AUUGN

Australian Unix User Group Newsletter

Volume 5
Number 6

Registered by Australia Post Publication No. NBG6524

The Australian UNIX* systems User Group Newsletter

Volume 5 Number 6

January 1985

CONTENTS

Editorial

Well, this is the last issue in volume 5 and I would like to take this opportunity to thank everyone involved in producing the newsletter. I will not mention names, they know who they are.

You may all look forward to a slightly different AUUGN starting in volume six, but rest assured that the quality will only get better.

Membership and Subscription Renewal

All Founding Members and volume 5 subscribers are reminded that, unless they have applied for Normal Membership of the AUUG or subscribed to volume 6, this will be the last issue of AUUGN they will receive. To find out whether you are one of these people, look at the address label on the envelope containing this issue. If the word "renew" appears in the top right hand corner of the label then fill in one of the forms at the end of this issue and send it, and your money, to the AUUG as soon as possible.

All correspondence should be addressed to

Greg Rose
Honorary Secretary, AUUG
8 Meadow St
Concord NSW 2137
Australia

Management Committee Report

The issue of receipts and documentary evidence of membership is progressing and these should be available by the time of the next meeting in Wollongong.

The Treasurers report will be presented at the general business meeting in Wollongong and should you have anything you wish to be discussed, please contact me so that it may be added to the adgenda. Members of the Management Committee are reminded that the committee will be meeting on the Sunday before the Wollongong conference.

A major undertaking for the group is the cooperation with Computerworld Expo in the organisation of UNIXWORLD 85. The group is organising the program and we have invited some very good speakers. UNIXWORLD 85 will be a yearly event and since it is aimed at managers, D.P. professionals, and people who want to hear about new equipment, we have an opportunity to increase the technical content of our regular meetings, and we can remove some of the emphasis on the equipment exhibition (which is a major headache to organisers). UNIXWORLD 85 offers a discount to AUUG members, and the speakers are worth hearing.

Greg Rose
AUUG Secretary.

Next AUUG Meeting

The next AUUG Meeting will be held in Wollongong in early February 1985. Further information appears later in this issue.

Contributions

Do you think I like writing all this myself? Come on you people out there in UNIX-land, send me your views, ideas, gripes, likes or whatevers.

Opinions expressed by authors and reviewers are not necessarily those of the Australian UNIX systems User Group, its Newsletter or the editorial committee.

Books

I have used information supplied by publishers and from other sources to compile the list of titles below. Should you know of any I have not listed or have listed incorrectly, or should you want to review a title, please contact me.

Books on UNIX

1. Using the UNIX System
   Richard Gauthier
   Reston Publishing Co.(1981)

2. A User Guide to the UNIX System
   Rebecca Thomas, PhD and Jean Yates
   OSBOURNE/McGraw-Hill (1982)

3. UNIX - The Book
   M.F. Banahan and A. Rutter
   Sigma Technical Press (Wiley)(1982)

4. A UNIX Primer
   Ann Nicols Lomuto and Nico Lomuto
   Prentice-Hall (1983)

5. UNIX Primer Plus
   Mitchell Waite, Donald Martin and Stephen Prata
   Howard W. Sams and Co. Inc. (1983)

6. The UNIX System
   Stephen R. Bourne
   Addison-Wesley (1983)

7. Concurrent Euclid, UNIX and TUNIS
   R.C. Holt, University of Toronto
   Addison-Wesley (1983)

8. The UNIX Operating System
   Kaare Christian
   John Wiley and Sons, Inc (1983)

9. The UNIX Programming Environment
   Brian W. Kernighan and Rob Pike
   Prentice-Hall (Software Series)(1984)

10. Starting with UNIX
    P.J. Brown
    Addison-Wesley (1983)

11. The UNIX Programmer's Manuals
    Bell Laboratories
    CBS Educational and Professional Publishing, NY

12. Introducing the UNIX System
    Henry McGilton and Rachel Morgan
    McGraw-Hill (1983)

13. The UNIX System Book
Peter P. Silvester
Springer-Verlag (1984) (Australian Distribution through D. A. Book (Aust) Pty Ltd)

14. Understanding UNIX, A Conceptual Guide
James R. Groff and Paul N. Weinberg
Que Corporation (1983)

15. UNIX System V: A Quick Reference Guide
William Wetzel
Prentice-Hall (1983)

16. A Practical Guide to the UNIX System
Mark G. Sobell
The Benjamin/Cummings Publishing Co. Inc. (1984) (Australian Distribution through Addison-Wesley Publishing Co.)

