September 1982

Vol IV No V

Australian UNIX Users Group

N E W S L E T T E R

This issue is devoted entirely to the abstracts of the Unix meeting held in the US in July 1982. This may seem a waste to our American readers and those that actually attended the meeting. However the vast majority of Australian Unix users won't have seen this information and we feel it will be of interest. Australia is isolated from the main Unixverse and most Australia Unix users have no idea of who is doing what in the USA. Printing the abstracts of the talks given at Usenix may help.

Bob Kummerfeld

ABSTRACTS

Tuesday, July 6, 1982
Copley Ballroom

8:45 AM <u>Welcome</u>
Alan G. Nemeth
Technical Program Committee Chairman

8:50 AM <u>Welcome</u>
Lou Katz
USENIX President

8:55 AM <u>Welcome</u>
Bob Marsh
/usr/group President

<u>Session Chair - John Donnelly, NCAR</u>

9:00 AM <u>Embedding UNIX in a Product (or, is "Real-Time" Real?)</u>

William R. Northlich Jr., T.D. McCreery
Zehntel, Inc.
2625 Shadelands Dr.
Walnut Creek, CA 94595
P.M. Powers
Megatest Inc.

Rampant buzzwordology has resulted in UNIX having a reputation as not suitable for "real-time" applications. First, some comments are made about the meaning of "real-time". Then an example is given of how UNIX was used as the basis of a computer system embedded in an electronic instrument. Some performance statistics and other facts about the system are given.

9:20 AM <u>Implementing a Multiple-Process Real-Time System Under Unix</u>

A.V. Hays, Jr., B.J. Richmond, and L.M. Optican
National Eye Institute
National Institute of Mental Health
9000 Rockville Pike
Bethesda, MD. 20205

REX (Real-time EXperimentation) is a system for real-time laboratory data acquisition and control that divides its functions among various cooperating processes. A running REX system includes a "comm" process to control keyboard interaction, a "scribe" process to write data on disk, an "int" process to respond to interrupts from clocks, analog-to-digital converters, etc., and a "display" process to generate online graphic displays. Laboratory control is accomplished with a state based interpreter in the "int" process. To implement REX under Unix it was necessary to modify the kernel. These modifications are tailored for smaller PDP11s (such as the 11/23 or 11/34) in an environment in which the processor is devoted entirely to the real-time tasks at hand. Capabilities added include mechanisms for shared data segments, interprocess messages, and user response to device interrupts. The attach() primitive allows the "int" process to answer device interrupts with minimum latency. For interrupts at priority seven on an 11/34 without cache the worst case latency to execution of a C language procedure in the "int" process is 90 microseconds, irrespective of whether Unix is running in user or kernel mode at the time of the interrupt. This time includes mapping the "int" process

into virtual address space if not currently mapped, and time spent by the C language procedure saving registers and adjusting the stack frame.

9:40 AM <u>Real-Time Performance</u>

Johann George
Mark Williams Company
1430 West Wrightwood Avenue
Chicago, IL 60614

One of the most important uses of operating systems is in real-time applications. Several real-time operating systems exist. The problems arise when one desires a real-time operating system that is portable. The presentation will contain a discussion of the problems involved and the efforts made to solve them. Some of the problems discussed will be reducing interrupt lockout time and allowing user programs more control over input/output operations.

10:00 AM <u>Real-Time Systems</u>

Jim Isaak
Charles River Data Systems
Natick, MA 01760

Definition: "A Real-Time System acquires data and processes it at execution time."

This is deliberately broad because the range of real-time is broad. High performance real-time systems deal with data rates at 100MHz or higher -- some of these must perform computations on this data and store it on media while sustaining these rates. Others take bursts, perhaps 1-2 MBytes in a second; then can process this in a lower priority batch mode. The other end of the spectrum includes environmental monitoring, energy management applications which may sample a few times per hour on a "n-minute" basis with computations or output every hour, day or week. Even "commercial" applications have real-time characteristics -- simple data entry and word processing are sensitive to response time.

For a person seeking a real-time system, a number of application characteristics need to be determined: (1) what data collection is involved -- rates, precision, interrupt and channel loads; (2) what memory capacity implications exist, filtering a large array (over 64Kb) may be a major effort and performance problem in a limited address space; (3) what computations are required -- data precision, performance, quantity; (4) what output requirements exist -- priority controls, critical response, data logging, display, etc., and finally, (5) what needs to be done at the same time (vs. "Batch" mode). These define a context for evaluating any system's real-time nature. Needless to say, some tasks will find few (if any) systems adequate, others will find many can do the job. If none can do the Job (with margins) then division into separate processors, custom hardware and custom systems are the techniques that are appropriate. (These same techniques permit the use of systems which would not otherwise be able to do the job, often with substantial savings in overall system cost if any quantity is involved.)

The functions a system (hardware and software) needs to perform a majority of real-time tasks include: address space greater than 128Kb per process; DMA channel bandwidth greater than 5MB/sec.;

multitasking/multiprocessing priority scheduling of tasks; user device drivers; reliable synchronization techniques; queuing mechanisms for messages; shared data space for high priority status information; exception processing for controlled recovery; high performance-high reliability file management; resident process locking, and processor/computational performance.

The final measure of real-time fitness is in actual application performance. Here, the quality of software (and hardware) engineering involved can literally make orders of magnitude difference in both performance and system cost.

10:20 AM <u>Coffee Break</u>

<u>Session Chair - Joseph Yao, Science Applications, Inc.</u>

10:40 AM <u>Optimizing Database Queries in SQL</u>

Dennis F. Meyer
UNIQ Computer Corporation
28 South Water Street
Batavia, IL 60510

The author describes a UNIX based optimizing compiler and run-time interpreter for database queries written in the Structured Query Language (SQL). This development centers on a slightly modified subset of the query facilities of SQL. The UNIX software tools lex and yacc are utilized to generate a lexical analyzer and syntax analyzer implemented in the parser phase of the compiler. The parser is responsible for diagnosing syntax errors, creating the initial abstract syntax tree and building the symbol tables that describe the database relations and their attributes. The optimizer phase of the compiler produces an intermediate file containing a version of the source queries from which faster object code can be produced. These improvements are realized by making transformations to the syntax tree created by the parser phase. The interpreter, which is designed with speed and efficiency as primary objectives, is responsible for executing the queries prepared by the compiler. Major design emphasis was directed to the interpreter data structures in order to achieve these objectives.

Several examples are included which show the details of a simple projection, selection and projection, a complex query and the join operator. This system provides a powerful and efficient query facility which has been effectively ported to several different operating environments.

11:00 AM <u>Interfacing UNIX to Backend Database Machines</u>

Michael E. Duffy
Consultant
65 Franklin Street
Allston, MA 02134

Along with growing interest in using UNIX for end-user applications has come the realization that standard UNIX is ill-suited for supporting multiple users requiring fast, concurrent access to large databases (to be fair, many operating systems possess this failing).

Database operations represent a substantial load on a UNIX system,

resulting in low system throughput and grinding response. This is particularly true of PDP-11 UNIX, which does not have the advantages offered by virtual memory implementations. Moreover, UNIX does not provide adequate facilities for ensuring that database integrity is maintained during concurrent access or machine failure. In a multiple processor environment, required when the number of users to be supported becomes large, the problems grow even more complex (synchronization, communication delays).

Use of a backend database processor offers an attractive solution to the problem of accessing large databases from UNIX-based host systems. In addition to offloading the host processors and providing concurrency control, a backend can also provide such facilities as audit trails, checkpointing, and crash recovery.

This presentation describes the interface between a Britton/Lee IDM-500 database processor and 5 PDP-11/70s running UNIX Version 6. A large legal firm intends to use this configuration as the base for a client information/accounting system. Among the topics to be covered during this presentation are:

- an overview of the IDM system
- a description of the UNIX driver
- problems encountered during implementation
- performance measurements
- possible enhancements to the driver for improved performance

11:20 AM <u>Time and Tuples</u>: <u>Concurrency Control in LOGIX</u>

Fred M. Katz
Logical Software, Inc.
55 Wheeler St.
Cambridge, MA 02138

A dated relation is a relation that carries its history around. Specifically, the file which holds the relation contains an entry indicating the date and time at which each item was inserted into the relation and when, if ever, the item was subsequently deleted. For the purpose of dating a relation, its history is divided into a series of discrete moments. When a relation is opened for reading, it is always opened "as of" some particular moment (present or past) and the reading process gets a consistent, though possibly stale, view of the relation. When a relation (or a part of the relation) is locked for writing, a new moment is assigned to the relation; updates made under the auspices of the lock have no effect on other processes which begin before the lock is released.

In addition to helping with concurrency control, dated relations provide retrospective views of a database and facilitate rollback to an earlier, and presumably more innocent, state.

The incremental space requirements for dated relations are prima facie, frightening. But, where dated relations are used to provide retrospective views, the space required has to be compared with what would be needed to keep more than one version of a relation. If dated relations are used only for concurrency control and recovery, LOGIX can be persuaded to retain only the recent history of a relation.

11:40 AM <u>Lunch</u>

<u>Session Chair - Bill Joy, U.C. Berkeley</u>

1:20 PM <u>UNIX at Lucasfilm Ltd. or Does Darth Vader Code in C?</u>

Bill Reeves
Lucasfilm Ltd.
PO Box 2009
San Rafael, CA 94912

This talk will present a brief overview of the many projects at Lucasfilm that are using the UNIX environment. The applications include: high-resolution three-dimensional color graphics, digital audio signal processing, video editing, database management, word processing, and computer games. Our work in bringing up UNIX Version VII on the SUN 68000 will be briefly described as well as a simple and clear implementation of an extended file system for the networking code of Berkeley's 4.1absd.

