering products and support.

The documentation on System V is better than that for System III: there is more and there are more types of documents. Licensing contacts for System V remain the same except for overseas which will be handled by AT&T International.

Training and technical seminars on System V will be offered; availability will be announced in the first quarter of 1983.

System V Support Offering

Dave Sandel
Western Electric

System V will be supported by Western Electric from

UNIX Systems Support Center
2600 Warrenville Road
Lisle, IL 60532
(312) 260-4880

Three different markets to be supported were named — domestic, international, and educational — although it was not stated what different treatment these markets might receive.

The basic System V product runs on the VAX 11/780 or 11/750 or the PDP 11/70. A complete list of the supported configurations can be obtained from AT&T Technology Licensing.

There are two levels of support offered. Level 1 includes:
• a monthly newsletter with product-related information and offerings, in-depth discussion of the most recent major problem uncovered, and support center activity and procedure updates;
• periodic updates with bug fixes (only);
• problem reporting capabilities including a guest login; and
• a periodically-updated known problem list with all known System V problems.

Level 2 includes all level 1 services plus more immediate support through a hotline 800 phone number.
One copy of each of the 12 System V documents comes with System V. Additional copies of each may be purchased. It appears from the list of documents that there is much more, and more useful, documentation available than with previous versions of UNIX.

The support center also offers an electronic mail facility for its customers. It is used with an automatic calling unit and uucp.

Licensing Activity and Pricing

Larry K. Isley  
AT&T  
Guliford Center  
P.O. Box 25000  
Greensboro, NC 27420

Five C compilers and three versions of phototypesetter software are being offered for vendors to offer to their customers. The end-user offering must be binary code only. The compilers are C/370, C/SEL, C/6000, C/DATA, and C/CRAY. The phototypesetter versions are typesetter-independent troff, phototypesetter/PWB, and phototypesetter/V7. Any C compiler or phototypesetter licensee may also provide their customers the C compiler or phototypesetter software from any of the UNIX systems. No up-front fee is required. The end-user fee for any of these is $200.

The following table shows the number of UNIX source licenses and installations worldwide as of December 1, 1982.

<table>
<thead>
<tr>
<th>Software</th>
<th>Commercial</th>
<th>Educational &amp; Admin.</th>
<th>Government</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lic</td>
<td>inst</td>
<td>lic</td>
<td>inst</td>
</tr>
<tr>
<td>Mini</td>
<td>6</td>
<td>8</td>
<td>128</td>
<td>376</td>
</tr>
<tr>
<td>V6</td>
<td>90</td>
<td>170</td>
<td>360</td>
<td>958</td>
</tr>
<tr>
<td>PWB</td>
<td>48</td>
<td>133</td>
<td>65</td>
<td>246</td>
</tr>
<tr>
<td>V7</td>
<td>144</td>
<td>237</td>
<td>359</td>
<td>1092</td>
</tr>
<tr>
<td>32V</td>
<td>71</td>
<td>121</td>
<td>206</td>
<td>435</td>
</tr>
<tr>
<td>SIII</td>
<td>176</td>
<td>265</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>TOTALS</td>
<td>535</td>
<td>934</td>
<td>1122</td>
<td>3111</td>
</tr>
</tbody>
</table>

Commercial source licensing fees are given in the following table along with the fee to upgrade to System V if you have the specified license(s).
Software  | Initial CPU | Additional CPU | Customer CPU's | Upgrade to System V
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mini</td>
<td>12,000</td>
<td>4,000</td>
<td>8,400</td>
<td>26,000</td>
</tr>
<tr>
<td>V6</td>
<td>20,000</td>
<td>6,700</td>
<td>12,000</td>
<td>16,000</td>
</tr>
<tr>
<td>PWB</td>
<td>30,000</td>
<td>10,000</td>
<td>11,700*</td>
<td>18,000</td>
</tr>
<tr>
<td>V7</td>
<td>28,000</td>
<td>9,400</td>
<td>18,000</td>
<td>6,000</td>
</tr>
<tr>
<td>32V</td>
<td>40,000</td>
<td>15,000</td>
<td>16,000</td>
<td>1,000**</td>
</tr>
<tr>
<td>SIII</td>
<td>43,000</td>
<td>16,000</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>SV</td>
<td>43,000</td>
<td>16,000</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>UNIVAC</td>
<td>30,000</td>
<td>10,000</td>
<td>12,000</td>
<td></td>
</tr>
<tr>
<td>TSS</td>
<td>100,000</td>
<td>20,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V6+V7</td>
<td>14,000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V7+PWB</td>
<td>4,000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* other schedules apply
** one time for all CPU's

Educational and administrative licensing fees to upgrade to System V are given in this table.

<table>
<thead>
<tr>
<th></th>
<th>Ed.</th>
<th>Admin.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial request covering all CPU's</td>
<td>800</td>
<td>16,000</td>
</tr>
<tr>
<td>Each additional CPU after initial request</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>Upgrade all CPU's from System III</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Upgrade all CPU's from other UNIX license</td>
<td>800</td>
<td>16,000*</td>
</tr>
</tbody>
</table>

* some credit allowed for previous UNIX license

The fees for System V support for a data center are:
level 1  $150 per month for the first CPU and $50 per month for each additional CPU.
level 2  $350 per month for the first CPU and $100 per month for each additional CPU.
A fee of $100 per hour is charged for out-of-hours service, or support services not covered in the level 1 or level 2 offering.

Further information on licensing and support contracts is available for domestic sites from AT&T Technology licensing at (919) 697-6530 and for foreign sites from AT&T International at (201) 953-7581 or TELEX-219365 ATT-I-UR.
The Plexus Networked UNIX

Monte Pickard  
Plexus Computers, Inc.  
2230 Martin Avenue  
Santa Clara, California 95050  
(408) 988-1755  

Plexus has designed and implemented a user transparent Distributed File System and Virtual Terminal capability under the standard UNIX System III.

The file system involves an extension of the mount(1M) command such that any branch of any file system on the network may be mounted on a branch of the local file system. Accesses to a path that crosses that mounted branch are satisfied on the remote system. All buffering is done on the system that contains the file system, as well as permissions granted based on the /etc/passwd of that system.

The Virtual Terminal capability consists of an implementation of the ‘tty’ characteristics over the network. All programs that utilize a tty interface may utilize the Virtual terminal capabilities. Local, as well as remote efficacy of ioctl provides expected response at both ends of the virtual circuit established.

The interface to the user is established at the system call level with the specific details of network media or technology transparent to the system code implementing the desired capability. This is done through the definition of a software architecture for the communication function that provides general virtual circuit and datagram capabilities through implemented networks.

The development was done utilizing the ethernet local area network technology. An X.25 implementation, as well as a broad-band implementation are next to be developed.

CSNET Status Report

G. Brendan Reilley  
Department of Electrical Engineering  
P S Dupont Hall NC12  
University of Delaware  
Newark, Delaware 19711  
302-738-1266  
Reilley@UDel  

The CSNET project is an inter-institutional effort to provide a nationwide network of computer science research departments. It includes both commercial and academic members, and is intended to be self-supporting, running on a pay-by-use basis. The initial funding has been provided by the National Science Foundation, and this seed money has carried the network into production. The initial implementation of CSNET supports mail transport only, though other services are planned for the future. CSNET currently supports a Coordination and Information Center, two mail relay machines, and a Service Host, which provides a nameserver for the network.

CSNET is a logical network rather than a physical network. This means that it uses existing transport mechanisms, rather than providing a new physical backbone. Currently it uses Telenet, Arpanet, and the dial-up telephone network. Approximately fifty member sites are currently being serviced.
SENDMAIL — An Internetwork Mail Router

Eric Allman
Britton-Lee, Inc.
1919 Addison, Suite 105
Berkeley, CA 94704
(415) 548-3211
ucbvax!eric

Routing mail through a heterogeneous internet presents many new problems. Among the worst of these is that of address mapping. Historically, this has been handled on an ad hoc basis. However, this approach has become unmanageable as internets grow.

Sendmail acts as a unified “post office” to which all mail can be submitted. Address interpretation is controlled by a production system, which can parse both old and new format addresses. The new format is “domain-based”, a flexible technique that can handle many common situations. Sendmail is not intended to perform user interface functions.

Sendmail will replace delivermail in the Berkeley 4.2 distribution. Several major hosts are now or will soon be running sendmail. This change will affect any users that route mail through a sendmail gateway. The changes that will be user-visible will be discussed.

W4B — Development Tools
Wednesday, 3:30—5:00
Bill Tuthill, chair

Reporter: Mark Bartelt
Research Development Corporation
170 Elizabeth Street
Toronto, Canada M5G 1X8

COBOL Compiler Construction Experiences Using lex and yacc

Robert E. Conant and Herbert G. Mayer
Burroughs Corp.
Bldg. MC03, Mail Station 703
16701 West Bernardo Drive
San Diego, CA 92127
Conant: (619) 451-4584; sdcsvax!sdcattb!xm95bc
Mayer: (619) 451-5175; sdcsvax!sdcattb!xm95hm

COBOL? Why even mess with it in the first place? Like it or not, among programming languages, COBOL pays the bills for most computer vendors. The speakers described their experiences with using lex and yacc to build a COBOL 74 compiler. It was hoped that by using these tools they would be able to reduce development time, and produce a compiler that would be easier to maintain than compilers produced by traditional methods.

A number of problems were described, some each in the scanner and parser designs.

The Scanner

COBOL permits any token to be split across lines! This, combined with COBOL’s fixed-format source lines, is something that lex has trouble dealing with. The difficulty was resolved by providing a pre-pass which concatenates split tokens.
The number of keywords is enormous: 300 or so, compared with 28 or so with C. A first attempt might be to provide one lex rule for each keyword. Unfortunately, a scanner built this way tends to run for hours, only to terminate abnormally by running out of space. The solution was to put all 300 keywords into the symbol table, and to provide a single rule which is capable of recognizing all COBOL keywords as a general class.

Another problem is the PICTURE clause, which is too complex in general for the scanner to be able to recognize whether or not it's well-formed. The solution was to provide a series of rules which recognize a PICTURE string and leave the real work of verifying its correctness for the parser.

Significant blanks are another problem. For example, arithmetic infix operators are required to have significant blanks around them. To deal with this, each infix operator has two rules provided, one for the operator properly surrounded by blanks, and a second for the same operator without the blanks. The latter issues an error message, but returns the same token as the first, so that scanning can continue.

The COPY statement is COBOL's equivalent of #include. The problem is that it can appear ANYWHERE in the source text. This is dealt with by having the same pre-pass which concatenates split tokens also handle the COPY statement.

The Parser

The first problem is that there is NO statement terminator. How do you recover gracefully after finding an error? How do you find the next statement? This problem was solved with a simple heuristic which involves using a global variable for communication between yacc and lex. A flag is set by yacc which tells lex to deviate from its normal mode of operation in which line terminators are ignored and tossed away, and instead watch for an end-of-line, and, when found, turn the error state off and continue scanning.

Other things that needed to be dealt with included 'dangling else' (and 'dangling other things') problems, and COBOL lists (some types of which can have zero elements, some require at least one element, and some require two or more!).

Development of refer: Bug Fixes and Enhancements

(or unofficially "Refer Madness")

Bill Tuthill
University of California, Berkeley
Computing Services
Berkeley, CA 94720
(415) 642-1307
ucbvax!g:tut or g.tut@berkeley

Refer is a bibliographic tool which, by itself, doesn't really fit the needs of a university community. Accordingly, a number of programs were created to make a bibliographic system easier to use. These include tools for creating and extending bibliographies, sorting them by arbitrary keys, formatting them (using nroff), creating indices, and so forth.

The -ms macros were revised somewhat, with a resulting increase in speed.

Refer itself was enhanced. Changes were made and options added so that it would provide the following new features [I couldn't write fast enough to get them all down; sorry.]:

1. Insert square brackets around references.
2. Permit setting of footnote numbers.
3. Sort arbitrarily large numbers of references without dumping core.
4. Turn off the automatic searching of /usr/dict/papers.
5. Produce endnotes instead of footnotes.
6. No longer behave anomalously in the presence of trailing white space.
7. Permit ‘corporate authors’ (i.e. names are sorted by first word of author’s name).
8. Collapse ranges of consecutive numbers (i.e. M-N instead of M,M+1,...,N-1,N).