17. A Practical Guide to Xenix
Mark G. Sobell
The Benjamin/Cummings Publishing Co. Inc. (1984)

18. A Practical Guide to UNIX System V
Mark G. Sobell
The Benjamin/Cummings Publishing Co. Inc. (1985)

19. The Business Guide To The UNIX System
Jean L. Yates and Sandra L. Emerson
Addison Wesley

20. The UNIX Guide (2nd Edition)
Pacific Micro

21. The UNIX Operating System Book
M.F. Banahan and A. Rutter

22. Operating Systems Pocket Guide: UNIX
Laurie Blackburn and Marcus Taylor
Pitman Publishing

23. UNIX For People
P. Birns, P. P. Brown and John C. Muster
Prentice-Hall (1984)

24. Real World UNIX
Halamka
(due for release late 84)

25. A Business Guide to Xenix
Yates et. al.
(due for release late 84)

26. UNIX on the IBM PC
Twitty
(due for release late 84)

27. Exploring The UNIX Operating System
Stephen G. Kochan
(due for release late 84)

Books on C

1. The C Programming Language
Brian W. Kernighan and Dennis M. Ritchie
Prentice-Hall (Software Series)(1978)

2. C Notes - A Guide to the C Programming Language
C.T. Zahn
Yourdon Press (1979)

3. Learning to Program in C
Thomas Plum
Plum Hall (1983)
(Australian Distribution through Prentice-Hall)

4. The C Puzzle Book
Alan Feuer
Prentice-Hall (1982)

5. The C Primer
Les Hancock and Morris Krieger
McGraw-Hill (1982)

6. C Programming Guide
Jack Purdum
Que Corporation (1983)

7. Programmer's Guide to C
Lees
Prentice-Hall

8. Programming in C:
For The Microcomputer User
R. P. Traister
Prentice-Hall

9. Programming in C
Stephen G. Kochan
Hayden Book Company
(Australian Distribution through Holt-Saunders Pty Ltd)

10. The C Programming Tutor
Leon A. Wortman and Thomas O. Sidebottom
Prentice-Hall (1984)

11. C Programming Guidelines
Thomas Plum
Plum Hall (1984)

12. A Book On C
Al Kelley and Ira Pohl
Benjamin/Cummings

13. C Programmer's Library
    Jack J. Purdum et. al.
    Que Corporation

14. Understanding C
    Hunter

15. Introduction to C
    Chirlian

16. C Programming Standards And Guidelines
    Thomas Plum

17. The C Programming Reference Manual
    Samuel P. Harbison and Guy L. Steele
    Prentice-Hall (1984)

18. C Programmer's Handbook
    Hogan
    (due for release late 84)

19. Programming In C On The IBM PC
    Pollack
    (due for release late 84)

20. C Language User's Handbook
    Weber Systems
    (due for release late 84)

Related Books

1. Software Tools
   Brian W. Kernighan and P.J. Plauger
   Addison-Wesley (1976)

2. Software Tools in Pascal
   Brian W. Kernighan and P.J. Plauger
   Addison-Wesley (1981)

3. Text Processing with UNIX
   nroff, troff and eqn
   D.W. Barron and M.J. Rees
   Addison-Wesley (1983)

4. The Bell System Technical Journal Vol57 NO6 Part2
   Edition on "UNIX Time-Sharing System" July-August 1978
   American Telephone and Telegraph Company

5. Pascal Under UNIX
   J.N.P. Hume and R.C. Holt
   Reston Publishing Company

6. Principles of Compiler Design
   Alfred V. Aho and Jeffrey D. Ullman
   Addison-Wesley Publishing Company (Talks about Yacc and Lex)

7. The Small C Handbook
   James Hendrix
   Reston Publishing Co, Inc (1984)
   (Australian Distribution through Prentice-Hall)

8. Comparing And Assessing Programming Languages:
   Ada, C and Pascal
   Alan R. Feuer and Narain Gehani
   Prentice-Hall (1984)

9. UNIX Applications Software Directory (2nd edition)
   Edited by Ray A. Jones
   Onager Publishing

10. The UNIX System Encyclopedia
    (This a vendor directory - Ed)
    Yates Ventures

11. Operating System Design:
    The Xinu Approach
    Douglas Comer
    Prentice-Hall (1984)

12. Using The Horizon(TM) Spreadsheet With
    The UNIX Operating System
    Don Beil
    Prentice-Hall (1984)

13. Word Processing On The UNIX System
    Kreiger
    (due for release late 84)

A General Purpose Multi Processor Kernel Written in C for a Unix Programming Environment

T.R. Cordingley and D.W.E. Blatt
Department of Mathematics, Statistics and Computer Science
University of Newcastle

ABSTRACT - This paper describes an approach being taken to build a programming environment for a multiprocessor system based on available microprocessor products and existing high level software. To the extent possible we are using available software and off the shelf board level microprocessor products. The development environment is a normal Unix system and therefore existing compilers, editors and linkers can be used and the applications will maintain Unix-like interfaces for the multiprocessor system.

INTRODUCTION

The concept of multiprocessor systems is quite old. The ability to achieve a linear performance improvement would make microprocessor based systems an attractive alternative in applications that traditionally have required large main frame computers. In the past the emphasis has been on large systems that have aimed at supporting large user populations. This requires sophisticated hardware and operating system support. An example is the Cm* multiprocessor project at the Department of Computer Science Carnegie-Mellon University where 50 DEC LSI 11 processors were connected and a sophisticated multiuser operating system was developed[1].

This is now simplified because microprocessors such as the Motorola 68000 have a 16M direct address space which will allow very large applications to be run with minimal operating system support. Essentially any microprocessor system that supports multimaster operations can be used for a multiprocessor system.

The approach taken in this project is based on the premise that most cost/performance advantage is to be gained from multi-microprocessors as opposed to large mainframe systems in applications which are single user compute bound. In such a situation, the high i/o bandwidth and terminal handling capability built into typical large systems is not cost effective, and this project aims to demonstrate that for jobs of this kind, a multiprocessor approach can compete to advantage.

The principal problem in such an approach would appear to be the software. It is a fairly straightforward matter, given the available board level products, to construct a multiprocessor system. Unless real user applications can be ported to this environment, the potential of the hardware cannot be realised. In this project, we are aiming to construct a kernel which will allow existing high level language compilers on a Unix system to be used directly on the multiprocessor hardware. The conversion of user applications will be used to develop guidelines for the use of tools for that purpose, the aim being to make the multiprocessor system accessible to a range of real user applications. Thus the approach has been towards a single user single application environment. This eliminates many normal operating system design considerations such as protection, resource allocation and memory management strategies.

The application areas that are being looked at are typically processor intensive requiring relatively small amounts of input and output, large amounts of memory and suited to parallel execution. Some typical examples are theoretical chemistry problems, simulations, combinatorial mathematical search problems and differential equations. These types of applications are usually done on larger computers. They are generally not limited by I/O bandwidth but by processor speed. The objective is to provide a programming environment in which a user with a large application of this type could use the tools and guidelines developed to take that program and break it down into parallel tasks that can be run on the multiprocessor hardware.

HARDWARE CONFIGURATION

The hardware configuration of the system is of two or more processors and one or more disks on a common bus. All these bus masters have access to a shared memory on the bus and each of the processors have access to private memory not accessed through the bus. This is an important consideration because it reduces bus traffic which is a limiting factor in the performance of this type of system.