1:50 PM <u>4.2BSD Overview</u>

Bill Joy
U.C. Berkeley
Computer Systems Research Group
Computer Science Division
Berkeley, CA 94720

An overview of the new facilities of 4.2bsd: virtual memory facilities, inter-process communication, networking and file system enhancements.

2:10 PM <u>4.2BSD Network Communications</u>

Sam Leffler
U.C. Berkeley
Computer Systems Research Group
Computer Science Division
Berkeley, CA 94720

An introduction to the networking facilities of 4.2bsd. Will describe the structure of the networking code, its place in the new system, and how new network protocols and network interfaces can be added to the system. Will include discussion of the currently available protocols and interfaces supported by the networking subsystem.

2:50 PM <u>Coffee Break</u>

3:20 PM <u>4.2BSD Interprocess Communications Primer</u>

Bill Joy
U.C. Berkeley
Computer Systems Research Group
Computer Science Division
Berkeley, CA 94720

Will describe the new interprocess communications facilities of the system and how they can be used to construct multiproces and distributed applications. Examples of the different styles of use of the facilities will be given.

ABSTRACTS

4:00 PM <u>4.2BSD File System</u>

Kirk McKusick
U.C. Berkeley
Computer Systems Research Group
Computer Science Division
Berkeley, CA 94720

Will describe the new file system organization and measurements of its performance. Impact on user programs of changes in the directory system will be described.

4:40 PM <u>4.2BSD Licensing</u>

David Mosher
U.C. Berkeley
Computer Systems Research Group
Computer Science Division
Berkeley, CA 94720

Will describe the current licensing situation with 4.2bsd and plans for 4.2bsd licensing.

4:50 PM <u>4.2BSD Questions and Answers</u>

Joy, Leffler, McKusick and Mosher

A give and take where we reveal all.

ABSTRACTS

Wednesday, July 7, 1982
Copley Ballroom

Session Chair - Mike Zuhl, Tektronix

8:40 AM **Ped - A Portable Editor**

Mario Ruggiero
University of Toronto Computing Services, CSS
20 St. George St.
Toronto, Ontario, Canada M5S 1A1

Over the last several years, the VIVA Project has been evolving at the University of Toronto as a student work station built from state of the art components. One of these components is Ped, a full screen editor.

Ped is designed to be portable to a wide variety of operating system environments. It is coded in Pascal to enhance portability. The target systems on which Ped is expected to run are UNIX, VAX/VMS, and the UCSD p-system.

A second design goal for Ped is that it be usable with a variety of different terminal types. This is accomplished by the use of a subset of the UNIX "termcap" mechanism.

9:00 AM **Portability of C Language Programs**

Chaim E. Schaap
Delft Consulting Corporation
392 Bleecker Street
New York, NY 10014

Frequent changes in hardware and rapidly increasing costs of software force us to write more portable code. The C language lends itself well to the task of writing portable programs if certain avoidable pitfalls are recognized.

This talk will focus on the following issues:

1. The de facto C standard and the growing variety of C dialects.

2. How standard is the standard library? C portability in general versus inter-UNIX C portability.

3. Differences in machines and compilers that affect the portability of C code.

4. Techniques, tools, and rules for writing portable C code.

9:20 AM **The Object Oriented Pre-Compiler: Programming Smalltalk 80 Methods in C Language**

Dr. Brad J. Cox
ITT Programming Technology Center
1000 Oronoque Lane
Stratford, CT 06497

This describes the Object Oriented Pre-Compiler, (OOPC, pronounced Oopsy), a pre-compiler and a library of routines for producing C language programs that operate according to the run-time conventions

of the Smalltalk 80 language.

Smalltalk 80 and OOPC are two programming languages that offer Object Oriented Programming, a technique in which both data and the programs which operate on that data are designed, built and maintained as inseparable units called objects. This is a powerful technique for decomposing difficult software design problems into more manageable pieces, and is of great interest as a conceptual basis, or organizing principle behind the ITT Future Programming Environment.

9:40 AM <u>Teaching AWK as a First Programming Language</u>

Bill Tuthill
Computing Services
University of California
Berkeley, CA 94720

Considered as a language, AWK is excellent for teaching students how to program for the first time. It requires no variable declarations, has no complex I/O or file-handling routines, and includes excellent control flow facilities. However, the current implementation gives unspecific error messages that are not very helpful for beginners. Furthermore, AWK cannot recover from certain syntax errors, and will core dump instead.

An new AWK preprocessor is available, which was used in a course on Computers in the Humanities. Patterned after Berkeley PASCAL, the new front-end points to syntax errors, giving a short explanation of the problem. It can be placed in a /bin that comes earlier than /usr/bin in the search path. If a program contains no syntax errors, the front-end will exec /usr/bin/awk and exit; AWK will in turn interpret the user's program. If there are fatal errors in a program, the syntax checker will exit without calling AWK.

One drawback of AWK is that it is non-procedural, so students never learn how to write their own functions. Also, AWK makes it difficult to access single array elements, and provides nothing for opening several files at once. Unfortunately for a "string processing" language, strings are limited in size, and the user has no control over these limits. Many programs fail because fields or records of input data are too long. If these limits were enlarged, or if they could be user-specified and dynamically allocated at run time, AWK would be far more usable.

10:00 AM <u>NUnix Window System Description</u>

Jack A. Test
Laboratory for Computer Science
Massachusetts Institute of Technology
Room 414
545 Technology Square
Cambridge, Ma. 02139

The NUnix Window System, developed for use with the MIT-RTS NU-Terminal, is implemented via a set of device-driver routines in the Unix kernel. Most of the window driver code is machine independent with the exception of two low-level routines for driving the raster-display and keyboard devices respectively. In keeping with the Unix philosophy of placing minimal-but-general functionality within the kernel, the NUnix Window System provides a basic window management mechanism that: (1) is transparent to the vast majority

of Unix user programs, (2) provides a clean user interface through the ioctl-call mechanism without the addition of any new Unix system calls, and (3) allows user processes to manage their windows independently and with minimal kernel-imposed limitations.

In the NUnix Window System: (1) windows are independent Unix teletype devices which respond to all TIOCxxxx standard ioctl-calls and can be frozen-thawed independently, etc., (2) windows also respond to a set of WIOCxxxx ioctl-calls used to manipulate their "windowness" attributes such as location, size, font-usage, exposure, and keyboard status, (3) windows may or may not have a one-to-one relationship with Unix processes, (4) every window has an independent multi-font-map, (5) every window belongs to a process-group to which a signal is sent whenever the window changes state, (6) only the super-user has control over all windows in the system, otherwise a window can be manipulated only by a process in its process-group or by a process with write permission on the window, (7) all windows, upon creation, are entered in the NUnix file system as character-special devices in the directory "/dev/wdev" with a filename specified by their user-supplied label, and (8) windows are dynamic in that their location, size, exposure, font-map, etc. can be modified at any time under user-program control.

10:20 AM <u>Coffee Break</u>

<u>Session Chair - Mike O'Dell, Lawrence Berkeley Laboratory</u>

10:40 AM <u>Design of an Intelligent Bitmap Terminal</u>

Rich Fortier and Tony Lake
Bolt Beranek and Newman Inc.
10 Moulton St.
Cambridge, MA. 02238

The benefits of flexibility, computing power, and resolution of a desktop computer display terminal must be weighed against the engineering and manufacturing costs inherent in a complex design. This talk will explore the design goals and resulting strategies employed in the development of a high performance, low cost, bitmap terminal for integrated text and graphics processing.

The BitGraph system design is based on a general purpose architecture which implements most of the design complexity in firmware executed by an integral MC68000 microprocessor. Standard interfaces include medium speed asynchronous, high speed synchronous HDLC, local printer, and pointing device or mouse interface. Emulation of a number of popular graphics and character-oriented terminals is supported indirectly in firmware, eliminating the need for procurement of different terminal types. Sophisticated host-resident development tools and ample RAM storage allow custom user programs to fully utilize the BitGraph's processing and display capabilities. Existing UNIX-based applications in word processing, computer aided logic and VLSI design, and distributed processing will be discussed.

11:00 AM <u>The SUN Workstation</u>

Andreas Bechtolsheim
Sun Microsystems
2310 Walsh Ave.
Santa Clara, CA 95051

The SUN workstation is a personal computer system that combines graphics and networking capabilities with powerful local processing. The workstation has been developed for VLSI design automation, text processing, distributed operating systems and programming environments. Clusters of SUN workstations are connected via a local network sharing a network-based file system.

The SUN workstation is based on the Motorola 68000 processor, has a 1024 by 800 pixel bitmap display, and uses Ethernet as its local network. The hardware supports virtual memory management, a "RasterOP" mechanism for high-speed display updates, and data-link-control for the Ethernet. The entire workstation electronics consists of 260 chips mounted on three 6.75 by 12 inch PC boards compatible with IEEE-796 Bus (Intel Multibus).

Keywords: Workstation, Personal Computer, Local Networks, Graphics

CR Categories: 6.21, 8.2

11:20 AM <u>Merging Bitmap Graphics and UNIX</u>

Rob Pike
Bell Labs 2C-521
Murray Hill, NJ 07974

UNIX is a multiprogramming system. A user can run several programs simultaneously, programs that can interact (such as programs in a pipeline) or can be independent (such as parallel compilations). The syntax and semantics of pipelined processes provides a powerful and convenient form of multiprogramming, but UNIX's current mechanisms for dealing with parallel independently executing programs are weak: there is no convenient way to control several processes. The "job control" mechanism of the C-shell merely provides a way to describe which of several processes the user wishes to work with now, and does not provide a capability for letting several programs run simultaneously without interfering with, say, each other's terminal I/O.