**RAPID: A Tool for Building Interactive Information Systems**

*Anthony I. Wasserman and David T. Shewmake*

Medical Information Science, Room A-16
University of California, San Francisco
San Francisco, CA 94143

(415) 666-2951
ucbvax!wasserman or wasserman@berkeley

How does one go about designing interactive information systems? It’s important, of course, that the user-program dialogue be friendly and usable, and that the software be reliable and reasonably efficient. The end-user, however, isn’t really concerned about the details of the underlying software. [Good quote: “But, let’s face it — Most of the users don’t care what programming language you use. They don’t CARE whether or not you’ve used ‘GOTO’ statements. They simply want the program to work for them when they want it to work. And they want it to be easy to use, and use terminology that’s comprehensible to them, and not muck up their screens with a lot of low-level error diagnostics.”]

In general, an interactive information system can be characterized by: (1) a user-program dialogue, (2) an underlying database, and (3) a set of operations on the database (e.g. queries, modifications) which are effected by the user-program dialogue. User Software Engineering (USE) is a methodology developed for the purpose of building systems of this sort in an orderly and rational fashion. The emphasis is on understanding the methodology involved in designing interactive information systems, and designing tools to fit that methodology, rather than the other way around.

One important goal is to ensure that the people who will ultimately be using the software be involved in its design from the very start. After some initial analysis and modeling, and identification of relevant user characteristics and skills, a decision is made — very early on — about what the user-program dialogue is going to look like.

RAPID is a tool which permits early prototyping by supporting a language which can be used to define nodes and transitions between them. A transition is triggered by a user input, and transitions have particular actions associated with them. RAPID’s ‘transition diagram interpreter’ can thus be used to create a prototype system which links a description of the dialogue with the database and with a set of actions on that database. The ability to do fast prototyping permits effective user involvement in the design of the overall system and allows designers and users to experiment with different user interfaces.

Moreover, the ability to experiment with alternative user interfaces to the same set of operations provides an opportunity to evaluate (both subjectively and objectively) users’ performance with different types of user-program dialogue, and thereby make some judgments about what sort of user interface is most appropriate.

Rapid prototyping also gets users actively involved in thinking about what the system will ultimately look like, with the result that they’re apt to start thinking about features or functions that might not otherwise have initially occurred to the developers, or, for that matter, to the users themselves.
The UNIX System: New Directions

Berkley A. Tague
UNIX Development Laboratory
Bell Laboratories
Murray Hill, NJ

This talk reviewed the history of UNIX and discussed the highlights of and new directions being taken with System V.

The ancestry of the C language was traced from CPL (Strachey) to BCPL (Richards) to B (Thompson) to C (Ritchie). The ancestry of the UNIX operating system showed BESYS (Bell Labs) and CTSS (MIT) influencing Multics (Bell Labs, GE, MIT) and Multics and Genie (UC Berkeley) influencing UNIX Versions 1–5 (Thompson). The Version 6 to Version 7 transition was centered around portability research while the V7 – 32V – System III transition was oriented toward improving program development tools.

Bell uses UNIX for a wide variety of applications including program development, document preparation, telephone network operations support, laboratory control, and manufacturing support. In phone support use UNIX is usually used with some data base system. In laboratory and manufacturing use they typically use satellite processors to handle data interrupts.

System III (and V) came from the integration by the UNIX System Developers of the systems developed by the research group (V6, V7, and 32V) and those developed by the Applications Developers (PWB 1.0). The offering of System V is the first time the current working Bell version of UNIX has been offered. In the past the version offered was typically two years behind that being used inside Bell.

A list of the features of System V was presented: file system enhancements, performance improvements, improved interprocess communications, many quality improvements (about 1600 fixes and enhancements), library improvements, uucp improvements, ex and vi editors, accounting enhancements (raw data collection), line printer spooling system, improved operator interface, new peripheral support, and improved documentation. [Most of these topics were covered in some detail later in the session.]

New directions for UNIX within Bell were listed as:

- control of system evolution through a system definition, the ANSI C standard, and the /usr/group and European UNIX Users Group standards efforts;
- meeting market needs of unbundling products and “small” UNIX systems;
- providing compatibility between versions;
- retaining portability and simplicity; and
- continued development and support.

In response to a question Mr. Tague stated that the research group that developed V6, V7, and 32V will not be offering any more complete UNIX systems.
UNIX File System Evolution

Larry A. Wehr
UNIX Development Laboratory
Bell Laboratories
Murray Hill, NJ

This talk enumerated the file system improvements made between Version 7 and System V. The objectives in this development were to maintain upward compatibility, increase performance and reliability, and to maintain a simple and uniform environment.

The file capacity has changed from 1mb files and 32mb file systems with V6 to 1gb files and 8gb file systems with V7 and System III to 16gb files and file systems with System V (with 1kb blocks). System V has a new compatible file type with 1kb blocks for improved performance. A flag has been defined near the end of the superblock to indicate the file system type. System V can have (different) file systems with both 512 byte and 1kb blocks existing on the same system; the utilities will handle either type. It can handle files from previous versions back to V7.

A number of changes have been made for file system reliability including ordered writes for create/link/unlink. There are also several performance enhancements besides the new 1kb block file type. These include an expanded range for the number of system buffers (up to 1000), a physical I/O buffer pool, and increasing the stdio buffer size to 1024 bytes.

Future work on the file system is being done in the areas of networking and data base applications, including record/file locking, archiving, and access methods.

Evolution of UNIX System Performance

Jerome Feder
UNIX Development Laboratories
Bell Laboratories
Murray Hill, NJ

This talk told of the changes in System V that make it run 25% faster than System III on a VAX 11/780 on their timesharing benchmark. The same benchmark showed very little performance change when run on a PDP 11/70. Very little was said of the 11/750; it appeared to be about the same as an 11/70.

The C libraries, and especially stdio, have had code refinements, have some better algorithms, use line-at-a-time output to terminals, and have had some routines recoded in assembly language.

Many changes have been made in the kernel. System call overhead and context switch are around 25% faster on a VAX. Inode search time is now constant on a VAX because a hashed search is used. (The 11/70 still uses a linear search.) Pipe bandwidth is approximately unchanged for 512 bytes per transfer but is about 2/3 higher for 1kb per transfer (VAX only). Disk I/O is reduced with the larger (1kb) buffers and the larger number of buffers available.

Raw and normal terminal I/O run at the same speed on a VAX with the programmed KMC/DZ! However, terminal I/O on a normal DZ on a VAX runs slower than on an 11/70 with a DH11.

Bell's timesharing benchmark measures system throughput for a standard set of commands under increasing load. It is modeled on the Bell Labs program development environment. There is no networking or system administration and user actions are read from files, not terminals. There are no swaps.

Several questions were asked:
Assembly language in the C library: every function that can be expressed in C is; for some there are also assembly language versions.
Demand paging: System V does not do it on a VAX and they will not commit as to when it might be implemented.

System V vs. 4.1: 4.1 is about 12% faster than System III and 12-13% slower than System V on their benchmark.

C Programming Environment

D. J. Kretsch
UNIX Development Laboratory
Bell Laboratories
Murray Hill, NJ

This talk discussed a proposed standard C environment based on System V and talked about software generation systems used with Bell Labs. The standards effort is aimed at increasing the portability of C code and making it more usable on small systems.

The proposed standard C environment has two levels: level one with core features that would be required for all C environments, and level two with additional UNIX-specific features. Level 1 would be a proper subset of level 2. Local features could be added to either level. The standard would cover the language (based on the Kernighan and Ritchie book), C library, stdio, the math library, and the C preprocessor.

[It was unclear to me how this standard would be formalized: through the ANSI C standard committee? Or would it be just a Bell definition? ...Ed.]

Mr. Kretsch also described the Software Generation System (SGS), a set of compilation and object file manipulation tools for developing software for the host UNIX environment or other computers. The tools are divided into machine dependent and machine independent pieces. The tools include a C compiler, assembly language optimizer, assembler, linkage editor, disassembler, various object file manipulation routines, and a library of object file access routines.

SGS runs on many host processors, and it is easy to change the target processor. The underlying feature that allows SGS to be portable is a common object file format. This is consistent across all processors and it allows such things as the definition of memory directives for the linkage editor.

UNIX System Definitions and Standards

Don Cragun
UNIX Development Laboratory
Bell Laboratories
Murray Hill, NJ

This talk described ongoing work within Bell to define a full and subset (Bell) UNIX system. The subset — level 1 — would be for small systems. Level 2 would be a full UNIX system — System V is an example.

At each level the definition would include hardware, operating system calls and drivers, and utilities and libraries. Each level would allow for the addition of packages including hardware to replace or add-on features.

The standards effort is based on System V. They are working with the /usr/group Standards Committee. The standard is to cover all level 1 system calls, selected level 2 system calls, and the level 2 C environment. Part of the intent of the effort is to control the evolution of UNIX and to provide UNIX releases with upward compatibility “wherever possible” and “advance notice of non-compatible changes.” [But see Mr. Guffy’s comments on compatibility in session W3B...Ed.]
Questions and Answers

[Just the answers were recorded; hopefully the questions can be inferred.]

Dr. Bob Fabry of UC Berkeley made several comments: Bell is aggressively importing good ideas developed outside the Labs [like the larger buffer]; support of UNIX by Bell as well as DEC and other vendors is important.

Not much work has been done on argument specification compatibility.

SGS is on the VAX only and not all the pieces are there.

There have been some improvements in real-time capabilities; they are interested in hearing what improvements are needed.

The symbolic debugger is called sdb; it works at the C statement level.

The question of defining whether things like nroff and yacc should be part of level 2 is being looked into.

System V handles both the old and new a.out formats; files in the old format need to be converted only if they are going to be processed by the link editor.

SGS is used at Bell for cross compilation to the 8086, 68000, and Western’s 32-bit processor. It has not been decided whether they will offer a 68000 or 8086 version of SGS.

System V has not been run on the PDP-11/44; Bell decided long ago to ignore it.

T1B — UNIX in the Office Environment

Thursday, 8:30—10:00

Keith Muller, chair

Reporter: F. Arlene Spurlock
Technical Information Department
Lawrence Berkeley Laboratory
Berkeley, CA 94720

A Uniform and Simple User Interface to UNIX

Spencer Rugaber
Interactive Systems Corporation
P.O. Box 3587
Estes Park, CO 80517
(303) 586-4463

This presentation described an interface to UNIX which has been under development at Interactive Systems Corporation’s Estes Park, Colorado facility. Their stated goal was to produce software that would be self-teaching and easy to use by non programmers. Their results, a full-screen interface that front-ends UNIX and a product upon which an entire office automation system has been based.

ISC’s solution was a program which runs on top of UNIX that utilizes a full-screen editor with a function key-pad to execute some basic operations. They have tried to limit the number of commands the user must confront, as well as bring continuity in command usage across the fields of usage, i.e., mail, word processing or database retrieval.

After running a prototype system for the last two years at their Colorado facility, they claim that someone could begin working with this new interface within a half-hour using the self-help that is included within the system.
UNIX Time-sharing Menu-driven Office System for Terminals (UTMOST)

James A. Neyer
Perkin-Elmer
Computer Systems Division
106 Apple Street
Tinton Falls, NJ 07724
(201) 870-5841

This presentation covered the enhancements made by a group at Perkin-Elmer to UTMOST which is a menu-driven office system developed by the Air Force Data Services Center.

Initially, they started this project by converting it from V6 to V7 UNIX, and then proceeded with "de-militarizing" to make it more suitable for commercial applications.

The present version has eight sub-systems: help, word processing, document management, electronic mail, electronic bulletin board, (these last two based on the Rand interface, though another could be used) an executive support package for things like do-lists, appointment schedules, a system information sections, and finally a facility for extending the system with your own personal menu or hierarchy of menus, without recompilation of source programs.

This package is available now and they have been looking for beta-sites. It will also be made available for the next software distribution tape from USENIX.

A Friendly Text Processing Environment

Arthur Zemon
Software Productivity Project
TRW R2/2009
One Space Park Drive
Redondo Beach, CA 90278
(213) 535-6166
ucbvax!randvax!trw-unix!trwsp!zemon

AUTHOR is a program developed as part of an ongoing software productivity project internal to TRW.

The user interface of the editor is similar to the Berkeley vi editor which takes AUTHOR based text and displays nroff'd like output on the user's screen.

Their approach was to put together something which looks like a dedicated word processor running under UNIX. AUTHOR's interface supplants the complexities associated with nroff/troff with a set of commands which the user must learn and some special terminal keys for formatting functions like paragraph, section headings, etc.

The text which the user sees approximates what one would get after it had been processed by nroff. An nroff/troff input file is produced by AUTHOR. A positive plus is that AUTHOR still allows raw n/troff formatting commands to be inserted into the AUTHOR file. If AUTHOR doesn't understand these, it leaves them as is for processing later by n/troff.