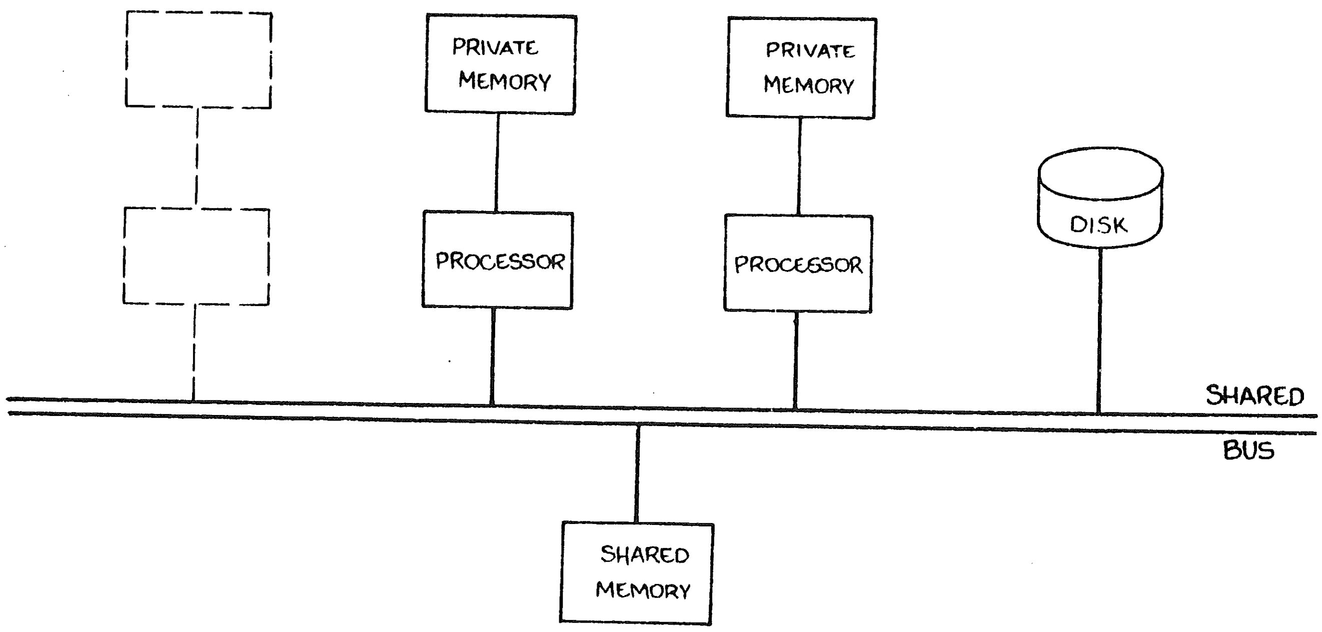


Fig 1. Typical Hardware Configuration

PROGRAMMING

Methods used in concurrent programming are well documented[2],[3]. There are a number of languages that specifically use these concepts such as ADA[4] and Modula[5]. Unfortunately these languages are rarely implemented for multiprocessor systems. The approach being taken is to build a set of primitives to enable construction of concurrent programs in the multiprocessor environment. This should allow a large set of existing algorithms to be used to construct a programming environment for the multiprocessor hardware.

Depending on the availability of memory most, of the application code is duplicated in the private memory of the processors. All shared process variables and stacks must reside in shared memory if more than one processor is allowed to execute them. Process synchronisation and message passing must also be through shared memory.

Each processor has its own scheduler task that searches a work pool after device interrupts are serviced or when the running task finishes with the processor.

The applications consist of a number of tasks that communicate with each other and a set of tasks that make up the kernel. Although the system is not primarily targeted at i/o intensive applications, a limited input/output subsystem is provided through the kernel tasks, a selection of which can be linked at compile time from object libraries. These kernel tasks include interrupt handlers, device drivers, file system managers and buffering tasks for serial devices. Once linked, a job becomes a stand-alone system, which can then be booted onto one or more processors using the normal Unix boot sequence.

The user interfaces maintain a high degree of compatibility with Unix so that existing software can be easily ported. The major difference between this and Unix is that file system and device interfaces are not through kernel traps but through inter-task communication primitives send and receive. There are also conditional send and receive primitives that are non-blocking. These essentially work the same way as read and write do in Unix but instead of a file descriptor a channel block address is used. The channel block contains the mechanisms necessary for task synchronisation and message buffer address passing. From these it is not difficult to implement read, write, pipes and even a Unix compatible file system as simple subroutines and tasks.

Fork and execute system calls are replaced with the subroutine task( ). This makes an entry for the schedulers from any C language subroutine, it allocates stack space and a task control parameter block in shared memory and passes additional parameters to the C routine as start parameters. Because C is reentrant 'task( )' can be called any number of times on the same subroutine. Obviously care is required when modifying static variables but the integrity of local variables can be assured.

```
task(C-subroutine,workspace,start params);
```

Fig 2. Task initialisation subroutine

The send and receive primitives can be used to simulate process synchronisation primitives that exist in languages such as ADA and Modula which cater for multiprocess activity.

* Blocking call - accept sequence send and receive may have to be used together to simulate in and out in a rendezvous call.

```
        X(...)          accept X(...)

 ----->

        send( );        receive( );
        receive( );     send( );
```

* Select loop - in ADA X and Y are task entries, here they are channel descriptors

```
        loop
          select
            accept X do ...
          or
            accept Y do ...
          end select
        end loop

 ----->

        loop:
                if (creceive(X,...) >= 0) { ... }
                else
                if (creceive(Y,...) >= 0) { ... }
        goto loop;
```

* Timeout accept call - this shows a busy waiting loop. This is not completely undesirable because the conditional send and receive primitives reschedule the processor before returning if no transfer can occur immediately.

```
        select
          accept X do ...
        or
          delay 1*MINUTE
          ...
        end select

 ----->

                t = time( );
        loop:
                if(creceive(X,...) >= 0) { ... }
                else
                if(time( )  -  t  >=  1*MINUTE)  {  ...  }
                else goto loop;
```

Fig 3. Construction of ADA primitives

TASKING

Tasking centres around a structure containing flags, a register save area and processor mask for giving processors execution access to tasks and a busy flag, which is a semaphore that is set when the task is being executed or another task has access to its data as in send or receive. A pointer to this structure is returned when the subroutine task is called. There are two subroutines that work on these structures 'saveenv(tsk)' and 'restorenv(tsk)'.

```
struct tcb {
        int busy;
        int flags;
        int regsav[NREGS];
        int procmask;
}
```

Fig 4. Task control structure

The subroutine 'saveenv' saves current working registers and returns immediately with '1' and the processor at high priority so interrupts can not occur. When the registers are restored the subroutine returns '0' at normal priority. This is similar to 'fork()' in normal Unix except no process duplication is made, the currently executing task has to decide in high priority state which task it is to restart, it can choose the scheduler task. The routine restorenv loads the registers that saveenv saved, it therefore does not return unless an error occurs. This functionally places operating system tasks into the user applications. These routines are not generally for user applications, they are primarily for the system interface library routines such as 'send' and 'receive'.

```
reschedule( ) {

        extern struct tcb *ctsk;

        if (saveenv(ctsk)) {
                restorenv(SCHEDULER);
                panic("No scheduler");
        }
}
```

Fig 5. Reschedule primitive

A typical usage of 'send' and 'receive' might be as illustrated in Fig 5. The variable 'ctsk' is a pointer to the task control parameter block of the currently executing task on this processor. The effect of this routine is to temporarily reschedule the processor that executes it.

LOW LEVEL SYNCHRONISATION CONSTRUCTS

A multiprocessor environment requires low level synchronisation constructs to exclude access from various resources. Most single processor operating systems use high processor priority for critical sections of code. This is not sufficient in multiprocessor systems. An indivisible instruction to test and set a flag in shared memory is required to inform other processors that a resource is currently being used. This means that the processors all have to conform to some kind of protocol. In the case of Multibus interfacing, a read-modify-write cycle can be done without losing control of the Multibus, making this primitive achievable. A subroutine 'test_and_set(&flag)' is implemented in assembler language so that this kind of protocol can be used. It returns '1' if the flag was reset and '0' if the flag was set, the flag is always set when the routine returns.

The test and set routine, saveenv, restorenv, low level interrupt primitives and run time initialisation are the only assembly language routines in the system. They also cover most of the machine dependent features of the processors. The rest of the interface can then be written in C and hence the environment could be easily ported to other machines.