Bitmap displays, as they have been traditionally used, take natural first step towards controlling a multiprogramming system. "Window systems" enable a user to store multiple program contexts on the same screen, but they have only (with a couple of exceptions) been static contexts, and are therefore incapable of handling multiprogramming.

The extension of the window system idea to supporting multiprogramming is simple conceptually, but surprisingly difficult to implement. There are interesting problems to solve in the areas of

- inter-process communication
- graphics support
- user interface.

This talk will address the problem and how they have been solved on the Blit terminal, a bitmap display built specifically for improving the user interface to UNIX. It will also discuss some of the extensions of the ideas developed to other areas, such as game playing and text editing.

11:50 AM <u>Lunch</u>

<u>Session Chair - Lauren Weinstein, Independent Consultant</u>

1:20 PM <u>The Multiple Device Queueing System</u>

Douglas P. Kingston III and Michael J. Muuss
Ballistics Research Laboratory
Aberdeen Proving Grounds, MD 21005

The Multiple Device Queueing System (MDQS) is designed to provide UNIX with a full function, modular, and consistent queueing system. The MDQS system has been designed with portability, expandability, robustness, and data integrity as key goals. MDQS is designed around a central queue which is managed by a single privileged daemon. Requests, delayed or immediate, are queued by non-privileged programs. Once queued, requests can be listed, modified or deleted. When the requested device or job stream becomes available, the daemon executes an appropriate server process to handle the request. Once activated, the request can still be canceled or restarted if needed. MDQS can serve as a delayed-execution/batch subsystem and replaces internally the functions of lpr(I) and at(I) as a minimum. MDQS provides the system manager with a number of tools for managing the queueing system. Queues can be created, modified or deleted without the loss of requests. MDQS recognizes and supports both multiple devices per queue and multiple queues per device by mapping input for a logical device to an appropriate physical output device. Anticipating the inevitable, MDQS also provides for crash recovery.

The MDQS system has been developed at the U.S. Army, Ballistics Research Laboratory by Doug Kingston and Michael Muuss to support the work of the laboratory and is available to other UNIX sites upon request or by the distribution tape. The MDQS system is designed to be compilable on a standard V7 UNIX system.

1:40 PM <u>Everything you wanted to know about System III* but Bell was afraid to tell you</u>

Brian Lucas and Mark Kampe
INTERACTIVE Systems Corp.
1212 7th Street
Santa Monica, CA 90401

System III is generally regarded to be V6 + PWB + V7. However, there are many changes and additions in the latest UNIX Software Release from Western Electric. This talk will focus on what's different in the various systems from a system programmer's point of view. A comparison of System III with it's predecessors, V7 and PWB will be given. Programs and kernel features will be compared. Problems that may be encountered in bringing the systems up and in porting V7 and PWB applications to System III will be discussed. Warnings and advice about the distribution will be given. This talk should be informative to those about to bring up a System III tape from Western Electric as well as to those contemplating a port of the UNIX system to another processor.

*System III is a trademark of Bell Laboratories.

2:00 PM **Bad Block Handling on the BBN C-Machine**

Steve Dyer
BBN Computer Corporation
33 Moulton St.
Cambridge, MA. 02238

This talk reviews the traditional methods implemented at various sites which provide bad block handling, and details the solution provided for BBN Computer's C machines.

UNIX has traditionally ignored the problem of media defects, requiring its users to specify "flag-free" disk packs. This was an appropriate solution for the early seventies' minicomputer disk technology, with its low bit densities and removable media. As storage densities grow higher and higher, and sealed disk systems become increasingly cost-effective, error-free media become more difficult and more expensive to obtain. What's more, a "flag-free" surface cannot be guaranteed over the entire life of a disk pack. Usually, due to normal wear and tear, data errors gradually will appear.

A reliable solution to the problem of bad blocks requires support of a level lower than file system operations. A scheme is needed that will transform a disk containing M sectors, some of which are bad, to a "virtual" disk containing N error-free sectors. For its late-model controllers, DEC has specified hardware and software standards which perform this transformation, and 4.1BSD has implemented DEC's standard for the RK07/RP06/RM03/RM80 disk drives.

Because of the C-machines' unique microcoded device controllers, it was relatively easy to add support for bad block detection and revectoring. The remainder of this discussion describes the solution we chose, its differences from the DEC standard, and the microcode and C-level changes which were needed.

2:20 PM **The New Curses and Terminfo Package**

Mark Horton
Bell Laboratories
6200 E. Broad St.
Columbus, OH 43213

Terminfo is a database describing many capabilities of over 150 different terminals. Curses is a subroutine package which presents a high level screen model to the programmer, while dealing with issues such as terminal differences and optimization of output to change one screenful of text into another.

Terminfo is based on the termcap database, but contains a number of improvements and extensions. Parameterized strings are introduced, making it possible to describe such capabilities as video attributes, and to handle far more unusual terminals than possible with termcap.

Curses is a CRT screen handling package. The package makes use of the insert and delete line and character features of terminals so equipped, and determines how to optimally use these features with no help from the programmer. It allows arbitrary combinations of video attributes to be displayed, even on terminals that leave "magic cookies" on the screen to mark changes in attributes.

2:40 PM <u>Programdb: Maintaining Symbol Use Data for Source Code Control</u>

Douglas I. Kalish
Logical Software, Inc.
55 Wheeler St.
Cambridge, MA 02138

Programdb is a collection of UNIX commands and LOGIX queries which provide information needed to debug, maintain, and update program modules. Programdb maintains a database which identifies program symbols and describes where symbols are used and defined. Program modules are identified by name and size, and comments embedded in the source functions. Programdb produces reports which relate symbols to the modules which define and use them. UNIX library calls and undefined or unused symbols are easily spotted. Dependency lists are generated describing which modules are required by other modules for symbol definitions. These lists can be used as input to makefile(1) to maintain program module inter-dependence.

A UNIX shell procedure collects name list data (using nm(1) and size information for object files, extracts identifying comments from source files and loads the data into LOGIX relations. Subsequent command procedures containing UNIX and LOGIX database commands build the database from joins, projects and selects of the stored data. LOGIX queries are provided which report symbol definitions and uses either individually for a given symbol, for all symbols defined in a given module, for all symbols with a particular lexical content, or for all symbols in the database. These lists are valuable for tracing symbol usage when symbol definitions are altered, or modules are restructured. The UNIX command makefile requires a current and complete dependency list to be useful. Programdb can help create and maintain these lists, ensuring their accuracy and freeing programmers from the tedious task of maintaining dependency tables.

3:00 PM <u>The Logical Softshell: A full-screen interface to UNIX</u>

Fred M. Katz
Logical Software, Inc.
55 Wheeler St.
Cambridge, MA 02138

The Logical Softshell, or "L-shell", allows users to invoke commands and specify parameters in either of two modes: scroll mode - in which the user furnishes the familiar, succinct, UNIX command lines; and full-screen mode - in which the user selects from menus, fills in forms, and toggles through options.

Shell scripts and programs direct the L-shell which invoked them to display screens and to read or write fields within those screens. The L-shell makes it possible to provide a full-screen interface for existing application programs and UNIX commands without changing the programs themselves. The L-shell provides new application programs with a generous addition to their functional capabilities at the cost of including a very small amount of additional code.

There are two implementations of the L-shell. In the first, the L-shell acts as a "shell's shell"; the L-shell runs as a separate process which intervenes between the terminal and some existing shell program (such as the Bourne shell or the C-shell). In the second, the L-shell assumes all of the functions of the Bourne shell and calls commands directly. The relative merits of these two implementations

will be discussed.

3:20 PM <u>Coffee Break</u>

<u>Session Chair - Mike O'Brien, Rand Corporation</u>

3:40 PM <u>Mascot and UNIX: Their Combination and Applications</u>

David M. Tilbrook and Ken Jackson
Systems Designer Ltd.
Systems House
1 Pembroke Road
Camberley, Surrey, England

UNIX is a general purpose, multi-user, interactive operating system for a wide variety of computers.

Mascot is a modular approach to software construction operation and test. It provides an approach leading to a system design based upon a particular formalism for the expression of real time or parallel processing systems. This formalism is represented using a precise diagrammatic notation (ACP Diagrams) which gives good design visibility. Furthermore, a system's ACP diagram is directly transformed into the construction commands that create that system, thus ensuring a congruence between the design and the product.

Office Systems is a catch-all term for systems used to aid office workers in performing their tasks.

This paper introduces ANGUS, a Mascot system built into UNIX for two major reasons: (1) to provide a foundation for a variety of office system applications, and (2) to provide a foundation for software engineering aids such as host/target development systems and configuration management.

The emphasis of this paper will be on Mascot. The need for the formalism is identified and then described, showing how the formalism meets the requirement. This is accompanied by a case study of the design and implementation of a variety of office automation systems, using Mascot on UNIX (Angus), to illustrate the strength and utility of the approach.

4:00 PM <u>How to use lots of memory</u>.

Michael C. Tilson
Human Computing Resources Corp.
10 St. Mary St.
Toronto, Ontario, Canada M4Y 1P9

It has been suggested that small-scale UNIX systems (e.g. PDP-11 based) can not effectively use more than about 1Mbyte of main memory. However, many modern small machines can address considerably more memory. This talk describes a few simple techniques which allow the UNIX system to make good use of several megabytes of main memory. The changes required to UNIX are very small. Substantial performance improvements have been noted, at very low cost. A handout will be available explaining how to install the necessary changes.