There are no commercial marketing plans at present for AUTHOR, though AUTHOR's authors will proudly show it off if you are interested.
Writing User Documentation for UNIX Systems

Jean Yates and Rebecca Thomas
Yates Ventures
P.O. Box 22411
San Francisco, CA 94122
(415) 753-5950
ucbvax!g:becca

A quick and lively rundown of Yates Ventures which covered a brief review of the recent marketing survey, along with short, verbal vignettes of the new UNIX user who is not the computer scientist/data processing professional of today’s UNIX.

They have several books forthcoming that are targeted for the business-oriented micro user, and programming oriented text targeted for the commercial applications programmer who only wants to know how to make the system work for their purposes and needs.

In a nutshell, “a system manual is just not enough for the new breed of end users” and Yates Ventures is actively filling this void.

T2B — New UNIX Implementations I
Thursday, 10:30—12
David Patterson, chair

[Notes for this session are not available so the abstracts for the talks have been printed...Ed.]

Hewlett-Packard’s Entry into the UNIX Community

Frederick W. Clegg
Section Manager
Technical Computer Group Engineering
Hewlett-Packard Company
11000 Wolfe Road
Cupertino, California 95014
(408) 257-7000
ucbvax!hpda!fc

After more than two years in preparation, Hewlett-Packard Company has made a major commitment to the UNIX operating system. The first public unveiling of HP’s support for this burgeoning industry standard occurred in November 1982 in conjunction with the announcement of the powerful new HP 9000 series of 32-bit computers.

The HP 9000 employs a new high-density, high-speed NMOS technology to achieve performance levels heretofore unapproached in a desktop workstation. The CPU chip alone of the HP 9000, for example, contains nearly a half-million transistors. The 32-bit system architecture provides features to support high-performance virtual memory, networking, and multiple CPUs. The resulting workstation is targeted primarily at scientific and computer-aided-engineering applications.

Shortly after the introduction of the HP 9000, HP will announce its support for UNIX on its Series 200 desktop line (two members of which are the popular HP 9826 and HP 9836 computers). These machines employ the widely-used MC68000 processor in a compact and cost-effective package aimed at instrument system control, “computer-aided-work,” and other applications not requiring the expandable performance of the HP 9000.
Many of the APSE requirements are similar in notion to UNIX — simple and consistent interface, tool concept, powerful command interpreter. The database APSE requirements are significantly beyond what is normally found on UNIX systems.

UNIX System V meets many of the APSE requirements. Significant extensions will be required to meet the database requirements.

**UNIX Logo**

*Brian Harvey*
Atari, Inc. and SESAME Group, UC Berkeley
Atari, Inc.
1196 Borregas Avenue
P.O. Box 427
Sunnyvale, CA 94086
(408) 745-0510 (Atari: Tuesday and Wednesday)
BH@SU-AI

This talk gave a brief introduction to the Logo programming language and its use in pre-college education. Mr. Harvey also announced the availability of Logo on UNIX and that he has submitted an updated version of it to USENIX for the 1983 Software Distribution Tape.

Logo was designed for learning programming and for learning in general. The ideas came from artificial intelligence research and LISP and Piaget's research into how children develop thinking skills. It is procedural, interactive (interpreted) and recursive. It has list processing and is not typed.

Part of Logo is the notion of turtle graphics, a simple and simply-expressed method of doing graphics. A turtle on the screen has a relative position and direction. It can be rotated or moved and can leave a trail of its movement (i.e., draw a line).

**LINUS (Leading Into Noticeable UNIX Security)**

*Steven M. Kramer*
Mitre Corporation
M/S B330
P.O. Box 208
Bedford, MA 01730
(617) 271-3874
...linus@sumk

This talk described various types of security problems with UNIX and two levels of solutions to the problems.

The major security problems with UNIX were listed:

- global (unrestricted) privileges like those available with `su`;
- authentication and simple passwords;
- "trojan horses" — utilities that look like normal commands but that perform other or additional things;
- integrity and denial of services (e.g., printing $10^6$ bytes);
- badly designed system programs;

---

1 available to 1983 Institutional members of USENIX with any UNIX license

# There is a informative article on Logo by Mr. Harvey on page 163 of the August, 1982, issue of *Byte*. 

April 1983

Volume 8, Number 2
• leniency in file mode settings; and
• lack of mandatory security levels.

Linus III is an add-on, portable solution to these problems that is built on existing UNIX security features and requires no kernel modifications. It tries to limit superuser use as much as possible, allows privileged command auditing, and does file system integrity checking. Linus III has been implemented but is not packaged. An accompanying database to provide additional controls has been designed but has not been implemented.

Linus IV is a more extensive package that is based on 4.1BSD. It adds Department of Defense security levels to UNIX as well as handling the other problems mentioned above more completely than with Linus III. It requires kernel modifications as different security levels must be in different hardware domains. Changes must also be made to about six programs.

A Global Optimizing C Compiler

George Powers
Zilog, Inc.
MS A2-5
1015 Dell Avenue
Campbell, CA 95008
(408) 370-8000, x4603

Zilog has implemented a global optimizer for C on their System 8000, which uses a Z8001 microprocessor. The optimizer works for either segmented (23 bit addressing) or nonsegmented (16 bit addressing) mode.

The compiler development was done on a descendental of the portable C compiler. Optimization is done in an optional pass after parsing but before code generation. The parser symbol table had to be retained until the code generation pass so the optimizer would have access to it.

T3B — Novel Applications of UNIX

Thursday, 1:30—3:00

Jim McGinness, chair

Reporter: Scott Thompson
2330 Plum
San Diego, CA 92106

UNIX Research at Lucasfilms

Jim Lawson
Computer Research and Development
Lucasfilms Ltd.
P.O. Box 2009
San Rafael, CA 94912
(415) 499-0239
ucbvax!dagobah!jrl

Lucasfilms is the creator of such films as Starwars and Star Trek. Mr. Lawson, head of the Computer Research and Development Division, related the current activities of Lucasfilms and the role UNIX plays in them. Deeply involved in the Systems Project whose chore is the maintenance of the many computer systems at Lucasfilms, Mr. Lawson outlined the increasing dependency the company
has on computers.

From administrative chores such as word processing, mail, and archiving, to the Industrial Light and Magic Division special effects and film post production, to the Computer Division graphics, game production in tandem with Atari, the computer and UNIX are integral to Lucasfilms.

Currently utilizing four 11/750's, one 11/780, and a network of 68000's, the company is awaiting VAX 4.2BSD support. John Seamons and Bill Reeves have created an extended file system with network—wide access to all files utilizing ethernet which requires no local discs.

Lucasfilms is active in digitizing as much of the process of movie production as possible; from their Audio Project which uses a 68000 as a controller for their audio signal processor, to their Video Project whose main goal is discovering better ways of utilizing video technology in a film oriented domain.

Lucasfilms has found UNIX an ideal operating system as it has allowed a uniform environment for the many different activities presently occuring in the company.

**ARIEL: An Experimental UNIX-based Interactive Video Information System**

*R. C. Haight and D. B. Knudsen*

Bell Laboratories

(201) 564-2241

ucbvax!allegra!bwkna!haight

ARIEL is an experimental UNIX-based interactive video information system designed by Bell Labs for Disney World’s EPCOT Center. The system consists of laser video discs, video overlay graphics, a tune synthesizer, touch screen input devices, phone connections, and up to fifty user terminals located throughout the Park.

Users interact with the system by touching the menu displayed on the touch sensitive screens; information which can be accessed ranges from current events to geographic history. Software for the system includes routines which interface to the above mentioned hardware via RS232 connections, shell level utility commands which use these routines to perform tasks on the devices, a music compiler written in C, a yacc-based script compiler used to compile the language which controls the devices, a script interpreter consisting of compiler output plus additional files, and high level control programs to handle interprocess and intermachine communications. The script language used is based on a single data type (strings) and includes many device primitives — video, audio, touch — as well as a C pre-processor used to define macros.

UNIX as a base allowed the development of the system in a timely manner with relatively few people.

**Development of a Digital Simulation System in a UNIX Environment**

*William Raves and James Cassidy*

Computer Automation

2102 North Forbes Boulevard

Suite 107

Tucson, AZ 85745

Computer Automation has been able to utilize UNIX in the development and implementation of an applications system used in the design and testing of printed circuit boards. The system consists of a slave processor "Digital Simulation System" which runs UNIX, linked to a resource manager hooked up to numerous single user cpu’s dedicated to testing. Simulation and testing involves defining the elements and topology of the particular circuit board, performed at the simulating stations; once simulated, the model is tested for possible faults at the testing stations.
In that much of the process of design and testing is based on feedback, the goal of the software engineer was ease of modification. UNIX lends itself well to this; by using shell script for control flow Computer Automation arrived at the flexibility that was necessary; by converting these shell scripts to C programs, increased processing speed was attained.

Many of the concerns of Production Management were quelled by the UNIX Operating System. The hierarchical file system aided in the complexities involved in a multi-shift production environment allowing the loading of appropriate file systems for each shift. In addition, the shell environment proved useful to the security needs of the management.

UNIX allows a flexible development environment and control of that environment; in addition, the portability of UNIX allowed the development of the software on a totally different machine, cutting a significant amount of set-up time.

Meeting the Coming UNIX Training Challenge

Jay R. Mosier
User Training Corporation
P.O. Box 970
Soquel, CA 95073
(408) 462-6527

Like many, User Training Corporation envisions an explosive future for UNIX; the already apparent influence this operating system has had on the industry could easily spread to retail sales systems, office automation, and even home computing, making UNIX the operating system choice among the 16 bit microprocessor systems.

A prerequisite for this growth is a cost effective means of training the large number of new users. The past has provided training via seminars, personal instruction, and commercial books. User Training Corporation has come up with a novel training technique utilizing the medium of Audiodigital Recordings.

Audiodigital Recordings pair up an audio track with a track of digital information to provide the illusion of someone using the system while explaining what he is doing. They call this "tutor simulator." Much more cost effective than a personal tutor, the Audiodigital method does not require a computer; once encoded, all that is necessary is a digital converter and a terminal.

Audiodigital recordings were devised at Stanford over five years ago but User Training Corporation is the first to exploit this innovative medium for demonstrating the use of interactive systems. User Training Corporation has a continuously growing library of UNIX tutorials and recognizes the potential for this medium in all sorts of training needs.
UNIX for the STD Bus

Luigi Cerofolini
Istituto Di Matematica Applicata
Facolta Di Ingegneria
Universita Di Bologna
Via Vallescura 2
40136 Bologna Italy
(051) 331588

The STD bus, jointly developed by Mostek and Pro-Log around mid-1978 and now being in the public domain, has gained wide acceptance among designers and, with over 70 manufacturers producing board based products for it, it is one of the fastest growing buses of the past ten years. So it was natural to put UNIX, one of the most popular operating systems available today, on the STD. The CPU choice was very natural. The STD data bus is 8-bit wide while UNIX was originally designed for a 16-bit CPU: The Intel 8088 CPU and associated coprocessors 8089 for I/O and 8087 for number crunching seemed to satisfy nicely the general system requirements. Because of the small size of the STD boards (6.5 in. x 4.5 in.) we have been forced to work on a very modular multi-processor system and this turned out to be the key factor in order to have a high performance system.

We divided the system into four main subsystems or Functional Units (FU): CPU, terminals, disk and tape, and memory. In order to offload the CPU from low level peripherals control activities we needed intelligent I/O controllers, but this was contrasting with the very small board size. So we opted for a two-board solution for every FU (except memory—that fits nicely into one STD board): one of the two boards, the Universal Processor Unit or UPU, is the same for all FUs, while the other board, the Special Unit or SU, is tailored to specific functions. The two boards of the same FU are connected together through a flat cable.

This possibility of sharing the same hardware between different FUs was a decisive fact for the success of the project: we had the same well known and debugged piece of hardware (the UPU) as the starting point for all new specialized SUs we needed for the system.

All the UPUs (Universal Processor Units) share the same hardware design and their main components are: the 8088 CPU, one free 40-pin socket (to be used for a coprocessor like the 8089 or the 8087 or for a Silicon Real Time Operating System Kernel like the 80130) whose use is application dependent, latches and STD buffers, logic for DMAing into the system STD bus, local ROM and RAM and a jumper configuration array.

The Special Unit structure can be usually considered as an I/O adapter: for the Terminals Controller it is simply an array of programmable UARTs; for the Disc and Tape Controller it is an adapter to a popular intelligent formatter/controller; for the CPU board it is a Memory Management Unit plus Timer and Interrupt controller logic.