INTERRUPT HANDLING

The interrupt handlers produce a single send every time an interrupt occurs. Interrupt handlers then become simple tasks that wait using a receive to the interrupt channel. A simple example is illustrated in fig 6. The ttyinput routine receives characters and sends then to the user application or terminal buffering task. The open routine simulates a normal unix open it allocates a channel 'buff' and starts a task 'ttyinput'. From here characters can be read using receive on buff which is functionally the same as read would be when used in a normal Unix programming environment.

CURRENT STATUS AND FUTURE DEVELOPMENT

The hardware being used is an Intel Multibus system consisting of a floppy and winchester disk subsystem and multiple Motorola 68000 processor boards. This selection of hardware was made with a view to the large available range of board level products for Multibus, as well as the C based software already available on the 68000. This has made it possible to commence the project with a good deal of the preliminary work done, and concentrate on the multiprocessor aspects of the development to a greater degree.

```
interrupt :       /* Machine Interface */
                  send(intx,0,0);
                  rti

ttyinput(ttych,intx)
struct channel *ttych,*intx;
{
        loop:
                  /* set up receiver for next interrupt */
                  ....
                  ....
                  receive(intx,0,0);
                  /* get character */
                  ....
                  ....
                  /* send character to buffer task */
                  send(ttych,&ch,1);
                  goto loop;
}

open( ) {
        struct channel *ch;

        ch = alloc(sizeof(struct channel));
        task(ttyinput,ch,intx);
        return(&ch);
}
```

Fig 6. Construction of simple I/O routine

The 68000 boards have 256k of local memory and there is 128k of shared memory on the bus. The multiprocessor system hardware is assembled and tested and drivers have been completed for the winchester disk and floppy.

The main work at the moment is a Unix port onto the system. The standalone support is running and a file system has been constructed. When the port is complete the multiprocessor system development can proceed on the system itself using standard Unix on a single processor and booting applications using the normal Unix boot sequence on to one or more processors. At this stage development is still based on another Unix host, but standalone C based applications can be run on a single processor of the multi processor system, and some limited testing of the full multiprocessor environment has commenced.

The multiprocessor application interface environment and development tools will be evolved from porting actual user applications to the system.

CONCLUSION

A Unix like programming environment can be constructed on multiprocessor hardware to support high level language applications without significant operating system or hardware development overheads. This should provide a portable and cost effective environment for evaluating various multiprocessor software strategies and also provide a useful environment for processor intensive applications.

REFERENCES

1. Department of Computer Science, Carnegie-Mellon University. "The Cm* Multiprocessor Project: A Research Review", July 1980.

2. Andrews, G.R. and Schneider F.B. "Concepts and Notations for Concurrent Programming." ACM Computing Surveys 15,1 March 1983.

3. Berztiss, A.T. "Synchronization of Processes." Preprint No 82-11, Department of Computer Science, University of Wollongong, Australia. May 1982

4. U.S. Department of Defence. "Programming Language ADA: Reference Manual." vol. 106, Lecture Notes in Computer Science. Springer-Verlag, New York 1981.

5. Wirth, N. "Programming in Modula-2." Springer-Verlag New York 1982.

A Mail Path Please
or
Something I get Asked A Lot

Robert Elz
Department of Computer Science
University of Melbourne

This article originally appeared as a mail message from Robert Elz to Glenn Huxtable. Robert was kind enough to mail me a copy suggesting that it might be included in the AUUGN. So be it -Ed.

There's no easy way to learn about mail addressing, it just comes from experience. Fortunately, things are slowly getting better. Eventually, all you will have to know is the (full) address of where you want to contact (and that is something that ought to be fixed, and available in a directory or something).

For now, you just take each segment (network) that the mail will pass through, starting with the last it will arrive at, and construct the address that would be used to get through that network, treating any later networks and addresses as if they were just the user name for the network you are now considering.

Of course, this doesn't work all the time, as sometimes the "user name" (which is really an address on a later network) is illegal on the network that you are now going through. Then you have to understand the transformations that are applied at the gateway nodes, and make use of one of those so you can have a legal address on the routing network, which will be transformed into a satisfactory address when the mail enters the destination.

Sometimes its foul, but fortunately, most of the time its comparatively easy. UUCP addresses are really very simple, except that you have to consider each node as a network gateway (each uucp hop is really a new network - uucp was originally designed to be used on one hop links only - if you want to talk to hirst1 you simply dial them, transfer your information, and hang up. Simple, and easy ...).

Uucp address syntax is then

```
host!user
```

So, to get to fred on hirst1 from ukc, address is

```
hirst1!fred
```

Then, from vax135, to get to ukc ...

```
ukc!user
```

and since the "user" is "hirst1!fred" it becomes

```
ukc!hirst1!fred
```

Similarly from mulga, leading to

```
vax135!ukc!hirst1!fred
```

Now that's a "user name" at mulga. You already know how to get mail to users at mulga, the syntax is

```
user:host
```

So, what you end up with is

```
vax135!ukc!hirst1!fred:mulga
```

Using uucp is really simple, provided that you know the path. Unfortunately, I think that you don't know the path - this is where uucp gets hairy. A while ago, vax135 stopped calling ukc (or I think they did, maybe they started again, I'm not sure), so at vax135 "ukc!..." is no longer legal.

That means that you have to dream up another path. In this case:

```
decvax!mcvax!ukc!hirst1!fred:mulga
```

should work (assuming that there is a hirst1 that is connected to ukc, that part I don't know about).

You should be aware that elecvax has a service to help you choose uucp paths; it was described in a past issue of AUUGN.

Also, sometime soon, I hope that mulga will be able to pick a path for you if you ask it to. ("Asking" will be done by simply using "host!user" as the address, where "host" is not a known ACSnet host (see below for why). Mulga will transform that into "a!b!c!host!user" for you (eventually ...). Of course, if "host" is decvax or vax135 then there won't be a lot for mulga to do. And also, "user" will still be able to contain anything that you like. That means that choosing your own route will definitely still work. Sometime, you may also be able to use "user@host.uucp" on ACSnet and forget mulga completely. This will be announced in the aus.netstatus news group when it is available)

Now for return addresses, you just do the same transformation. On ACSnet, your address is

```
glenn:wacsvax
```

so, you need to get mail to an ACSnet host, with that as the "user" part. Mulga is probably the ACSnet host to choose (probably == certainly!). You probably want to reverse the above path (not necessarily, there are alternatives) so, you would want to be at decvax before mulga, to get to a user at mulga from decvax, the syntax is

```
mulga!user
```

so your address is mulga!glenn:wacsvax.