4:10 PM **The Ultimate UNIX System**

Robert Ragan-Kelley
Executive Vice President, Development
Pyramid Technology Corporation
2471 E Bayshore Rd., Ste 600
Palo Alto, CA 94303

We believe the ultimate UNIX system should include:

- workstations
- local and long-distance networking which is transparent to the user
- interprocess communication through message passing
- redundant processes for reliability
- intelligent mapped file system to support large address space for virtual memory
- multiple processors
- specialized hardware (e.g. database accelerators)
- high-level functions implemented in hardware such as: scheduling, library functions, signal/wait mechanisms, procedure calls, commonly used instructions, support for multiple processors.

We will outline the integration of UNIX on the Pyramid computer, a distributed-function architecture built around a very high-speed bus, and optimized for high-level languages such as C. The initial implementation of UNIX will be standard System III with a subset of the Berkeley enhancements, but we will discuss how we will extend UNIX to accommodate the above features.

4:30 PM **A High-Performance Computer System Suited to UNIX**

Charles Minter
INTERACTIVE Systems
1212 7th Street
Santa Monica, CA 90401

This paper describes a computer system designed to run UNIX. The system consists of a CPU with memory management and a cache, error correcting memory, and intelligent I/O processors.

The CPU uses a Motorola 68000 running at 12.5 MHz. The memory management unit uses a segmentation system optimized for speed. The grain size is 4 Kbytes and the segment size is between 4 Kbytes and 1 Mbyte. Other memory management organizations were considered but were rejected for performance reasons.

The cache size can be 4 Kbytes, 8 Kbytes, or 16 Kbytes. Both the cache and the data path to main memory is 32-bits wide. The cache uses a one-way set-associative design with write-through. Physical addresses are used in the cache address match logic to minimize cache flushes. The cache is highly integrated with the memory management unit to maximize performance. A 68000 memory read that produces a cache hit requires no CPU wait states.

One, and optionally two, data channels can support simultaneous data transfers of over 1 Mbyte per second each. These transfers are passed through the cache logic so that the cache contents remain valid after DMA transfers. These channels would be used by two disks or a disk and a high-speed network. Off-card adaptors connect these channels to either an ANSI X3T9 8-inch "Winchester" interface or an SMD interface.

Slower speed I/O devices are connected through a special I/O bus designed for easy interfacing. The expectation is that all devices on this bus will be controlled by a local processor that preprocesses and buffers data. The main CPU can transfer data from the I/O bus either with programmed transfers or with DMA transfers. These transfers also go through the cache logic so that, like the high-speed channel interfaces, cache data remains valid.

We have run some simple benchmarks written in C on this system. The performance varied from between 40% to 130% of a VAX 11/780. The only benchmark where the performance was not very near to or somewhat better than that of a VAX 11/780 was one that used 32-bit integer multiplies. The 68000 has no such instruction while the VAX does. It would certainly be possible to construct programs where the performance relative to the VAX would be even worse than the 40% number above. Most benchmarks we ran showed a performance of from 110% to 120% of a VAX 11/780. We have run no floating point benchmarks but since the system has no floating point hardware the performance should be substantially worse than the performance of the other benchmarks.

4:50 PM <u>Perkin-Elmer's Hardware/I-O System: Flexibility That Matches UNIX</u>

Joel R. Carter
The Wollongong Group, Inc.
1135A San Antonio Road
Palo Alto, CA 94303

Anyone who has worked in the UNIX environment knows there are a vast set of choices available to perform any single task. Perkin-Elmer's hardware offers to the system designer a similar array of choices for system configuration. The hardware alternatives this talk will cover are bus switches, shared memories, multi-processor systems, and a complete range of low to very high bandwidth I/O options. The I/O options are programmed I/O, the auto driver channel, the DIOs, and the selector channel. A brief description of the capabilities and limitations of each alternative will be included. We will discuss specifically how system designers have and are integrating and using these hardware options in the UNIX environment. We will present some exemplary and unusual case studies. The conclusion of the talk will hint at future directions.

5:10 PM <u>Introduction to UNIX - videotape</u>

Fred Gerkin
Bell Laboratories Public Relations Office
150 JFK Parkway
Short Hills, NJ 07974

This 22 minute videotape provides some discussion of the philosophy behind UNIX and its major features as viewed by its designers.

ABSTRACTS

Thursday, July 8, 1982
Copley Ballroom

Session Chair - Bob Marsh, Plexus Computers

8:30 AM News from USENIX
Lou Katz

8:45 AM News from /usr/group
Bob Marsh

9:00 AM News from AT&T
Larry Isley
Technology Licensing Manager
American Telephone and Telegraph
P.O. Box 2500
Greensboro, N.C. 27420

9:20 AM News from DEC
Armando Stettner
Digital Equipment Corporation
Continental Blvd.
Merrimack, NH 03054

9:40 AM Introduction of the EDUCOM-UNIX Task Force
Carolyn Autrey-Hunley and Jack McCredie

9:50 AM What's New at Purdue/EE Dept.
George Goble
Purdue University
Electrical Engineering Department
W. Lafayette, IN 47907

10:00 AM Rogue: Where It Has Been, Why It Was There, And Why It Shouldn't Have Been There In The First Place
Michael C. Toy, Esquire and Kenneth C.R.C. Arnold
University of California-Berkeley
Computer Science Division, EECS
Berkeley, CA 94720

A not-too-serious trip through the history of the computer game rogue, where it is now, and where we would like it to go. Looking at the past will include such topics as: why rogue is necessary; why monsters aren't affected by traps; ways to cheat we have fixed; and who is McTesq, anyway? Plans for the future will include multiple player rogue, rogue servers, home rogue, hologram graphics rogue. . ..

(Rogue is a visual CRT based fantasy game which runs under the UNIX time-sharing system. Rogue differs from most computer fantasy games in that it is screen oriented. Commands are all a few keystrokes (as opposed to pseudo English sentences) and the results of your commands are displayed graphically on the screen rather than being explained in words.)

The talk will end with our rationalization of why we aren't responsible for all the money and time people have spent playing rogue when they "ought" to have been doing work, and what the role of games is in computing environments.

10:20 AM <u>Coffee Break</u>

10:40 AM <u>UNIX etc. at National Instruments</u>

Jeffrey L. Kodosky
National Instruments
12109 Technology Blvd.
Austin, TX 78759

This presentation describes National Instruments' use of UNIX for integrated operations support, networking and real-time instrumentation systems. Also discussed are products currently being marketed, those soon to be marketed, current UNIX based projects, in-house activities, and software contributions offered at this meeting.

National Instruments was founded by employees of the Applied Research Labs UT Austin at the conclusion of a large hardware and software project for the Navy: an interactive multiprocessor UNIX-based real-time measurement system. The company began by introducing a line of GPIB (aka IEEE Std. 488) interfaces for DEC processors and also supplying comprehensive software for the DEC operating systems and UNIX. This product line serves as the basis of the recently announced software product dubbed NET488, which supports file transfers between any combination of processors, operating systems, and interfaces on the GPIB. Other soon-to-be-announced software products include a UNIX-based electronic spreadsheet program and a business graphics package.

Software development for National Instruments hardware projects relies heavily on the GPIB, UNIX, and C. The in-house GPIB network currently consists of a PDP11/34 running Version 6, a PDP11/04 running RT, an S100 CP/M system, and an IBM personal computer. Each satellite is a development system (of sorts) for a current product or special project. Software development consists of writing a minimal GPIB download program (sometimes in parallel with hardware development of the GPIB interface), procuring a C cross compiler to run on the PDP11/34, and writing a library of routines to interface to the special hardware on the target system (e.g., fuel pumps and credit card readers, or waveform synthesizer and timing hardware).

11:00 AM <u>News From Perkin-Elmer</u>

David J. Preston
Perkin-Elmer
Computer Systems Division
106 Apple Street
Tinton Falls, NJ 07724

The talk will discuss the history and current status of Perkin-Elmer's UNIX offering, Edition VII Workbench*, and our typical customer, configuration, and some of our future plans and directions regarding UNIX. The intent of the presentation is to position our product line with respect to the targeted UNIX customer base we have selected, and provide a short-term view of Perkin-Elmer's planned enhancements to Edition VII Workbench.

*Edition VII Workbench is a product of The Wollongong Group, Inc.

11:20 AM <u>Current Database Research at the Computer Systems Research Group, University of Toronto</u>

John Z. Kornatowski and Ivor Ladd

Rhodnius Incorporated
P.O. Box 1, Station D,
Scarborough, Ontario, Canada M1R 4Y7

This paper reports on current database management research as part of the unified Message Management Systems project under the auspices of Profs. D.C. Tsichritzis, F.H. Lochovsky, and A.O. Mendelzon of the Computer Systems Research Group at the University of Toronto. First, an overview of the Message Management Project is given. Next, three M.Sc. projects which deal with interfaces to the Mistress Relational Database Management System are described in some detail. These are:

1. A Query by Example-like Interface, using a full-screen display

2. An editor-like Interface; instead of editing text, the user edits a relational database.

3. A preprocessor that allows embedding SLQ queries in C programs in a manner similar to EQUEL.

Finally, future plans for the Project are discussed.

11:40 AM <u>Lunch</u>

<u>Session Chair - Lou Katz, U.C. Berkeley</u>

1:20 PM <u>The UCSD MSG System: Iterative Design in the UNIX Environment</u>

Philip J. Mercurio
Cognitive Science Lab, UCSD
C-015
La Jolla, CA 92093

For the past year I have been working on re-designing the human interface to our computer-mail handling software package, MSG. (The original version of MSG was written for our research group by Greg Haerr in 1980, who modified a system obtained from the RAND Corporation.) The project has proceeded in two phases:

1) the user community was polled informally for suggestions and complaints regarding MSG. Many of these ideas were then incorporated into the first new version of the system.