We are now developing controllers for a Laser Printer and for a Graphic CRT and we found the two boards-set approach very powerful and flexible.

All FUs are the same to the operating system and this uniformity has various nice consequences like easy maintenance, short design time, good performance and low cost.
Ctrace - A Portable Debugger for C Programs

J. L. Steffen
Bell Laboratories
Room 2C-331
Naperville-Wheaton Road
Naperville, IL 60566
(312) 979-5381
ucbvax!ihnss!ihuxs!steffen

Ctrace is an easy to use, portable C language debugging tool that lets you follow the execution of a C program, statement by statement. The effect is similar to executing a UNIX shell script with the -x option. Ctrace is far easier to use than adb or sdb and, unlike these debuggers, it is completely machine independent so it is highly portable. It has even been used to debug C programs in non-UNIX environments.

A PDP-11 UNIX System with 4.1BSD VAX-11-style Filesystems
(or one more fun thing to do with your old PDPs)

Donn Seeley
University of California at San Diego
Department of Chemistry B-014
La Jolla, CA 92039
(619) 452-4016
ucbvax!sdcsvax!schemc!donn

8.2 is a UNIX operating system for the PDP which has filesystems that conform to the structure of the filesystems of the VAX running 4.1BSD, thus allowing a PDP to mount VAX filesystems or vice versa in a dual-ported disk or controller situation. The system consists of a modified Ritchie C compiler and C library, a 2.8BSD kernel and debugging tools, certain 2.8 utilities for the small PDP such as a small csh and vi, and as much 4.1BSD software as will run without modification, including fsck, df, getty and so on. Advantages of the system include rapid and trivially simple file transfer between PDP and VAX, reciprocal filesystem construction and maintenance, and the capacity to spread the availability of peripheral devices to more than one CPU (e.g. PDP files may be backed up to tape from the VAX, all printing can be channeled through one PDP, etc.). A side benefit of the particular implementation is more complete binary file compatibility between PDP and VAX.

The IS/1 Workbench for VAX/VMS

William Torcaso
Interactive Systems Corporation
1212 Seventh Street
Santa Monica, CA 90401

As the successor to the DEC PDP-11, the VAX 11/780 is obviously a suitable machine on which to implement UNIX. VMS is the VAX operating system supplied by DEC, and it clearly was influenced by UNIX. But VMS differs from UNIX in several important aspects of the file system, the process environment, and the user interface.

Since VMS was initially the only VAX "game in town," we implemented a UNIX emulator for it. This allowed us port to the VAX a large collection of UNIX tools (csh, make, etc.) and have them run under VMS. Each successive major release of VMS has been more friendly to this effort.

This talk covers the major obstacles presented by VMS and show how we adapted to them. We also discuss the trade-offs between performance, compatibility with VMS utilities, and faithful UNIX
emulation, as well as the lessons we and our many users have learned in the process of using such a UNIX emulator.

**T4B — Database Systems**
Thursday, 3:30–5:00
Roger Sippl, chair

Reporter: *Mark Bartelt*

**Microprocessors for Biomedical Research Database Management and Analysis**

*F. W. Stitt*
Clinical Data Research Services, Inc.
340 Lombard Street
San Francisco, CA 94133
(415) 398-3193

Good biomedical database systems are hard to find and hard to design, because of the very nature of the data (i.e. noted for its diversity and complexity) which the databases need to deal with. For example, it is often necessary to store, retrieve, and analyze data from surveys, clinical trials, and so forth. Much of this data is not of a sort that is easily dealt with by the standard relational model popular today. There exists some software on large mainframes, but what is available is not entirely adequate. One often finds that the tools for data entry and validation, if available and usable at all, provide far less functionality than what's required.

A system comprising two VAX 11-750's with UNIX plus the 'Unify' database system and the BMDP statistical analysis package was described. An Onyx system is also being acquired to run the database software.

Unify was chosen over several other database packages because of the fact that it is actually a hybrid network and relational database system, and therefore better suited to the requirements. It provides efficient data access by using hashing and B-trees. It also includes an extensive C application library, and an interactive query-by-example facility. The package itself consists of something called Unitrieve (the database handler), plus a screen handler and menu processor.

The other database packages evaluated were LOGIX and Informix (purely relational), and IMP (an ISAM-based system). The speaker felt that none of the query languages with any of the four products were truly English-like (despite claims to the contrary by the vendors), though all were adequate. All provided an adequate C language interface.

It was also pointed out that Unify's hashing algorithm for record location results in excellent performance, but requires that the database be rebuilt as it grows.
The DB Relational Database Management System

J. Robert Ward
International Institute For Applied Systems Analysis
Schlossplatz 1
A-2361 Laxenburg
Austria

The speaker described the design philosophy of a database system which currently runs on the VAX and the 11/70, and which will soon run on one or more 68000-based machines. The basic design goals were: (1) flexibility, and (2) that it be efficient for query processing.

The system provides a query language with syntax and typing rules similar to those of C. A user can specify the format of output in a printf-like way, to facilitate coexistence with other UNIX programs, such as awk and sed. It's also easy to combine this database system with existing report generators.

The query processor does its work by parsing a query, and converting the parse tree to a pipeline of action-specific nodes. Each node in the pipeline contains an internal representation of a selection constraint, and a pointer to a coroutine which acts on tuples to perform the required selection operation. This approach permits the selection procedure to run without the need for temporary disk files. The coroutines themselves are simulated by recursive function calls.

The Informix Commercial DBMS for UNIX

Laura L. King
Relational Database Systems, Inc.
1208 Apollo Way, Suite 503
Sunnyvale, CA 94086

[Notes on this talk are not available so the abstract has been printed...Ed.]

In this paper the informix relational database management system is described as the ideal tool for the building of computer applications which are designed to gather business, engineering or scientific data and produce informative reports. Use of informix by end-users, system integrators and application builders is discussed.

The c-isam B-tree based retrieval method is described as being informix's software search and sort engine, whose features are critical for end-user ease and flexibility. C-isam is also described as the up and coming de facto ISAM for UNIX operating systems, and the C language.

The ace relational report writing language is described as an English-like and non-procedureal tool for producing 100% of the needed reports from a database, in exactly the format the user desires. The fact that ace is a non-procedureal language is cited as the reason for the greater than ten-to-one gain in man-hours needed for report development and maintenance.

The new perform custom screen building system is now being shipped to informix users. The user or system builder can construct perform screens for data entry and maintenance, as well as "query by example" style database interrogation.
The Intel Data Base Processor (iDBP) and IS/3

John R. Levine
INTERACTIVE Systems Corporation
P.O. Box 349
Cambridge, MA 02238

The Intel 86/440 is a microcomputer-based back-end relational database machine. A host computer stores data into and retrieves data from a database by communicating with the iDBP, making use of the iDBP's set of database primitives. (The term 'primitive' may be a bit misleading, though, since database operations such as 'join' and 'select' are provided directly by the iDBP. In other words, some of the 'primitives' are rather powerful.)

The iDBP permits a host system to define, select, retrieve, update, and delete data. It provides access control on both a record and an item level. It also provides all necessary concurrency control, to ensure the integrity of the data stored in iDBP databases. It supports both structured and unstructured data, and relationships between structured and unstructured files. Features such as transaction logging are also provided.

What is necessary to make effective use of the iDBP is an appropriate software package in the host machine(s), which provides an interface between users and the iDBP itself, and which translates commands and queries into equivalent host—iDBP message packets which execute the iDBP database primitives.

The speaker described a set of tools running under IS/3 (Interactive's version of UNIX System III), which permit a user to utilize iDBP functions. For example, there is an interactive inquiry language, 'ddml', which permits a user to enter iDBP commands and immediately see the response. There is also a precompiler which processes programs written in an 'extended' version of C (consisting of normal C plus embedded iDBP interface language) and produces a pure C program (intended to be system-independent) which can then be compiled and run to interact with databases in the iDBP.

In the work-in-progress category are other programs, designed more for the casual user (i.e. non-programmer). For example, there is a screen-oriented query language, modeled after IBM's Query-by-Example language.

/rdb: A Relational Data Base Management System for UNIX

Rod Manis
Computer Consulting
Suite 305
75 Buena Vista East
San Francisco, CA 94117
(415) 967-7770

With all the database packages available for UNIX these days, why write yet another? The speaker gave three reasons:

1. He wanted one he could afford; if this meant writing it himself, this implied that it should be easy to write.
2. It should be VERY easy to use (and able to be ported, say, to a personal computer).
3. It should nonetheless be very powerful.

The /rdb database system permits a user to create tables (including column headings) with a standard text editor. Columns are separated by tabs, and the first two lines of the table consist of, respectively, the column names and fields of dashes which define field widths.
All commands which query the database are simple programs invoked from the shell. This makes it easy to combine them with other UNIX utilities (e.g. *awk*, which can be used to perform selects).

**FOCUS: A Screen Interface to TROLL Databases**

*Anthony I. Wasserman*
Medical Information Science  
University of California, San Francisco  
San Francisco, CA 94143  
(415) 666-2951  
ucbvax!waserman

*Martin Kersten*  
Wiskundig Seminarium  
Vrije Universiteit  
de Boelelaan 1081  
Amsterdam, the Netherlands

It’s often the case that, while query languages associated with database systems may be powerful enough to perform necessary operations, they are nonetheless not totally satisfactory because they are often overly verbose (i.e. require a lot of typing), and many people find them difficult to learn. Focus is a simple, low keystroke interface to the TROLL database system (both of which are outgrowths of the USE project [see the summary of RAPID elsewhere in these notes]). It was intended to facilitate both the addition and updating of records in a relational database system, and also browsing through the database and retrieval of records.

One of Focus’s more interesting attributes is its ability to permit a user to ‘focus in’ on a desired set of records. One can select a set of tuples based on a specified condition, then select a subset of that set based on another condition, and so on, until one has gotten the set of records one wants. It is also possible to ‘pop up’ to previous sets which were selected during the focusing process (and which are therefore supersets of the current subset).

**F1A — Graphics**

Friday, 8:30—10  
Phil Cohen, chair

[Notes for this session are not available so the abstracts for the talks have been printed...Ed.]

**Device Independent Graphics Enhancements**

*Greg Hidley*  
ITT Defense Communications Division  
10060 Carroll Canyon Rd.  
San Diego, CA 92131  
(619) 578-3080  
ucbvax!sdcsvax!cdwest!greg

As of the 4.1 Berkeley Software Distribution the graphics packages available under UNIX had certain limitations: *vplot*, the Versatec interface, did not run on the 32 bit machine and did not share the Versatec as did other support software (*vpr, vroff, vgrind*, etc.). The UNIX plot library (-lplot) did not do any form of automatic scaling and did not allow optimized scaling for plots which were overly X or Y range dominated. Graph offers additional flexibility as a front end for standard two dimensional
Cartesian plots but does not allow user defined subroutines for the generation of non-standard representations often needed in a research environment.

In addition to modifying vplot to work on the 32 bit machine, queuing plots for the Versatec, and adding “plot handlers” for the HP Penplotter, Matrox graphics monitor, HP 2648a, Dec VS11, and Digital Engineering VT100 Retro graphics terminals, we modified the plot library to allow automatic scaling, user defined scaling, and implemented the functions circle and arc. There is also a plot handler for a non-graphics terminal which uses curses and termcap for a rough approximation of the plot.

We enhanced the UNIX plot command’s inherent device independency by using user-assigned environment variables to control the run time output of plotting programs. Thus, a user could write a single C program to plot a graph and, when loaded with the ITT plot library, could execute that single program and get plots on any of the devices listed above. The environment variable PLOTTER would determine at runtime the destination of the plot. If hardcopy devices were targeted automatic queueing would be invoked.

The package is compatible with the standard UNIX plot library routines and the plot handlers are usable with the standard UNIX intermediate plotfiles. Scaling is handled by scanning the coordinate ranges while building the intermediate plotfile and then backpatching appropriate deltas at the beginning of the file. The plot handlers use independent scaling macros for the x and y axis which allow both scaling that either retains shape or that optimizes the plotting surface of the target device. The former scaling is default but can be overridden by the user.

We are currently implementing an interactive query system which would allow the user to view a sequence of plots on a graphics terminal and redirect copies of selected plots to a hardcopy output.

Terminal-Independent Plotting Packages

An Example and Suggestions for Standards

Don Mackay
University of California at San Diego
Department of Chemistry B-014
La Jolla, CA 92093
(619) 452-2572
ucbvax!sdcsvax!sdchemc!dm

Crude plotting on character terminals is a quick and convenient way to display data when a high resolution device is unavailable or inappropriate. As enhanced graphics capabilities make their way into low priced terminals, it becomes desirable to take advantage of line drawing and other special characters for low resolution plots on terminals. A sample set of terminal independent plotting programs currently in use at UCSD is discussed in terms of the variety of enhancements used and the method of implementation. Suggestions for standardization of graphics attributes in the termcap or terminfo files are presented.