Continuing this backwards till origin host is hirst1 you end up with

```
ukc!mcvax!decvax!mulga!glenn:wacsvax.
```

Now, here is where it starts to get interesting. Some hosts can't handle ':' as a legal part of a user name. The ARPA RFC822 standard uses ':' for other purposes (various forms of grouping), so if any of the hosts along this route dislike the ':', then mail will not get back. So, you need to (or might need to) find an alternative syntax to use at mulga.

This is where you need to understand gateway transformations: mulga is one such gateway.

It happens, that mulga will accept any of

```
user@host
user@host.oz
user:host
user:host.oz
user.host
user.host.oz
host.oz!user
```

and sometimes

```
host!user
```

and sometimes even

```
otherhost!andanother!host!user
```

to mean the same thing. The "sometimes" in the last two, is because obviously "decvax!user" is not meant to be taken as "user:decvax" so only hosts that are known at mulga as acsnet hosts (in a table that is distinct from the acsnet statefile) are recognised for the last two. In the last form, all of the hosts in the path must be known. (Reason for last one is that news leaves here with a path like

```
mulga!munnari!basser!wacsvax!glenn
```

if you sent it. While the path isn't meant to be used for return mail, people often use it that way).

So, there are lots of possibilities. (In the above list of alternatives, except for the last two, "host" can have internal ACSnet domains combined in, as "basser40.su").

It happens, that the "from" address that mulga generates (these days) is

```
mulga!host.oz!user
```

so, when mail arrives at hirst1 from you, it will look as if it was sent by

```
ukc!mcvax!decvax!mulga!wacsvax.oz!glenn
```

That format is supposed to be legal anywhere.

The "@" forms are quite unlikely to work, as lots of places parse '@' as the highest precedence address separator, so

```
xxxxx@yyyyy
```

always means user "xxxxx" at host "yyyyy", whatever characters happen to be in the x & y strings. So at mcvax (for example) an address like

```
        decvax!mulga!glenn@wacsvax
```

might be

```
        (decvax!mulga!glenn) @ wacsvax
```

and they would say "we've never heard of wacsvax - send this back".

Things get even more foul if you try inserting more than one '@' in the address. Strictly, that's illegal, that is, there is only allowed to be one unquoted '@' in an address.

Here we get into the ugly world of address quoting. According to RFC822, its legal to put (almost) any char in the local part of an address, provided that its quoted. There are various ways of doing the quoting, but one would be (at mcvax)

```
        "mulga!glenn@wacsvax"@decvax.uucp
```

(assuming that mcvax treats "decvax.uucp" as a legal host.domain pair. Strictly its not legal by RFC822, the only legal domain at the minute is "ARPA").

You would like then to generalize this to

```
        """glenn@wacsvax.oz""@mulga.uucp"@decvax.uucp
```

unfortunately, the chances of it working are very slight indeed. Almost no UNIX hosts, and probably not many arpanet hosts, handle quoted addresses correctly.

Even though you didn't ask about this, I'm going to continue here and describe addressing to a couple of other networks that people sometimes want to contact.

First ARPANET. This one is easy, to address it, you simply use

```
        user@host.arpa:mulga
```

Mulga sends the mail to an arpanet host via uucp. Which host that might be can change, at the minute its Berkeley. That host sees the address "user@host.arpa" and forwards your mail directly.

Your return address would be one of

```
        decvax!mulga!wacsvax.oz!glenn@Berkeley.arpa
```

or

```
        decvax!mulga!glenn.wacsvax@Berkeley.arpa
```

The former is the address that will appear as the sender when your mail arrives at the remote end. (Though "Berkeley" might be "UCB-VAX" or something, it doesn't matter, they are aliased).

Next, CSNET. This isn't too hard, given that csnet is reached from arpanet, as

```
        user%csnet-host.csnet@csnet-relay.arpa
```

so, from here, you just append ":mulga" to that, and you have a legal address. Your return address can probably be the same as from arpanet as most csnet hosts can handle mail to "user@host.arpa" without needing additional information. Alternatively

```
        user%Berkeley.arpa@csnet.relay.csnet
```

The DEC Engineering net (worldwide internal dec decnet) can be reached as

```
        user@host.dec:mulga
```

And, a return address is (here I'm guessing a bit - decnet isn't very familiar to me)

```
        RHEA::DECWRL::user@host.oz
```

Anyway, the return address on the mail that arrives there should work.

Last, BITNET. Berkeley acts as a gateway here as well, so you should be able to get mail to bitnet as

```
        decvax!ucbvax!user=host:mulga
or
        user%host.bitnet@Berkeley.arpa:mulga
```

And one or two other combinations. I have no idea what the return address would be.

There are lots more networks, there are gateways to some of them, but by no means all of them.

I hope that this helps a bit, and perhaps if peteri decides that its worth publishing in AUUGN, it might help others too.

But beware, as I said right near the top, all this is changing, and in a few months the syntax might be quite different.

Robert

EUUG
European UNIX† Systems User Group

Newsletter Vol 4 No 2

(Late) Summer 1984

† UNIX is a Trademark of Bell Laboratories.

;login:

The USENIX Association Newsletter

Volume 9 Number 4 **September 1984**

CONTENTS

* Reprinted here with permission.

The deadline for submissions for the November issue of *;login:* is October 26

The UNIX Magician's Handbook

Elizabeth Bimmler°
Murray Hill, NJ 07974

ABSTRACT

A review of "The UNIX Programming Environment," by Brian W. Kernighan and Robert Pike, Prentice-Hall Software Series (1984), ISBN 0-13-937699-2.

The UNIX Wizard

UNIX† is traditionally taught by 'wizards.' Every installation, and there seem to be well over 3000 now, inevitably comes with its own set of gurus where UNIX freshmen can learn the art of UNIX programming.

Until recently, the prospective UNIX programmer had to find a guru and seek to become his apprentice. After a long training the student could then become a master and as a token of his excellence would be issued the superuser password. UNIX, to be sure, is not a trivial system, and as Kernighan and Pike note in the preface to their book: "as the UNIX system has spread, the fraction of its users who are skilled in its application has decreased."

The UNIX operating system has been acclaimed for its conciseness and structure, its portability, and perhaps more important still: for its availability. Commenting on the 6th Edition UNIX, John Lions made the following, often quoted, observation: "the whole documentation is not unreasonably transportable in a student's briefcase." As Kernighan and Pike have aptly countered in their book: "this has been fixed in recent versions." Considering that the kernel of the first edition UNIX was only 8K in size, on the average, the size of the system has more than doubled with every new edition issued. In comparison to other commercially available operating systems, the documentation that comes with the newer versions is still modest in size, but it certainly will not fit a student's briefcase anymore, unless, of course, it's left on the magnetic tape.

UNIX was necessarily taught by wizards, simply because those wizards never wrote down their art. Now that the UNIX system has become so popular the number of publications on UNIX is steadily increasing, and a small set of textbooks has finally appeared. In some cases these textbooks convey little more than the information already available from the system documentation, though perhaps presented in a more friendly way. Oddly enough, the people responsible for the development of the UNIX system have long hesitated in providing us with good quality textbooks on their system. "Well," said Rob Pike, "if somebody is going to do it anyway, why let outsiders do a bad job when insiders can do a bad job ...."