2) a pool of volunteer test-users was drawn from the user community. This pool has subjected each new version of the system to immediate and stringent testing over the past 8 months.

This iterative design technique has produced a human interface preferred by many users of the original version. Some of the features developed as a direct result of interaction with the test-user pool are:

- the addition of feedback to avoid performing unwanted irreversible actions

- customized interfaces via individual user option files

- context-dependent "help" messages

Some additional features are:

- compatibility with UNIX's "delivermail" and ARPA's "RFC 733" formats

- establishment and maintenance of a set of mail aliases across a number of networked (via berknet, uucp, etc.) machines, called "MSGNET". Individual accounts have the same alias on all of the machines in the MSGNET.

- machine aliases, called "mailroutes". A table containing the uucp/berknet routes to many machines allows the user to direct mail to a distant machine by giving only that machine's name, without requiring knowledge of the intervening machines.

1:40 PM <u>ATLAS Test Language</u>

Mark T. Horbal
UNIQ Computer Corporation
28 South Water Street
Batavia, IL 60510

The paper presents a case study of an implementation of a compiler for the ATLAS test language under the UNIX operating system. The paper discusses the design objectives and decisions, the architecture of the resulting compiler development under the UNIX operating system are presented as well. The paper concludes with a discussion of the compiler performance, possible enhancements, and acceptance by the user community.

2:00 PM <u>UNIX and Manufacturing Testing</u>

Jack Dixon
UNIQ Computer Corporation
28 South Water Street
Batavia, IL 60510

The paper describes an application of the UNIX operating system in a CAM (Computer Aided Manufacturing) environment. The application involves the use of UNIX as the operating system on a series of computers used to test complex electronic assemblies. The paper presents the tradeoffs encountered during the design phase and the resulting application software ultimately developed. The paper comments on the intrinsic limitations of the operating system, problems related to the real-time nature of the application, and ways in which these were solved.

2:20 PM <u>Development of a Commercial Applications System Under UNIX</u>

Curtis Sanford and David Walden
BBN Computer Corp.
33 Moulton St.
Cambridge, Ma. 02238

UNIX is well known as a systems development environment. The suitability of UNIX as an environment for the development of commercial applications packages is less well known. Under BBN's version of UNIX over the past two years, we have developed InfoMail, a large, sophisticated, highly user-oriented, production applications package which is sold to run under IBM and DEC operating systems as well as under UNIX. In this paper we describe our use of UNIX facilities for (a) building a high performance application (b)

building a thoroughly tested and reliable application, (c) software configuration control and distribution, (d) highly productive and controlled development by a multi-person group, and (e) building a portable system. Many of these tools and the techniques of their use would be applicable for other groups developing commercial applications software packages.

2:40 PM <u>Application Programming Environment on UNIX</u>

Masatoshi Kurihara and Yukio Ikadai
Software Research Associates, Inc.
9314 Cherry Hill Road #519
College Park, MD 20740

Software Research Associates, Inc. (SRA) is one of the oldest independent software houses in Japan. It was established 15 years ago, and has about 350 programmers now. SRA's main business includes varieties of application software developments for big computer users such as banks, insurance companies, steel manufacturers, etc. Roughly speaking, 70% of those are developments of business data processing application programs using COBOL, and the rest are developments of process control application programs in FORTRAN or assembly languages.

Two years ago, the company decided to change the programming style from the traditional paper-and-pencil approach to the online interactive environment, and installed VAX/UNIX (Berkeley version) in August 1980. Since then, various experiments have been made to construct a practical environment for average application programmers. Some of those are: (1) "Module Testbed for FORTRAN", which was used by a pilot project to increase the programmers' productivity and to improve the quality of resulting programs; (2) "Strictly-Controlled Environment for Novice Programmers", whose purpose was to provide a group of easy-to-use commands under the assumption that novice programmers had no knowledge about UNIX; and (3) "COBOL Interpreter and Testing Tools", which attacked a field that UNIX community usually ignores.

Installation of UNIX environment has affected the life-style of both programmers and project managers. In some cases, it resulted in great increase in productivity and/or reliability. But in other cases, serious "cultural" conflicts and troubles have been introduced.

3:00 PM <u>Managing A Roomful of UNIX Systems</u>

Benjamin J. Woznick
BBN Computer Corporation
33 Moulton Street
Cambridge, MA 02238

In January 1981, the prototype BBNCC C/70 was delivered to the BBN Computer Services Division's UNIX Cost Center, joining a PDP 11/70 and a VAX 11/780. During the last 18 months, this data center has added 15 C/70's, 5 C/30 IMPs and 3 C/30 TACs to become one of the largest single collections of UNIX systems. The UNIX network is an extension of an BBN network, and some of the machines also have direct access to the ARPANET, and there are conventional gateways to that network and to another internal network. The evolution of this network, which supports the ARPANET protocols, and provides access for each user to that wider network's facilities, including electronic mail, is described. Some of the technical and administrative problems are the same as those in smaller operations, especially the ones involving a

single machine with special software. Others are quite different, and one of the objectives was to provide as uniform a service as possible from machine to machine. It was necessary to invent procedures for almost everything: handling administrative requests, doing backups, receiving notices of bugs, transferring software updates to a dozen systems, and so on.

3:20 PM <u>Coffee Break</u>

<u>Session Chair - Marleen Martin, 3COM Corporation</u>

3:40 PM <u>On Ring Architected Local Networks</u>

Howard Salwen
Proteon Associates, Inc.
24 Crescent St.
Waltham, MA 02154

4:00 PM <u>A Family of Portable Systems Based on System III*</u>

Heinz Lycklama and Steve Zucker
INTERACTIVE Systems
1212 7th Street
Santa Monica, CA 90401

INTERACTIVE Systems Corporation is developing a line of products based on UNIX System III. The products, generically called IS/3, are targeted to run on a broad range of processors including the Zilog Z8000 and Motorola 68000 as well as the DEC VAX-11 and PDP-11. Goals for the systems include:

1. Full System III compatibility at the programmer, as well as the user, interface.

2. Ease of portability of applications not only across systems running IS/3 but to Version 7 and other selected variants of UNIX as well.

3. Enhanced performance, robustness, and generality.

4. Improved system management tools.

In order to avoid the "least common denominator" syndrome, which makes it possible to take advantage of features that are not available on all the systems, we have defined a programming environment and a set of programming standards for applications that masks the differences between the systems with porting libraries and include files. The decision-making process that went into the selection of features to be included and excluded provides an interesting case study for anyone interested in developing portable software for UNIX systems.

The major performance enhancement to date has been the parameterization of disk block size to permit the throughput improvements achieved by Berkeley on systems where the space lost to fragmentation is tolerable, the memory for larger buffers is available, and the hardware can support larger transfers. For the VAX, additional device and adapter support has also been added based on the work at Berkeley.

The system management tools have been improved not so much by changing

what they do but how they are told what to do. For example, all the file system utilities -- fsck, df, mount, mkfs, dump, restor, etc. -- make use of an attribute-value file (/etc/filesystems) which specify all defaults on a per-filesystem basis. This makes it much easier to change file system layout, lessens the burden on the system manager to remember details such as device names and disk interleaving factors and the like, and consolidates the information that is normally scattered about in separate files for each utility, in various manual entries, and in the system manager's head. Another attribute-value file consolidates the information associated with ports and provides greater control over port modes and usage than other systems.

*System III is a trademark of Bell Laboratories.

4:20 PM <u>Current Status of Mistress (Version 2) and Future Plans</u>

John Z. Kornatowski and Ivor Ladd
Rhodnius Incorporated
P.O. Box 1, Station D,
Scarborough, Ontario, Canada M1R 4Y7

Current Mistress Database Management System enhancements at Rhodnius are described briefly, and an overview of the new version of Mistress due for release June 1982 is presented.

Highlights of the features of Version 2 include: built-in sorting capability; creating relations on the fly (results of queries placed in new relations); a simple macro facility to save tedious typing; range checking on data entry; and a new text data type (variable-length record).

The long-awaited Report Generator has arrived and is available for our customers' delectation. A new set of Host-Language Interface routines has been written, simplifying the low-level programming language interface considerably. Programmer productivity is increased by up to 250%, and the amount of code necessary for applications is cut by an average of 50%.

Mistress has been ported to several new machines, including a Eastern Electric Wringer Washer, Serial# AJ203774K, running Cistern 3 BRANDX. (Washer version available only under special un-educational profit-only license.)

4:40 PM <u>"C" Compiler for Data General AOS/VS</u>

Robert Weisman and Mike Meissner
Data General Corporation
4400 Computer Drive
Westboro, MA 01580

The presentation will be centered around the implementation of a "C" Compiler under Data General's AOS/VS operating system. Discussed will be the compiler and runtime extensions, the AOS/VS system call interface, built-in functions, and listing and crossreference files. Also discussed will be the architecture of the compiler, the I/O implementation, and some of the differences between the Data General compiler and others currently on the market. Lastly, compiler performance and associated information will be presented.

The compiler was implemented using state-of-the-art development techniques, and is part of an ever expanding family of high-level

languages, programmer productivity tools, and operating systems for Data General's Eclipse MV/6000 and MV/8000 family of computers. The presentation will be of interest to all current Data General users, users of "C", and the "C"/UNIX community as a whole. Questions will be addressed during and after the presentation.