Graphics Standards for Personal Workstations

Michael Shantz
Sun Microsystems, Inc.
2310 Walsh Avenue
Santa Clara, CA 95051
(408) 748-9900
ucbvax!ucbarpa!sun!shantz

Graphics workstation architecture. A modern workstation architecture with local area network support provides an enhanced environment for engineering graphics applications. The Sun workstation has a 1K by 1K bitmap display with rasterop hardware, a 640 by 480 color frame buffer, one or two Mbytes
of main memory, a roughly 1 MIP MC68000 processor, an Ethernet interface, and vector drawing at 3 to 4 microseconds per point. The 4.2BSD UNIX operating system provides an advanced network interface with remote shell and remote login, 10 MHz network and a hardware graphics output pipeline will complete the low-cost high-performance graphics workstation environment.

Core, GKS, and VDM. Adherence to standards for both hardware and software is central to the design philosophy of the Sun workstation. Hardware “standards” for this workstation include the Multibus which facilitates system expansion, the Ethernet which facilitates multi-host local networks, the MC68000 processor, and SMD disk interfaces. Software “standards” used include UNIX, the C programming language, and for graphics, the ACM SIGGRAPH Core system, the GKS graphics standard, and the Virtual Device Metafile (VDM) draft proposed standard. Graphics standards are important for device independence, software portability to future products, and for communication of graphical data.

We have implemented the ACM SIGGRAPH Core graphics package and a GKS implementation is scheduled for completion in mid ’83. Each of these will have a VDM standard interface pending finalization of the standard. SunCore and SunGKS comprise a set of low level graphics output primitives such as lines, text, and filled regions, facilities for grouping primitives into segments which may be manipulated as a unit, and a set of input primitives that use the mouse and keyboard for dragging, drawing and transforming segments. Device independence allows the user program to direct graphics output to the bitmap, color display, VDM, or to a process on another workstation on the ethernet. Hardcopy output requires piping VDM data over the network to a process on the printer server.

Extensions to the standards. Planned extensions to the graphics standards implementation include integrating the graphics processes and the window system so that graphics processes and other UNIX processes can execute simultaneously in multiple windows. Shaded surface polygon primitives with hidden surface elimination via a virtual memory z-buffer and bicubic patches with image and texture mapping are planned for ’83. Other raster extensions will be added for handling bitmaps and imagery in a Core compatible manner. Interprocess communication and file transfers between workstations would permit sharing of large data bases and utilization of compute power or special hardware on other “file servers” or “crunch servers.”

Windows with 4.2BSD

Steven R. Evans
Sun Microsystems, Incorporated
2310 Walsh Avenue
Santa Clara, CA 95051
(408) 748-9900
ucbarpa!ucbvax!sun!sevans

The presentation describes the user interface and technical highlights of a UNIX-based multi-window development environment. The hardware environment of this system consists of the Sun workstation which utilizes a 10 MHz 68000 cpu, 1 MByte RAM, large bitmap display, optical mouse and rigid disk (possibly shared over a 10 MBit Ethernet). The software environment of this system is the new Berkeley 4.2BSD version of UNIX.

One of the most interesting problems facing a designer of what is intended to be a multi-window system for UNIX is how to achieve a reasonable level of cooperation between windows. This is troublesome because an application running in a window often knows nothing of the existence of other windows, to say nothing of the window system itself. Furthermore, a process can’t communicate with the other processes in other windows due to separate address spaces unless a lot of Kernel support is utilized.

This talk describes what a “reasonable level of cooperation” is thought by the author to be. Also, the talk describes how the new facilities of 4.2BSD are utilized to implement the user interface ideas without requiring a great deal of operating system support.
Computer Animation at UCSD

Jeff Loomis
Seacoast Software and Salk Institute
ucbvax!sdcsvax!jeff or loomis@nose

Phil Mercurio
Quantitative Morphology Lab
University of California at San Diego
ucbvax!sdcsvax!mercurio or sdcsla!mercurio@nprdc

At UCSD's Quantitative Morphology Lab, UNIX is used for the control of high performance computer graphics hardware interfaced to a film animation bench. Our animation system has been used to produce scientific films in the areas of neurosciences and linguistics. Several artists have also been turned loose on the system with interesting results.

The presentation will include a brief description of the evolution of the hardware and software and how we took advantage of UNIX to save programming effort. We'll also be showing a short film made especially for the conference.

F1B — New UNIX Implementations II

Friday, 8:30—10:00

Joel Carter, chair

Reporter: James Persky
3765 #G Miramar Street
La Jolla, CA 92037

REGULUS, a Real-Time UNIX Lookalike

Bill Allen
Alcyon Corporation
8716 Production Avenue
San Diego, CA 92121
(619) 578-0860

REGULUS is source level system call compatible with UNIX. All UNIX system calls are implemented and have the same names, same arguments, and same return values. REGULUS supports standard UNIX features: multiuser, multitasking; tree structured file system; device and file independent I/O; pipes and filters; runtime I/O redirection; shared code programs; and a full complement of software development tools.

REGULUS has many extended capabilities. Real-time tasks can be locked in memory, preemptively scheduled by priority, and prevented from being time-sliced. Inter-task communication includes 32 'user' signals which have no default meaning to the system; the default action is ignored.

A large improvement went into the design of the file system. These include dynamic allocation of file index records, fewer disk accesses for sequential data, and index records stored next to file data blocks. Tasks can lock multiple file segments as well as choose the response to lock failure. The file segment can be locked for reading, writing, or both.

One of the big advantages of REGULUS is the minimal hardware support required. Only 128kb of RAM is needed for a complete system. With one megabyte of disk storage and an OEM—provided dedicated kernel as small as 32kb REGULUS requires very minimal hardware.
UNIX for the National 16032

Paul Neelands, Richard Miller, and Chris Sturgess
Human Computing Resources Corporation
10 St. Mary Street
Toronto, Canada M4Y 1P9
(416) 922-1937
dcvax!hcr!hcrvax!paul

The NS 16032 is a true 32-bit processor. Although its current hardware interface to the outside world is via 16 multiplexed bits, the instruction set was designed from the start to look like a big machine. It has VAX-like memory management unit and also supports page reference bits. Full floating point support is available. The 16032 instruction set allows high density code. The machine provides support for high level languages such as Pascal and C without being restrictive.

The base hardware for the porting effort was a prototype workstation with the 16032 CPU, the 16082 memory management unit, and the floating point chip. The machine had 256kB of main memory, a Winchester disk, a streaming tape drive, a GPIB interface, RS232, a bit-map display, and limited audio output.

The C compiler for this effort was based upon the VAX version of the portable C compiler. The assembler and loader were difficult because the machine has a complex instruction set.

The UNITY kernel for the 16032 is initially based upon Berkeley 4.1BSD. This was pulled off with only 256kB of main memory! Much grunge and a number of machine dependencies had to be removed. Added were multiwindow graphics support, audio output support, and, in the future, GPIB support. The major problem in the porting effect was to circumvent minor hardware bugs in the early versions of the chips.

The system was first demonstrated in public at the November 82 Comdex meeting. Future plans call for:

- first OEM versions in March of this year
- a full rewrite of the demand paging algorithm
- unification of paging and the buffer cache into one concept
- increased standardization of the utility programs in order to conform to industry standards
- network support
- move the enhanced demand paging code back to company’s VAX product.

The Port of UNIX to the Gould 32/27

Jack Blevins
Gould, Inc.
S.E.L. Computer Systems Division
Post Office Box 9148
Fort Lauderdale, FL 33310-9148
(305) 587-2900 ext. 5143

The porting effort began in October, 1981 with a team of five programmers. The Gould S.E.L. MPX operating system was the development environment. Two unique features of the Gould S.E.L. hardware are an 8192 byte page and an I/O system that uses a 26.6 Mbyte per second SelBUS and a 1 Mbyte per second MPBUS. The SelBUS interfaces high speed peripherals like disc, tape, and memory while the MPBUS interfaces terminal controllers, floppy discs, line printers, and communication controllers. An I/D Processor “IDP” connects the MPBUS to the SelBUS.
The actual port was performed from UNIX Version 32. Version 32 was selected over Version 7 because it had been converted to a 32-bit processor. The portable C Compiler had previously been ported to Gould S.E.L. computers.

Armed with these tools mfks, fboot, tboot, tdcopy, and the kernel were tackled. Naturally many obstacles had to be overcome. The team learned about Ld and the difference between the object code generated by the assembler and that expected by Ld. A C language program called oc was developed to convert the object code format and port Ld to MPX.

The first kernel was linked and tested in early February of 1982. An MPX debugger was linked into the kernel to provide the means to determine the errors and possible patches for them. This debugger was only useful for one person testing at a time.

In late May 70 utilities were operational and the system was stable as long as there was no swapping of user programs. The swapping problem was corrected by mid-June and a very stable UNIX was usable by the development team and software quality assurance.

The source code received from Bell Laboratories had no major bugs in the operating system, and the gamble on the code from Bell paid off. The compiler errors could be avoided by changing C source. The utility programs were the major source of porting difficulties because differences in code and documentation were found in 25% of the utilities.

All in all, the package supplied from Bell Laboratories was good, as demonstrated by the fact that a team of five and later nine could port it with no outside assistance.

UNIX Enhancements for Real-time Applications

Kent Blackett
MASSCOMP
543 Great Road
Littleton, MA 01460
(617) 486-9425

A laboratory computer system based on UNIX was developed which can reliably and predictably achieve two Mbytes per second throughout with interrupt latencies as low as 2 micro-seconds. The system is based on UNIX system III and augmented by UCB's 4.1BSD. The real-time extensions that were developed were:

- fixed priority schedules
- memory locked processes
- contiguous disk files
- priority-delivered user interrupts
- high performance pipes

A high speed bipolar front end processor offloads the host. Modifications to UNIX improve system throughput and file integrity. Layered software was added to provide user ease-of-use through window management and “quick-choice” menu selections.
EUNICE

Ellen Williams
The Wollongong Group, Inc.
1135A San Antonio Road
Palo Alto, CA 94303
(415) 962-9224

EUNICE is software that provides the capability to run UNIX programs under VMS. This gives VAX users both UNIX and VMS on the same machine. They can operate in either environment with all facilities of the other immediately accessible. The user can enter file specifications in either UNIX or VMS forms. UNIX programs running in the EUNICE environment can read and write both VMS and UNIX files, and read VMS directory files as though they were UNIX directories. When using UNIX compilers under EUNICE the user can generate either VMS or UNIX object files. VMS object files may be linked with other modules from other VMS languages.

VMS users benefit from additional features and software available through EUNICE/UNIX, including:

- Escape to the UNIX environment
- Any of the more than 200 UNIX utilities
- High level languages, such as C, FORTRAN 77, RATFOR
- Compatible file, device and interprocess I/O
- Shell command language (sh and csh)
- I/O re-direction
- Pipes for interprocess communication
- Command chaining
- Networking
- Program development tools:
  - Program maintenance system
  - Compiler-compiler
  - Lexical analyzer
- Text processing and document preparation software:
  - Full screen and line editors
  - Formatter for correspondence quality output
  - Phototypesetter formatter
  - Preprocessor for mathematical equations
  - Preprocessor for tables
- Communications software
- On-line documentation

The EUNICE package is available for all VAX/VMS systems (11/730, 11/750, 11/780, and 11/782).
Porting UNIX

P. Verbaeten and Y. Berbers (main authors)
(‘‘A. Wupit’’ collective name of the participants of the TWIX project)
Katholieke Universiteit Leuven
Dept. Computerwetenschappen
Celestijnenlaan 200A
B-3030 Leuven
Belgium
016/20 06 56, x1564

In our department we have ported the UNIX V7 computer system, and have gained much experience about many different aspects of porting an operating system to a new hardware environment, especially about porting UNIX.

The emphasis here will be on the porting strategy and the working environment; we will briefly discuss some alternatives and the major problem areas of a UNIX port.