Inside Look at UNIX

Two textbooks on the UNIX system written by the 'insiders' have now appeared, and they haven't done such a bad job at all. A first book was written by Steve Bourne,‡ the author of the 7th edition UNIX 'shell' (the command interpreter). Bourne's book gives an excellent overview of the system, with one of the best introductions to the C language and UNIX system programming I have seen so far.

The second book, and the one we will consider more closely here, was written by Brian W. Kernighan and Rob Pike.

° Disclaimer: Ms. Bimmler is a colleague of the authors of the book reviewed. However, the ideas expressed herein may not be credited to Messrs. Kernighan or Pike.

‡ "The UNIX System" by S. R. Bourne, Addison Wesley, Computer Science Series, (1983), ISBN 0-201-13791-7.

Brian Kernighan is a long-time member of Bell Labs Computer Science Research Center where UNIX was born, nearly fifteen years ago. With others he is jointly responsible for many of UNIX' utilities. He was there when Ken Thompson and Dennis Ritchie experimented with the first versions, and it was Brian who, paraphrasing the name 'Multics,' suggested the term 'UNIX' as the final name for the system they developed.

Rob Pike is a true UNIX 'wizard' who joined the Bell Labs group more recently. In the three years he has been there, he has already left an impressive trail of accomplishments, most notably the development (together with Bart Locanthi) of the popular 'Blit†' multi-window terminal, which is perhaps best described as a novel type of programmable workstation for UNIX systems.

Kernighan and Pike's book is quite different in approach from the earlier book by Steve Bourne, though there definitely is some overlap in the material presented. Both books cover 'shell programming,' and 'document preparation' in some detail. Still, the book by Kernighan and Pike is clearly not aimed at the casual user of the UNIX system. On the contrary, they aim for the reader who has mastered the basics and who wants to expand his or her knowledge.

This approach is perhaps best illustrated by the terse reminder at the start of their chapter on 'program development' (chapter eight, p. 234): "We are assuming that you understand C." For sure, not a very gentle introduction to the C language, but then again, the subject was covered at length in the earlier "C reference manual" by Kernighan and Ritchie. For those of us who do already understand C, though, the fifty pages that follow give an outstanding exposition of the development of a substantial and extremely useful program: an interpreter for a Basic-like language, written with the aid of the UNIX compiler writing tools *lex* and *yacc*°.

Tools

The book by Kernighan and Pike is crowded with many small examples of useful tools. Shortly after I had circulated a draft of the book at the University where I worked my system was literally swamped with all the little 24-hour watchdog programs that students and co-workers had discovered in the text. As an example, here is the shell script for a small program 'watchfor':

```
# watchfor:  watch for someone to log in

PATH=/bin:/usr/bin

case $# in
0)echo 'Usage:  watchfor person' 1>&2
exit 1
esac

until who | egrep "$1"
do
sleep 60
done
```

In this tiny shell script the use of while loops, comments, search paths, case switches, duplication of file descriptors, exit status and shell arguments is illustrated. Let us briefly go through it.

A sharp sign in column one indicates the start of a comment: the shell will simply ignore everything that follows up to the end of the line. The search path is stored in the predefined shell variable 'PATH', which stores a list of directory names, separated by colons. The listed directories are the only ones checked by the shell in an effort to locate a command given by the user. Traditionally UNIX commands are stored in directories */bin* and */usr/bin*, so it suffices here to set the search path for this script to these two directories.

† The terminal will be marketed by the Teletype Corp. as the model 5620 'dot-mapped display'.

° *lex* is a general tool for writing a lexical analyzer. *yacc* can be used to write parser generators.

Next we find a case switch on the number of arguments passed to the script. This number is squirreled away by the shell in a variable named '$#'. If the number of arguments is zero the script will print an error message and exit, returning an error status '1'. With the magic '1>&2' we combine the standard output channel (by convention file descriptor 1), where the user will expect the output from the script to appear, with the standard error output channel (file descriptor 2). The script as presented does not provide for an appropriate reaction if the user erroneously types more than a single argument, though the change is in fact trivial:

```
case $# in
1)until who | egrep "$1"
do
sleep 60
done ;;
*)echo 'Usage:  watchfor person' 1>&2
exit 1
esac
```

This time the loop is only executed if the user gives exactly one argument, hopefully the login name of someone who is likely to login to the system. The argument itself is available in the variable '$1'. The condition of the loop consists of two commands: *who* and *egrep*, which are connected via a 'pipe'. The pipe will direct the output from the first command to the input of the second. *who* will list all active users, *egrep* will select lines that match the pattern it is given in '$1'. The condition of the until statement is set by the exit status of the pipeline. Again 'by convention' this is the exit status of the last program in the pipe: in this case *egrep*. *egrep* will return 'true' (oddly enough this is the numerical value 0) if it found a match and 'false' (some value other than 0) if it didn't, which is exactly the effect that we wanted here.

Other Examples

None of the examples in Kernighan and Pike's book are artificial: they all turn out to be instructive, practical and, unfortunately, highly addictive. Here's a short list of examples they discuss:

- simple aliasing functions, such as a little shell script named 'cx' as an abbreviation of 'chmod +x' to turn files into executable commands, or 'lc' as an abbreviation of 'wc −l' to count the number of lines in a file;
- a small computerized telephone directory server 'tel', that uses *grep* to locate names and numbers in a list;
- a little C program 'vis' that makes non-printing characters in files visible;
- the following very original script that can be stored in a file named '2', and linked to files named '3', '4', '5', ... etc.:

  ```
  pr −$0 −t −l1 $*
  ```

  The first argument to *pr* becomes the number of columns in which it will print its output. '−t' turns off the usual page headers, and '−l1' sets the page length to 1. '$0' is the name of the script itself. (If you have access to a UNIX system, try: 'who | 2' or 'ls | 5' and see just how useful this is.)
- straightforward implementations of simple database managing tools, 'put' and 'get', that can be used to keep track of the revisions that, for instance, popular programs incur;
- an especially elegant program named 'bundle' for packing and unpacking sets of text files (e.g. programs) for distribution via ordinary system mail.

All tools, and there are many more in the book itself, are examples of small, well structured scripts or programs that solve practical problems. In nearly all cases the book includes sample runs of the programs showing typical system reponses.