5:00 PM <u>Systems Designers Limited Vendor Presentation on Angus</u>

Elwyn Wareham
Systems Designer Ltd.
Systems House
1 Pembroke Road
Camberley, Surrey, England

The presentation will outline a new product from Systems Designers Limited, which extends the facilities available under UNIX by adding new tools aimed at the development of real-time, parallel processing systems.

The background of the company, particularly in the development of such tools, will be outlined with particular emphasis on involvement in Mascot (Modular Approach to Software Construction, Operation and Test) the method which underpins the tools being offered, and which is widely used for real-time systems development in Europe.

Example applications, and the associated benefits of using the new tools will be outlined. Extensions to the initial product range and the additional capabilities offered will be discussed.

5:20 PM <u>UTS: UNIX on the Amdahl 470</u>

Daniel Walsh
Amdahl Corporation
P.O. Box 470
Sunnyvale, CA 94086

The talk will discuss the UTS product currently being licensed by Amdahl. Major points include:

1. features we have added to the system

2. compatibility with existing UNIX software, including difficulty in porting existing UNIX software

3. requirements for installation

4. licensing arrangements

ABSTRACTS

Friday, July 9, 1982
Sheraton Constitution Room

__Session Chair - Clem Cole, U.C. Berkeley__

8:40 AM __Compact Data Analysis Programs for UNIX__

Gary Perlman
Dept. of Psychology, C009
University of California, San Diego
La Jolla, CA 92093

Most data analysis packages have programs that can do more than the analysis desired, including data validation, data transformations, and specific analyses. A problem with this approach is that individual programs with all such capabilities built in are so large as to be unusable on mini-computers without significant space and time (for overlaying) penalties. In the spirit of the UNIX philosophy of combining special purpose modules via pipelines, I designed a small set of data analysis programs. Instead of incorporating elaborate data validation and transforming procedures into every program, a few small programs do the job. A typical data analysis session repeats the following abstract shell script:

```
validate < data
transform < data | analysis
```

After validation and transformations, data can be analyzed by an analysis of variance, multiple regression, or descriptive statistics for univariate or bivariate distributions. Because of some serious human engineering, all these programs have elegant user interfaces that for the most part remove the requirement of specifying design information. These programs have acquired acceptance over the past two years of use at about 40 UNIX installations.

9:00 AM __MENUNIX: An Interface to UNIX Programs and Files__

Gary Perlman
Dept. of Psychology, C009
University of California, San Diego
La Jolla, CA 92093

MENUNIX is a menu driven interface to UNIX in which both programs and files are presented in windows or users' terminal screens. The FILE MENU presents a window into the files of the current working directory and displays the file names, sizes, and a special coding of the access protection that indicates file ownership. Files and directories are distinguished by highlighting as well as the protection code. The PROGRAM MENU displays an extensible content hierarchy of the programs of UNIX with clusters of related programs (called workbenches) for tasks such as manipulating the file system, programming (with sub-workbenches for C, FORTRAN, Pascal, and LISP programming), writing papers, sending and receiving electronic mail, and so on. Because the PROGRAM MENU hierarchy is extensible, new programs and workbenches can be incorporated with ease (for example, we have a workbench for doing statistics). One of the biggest advantages of MENUNIX is being able to search through workbenches to find commands of interest, a task not well supported by the UNIX manual. Workbenches contain all and only those commands related to a task, and these workbenches are a cognitively manageable size, making it possible to become expert in limited areas of UNIX. MENUNIX is an experimental interface in which the psychological problems of user-interface design have been studied. Both psychological theory and data collected under controlled experimental conditions were applied to its design.

9:20 AM <u>Description of a Menu Creation and Interpretation System</u>

Michael J. Heffler
Delft Consulting Corporation
392 Bleecker Street
New York, NY 10014

The Menu Creation and Interpretation System (MCIS) is a menu system development tool that can provide the high-level control structure and the user interface for any application program or software system. MCIS simplifies the task of building menu-driven systems while presenting a user interface that is easily learned and used.

Human factors considerations have been stressed in the system design. A user can move from menu to menu, obtain help when needed, and easily recover from errors. Menu systems created with MCIS are extendable and easily modified, enabling them to meet changing user needs. In addition, through the specification of user profiles, menu systems can be customized to meet the needs of individual users.

MCIS is written in the C programming language under the UNIX operating system. It can be used in any environment that includes a C compiler and the UNIX standard I/O library.

9:40 AM <u>Benchmarking to Eliminate the Benchwarmers</u>

Eugene F. Dronek
Aim Technology
3333 Bowers Avenue, Suite 199
Santa Clara, CA 95051

A portable suite of benchmark programs has been developed to measure UNIX performance on different architectures. Issues behind the design of these programs and interpretation of their results will be presented, highlighted by benchmarks on VAX 750, PDP 11/70 and M68000 machines.

These benchmark programs measure and display system throughput in specific areas -- disk, memory, interprocess communication, and floating point -- as well as plot user response (stretch) under simulated multi-user loads.

10:00 AM <u>A UNIX Benchmarking Tool, and Results from the PDP-11/44, VAX 11/780, and Perkin-Elmer 3242</u>

Martin Tuori
D.C.I.E.M.
P.O. Box 2000
Downsview, Ontario
Canada, M3M 389

A UNIX benchmarking tool has been developed, which can run variable numbers of editors, nroffs, C compiles, and floating point processes. The implementation is based on command files, and is easily modified or extended. It can be used to exercise a system, providing timing information, and checking the consistency of the results obtained from each process. A brief description of its design and use will be given.

This benchmarking tool has been used to compare several UNIX systems, including a PDP-11/44, and VAX 11/780, and a Perkin-Elmer 3242. The two 32 bit systems are configured almost identically, having 3MB main memory, three 300MB discs on two controllers, and floating point processor. The VAX runs 4BSD, while the P-E runs TWG's UNIX Edition VII. Our current results suggest that the P-E is slightly faster at the user component, while the VAX's speed at the system component brings its real time to about 10% less than on the P-E. More detailed results will be available at the time of the conference.

This tool will be made available to all UNIX users through USENIX.

10:20 AM <u>Coffee Break</u>

<u>Session Chair - Mark Seiden, Valid Logic Systems</u>

10:40 AM <u>UTS: UNIX on the Amdahl 470</u>

Daniel Walsh
Amdahl Corporation
P.O. Box 470
Sunnyvale, CA 94086

This talk will discuss the porting of UNIX to run on the Amdahl 470 computer. Major points include:

1. the history of the port, going back to 1975

2. the design decisions that had to be made in transporting UNIX to this architecture

3. efficiency improvements we made, some particular to our architecture, some of more general interest

4. our interpretation of the successes and failures of the port

11:00 AM <u>Porting UNIX to a Personal Computer</u>

Gregory J. O'Brien
Digital Equipment Corporation
110 Spit Brook Rd., ZK1-3/B21
Nashua, N.H. 03062

This paper describes the results of a project to port UNIX V7 to Digital's Professional 350 personal computer. The paper will cover two areas of UNIX portability: How is a personal computer different and how does this affect UNIX? How hard is it to port UNIX to a computer with the same ISP but very different peripherals.

The operating system for a dedicated personal computer must meet different requirements than the OS for a timesharing computer. The paper discusses parameter tuning, and exclusion of system features for a single user system. Personal computers often feature a bit-mapped display and rarely contain a tape drive. The impact of such hardware is described.

Previous reports on the portability of UNIX have discussed the effort needed to port UNIX from a PDP-11 or VAX implementation to another processor. The PC-350 uses the same instruction set processor as a PDP-11/23. The paper will outline the amount of work needed to port to a system with the same ISP but a new bus, new interrupt controller, and all new peripherals.

11:20 AM <u>UNIX/Prime: Porting the UNIX operating system to Prime machines</u>

James L. Weiner and Brian L. Johnson
Computer Science Department
University of New Hampshire
Durham, N.H. 03824

A project is under way at the Computer Science Department of the University of New Hampshire to port the UNIX operating system to run on Prime Equipment. UNIX will run on top of a kernel of the Prime operating system (PRIMOS*), rather than either on a bare bones machine or on top of the full operating system. UNIX will have the same access to kernel operations that PRIMOS has and thus its implementation will not suffer from having to be simulated. Our goal is to develop a standard UNIX without restrictions. Of course, due to the differing hardware there will be slight exceptions.

The Prime hardware makes our project interesting outside of just having UNIX available on a different machine. Memory on the Prime is partitioned into 65K segments which can be in any one of three rings of protection. Every user runs a virtual machine with security provided by the rings rather than changing context and running kernel mode. Prime also has hardware that supports many of the same features as does the UNIX notion of signals. They are actually more flexible since users may define their own. Along with the hardware support for signals is hardware support for semaphores which are used in our implementation as well as available to users.

As part of our project we are developing a general mechanism for treating segments of memory as an abstract data type that supports operations such as allocate, deallocate and share. This will allow an additional form of interprocess communication with no additional overhead as well as a start to incorporating datagrams into a UNIX network.

*PRIMOS is a trademark of Prime Computer

11:40 AM <u>Lunch</u>

<u>Session Chair - Richard Bratt, BBN Computer Corp.</u>

1:20 PM <u>UNIX Emulation, Again</u>

Sanand Patel and Richard Sniderman
Human Computing Resources Corp.
10 St. Mary Street
Toronto, Ontario, Canada M4Y 1P9

VX is a software package which emulates the UNIX environment under another operating system. The UNIX shell is invoked from that system's standard command interpreter, providing access to all UNIX utilities. Essentially all of the UNIX system calls are provided, enabling most UNIX C programs to run with no modification. No changes to the host system are necessary.

VX is a substantial improvement over similar packages in the degree to which UNIX is emulated. In particular, "fork", "exec", inherited file

descriptors, and full UNIX file names are all handled as in a true UNIX environment. Many difficult implementation problems had to be solved.