UNIX on the National Semiconductor NS16032

Glenn C. Skinner
National Semiconductor
Microcomputer Systems Division
M/S 7C-266
1135 Kern Avenue
Sunnyvale, CA 94086
(408) 733-2600, x368
ucbvax!menlo70!nscl!glenn

Bill Jolitz
Symmetric Systems
21467 Aldercroft Heights
Los Gatos, CA 95030
(408) 353-3259
ucbvax!william

We have recently ported UNIX to the NS16032 microprocessor. The port is based on the highly successful Berkeley VAX system and has been tailored to the microprocessor environment. The NS16000 chip set family provides hardware support for full demand-paged virtual memory, modular software, runtime debugging, and high-performance floating point. These features are well-suited to UNIX and greatly speeded the porting effort; it took a month to get a “bare bones” kernel running, another month to add protected virtual address spaces, and only an additional month to implement demand paging.

These features have been exploited to provide a high quality software environment for the design and construction of large software products. This system (both hardware and software) allows for very low cost computers with state-of-the-art software.
The kernel is designed to exploit fully the virtual memory architecture of the NS16000 family. Each process runs in a protected linear virtual address space of up to 16MB. Page tables for a full size process occupy 129KB, a substantial fraction of the system's physical memory. To reduce this mapping overhead, the kernel can page user process page tables (both first and second level) as well as user process text and data pages. The kernel can also page parts of its own text and data spaces. Thus, large address spaces can be supported on much lower cost machines than was previously possible.

The kernel provides additional virtual memory features. There are new vspy and vlock system calls for mapping parts of one address space into another and for locking pages into physical memory. These features support time-critical applications by allowing processes to take direct control of device registers and buffer areas. The kernel virtual memory implementation is also designed to support memory-mapped files, explicitly shared segments, and copy-on-write shared segments.

In addition to virtual memory, the NS16032 supports modular software. The processor includes several registers that mediate access to local data, to function and procedure arguments, and to references to external data and routines. This organization has several implications for UNIX. Code size is reduced, since absolute 32-bit addresses can be replaced with small local offsets. Relocation and loading are simplified; code itself need never be modified when loaded and can easily be made ROM-resident (one intriguing possibility is a ROM-resident, dynamically configurable UNIX kernel). However, object files no longer consist of homogeneous text and data segments; instead these segments are partitioned into modules. One of the more challenging parts of the port has been to devise an a.out format that captures module-related information without violating existing UNIX conventions on address space organization.

A major aspect of the NS16032 UNIX system is its hardware debugging features. The memory management unit supports two breakpoint on reference address registers and limited program flow tracing. The system has been enhanced to support these features, significantly easing process debugging. For example, one can locate code that might be destroying the contents of a given data cell by trapping writes to that cell. Similarly, a better picture of a program's abnormal behavior may be found by recording its last few (possibly conditional) jumps with the program trace registers. A major step of the port has been to implement a UNIX version of ddt that grants user access to this enhanced debugging environment. We have found it invaluable for interactively debugging the kernel itself as well as for the more prosaic purpose of debugging user programs.

Finally, the system has a fast floating point instruction set with addressing modes identical to those of the standard instruction set. This instruction set symmetry makes for straightforward and efficient code generation by compilers; no special addressing cases need be considered merely because one or more operands are floating point.

UNIX for the Computer Automation 4/95

Steve Pozgaj
Human Computing Resources Corporation
10 St. Mary Street
Toronto, Ont. M4Y 1P9 Canada
(416) 922-1937

This talk discusses an implementation of UNIX on the Computer Automation 4/95 minicomputer. The 4/95 is a high-performance 16 bit computer featuring a modern memory management unit and a baroque instruction set. The major problems with this implementation involved 1) building a C compiler which handled word addressing properly, and 2) dealing with the fact that many of the machine features were not documented. This talk will be of special interest to those about to embark upon a UNIX port to any less well-known machine architecture. This talk will give tips on avoiding the many pitfalls in such a task.

;login:

The kernel is designed to exploit fully the virtual memory architecture of the NS16000 family. Each process runs in a protected linear virtual address space of up to 16MB. Page tables for a full size process occupy 129KB, a substantial fraction of the system's physical memory. To reduce this mapping overhead, the kernel can page user process page tables (both first and second level) as well as user process text and data pages. The kernel can also page parts of its own text and data spaces. Thus, large address spaces can be supported on much lower cost machines than was previously possible.

The kernel provides additional virtual memory features. There are new vspy and vlock system calls for mapping parts of one address space into another and for locking pages into physical memory. These features support time-critical applications by allowing processes to take direct control of device registers and buffer areas. The kernel virtual memory implementation is also designed to support memory-mapped files, explicitly shared segments, and copy-on-write shared segments.

In addition to virtual memory, the NS16032 supports modular software. The processor includes several registers that mediate access to local data, to function and procedure arguments, and to references to external data and routines. This organization has several implications for UNIX. Code size is reduced, since absolute 32-bit addresses can be replaced with small local offsets. Relocation and loading are simplified; code itself need never be modified when loaded and can easily be made ROM-resident (one intriguing possibility is a ROM-resident, dynamically configurable UNIX kernel). However, object files no longer consist of homogeneous text and data segments; instead these segments are partitioned into modules. One of the more challenging parts of the port has been to devise an a.out format that captures module-related information without violating existing UNIX conventions on address space organization.

A major aspect of the NS16032 UNIX system is its hardware debugging features. The memory management unit supports two breakpoint on reference address registers and limited program flow tracing. The system has been enhanced to support these features, significantly easing process debugging. For example, one can locate code that might be destroying the contents of a given data cell by trapping writes to that cell. Similarly, a better picture of a program's abnormal behavior may be found by recording its last few (possibly conditional) jumps with the program trace registers. A major step of the port has been to implement a UNIX version of ddt that grants user access to this enhanced debugging environment. We have found it invaluable for interactively debugging the kernel itself as well as for the more prosaic purpose of debugging user programs.

Finally, the system has a fast floating point instruction set with addressing modes identical to those of the standard instruction set. This instruction set symmetry makes for straightforward and efficient code generation by compilers; no special addressing cases need be considered merely because one or more operands are floating point.

UNIX for the Computer Automation 4/95

Steve Pozgaj
Human Computing Resources Corporation
10 St. Mary Street
Toronto, Ont. M4Y 1P9 Canada
(416) 922-1937

This talk discusses an implementation of UNIX on the Computer Automation 4/95 minicomputer. The 4/95 is a high-performance 16 bit computer featuring a modern memory management unit and a baroque instruction set. The major problems with this implementation involved 1) building a C compiler which handled word addressing properly, and 2) dealing with the fact that many of the machine features were not documented. This talk will be of special interest to those about to embark upon a UNIX port to any less well-known machine architecture. This talk will give tips on avoiding the many pitfalls in such a task.

;login:
Architectural Implications of UNIX
(or Pitfalls for UNIX Porters!)

Matt Dickey, Greg Noel, Bob Querido et. al.
NCR Corporation, SE-TP
11010 Torreyana Rd.
San Diego, CA 92121

Bill Appelbe and Jim McGinness
University of California at San Diego
La Jolla, CA 92093

UNIX is being ported to an increasing variety of computer systems. The difficulty of porting an efficient UNIX implementation is highly dependent upon detailed characteristics of the target architecture.

The presentation discusses the constraints upon a UNIX-compatable architecture, as a guideline to new porting projects. The major architectural impact of UNIX upon memory management, address modes, primitive data types, operators, protection, and context switching is discussed.

UNIX assumes a low-level architecture (not necessarily a PDP-11!). Hence, it is usually easier to port UNIX to a simple architecture than to a high-level architecture which provides descriptors, and high-level task and memory management.

F2B — Marketing and Venture Capital

Friday, 10:30—12
Marlene Martin, chair

Reporter: Sam Penny
Penny Penny & Strong
El Cerrito, CA 94530

Getting Venture Capital

Henry Wilder
Dougerly, Jones and Wilder Venture Capital

This talk was a concise summary of what it takes to get venture capital. The comments were also an excellent guideline for new entrepreneurs for choosing markets, products and business partners.

The steps followed by a VC (Venture Capitalist) in considering a proposal are the “path of greatest resistance” that eliminates unacceptable proposals as soon as possible. 5% of the proposals fail on introduction, 45% more after the VC has spent about 20 minutes considering details of the plan. After an initial presentation only 25% remain, and 10% more are eliminated during in-depth interviews. Another 10% fail when the VC and entrepreneur can not reach a mutually acceptable deal. The VC will then eliminate 3% more by “due diligence” — researching why the deal should be turned down business-wise and legally, leaving only 2% of the initial proposals that result in any investment.

The highest priority is that the proposal offer an inviting growth market with new customers available, little resistance to new entrants, positive economics (enough gross margin to support the business, distributors and dealers), and the possibility to become a large independent company. There should be evident ways to increase sales efficiency, and distribution channels should be available. The market should allow for products to be focused so that the company can pick the best products and sell in volume.
The third priority is a proprietary product plan. There should be something special and compelling about the products (capability, price, geography, reliability, etc.). There should be a barrier to competition (a learning curve head start, a lock on customers and suppliers, name recognition, etc.). The plan must offer a reasonable longevity to provide cash for establishing distribution and developing the next product. Finally, there must be some related future growth area for the new business to move to when the initial plan is accomplished.

The final priority of the VC that was discussed is to develop a reasonable deal. This includes having a step-wise investment schedule based on milestones to limit risk and having a reasonable cost for the profit potential. It is typical that terms for vesting the management in the resulting firm be based on seniority. The VC will be concerned about who else is investing and how much help and influence will be expected from the investor group. It is important to keep the styles of all investors in the business consistent.

UNIX Markets and Competition

Bob Katsive
Gnostic Concepts

Gnostic Concepts is a “professional reconnaissance unit” and has developed a strategic view of UNIX. UNIX is the result of a “demand/creation critical mass phenomena.” The university environment produced people who wanted a transportable operating system for inexpensive computers. There was a need for a standard interface that would be stable for third party vendors. AT&T supported this concept, and UNIX was born. Mr. Katsive declared, “If UNIX had not existed, it would have been necessary for man to invent it.”

UNIX is a fourth generation operating system with key features that reduce development times and costs. Hardware is becoming an inexpensive commodity, and inexpensive software is on its way. Some of what is coming includes invisible UNIX, silicon UNIX, distributed UNIX, very fast UNIX and secure UNIX. Mr. Katsive pointed out that the world’s largest processing system is composed of all the UNIX sites on the ARPAnet.

UNIX-related EDP expenditures grew from $151m in 1980 to $1.38b in 1982. It is projected to be $5.28b in 1985 with about 23% for support, 37% for personnel, 36% for hardware and 4% for systems. UNIX will be the fastest growing operating system on personal computers.

The greatest pressure on the future development of UNIX will come from the users of microcomputer hardware. In 1982 65% of the UNIX systems shipped were installed on microcomputers, 34% on minicomputers and 1% on mainframes. By 1990, 94% will be on micros, 3% on minis and 3% on mainframes. These estimates apply to the combination of Bell UNIX and UNIX look-alikes.

Delivering UNIX to the End-User Market

Michael Denny
BASIS, Inc.

This was a delightful sermonette you had to hear in person to fully appreciate. In the words of the preacher, “I have been to the promised land, and it is a wilderness... What does it take to bring UNIX applications to the user market? Not Bill Joy.”

Personal computer software is producing a groundswell of frustration. The end-user may be naive, but she/he is not dumb. CP/M is being pushed by the “media,” but it is not the right answer. UNIX offers the real key, but care must be taken to meet the needs of the end-user.

No software is friendly enough. Users try to do their jobs, and the computer is supposed to help. You cannot expect the bulk of the users to learn operating systems — they need turnkey systems. With the computing power now available, large and expensive systems are unnecessary. There is no
need to integrate the computer system and the end-user. You can broker application services to them so they can get on with their jobs without worrying with the operating system.

To have a good menu for an application, it is necessary to implement the user jargon for the job. Keep the menu from being too dry. Maybe conversational computing is better. And don’t try to design the “perfect generic vocabulary.” Make the software user-ready. It should be living software that is easy to reimplement and revise.

UNIX provides an unusually appropriate place for developing user-friendly applications programs. Mr. Denny concluded, “[Using this method means] the user can do his job, and we can do ours.”

Distribution and Differentiation

*Marlene Martin*

The Marketing Network

This was a discussion of product and marketing considerations for selling in the UNIX market. The UNIX market is already a mature market with many kinds of applications available under UNIX. There is a shakeout coming in the market, and it is important for any young company coming out with a new UNIX-based product to be unique.

Differentiation of your product from others is important. One method of differentiation is the amount of value-added given to the product. UNIX plus a few tools may be sufficient for a barebones offering to an OEM. More tools may need to be added to produce a basic end-user product. Adding still more tools and features increases the product’s value as an enhanced end-user product. You must choose just where you want to be in this differentiation.