UNIX Principles

The strength of the UNIX system, which characterizes the true "UNIX programming environment" can be summarized in a few points (see e.g. Kernighan and Pike, p. 131, or Bourne, p. 4-5):

- Files have no predefined format. The interpretation of a file's contents depends entirely on the program that reads it.
- Programs are tools. Each tool should perform one clearly defined function, and it should be optimized for that single function.
- The output from any program should be understandable as input to other programs. One should avoid fancy headers, trailers or spurious blank lines.
- Conversely, one should be able to replace the input of any program with the output of another. This rules out interrogative programs that prompt the user for arguments. All the information a program needs to run is given on the command line. The program either runs or fails, and returns the appropriate exit status.
- If no arguments are given a program should read the standard input and write the standard output (keyboard and screen by default), so that the program can always be used as a filter. Optional arguments precede filename arguments. Filename arguments may specify additional *inputs,* but never an output unless properly protected (e.g. with a '−o' flag).

In an early version of UNIX one of the disk maintenance commands (a predecessor of *fsck*, named *check*) violated the last principle. When called without an argument this program would faithfully read and check the default filesystem (there was only one) and list the errors on its standard output. If an argument was given, the corresponding file would be used to store the error list. The next version of the command, made for systems with more than just a single disk, expected the name of the filesystem to be checked to be specified in its first argument (an input instead of an output). One day an unfortunate user went back to a precious copy of the old system, typed:

```
check /dev/usr
```

and destroyed the user file system completely. Note that the all-important 'superblock' is one of the first blocks on disk and therefore also one of the first destroyed.

The File System

One of the good things about the book that Kernighan and Pike have written is that it gives access to information that one could obtain only from the gurus before. In very few other books you can find a succinct explanation of the implementation of the *file system* revealing the secrets of inodes and superblocks. There are chapters that explain how to use and how to *program the shell,* revealing the mysteries of meta-characters, quoting, shell variables and traps. Other chapters focus on the use of *system calls* in user programs, building up to the knowledge and insight required to make system programs that dig somewhat deeper into the bowels of, for instance, the file system.

Again, every topic discussed is illustrated with numerous small example programs, such as:

- a program 'waitfile' that waits until a file stops changing for a given period of time (say 60 sec.);
- a program that can correct typing mistakes in filenames by comparing the existing filenames to the user-specified filenames;
- an extremely useful program called 'readslow' that waits for the writer of a file to produce more output when the end of a file has been reached;
- a program called 'timeout' that sets an upper limit to the run time of an arbitrary other user program.

The book includes excellent tutorials on the use of esoteric programs such as *sed*, *awk*, *lex* and *yacc*. However, it is not meant to be a reference manual. "We feel it is more important to teach an approach and a style of use than just details," the authors write. The problem is of course that you can only illustrate an 'approach' to programming by either giving examples or by discussing the underlying principles. Some readers may feel that there is a little too much of the first and too little of the last in "The UNIX Programming Environment." I, for one, regret that not more material was included on the design and study of the algorithms that have been used to built the UNIX programming environment. Pattern matching tools such as *grep*, *egrep*, and *fgrep*, for instance, are not very well understood. There is a lot of folklore around these programs, which has spurred many an interesting lunch discussion (which program is fastest for which problem and why). It would certainly have been revealing to see some of this folklore substantiated or rejected by a thorough discussion of the algorithms used. But then, we have at least something to look forward to in a next book on the UNIX system. "The UNIX Programming Environment" is not a book for utter beginners. For those, however, who not only want to be able to *use* UNIX, but who want to be able to use it *well*, their book is an invaluable guide.

The 1984 Summer USENIX Convention Report

Rob Pike
AT&T Bell Laboratories

The 1984 Summer USENIX convention was held June 13-17 in Salt Lake City, Utah, home of the Mormon religion and anchovies. The convention base was the Hotel Utah, owned by the Church but not decorated like it. The center of the hotel is an ornate two-story lobby, with a great chandelier, gold trim, whorled carpets and illuminations, and a grand piano played in the afternoons while tea is served by two hotel maids. As many times before, USENIX was accommodated by a hotel of incongruous style.

The rooms were pleasant (despite the lack of Gideon Bibles), but mostly less magnificent than the lobby. From rooms on the top floor, one could hang out the window to admire the tin and concrete structures adorning the roof. The finest room, however, was that given to Lou Katz, who was attending his last USENIX convention as the organization's president. His room, or rather rooms, included a bedroom, sitting room, living room and dining room, not to mention the servants' quarters, all decorated in the Edwardian opulence of the lobby.

The technical sessions were held at the other end of Temple Square from the Hotel Utah, at Symphony Hall, a modern building with a glass-enclosed lobby that was sunny and well-populated even when refreshments were not available (and refreshments *were* available even at the end of coffee breaks, for the first time in my experience).

Although none of the talks in the technical sessions was outstanding, there were many good ones, few poor ones, and overall the sessions were significantly more interesting than at previous conferences. The main improvement was in the technicality of the talks. There was less marketeering and more attention to design and implementation of the various programs and systems described, as befits a technical conference (at last!). I needn't describe the technical sessions further, as the conference proceedings, which were available on-site, do so admirably.

Near Symphony Hall, but considerably harder to find, was the vendor show, which was poorly attended by the conference goers. Among the factors affecting the low turnout were its distance from the technical sessions (you had to go outside), the quality of the technical sessions, the busy conference schedule, and a waning interest among USENIX attendees to see yet another dozen UNIX systems-in-a-box. Although the conference organizers thought it was their fault that the turnout was poor (in what was otherwise an outstandingly organized conference), the show was simply uninteresting and this was made known quite quickly through the conference rumor mill.

A novel event at the conference was the computer GO tournament. GO is an ancient Oriental board game with deceptively simple rules; most of its proponents claim it is much more difficult than chess. Because of the large board (19×19), the importance of strategy, and the extensive use of large, loose patterns in good tactics, computers have never done well at GO, and it was to encourage work in computer GO that the tournament was organized.

Only four programs entered, but they spanned a wide range of playing abilities. It was a foregone and correct conclusion that the winner would be NEMESIS, Bruce Wilcox's entry, because it was the only serious program in the competition. The other three were shams, and NEMESIS would have been unbeaten if it had not crashed while soundly trouncing Bruce Ellis's goanna, the second-place finisher. Third place went to Peter Langston's entry, ogo, which was a converted test program that played a famous sleazy strategy: playing moves symmetric with the (supposedly cleverer) opponent until the symmetry is broken by someone occupying the central point on the board. Last place went to jim, a program by Hank Dietz, that always committed suicide before its playing skill could be determined.

The tournament was played on a full-sized board, with a time limit of one hour of user VAX-11/780 CPU time per player per game. When the rules were announced a few months before the tournament, prospective entrants felt that was insufficient time, but when the tournament was played, no program came near the time limit, although jim might have had it finished a game. NEMESIS used a total of 36 minutes in 5 games, and the simplistic ogo took 6 minutes 31 seconds, just over a minute of CPU time per game. But the speed record must fall to goanna, which placed second in the world's first full-board computer GO tournament, using a total of 12 seconds of CPU time, an average of 2.4 seconds per game, or 2.7 milliseconds per move. Goanna was certainly the favorite of the surprisingly large and appreciative crowd at the tournament.