The major parts of the implementation - process management, file management, and file name translation, will be discussed.

1:40 PM <u>A UNIX Emulator for Vax/VMS</u>

Michael Caplinger
Dept. of Math Sciences
Rice University
PO Box 1892
Houston, TX 77001

Software performance and portability issues, (particularly, the poor performance of the UNIX Fortran compiler) have forced many Vax sites to reluctantly choose VMS over UNIX. However, the programming environment provided by VMS is much less convenient and user-oriented than the UNIX environment, as well as being unfamiliar and restrictive to someone familiar with UNIX.

One solution to the problem (the "Virtual Operation System" (VOS) of Lawrence Berkeley Labs) provides a UNIX-like interface on top of an already existing operating system. It does not, however, address the problem of porting UNIX programs. Using a different approach, David Kashtan of SRI wrote "Eunice", a UNIX simulator. He attempts to solve the porting problem by emulating UNIX at the system call level. Eunice has several problems. Some of the system calls are implemented inefficiently (and some incorrectly.) Also, many factors make large amounts of source-level modification necessary to run arbitrary UNIX programs.

Faced with this problem, we at Rice University are in the process of designing and implementing a complete emulation of UNIX, called Phoenix, consisting of a set of system calls and a new loader developed for VMS. The system calls provide exact simulations of the facilities provided by the UNIX kernel, even providing functions that are not provided by VMS. The loader takes UNIX object modules and produces a VMS executable. This gives Phoenix complete object-level compatibility with UNIX.

The development of Phoenix is being done in a somewhat-enhanced version of Eunice (which already uses the new loader). The completion of this project should make porting of nearly all UNIX programs to VMS a trivial exercise, and will extend the benefits of the UNIX community to VMS sites who, up to now, have been denied access to them.

2:00 PM <u>Selecting a DBMS for a Super Micro</u>

Mike Bender
ZILOG
Building B1-3
1315 Dell Avenue
Campbell, CA 95008

At Zilog, we have been considering various approaches to Database Management on the System 8000 (a 16-bit UNIX machine). Among the DBMSs studied were Oracle, Ingres, Marathon, Mistress, Unify, Logix, Sequitur, MDBS III, and Data Base Plus.

A Super Micro must support a large range of sizes and have full networking capabilities. Some users may want to have a Data Base Machine residing on a network, others may want to distribute their data, and finally there will be those that simply want a stand-alone database. It is important that the DBMS support these possibilities.

To fulfill these capabilities a DBMS has to be more than just efficient, easy to use, etc. There are a few unique features that it should have:

- It must support a relational model, so that the data can be logically divided between machines.
- It must be divisible into a disjoint front-end and back-end so that the user can sit at a different machine from the data.
- It must be dictionary-driven, to help the user with the addressing, and so that integrity control can be centralized.
- It should be able to work as a stand-alone system, independent of the network.

2:20 PM <u>A Business-Oriented File Manager under UNIX, with Contention Control and ISAM</u>

Gary Williams
Durango Systems, Inc.
3003 North First Street
San Jose CA 95134

Many standard applications programs require record-oriented sequential and random access files, and indexed sequential access (ISAM) files, with extensive record-contention control not provided by the standard UNIX file management system. The addition of a pseudo device driver, /dev/locker, and the use of a set of file management routines written in 'C' provide file and record level access control without modifying the standard UNIX file system or penalizing nonapplications users. Support for byte length-mark variable record length sequential access files, fixed record length random or sequential access files, and a combination tree-structure and linked-list ISAM is described. Generalized locker usage for full system resource control is also described, with a sample usage protocol.

2:40 PM <u>IAFORM, An On-Screen Definition Package for Data Retrieval Forms</u>

T. Scott Pyne
Computer Systems Technology Division
Science Applications, Inc.
1710 Goodridge Drive, P.O. Box 1303
McLean, VA 22101

The usefulness of database management systems to computer-naive users has been significantly enhanced by the availability of on-screen query processors, wherein a user can fill in some items of a form and initiate queries based on the filled-in values, and view retrieved records using the form. ORACLE, a relational DBMS from RSI, provides such a facility in its IAP (Interactive Applications Processor), but the forms used must be defined via a tedious question-and-answer process that is decidely not appropriate for those naive users who may be using the forms. SAI has developed a package, called IAFORM, which allows a user to define a form to be used with the on-screen query

system by drawing a picture directly on the screen using a subset of a standard text editor (the Rand editor). The editor takes input in a manner such as "32c" (meaning an alphabetic field of length 32) and draws a field on the screen consisting of the appropriate number of lower case characters (in the example case, 32 'c' characters would be shown). This method allows the user to see exactly how the finished form will appear in use. The editor allows field definitions to be picked up and moved around, so the elements of a form may be juxtaposed after initial definition. Once the user has defined the form to his/her satisfaction, the template is translated into a set of description structures from which a question and answer file is built. This file is submitted to ORACLE's forms definition utility and produces an input file for use with the on-screen query processor.

3:00 PM <u>Tabstar - Information Data Base Management</u>

Gordon W. Waidhofer
The Wollongong Group, Inc.
1135A San Antonio Road
Palo Alto, CA 94303

Tabstar is a collection of utilities for simple data base manipulation, and report generation. Tabstar addresses the range of problems that are technically infeasible with standard UNIX utilities (e.g. grep, uniq, sed) alone, and are economically infeasible with full scale data base systems (e.g. oracle, ingres). A data base is a text file, manipulated by the Tabstar and existing UNIX utilities. Data base views are created as shell command files, and may be used in the same way as data base files. The design strategy of Tabstar is to take as great advantage of existing UNIX text manipulation and test processing facilities as possible. Some of the existing applications are marketing prospect tracking, action item tracking, and documentation control. This presentation describes Tabstar and its use.

3:20 PM <u>Coffee Break</u>

<u>Session Chair - Doug Michels, The Santa Cruz Operation</u>

3:40 PM <u>Is UNIX as a Standard Doomed</u>?

Robert B. Greenberg
4318 Collins Ct., #3
Mountain View, CA 94040

A tremendous ground swell of interest in UNIX by the commercial community began late last decade. The continued growth of excitement is due to UNIX's two principal assets: it has a technically superior design for software development environments, and it is a very likely choice for becoming the operating system standard for low-end business systems.

Last summer, this author discussed many of the shortcomings of UNIX from a technical standpoint. In this talk, he pursues the second point: just how realistic is it that UNIX will be THE operating system standard. The talk concerns itself with the lack of guidance in the UNIX community, the over-simplicity of the desired goals. The relative immaturity exhibited in such a new marketplace, and the current and anticipated practices and events that will ultimately determine the outcome of the current UNIX rage.

The talk is intended to dispel the implicit assumption that UNIX will automatically become the standard (or that it needs to be), criticize some current marketing claims, and remind the newly initiated as to some of UNIX's weaknesses. The focus is to discuss what needs to be done to insure that UNIX has a healthy future in the non-computer environment. The author has a great love for UNIX, but feels this love is best served by prophesying warnings against over-optimistic views of UNIX as the panacea of computing for the masses.

4:00 PM A Survey of UNIX Usage in Scientific and Business Applications

James R. Hanley and Jeffry A. Scott
Laboratory for Information and Science in Agriculture
Room 302 Aylesworth Hall
Colorado State University
Ft. Collins, CO 80523

Increasing attention is being focused on UNIX to determine its appropriateness as a general purpose operating system for scientific and business applications. Unfortunately, little of this information has made its way into the literature. As a result, a survey was conducted among UNIX users and followers to determine:

1. To what degree UNIX is currently being used for scientific and business applications within the United States and at what level of expertise.

2. How the UNIX user community views the strengths, weakness, and competitive position of UNIX as a general purpose operating system for scientific and business applications.

The survey's sample was drawn from the ranks of the /usr/group and USENIX memberships. This paper presents the results of the survey and the conclusions drawn.

4:20 PM The Coming UNIX Crash

Roger McKee
Vice President
The Wollongong Group, Inc.
1135A San Antonio Road
Palo Alto, CA 94303

The history of data processing abounds with examples of companies and their offerings that were valuable and useful. Some of these companies did not survive; their offerings did not become industry standards. The reasons for failure vary. This paper examines the reasons and applies them to the world of today's UNIX companies. We present several obstacles that have to be overcome before UNIX can become a standard in the data processing community.

4:40 PM The Commercialization of UNIX

Dr. Rebecca Thomas and Jean Yates
Yates Ventures, Inc.

UNIX is evolving rapidly from a research-oriented product with limited commercial appeal to a product with overwhelming impact on the data processing industry. In our presentation, we will show the implications of this at two levels:

- Technical direction of UNIX -- what kinds of features will be developed -- will they leave universities out in the cold?

- Market impact -- what will UNIX's place be in the office automation/microcomputer/minicomputer markets.

At the technical level, Dr. Thomas will look at the enhancements and improvements that the usenix community wants, and compare them to what commercial vendors are doing. She will look at the possible directions that Bell Labs will take in view of a more commercial orientation for UNIX inside Bell.

At the market level, Ms. Yates will present results of Gnostic Concepts' intensive market analysis of existing and new markets for UNIX. This data is part of an $18,000 study and has never been presented to the public before.

ABSTRACTS

Friday, July 9, 1982
Copley Ballroom

8:40 AM <u>Welcome</u>
David Stoffel, Users Group Co-ordinator

9:00 AM <u>Software Tools Bulletin Board</u>
CompuServe

CompuServe will present an overview of their nationally accessible computer system and their ability to provide a clearinghouse for exchange of software tools information. Subjects to be covered include their electronic mail, source code archives, tools on their machine, pricing structure, and tape distributions.