Packaging of the product is vital. You must make your product look good to your customer. The amount of resources required to successfully package a product for the market will increase as its value-added increases and as you try to sell to the higher level markets.

Delivery is a commitment that must be met. Startup companies sometimes mistakenly see their goal as making money. They should see their job as offering a service that provides value to the customer. Timely delivery is one such service.

Distribution of UNIX-based products can be done through bundling with hardware systems, OEMs, value-added resellers, master distributors, industrial distributors, retailers or direct to the end-user. UNIX products have not been an easy sell in the retail markets because of the difficulty of training sales personnel.

Some suggestions for OEM marketing are to emphasize reliability, delivery and features. Do not compete with your OEMs and avoid dealer over-crowding. In conclusion Ms Martin said, “Put your marketing strategy in place and stick to it.”

New UNIX Markets in Engineering

*Camran Elahian*

Computer-Aided Engineering

The right application software using the features of UNIX represent a winning combination in the CAD market. This talk gave an example of the application of UNIX to a particular vertical market.

CAD was spawned by the breakthrough of minicomputers and inexpensive graphics hardware. The market will grow to $4b in 1986. First generation CAD systems have about 80% of the current market, but their lack of standard components has minimized portability, diluted efforts and lengthened product cycles.

At least 27 second generation CAD systems have emerged in the early 1980s as a result of the introduction of microcomputers, better graphics and Local Area Networks. Some standard components
have emerged, but portability remains a problem. By using UNIX as the base, the speaker’s company has developed a portable CAD system featuring a common data base for CAD as well as CAE and CAM. The product is a good example of how UNIX can be the basis for vertical-market systems that are portable and that make efficient use of design efforts.

F3A — Portability
Friday, 1:30—3
Mike O’Dell, chair

[Notes for this session are not available so the abstracts for the talks have been printed...Ed.]

Portability in the UNIX World
What UNIX Can Learn from the Software Tools
Mike O’Dell
CSAM 50B/3238
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720
(415) 486-5583 (FTS: 451-5583)
ucbvax!bl-csam!mo

UNIX is a living, breathing, growing system which has become famous for its ability to change and adapt. Now people are beginning to fret because this growth and change inevitably leads to non-portability of programs, even within the UNIX world. Luckily, the Software Tools provide a well-developed and tested technology for dealing with diversity of environments. The Tools technology is built on the shoulders of the ideas central to ‘‘UNIX-ness,” but in a highly portable way. This technology will be explored in some detail and its implications for portable UNIX programming will be examined. This confluence is particularly fortunate because of a growing interest within the Software Tools Group to adopt C as the successor to Ratfor.

If these ideas do not prove sufficiently controversial, the speaker will advocate adoption of EBCDIC as the standard character code for all programming.

A Tutorial on C Portability
Michael Tilson
Human Computing Resources Corporation
10 St. Mary Street
Toronto, Ont. M4Y 1P9 Canada
(416) 922-1937
dcvax!hcr!mike

C programmers have always been aware that it is possible to write programs in C which depend on the underlying hardware. The C language allows portable programming, but does not require it. A number of people have experience with 11-to-VAX-to-68000-to-Z8000 portability. Indeed, there are a number of serious issues in moving between these machines, many of which have been discussed in talks and articles by other authors. Recently, the UNIX system has been moved to a number of machines which are less “friendly” to C. This talk is a tutorial on C portability, illustrated by examples. It will discuss issues of program portability and style which are important to anybody producing widely portable UNIX software. Most C programmers will not have been aware of all of the specific
issues discussed. The talk draws upon the author's experience with the PDP-11, VAX, Z8000, 68000, CA 4/95 (a word-addressed machine), Three Rivers PERQ (a word-addressed stack architecture), and NS16032. The talk will contain some advice for compiler writers, and will also discuss some flaws in the C type rules.

IS/3: A Compatible Extension of UNIX System III

Steven Zucker
Interactive Systems Corporation
1212 Seventh Street
Santa Monica, CA 90401

Interactive System/Three (IS/3) is a family of UNIX-based systems for micro to main-frame computers. The members of the family are compatible extensions of UNIX System III. This presentation discusses the nature of this compatibility, its benefits, and its costs.

Some of the topics we discuss are:
- Why standardize?
- Why standardize on System III?
- Supersets versus "vanilla-flavor" implementations.
- Some hard lessons from the old days.
- Compatibility of the user and program interface.
- Programming standards and porting procedures.

We also discuss some of the more recent lessons we learned in the process of porting IS/3 to a variety of hardware environments.

F3B — Languages and Programming Environments

Friday, 1:30—3pm

Reporter: Jeffrey C. Bruce
Joseph Yao, chair

VMS C Compiler

Jean Wood
Digital Equipment Corporation
Technical Language Group
110 Spitbrook Road
M/S Zk02-3/N30
Nashua, NH 03060
(603) 881-2080
decvax!jean

DEC's new VMS C compiler began with an existing PL/1 compiler which was restructured and implemented with a switch to generate C code when desired. Five languages (C, PL/1, Macro, Bliss and TBL) were used in coding the various modules. An LALR parser, interacting with a lexical analyzer, passes the processed input through a semantics routine which produces highly optimized native-
mode VAX code.

The primary design criterion for the compiler was portability with UNIX. It was intended to promote usage of VMS on the VAX. However, some small portability problems still exist. Not all, but a large portion, of UNIX functions are supported.

The second version of this compiler will not have a C device-driver, but may have full debugging support. It will be packaged as a set of four rather than five disks, and hopefully will also be available on magnetic tape. No serious bench-mark studies have been done yet, but it was reported that several programs studied thus far have run at an improvement of around fifteen percent. Higher compilation rates were also in evidence.

UNIX APL

Joseph Yao
Science Applications Inc.
1710 Goodridge Drive
McLean, VA 22102
(703) 734-5957

APL is popular in scientific programming due to the power of its large operation set. Most of its operations can be performed on either arrays or atoms. One portability issue with APL is that most versions compile code for the character set of a specific terminal. APL's character set is very large, including underscored characters, so this is a great problem for programs that run on different terminals. In this IBM UNIX version this problem has been solved — the system will set the environment for a specific terminal when the user inputs a terminal-type code.

I/O with a file variable can be achieved using the shared-variable function, SVO, which is also implemented in this version.

The yacc parser, APL.g, and lexical analyzer, lex.c, parse the program text to strings which are later interpreted. Some difficulties existed in the areas of statement printing within functions and in file I/O, but, for the most part, they have been resolved.

Q'NIAL A New Interactive Programming System for UNIX

M. A. Jenkins
Queens University
Department of Computing and Information Science
Kingston, Ontario
Canada K7L 3N6
(613) 547-2784

Q'NIAL is a Queen's University implementation of Nested Interactive Array Language. The primary goal of the language is to match the data-structures and operations to the conceptual structures of the mind.

All data-structures are arrays with zero to n-dimensions. (A zero dimensional array is considered to be an atom.) Any arrangement of these arrays can be declared, and the frames within the arrays may be of different types, allowing the power of a record. The transformer, "EACH," can be used to apply one of a vast selection of functions to each array element.

The implementation of Q'NIAL is in C, aiding portability. It contains many of the list operations of LISP, as well as a powerful operation-set similar to that of APL. The VAX-UNIX version is in test-site use currently. Packaging and licensing arrangements are still being made, and the product is expected to be ready in late Spring.
SOLID: for On-Line Systems Development

John R. Mashey
Bell Laboratories
WH 1B-221
Whippany Road
Whippany, NJ 07981
(201) 386-6844

Solid/UNIX is an update of the 1982 version of Solid. Solid is portable, has a common information database, and is group oriented. The system interfaces individual modules perfectly, eliminating a need for customization between group programmers. It also provides an excellent environment for individuals.

A document universe is set-up by the system, and modularity is promoted by re-use of as much documentation as possible. Files under the system are protected from collisions (simultaneous update of a file). Mostly implemented in C-shell, the system is fairly mobile.

The primary storage mechanism for a file is sccs — the source code is never discarded but kept in a "dust bin." In general, SOLID provides more structuring than make but make still has advantages for certain applications.

SOLID is currently in use by approximately thirty BTL projects of various sizes.

F4 — Standardization

Friday, 3:30—5
Heinz Lycklama, chair

[Notes for this session are not available so the abstracts for the talks have been printed...Ed.]

The /usr/group Standards Committee

Heinz Lycklama
Interactive Systems
1212 7th Street
Santa Monica, CA 90401
(213) 450-8363

The /usr/group Standards Committee was formed in the summer of 1981 by the Board of /usr/group in response to the need for a standard in the commercial UNIX marketplace. The existence of a standard will benefit both producers and consumers of UNIX-based software. The committee has put great emphasis on establishing the proper organization for this effort so that the proper procedures are in place when the time comes to vote on a standard. The committee is sensitive to legal and competitive issues involved with attempting to establish a standard. Efforts are currently being focused on identifying the set of system calls and subroutines which will comprise the system interface standard and on identifying a set of extensions to the system interface which are frequently required by commercial applications.

This paper will discuss the purpose, charter and procedures used by the /usr/group Standards Committee. It will discuss the current status of the standardization effort as well as the schedule for adopting a standard. Some details of the proposed standard will be described.

The draft standard may be purchased for $50 from
/usr/group
The History and Purpose of Standards

Eric Petersen
P2/I
Suite 202
5345 Wyoming Blvd. N.E.
Abuquerque, NM 87109
(505) 822-8585

This talk will present a brief history of the development of standards with emphasis on computer related standards. Characteristics of both successful and unsuccessful past standards efforts will be identified and discussed along with the general purpose of standards and their relationship to both consumers and suppliers.

Standards Organization: Levels and Measurement

Jim Isaak
Charles River Data Systems
4 Tech Circle
Natick, MA
(617) 655-1800

Why break out levels of implementation?

• What is called “standard” and how does a developer and/or end user know what he is getting?
• Portable programs, portable operations, portable media
• Sub-sets (diskless, workstations)
• Environments versus operating systems
• Extensions: record locking, real time, keyed access files

How to tackle the task

• Sort out the logical groups of routines (file management, process management). These define the “modules.”
• For each group, identify levels. Can these be left out altogether and leave a meaningful subset? (Level 0 = Null) What does every program need? (Level 1 = Nucleus) What does the next level of capabilities require? (Level 2 = extensions) Are there more esoteric operations that few will require but are logically connected with this module? (Level 3 = options)
• As new elements are brought forward, they can be fit into existing modules, or defined as new ones.

Timing and commitment

• By clearly setting this concept up as an objective, the standards group can start evaluating the approaches and the developers can start monitoring their products to know what module/levels they require or provide.
• Early commitment will also signal the possibility for subsets and encourage a wider spread level of compatible implementations.

An example
Modules and levels for the subroutines presented as part of the proposed standard (not complete, nor
for discussion, but a few obvious examples).

*Measurement, presentation and validation*

The real payoff is in making it easy for a vendor to define what capabilities his products offer, and for an application developer to know what range of facilities to expect, or require for his environment.

And most importantly, the end-user can make a quick and straightforward evaluation of the systems available-and applications available to make sure he gets what he needs.

Validation suites from University and/or governmental agencies will lend to a higher quality of "standard environments."

And ANSI standardization with NBS validation would be best approached with a multi-level multi-module standard that would span a wide range of systems and applications.

**Criteria for Standards**

*Robert Swartz*
Mark Williams Co.
1430 West Wrightwood
Chicago, IL 60614
(312) 472-6659

The operating system standard is one "which will assist persons producing or acquiring products based on these systems in accurately predicting the behavior of the product in the context of a specific implementation." (/usr/group Standards Committee Draft Proposal for Charter, December 10, 1981.)

What will be discussed is how to accomplish his goal. The criteria for inclusion of modules in the standard or their exclusion will be covered. The major criteria should be that the standard should include the smallest number of elements which support the largest number of user requirements.
Summary of USENIX Board Meetings at San Diego

To: Members of USENIX
From: Lew Law, Secretary of USENIX
Subject: Report on Board meetings held during the Unicom, San Diego Conference.
Date: March 1st, 1983.

Three Board meetings of the USENIX Association were held during the week of the San Diego Conference, the first on Monday January 24th to deal with items concerning the meeting, the second on Wednesday January 26th and the third on Friday, continuing into Saturday, January 28th and 29th. During the San Diego Conference there was also a meeting of the Board with members.