9:20 AM <u>The Software Tools On The Data General NOVA</u>
Jon Hanshew
CompuCode
6147 Aspinwall Road
Oakland, CA 94611

I have ported most of the Tools to a Data General Nova under RDOS. Specific problems I encountered will be presented as well as general techniques required to fit the tools into a small unmapped machine.

I gave a talk on my progress at the Winter, 1982, Software Tools Conference. This presentation will relate problems encountered since then; specifically in bringing up RATFOR and the SHELL. Additionally I will present the requirements imposed in order to maintain portability between the various Data General FORTRAN compilers and operating systems.

9:50 AM <u>Software Tools for TOPS-20</u>
Steve Hathaway
Tektronix, Inc.
Delivery Station 63-333
P.O.Box 500
Beaverton, OR 97077

The portable software tools package is now installed on the TOPS-20 operating system. Like most implementations, the hardest utility to install is the SH command processor. Our implementation at Tektronix, Inc. does not allow background processing, but it handles all the remaining functions of the portable software tools package. SH command files can be nested to arbitrary depth with default redirected I/O acquired from the parent process.

Additional features of the TOPS-20 implementation include a gateway utility (XCOM) that can execute TOPS-20 commands and system programs. Software tools are written in RATFOR and linked with a library of software tools primitives. TOPS-20 system programs are written in any arbitrary programming language and not directly compatible with the software tools interfacing conventions.

The software tools programs for TOPS-20 are written so as to be callable by the batch processor and the programmable command language, PCL. Because of system architecture differences, the calling conventions outside the shell for software tools had to be modified.

ABSTRACTS

10:20 AM Coffee Break

10:40 AM Portability in the Virtual Operating System

Bob Upshaw
Real Time System Group, Build. 46A
Lawrence Berkeley Lab.
1 Cyclotron Rd.
Berkeley, CA 94720

The software tools effort is unique in the computing world because it attempts to make practical use of software portability. "Portability" is one of those terms which is difficult to define, the reason being that portability means different things under different circumstances.

For example, a portable Virtual Operating System offers a mechanism by which users of that system can move from machine to machine with little or no effort. This is known as "people portability". Programs in the tools environment can also be portable, because they are written in a portable language (RATFOR or Pascal), and depend on primitives which exist in all software tools environments (e.g. open.) This is known as "program portability". Lastly, primitives in the tools environment are (should be) defined to have a portable user interface even though their implementation is not portable (for example, 'close' or 'remove'.) This is known as "functionally portable".

This paper will discuss portability in relation to the software tools. In particular, it will show how easy it is to violate the principles of portability and attempt to give some easy rules to follow to achieve one of our major goals - portable software.

11:00 AM Proposed Extensions to the Primitives and Library - Panel Discussion

11:40 AM Lunch

1:10 PM Spelling Checkers, Compound Words, and Variant Spellings

Eric S. Rosenthal
IMI Systems
1500 Broadway
New York, NY 10036

The ability to check the spelling of compound words and words with variant spellings can be added to existing spelling checks or provided by a separate program. Although these words cause the greatest spelling difficulties for authors and editors, they are not adequately treated by existing spelling checkers.

Both classes of words are difficult because authorities disagree on their preferred spellings. Compound words are also difficult because there are too many potential compounds to fit in a dictionary.

A rule for the spelling of a compound word or word with variant spellings can be expressed as a restriction on the combinations of words or sequences of consecutive words which may appear in a document. Such rules cannot prescribe the use of different forms of a word for different meanings or grammatical functions or distinguish an incorrect open compound from a correct use of its components as separate words. These distinctions would require much more sophisticated processing of natural language than performed by existing spelling checkers.

A spelling rule should be specified for each compound word or word with variant spellings in a document, either by the user or by the checker's original dictionary. There exist search procedures which can identify many of the words in a document requiring such rules, even if the checker's original dictionary, if any, does not indicate which of its words require them.

1:40 PM <u>QDP - A Quick Plotting Tool</u>

Kenneth R. Anderson
M.I.T.
Applied Seismology Group
Lincoln Laboratory
42 Carleton Street
Cambridge, MA 02142

QDP is a program that allows a user to plot data from data files easily. When combined with other software tools it forms a convenient data analysis environment. It is also flexible enough that it can produce complex plots of publishable quality. A good graphics tool should (1) be easy to use, (2) interface easily with other tools, and (3) be easily adapted to a users needs. Unfortunately, these goals conflict and one must compromise between them.

QDP was motivated by a desire to look at various forms of data quickly and easily. Since such data was typically the input or output of other programs, the goal was to plot the data with little editing or format conversion. Although flexibility was reduced in favor of ease of use, QDP is flexible enough to be used in many applications that would otherwise require specialized graphics software. Examples of how QDP can be used in data analysis will be given and a comparison between it and other plotting programs will be made.

2:00 PM <u>NBS Projects On Software Technology and Computer Based Office Systems</u>

Mike Chernick
National Bureau of Standards
Building 225, Room B-226
Washington, DC 20234

The purpose of the Software Technology Project is to develop software engineering guidelines and standards for the Federal Government in the area of advanced program development techniques. The development of guidelines and standards is re-enforced by actual hands-on experimentation with software tools, programmer workstations, and user environments. Products developed within the project include a taxonomy of software tool features, a software tools database, and a guide for introducing tools into medium and small programming environments.

Computer Based Office Systems (CBOS) represents a major application area for computers and networks. The CBOS program at NBS includes development of Federal Information Processing Standards (FIPS) for such applications as Computer Based Message Systems (CBMS). The first standard is a message format for CBMS and the second is a message transfer protocol. The CBOS program also has a laboratory for testing proposed FIPS protocols and for implementing developmental applications.

2:20 PM **Navy Software Development With RATFOR-T and Software Tools**

Neil P. Groundwater
Analytic Disciplines, Inc.
8320 Old Courthouse Rd. #300
Vienna, VA 22180

An overview of the software development tools recently developed by ADI under contract to NAVSEA 63D will be presented. These tools have been developed to reduce software development costs within the Acoustic Performance Prediction (APP) Program at the Naval Sea Systems Command.

RATFOR-T is a structured FORTRAN preprocessor based upon a version of RATFOR developed at Lawrence Berkeley Laboratories. The two preprocessors differ in that RATFOR-T discerns enough about the program structure to provide handles for determining test data coverage on a given set of program modules.

RATFOR-T has been designed in an effort to assist the user in verifying that each program predicate path has been exercised during the testing process. A program predicate path may be defined as a sequence of program statements which originate at a decision point or at a program entry point and terminates at the next sequential program or at the module return or exit point.

Both PRODOC and TREE are used to enhance the documentation process and to provide further insight into the program structure. PRODOC is an interactive program that knows the program structure and requests information from the programmer to supplement what it has determined automatically. TREE displays the program structure and thus the calling program relationships.

2:40 PM **Rocky Mountain Area Implementors Group**

Ben Domenico
NCAR

3:00 PM **Coffee Break**

3:20 PM **Future Directions - Discussion**

3:40 PM **Implementor and SIG group meetings**

Commercial Support of the Tools
Text Processing Tools
RATFOR
The SHELL
Primitives and Runtime Library

Winter UNICOM 1983

Pre-announcement

UNICOM is the conference and exhibition of the *UNIX** community sponsored jointly b*y* */usr/group*, the *USENIX Association*, and the *Software Tools Users Group*.

Place:	Town and Country Hotel San Diego, California
Dates:	Tuesday through Friday, January 25-28, 1983
Vendor Exhibition:	Tuesday through Thursday

/usr/group is a non-profit corporation formed by and for people who have a commercial interest in *UNIX* and *UNIX*-like operating systems and associated software tools. Key */usr/group* activities include sponsoring twice-yearly conferences where, in addition to meeting contacts, attendees have the opportunity to learn about software and hardware products being offered in the *UNIX*-related marketplace. Also provided are products catalogs and a bi-monthly newsletter.

The *USENIX Association* is an organization providing a forum for the exchange of technical information about the *UNIX* system and associated hardware and software. Such services are provided through semi-annual conferences, bi-monthly newsletters and software exchange.

The *Software Tools Users Group* is focused on a set of license-free, *UNIX*-like utilities and system calls, written in Ratfor and Pascal. When run in conjunction with almost any local operating system, the Tools package presents a virtual operating system interface consisting of a virtual machine (system calls or "primitives"), utility programs, and a command language, thus achieving inter-system uniformity over a variety of operating systems. Originating from Kernighan and Plauger's book *Software Tools*, the enhanced package now includes programs for text formatting, mail systems, enhanced Ratfor preprocessors, a source code control system, command line interpreter similar to the *UNIX* Shell, and many other utilities.

If you wish to receive the pre-registration packet (and did not get a copy of this announcement by mail) send your name and address to:

UNICOM
P O Box 385
Sunset Beach, CA 90742

Local Arrangements Coordinator is
Judith DesHarnais
ucbvax!sdcsvax!sdchema!jfd
(213) 592 - 3243

* *UNIX* is a trademark of Bell Laboratories.

AUUGN is produced by volunteers from the Australian Unix Users Group. Material on Unix is solicited from readers. Please send your contributions to the editors at the address given below.

The subscription fee for AUUGN is $24 for six issues over one year. Overseas subcription is $30 Australian dollars. Please do not send purchase orders, only money! The volunteers running AUUGN don't like paper warfare! Your subscription fee and contributions should be mailed to:

AUUGN
c/o Bob Kummerfeld
Basser Department of Computer Science
University of Sydney
AUSTRALIA 2006