Monday January 24th, 1982

The meeting was to be divided into three sections, the first dealing with San Diego conference matters, the second a meeting with representatives of Western Electric, and the final part to discuss general business. The first part of the meeting was brought to order at 5.17 pm. Board members present were: Katz, Law, Nemeth and Scherrer. Donnelly and Wedel joined the meeting at 5.40 pm. There were seventeen invited guests: Judy DesHarnais, Local Arrangements Chairwoman of the San Diego meeting; Harry Kerman, technical arrangements; Tom Uter, Chairman of the Technical Program Committee; Bill Applebe and Jim McGinness of the University of San Diego; members of /usr/group Bob Marsh, Lizabeth Riley, Mike Florio, Judy Ross and Joel Carter; members of Software Tools Users Group Neil Groundwater, Barbara Chase, Dave Martin, Skip Egdorf, Bob Upshaw, Dave Stoffel and Tonia Cantrell.

The forthcoming meeting was a joint meeting of three groups: the USENIX Association, /usr/group and Software Tools.

The San Diego meeting

DesHarnais reported that there were 1500 registrations at that time, with approximately 2000 expected. Everything seemed to be under control; the Town and Country and several other local hotels were booked to capacity and bookings had gone smoothly; expenditures for the conference were well within budget for those items over which she had control — most of the remainder were per person expenditures which would automatically be covered by registration; the introductory tutorial was not fully subscribed; eight rooms had been reserved all week for birds-of-feather sessions and other uses on a first come, first served booking basis.

There was one major problem which did not directly affect the meeting, but warranted attention for future meetings, and would cause much added effort after this meeting — registration for tutorials. Several of the tutorials were vastly oversubscribed, and money had to be returned to unsuccessful applicants. There was discussion as to why tutorials could not be replaced by equivalent technical sessions, but it was felt this was impossible when dealing with licensed material. The limitation on numbers of attendees were placed by the people giving the tutorials.

Minor problems concerning the vendor space were discussed, and Florio was asked to negotiate with the parties involved and settle the issues. The question of the possibility of local taxes on admissions to the vendor exhibition was raised — separate accounting of the monies taken would be kept. There was agreement on what credentials were required for access as press representatives, and that the press would be admitted to the reception; Ross affirmed a press room was available and ready.

Four hundred copies of the proceedings of the Boston Conference were available for sale at the conference for $25 each; proceedings of this conference would be produced by Penny, Penny and Strong under contract to the Software Tools Group at a price of $15 for members, which was
The meeting recessed at 6.30 pm., reconvening at 8.17 pm with all members of the Board and Tom Strong present.

*Meeting with AT&T*

Before the guests from AT&T arrived, Strong stated there had been many problems verifying licenses. AT&T had been very helpful, but further clarification as to requirements was needed. Information was also needed on what manuals the USENIX Office could reproduce and sell.

Bob Guffey, Larry Isley, and Otis Wilson from AT&T joined the meeting at 8.39 pm. Procedures for license verification and distribution of licensed material were discussed and agreed upon, as were procedures for verifying that people attending "licensed material" tutorials were covered by the appropriate license. Help was offered in reducing problems resulting from licensing requirements.

Wilson stated that the USENIX Office will be provided with a license to cover all administrative functions connected with licensed material.

AT&T has no objection to others reproducing and selling manuals, subject to copyright permission being obtained. System 5 manuals, of which there will be many, will be available from the AT&T Indiana Publications system.

The AT&T guests left the meeting at 9.45 pm. There was a general feeling that it had been a very productive session.

*General business*

Katz reported that he was informed at a meeting in mid January with Bob Marsh (President of `/usr/group`), that `/usr/group` had decided at their last Board meeting in October that future conferences would be held separately, and that there would be an Oct. 1983 conference run by the group in Washington using a professional organization. The remainder of the meeting was largely taken up with discussion as to the effect of this, the reasons for it, and whether the USENIX Board wanted to hold future conferences with `/usr/group` if the opportunity arose. The Toronto meeting in the summer of 1983 would proceed as a USENIX meeting but it was felt that prediction of the size of the conference and the effect on the vendor exhibition would be difficult.

A revised proposal from Rogal (a conference services company, which was used for the Boston meeting) had been received. Donnelly was due to meet with Rogal and Human Computing Resources (the Toronto host) the following day. There was discussion as to what company to use for provision of meeting services in the future, but this was not settled.

Further discussion revolved around relations with `/usr/group`, resulting in a motion that the "Board of Directors of the USENIX Association regrets the decision of `/usr/group` to discontinue the holding of joint meetings".

The meeting adjourned at 11.02 pm.
Meeting of the Board with the Membership Wednesday January 25th, 1983, at 6 pm

The Board meeting was called to order at 6.10 pm. All Board members were present, and there were approximately 30 attendees, with numbers fluctuating during the meeting. Katz presented a summary of the situation concerning sale of manuals.

The main topic of the meeting was the subject of joint conferences with /usr/group. Katz outlined what had happened, what was in process in terms of negotiations with /usr/group, and the alternatives as presently seen. These appeared to be:

1. Two meetings per year, similar to UNICOM
2. One large joint meeting, following the Comdex model
3. One large meeting, as above, with one additional small meeting having no vendor exhibition.

There were a few strongly voiced opinions in favour of reducing the size and scope of the meetings, getting back to solid technical sessions only. In general, there seemed to be sentiment for a loosely coupled joint meeting, with the opinion that the technical sessions and the commercial activities are largely orthogonal. There was strong sentiment that the time allocated for technical presentations was too short.

The meeting adjourned at 7.30 pm.

Meeting of the USENIX Board during UNICOM

Wednesday, January 26th, 1983.
Present: Katz, Ferrin, Law, Borden, Nemeth, Scherrer, Wedel, Strong, and Randy Frank (University of Utah) as an invited guest.

The agenda for the meeting was discussion of relations with /usr/group. The meeting was brought to order at 11.34 pm.

Katz summarized the present position: /usr/group was planning a September meeting in Washington, and were pressing USENIX to hold a solely technical conference without vendor participation at the planned time in Toronto. There was much discussion of the effects of this decision on the Toronto meeting. It was felt that the decision that had to be made before the end of the meeting was whether there was any prospect of cooperation or not; that any such decision should not be made on the basis of economics of the meeting, but rather that the interests of the members should be paramount.

There were diverse opinions amongst the Board members, from "what is to be gained, if anything, by cooperation" to "there is a lot to be lost" by a total separation. Organizing conferences is far more complicated when done in cooperation with other organizations, unless responsibilities are well defined and committed to paper. Nothing could be decided in isolation, but the motion that "USENIX should try to reach an agreement with /usr/group for holding joint meetings" was passed, with conditions that the next meeting should be in Toronto as scheduled, Software Tools should participate, and that any present financial commitments would become an expense of the joint meetings.

Nemeth and Katz were designated to negotiate with /usr/group, and report on results so that decisions could be made at the scheduled Friday evening Board meeting.

The meeting adjourned at approx. 1.15 am.
USENIX Board meeting

Friday Jan 28th, 1983.
Present: All USENIX Board members and Strong.

The meeting was brought to order at 3.51 pm.

As the agenda for the Monday Board meeting had been scrambled by the overriding question of joint meetings, the agenda for this meeting was reorganized.

Scherrrer reported that there was a lot of work sorting out the information received from the now defunct New York USENIX Office. License information seemed to be available for about half of the installations requesting tapes. Of 297 Members eligible to receive tapes, 197 had been sent a tape, and 77 had not submitted a release form. Licensing was a problem, although AT&T had been most cooperative. As there are many different categories of license, after some discussion it was decided to produce a different tape for each class of license, and that organizations should get the appropriate tapes for which licenses were held. There was material submitted at the Boston conference; Release forms are to go out after all submissions have been processed. A problem exists with tape reproduction: Strong was asked to find a commercial duplicating service.

Minutes of the previous meeting were adopted.

Katz reported on newsletter production. There were six issues in 1982, the last actually appearing in Jan 1983 so that conference information could be included. A technical editor is needed, as are technical contributions. O'Dell has agreed to "beat the bushes". The printing schedule for 1983 has been set.

Ferrin gave a budget report. Expenses for closing out the New York office were final, and the bank accounts had been closed. The New York accounting firm of Breinner and Bodian would be asked to prepare and submit the 1982 income tax returns, and also generate a financial statement for publication in ;login:. Accounting and income reporting from the USENIX Office has been good.

Donnelly gave a summary of the finances of the Boston conference: all bills have been paid, and a final report will be presented to /usr/group and Software Tools Users Group as soon as it has been reviewed by Donnelly, Ferrin and Katz.

Strong gave a report on office operations, starting with membership: in 1982 there were 279 Institutional members (149 Educational), and 399 Individual and Public members. Tape distribution was under control, but the question was raised as to whether USENIX was legally liable for exporting software. Katz will get a legal opinion.

Procedures for processing and acceptance of membership are not well defined; a motion was made that the Office Committee define and set up procedures. Modifications were made to the proposed new application forms. It was decided that the table of contents and an index for each issue of ;login: would be published on the net.

Insurance coverage for the Board has not yet been obtained, as information is needed from the original papers of incorporation. Information is also needed as to how to extract the Association from New York State — Katz will follow up.

Katz reported on manuals and publications: it was decided that Berkeley should be asked for copyright approval for reproduction and distribution of 2.9 and 4.1 BSD manuals. It was understood that rights for 4.2BSD manuals would not be available. Availability of other manuals via the Association Office will be announced in the next issue of ;login:.

Ferrin presented a 1983 budget. Working capital is needed for cash flow prior to each conference, and it was felt that it would be prudent to have approximately one years operating expenses as a reserve to cover these requirements. It was not clear whether transient expenses for office start up had died away, and whether equilibrium had been reached. As items for legal and other sundry expenses had not been included, it was agreed an amended version would be produced for the continuation of the meeting on Saturday. There was concern on the part of several Board members about income, as this
is partially derived from conferences, and it is not clear what effect additional /usr/group meetings will have on conference revenues.

Strong, in a report on office operations, said that computer usage at Berkeley was running at about $700 per month. It was thought that at this rate of expenditure, it was not yet time to purchase computing equipment. Concern about security and privacy were expressed — it was decided that all membership files and other Association business records should be kept encrypted. Minutes of Board meetings for which there are final paper copies should be deleted. He asked for guidance on publicity when approached by the press: it was decided that all press contacts should be logged, and that comparisons, either between software or other user organizations were to be avoided; if there were questions or problems, advice should be sought from Wedel or Katz.

The next agenda item was future conferences. A postmortem was to be held in about one months time on the San Diego meeting, with two representatives from each group. Katz was to organize this.

A tentative agreement had been worked out with /usr/group that the there would be two conferences per year, one organized by USENIX during the summer, and a second by /usr/group during the winter. USENIX would be invited to participate in the winter meeting, possibly organizing the technical presentations. /usr/group was invited to join with USENIX in the Toronto conference but declined. This cleared the path for decisions on the Toronto conference. Donnelly was authorized to get legal assistance on the contract with Rogal for conference services, and to engage Suzanne MacNary as Local Arrangements Chairwoman, on the basis of her proposal with the modifications that were discussed and added. Law and Nemeth are to supervise budgets and expenses, and arrangements were made for funding of initial conference expenses.

Ferrin asked whether there might be problems with taxes etc. in Canada. This would be investigated. Insurance coverage was questioned on the same basis — did coverage extend to Canada? Strong was asked to check, and also ensure that MacNary was covered.

The meeting following Toronto is presently scheduled for Salt Lake City in January 1984. /usr/group was asked to consider this location for the winter meeting, but it was decided that options for cancellation of commitments for a meeting at this site in January 1984 should be explored. Wedel was authorized to pursue this. Host organizations in Portland and Chicago are willing to host future conferences.

The meeting recessed at 10:36 pm, until 9 am the following morning.

Present: all Board members plus Strong. Ron Baecker, Mike Tilson and Ed Borkovsky all of HCR, the host company for the Toronto meeting, and Suzanne MacNary joined the meeting as guests.

The agenda for this meeting was arrangements for the Toronto conference. Nemeth summarized the Board assignments for the conference: Law as conference treasurer; Nemeth as technical program coordinator; Borden to organize tutorials; Katz to coordinate logistics, A/V and physical plant; and Donnelly other things not specifically designated above. Tilson requested a summary in writing, and asked for a handbook for conference organization. Many necessary details for the conference were discussed and decided upon, in particular a schedule for mailing of information, registration materials, and information concerning vendor exhibits. MacNary was asked to pursue questions concerning Canadian income and sales tax implications. It would be Rogal’s responsibility to handle all vendor problems, including the Canadian border import/export questions.

The date for the next Board meeting was set as Sunday and Monday March 27th and 28th, 1983.

The meeting was adjourned by common consent at 11.15 am.