Another unusual feature of the convention was a trivia quiz. Contestants were given less than two days to answer 84 pointless questions about UNIX history (sample: What was the first edition of the manual to be typeset?). As the deadline for submissions neared, inquisitive groups swarmed around dignitaries and people known to have *been there* in the early days, ready to scribble down whatever clues might be revealed ("Mr. Chesson, what was your phone number at Bell Labs?").

The best entry — 60 correct — was under the pseudonym I. P. Stubbies, although none of the actual contributors to the entry were Australian. The best individual entry was submitted by Jim McKie, and had 57 correct answers; Ron Gomes had 56. Of course, neither of these really did the research individually, but the contest rules were deliberately vague about ethics. Because of the capricious judging, the best prize — an authentic UNIX Second Edition boot DECtape, went to McKie, Stubbies got a silly trophy and Ron Gomes got an official Sun Microsystems Bill Joy name badge. (The GO tournament prizes were even less interesting.)

Off-site activities: There was a dinner do at the Snowbird ski resort, where people saw snow for the first time since they went skiing the previous day; some people wandered over to the University of Utah, where they saw how to house a tertiary educational institute in secondary school buildings; and on the last night there was a party that I have been asked not to discuss.

Overall, a good conference, for the right reasons, but nothing to write to *;login:* about.

Is This System Mannerist or Rococo in Style?

A Visit to the Promised Land
Salt Lake City and USENIX

P. Rayonnant

Sunday

Five o'clock, the new frontier beckons. Goodbye Dick, goodbye Doughball; don't eat any cables while I'm away. Why isn't the bus here? What, they changed the timetables last week? And this is a holiday anyway? Taxi. No, I don't know which travel agency ordered the tickets, don't you have them indexed by traveller-name? "For operational reasons we are flying this morning in a Bandicoot 20 seater across the North Sea. I'm your co-pilot, and once we're airborne, I'll be serving coffee."

London - yick. Gorky Park - the book was better.

I visited my first two U.S. airports, and got lost in both. There seems to be an assumption that the traveller knows which airline he is flying with. For the connecting flight from St. Louis to Salt Lake City my ticket just said WA691. Each airline has its own monitors, departure lounges, etc. and there is no information desk. Could be *Wright Bros. Airlines*, I suppose. If in doubt, ask a policeman.

Had a beer while waiting for the flight. Not promising.

Flying across the central U.S. with *Western Airlines* was dull, the country is flat, just like The Netherlands, except no one has bothered to make the waterways run in straight lines. How untidy.

In Salt Lake City I just followed the signs to "Baggage Reclaim." Of course, there are two....

This hotel is IMPRESSIVE. Must be old, the lifts are just like in The Netherlands, *slooooooow*. The single-room has a bed wider than it is long, three pillows wide, a Mormon-bachelor occasional-double? Speaking of Mormons, the Hotel Utah is right next to the centre of Salt Lake City, where the Temple is (I can look down into it from my room); the streets in Salt Lake City are not named, but numbered on a Cartesian grid ("100 South 300 West"), and yes, Temple Square is at 0,0.

The clock says eight in the evening, the body says "It's almost time to get up tomorrow", so it must be time for a wander around (short, everything is closed), and something to eat. "Hi *Sir!* Have you made your choice yet?" Mumble mumble. "Thank you, *Sir!* Will that be all?" Mumble mumble. "Thank you, *Sir!* My name's Larry, and I'll be your waiter for this evening." Mumble mumble ZZZZZZZZZZZzzzzzz.

Monday

No! Not yet! Not Yet! OK. I suppose so. Lunchtime. Wandered around. Still doesn't look very alive (I don't either I suppose).

Registration opens in the late afternoon, so I get registrated. Along with the registration packet comes the proceedings, a really handy thing to have *before* the conference; now I can ask even more interesting questions since I'll know it all beforehand. Why did I bother coming? It also means that I don't have to write everything down in a boring trip report, I only need to give the undocumented features not in the manual.

Finally bump into some people I know, time to eat and drink. Well, more eat than drink, Utah has some strange laws about the sale of alcohol. Fortunately beer is easy, but were we foolish enough to want wine with our meal, the restaurant has a *Mini State-Liquor Store* at the front (within and part of the restaurant). Customers who want wine have to go there themselves, buy the evil stuff and return to their seats. The restaurant is allowed to provide glasses and open the wine for you, but is *not* allowed to pour it, you have to do that yourself. Weird.

Back to someone's hotel room for a wine tasting, and very nice they tasted too.

Tuesday

Wandered around some more. Saw *Star Trek III - The Search for Spock* in the afternoon, for want of something better to do. I was looking less like Spock by time I came out.

There were some tutorial sessions on *C-Style*, *4.2BSD Internals*, *Networking*, etc. Some were sold out, some had a handful of attendees. USENIX seems to find these tutorials popular, and wants to continue them at future meetings. The problem is finding people who know the the stuff *and* can communicate it well. Suggestions are welcomed.

The exhibition was also under way, less than 100 stands I would guess. Nothing really new. AT&T wouldn't let anyone touch their machines. Some of the 68K boxes are becoming quite nice; Integrated Solutions were claiming theirs was "what the Micro-Vax should have been".

```
<<`echo "Goodbye World!"`
```

to the Bourne Shell was popular amongst the vendors I tried it on, some machines just crashed, others needed to be powered off.

The most interesting thing about the exhibition was the Ethernet. There was an Ethernet cable to which vendors were invited to attach their machines. I saw about six different brands of hardware all chatting away nicely (using TCP/IP protocols), including a laser printer station from Imagen. Apparently there was hardly any preparation for this Ethernet demonstration, all the more remarkable.

Tuesday Evening

Things are beginning to move now, people are beginning to arrive. And the hospitality suites are open!

Hospitality suites are places where, under the influence of free food and alcohol, you are scrutinised and pestered by representatives of the company supplying the food and drink with a view to them weaning you away from your current blissful employment.

But wait! There is something wrong! Here I am, eating and drinking in a hospitality suite, *and the reps are ignoring me!* What is wrong? Why is everyone looking at the T.V. set? I ask, and it seems there is a *basketball* game on, the last one of the season or something. It looks exceedingly dull, everyone runs to one end of the pitch, the ball may or may not go through a hoop attached to a board, then everyone runs back to the other end; the team which loses seems to be the one which fails to put the ball through the hoop most, as far as I could see each team gets an equal number of tries. If it was profiled, all the running around would probably be moved out of the loop‡.

Other hospitality suites were the same. Refreshing (at the time).

‡ This wasn't the only strange game I saw on the television, there was also *American football* and *baseball*. American football is remarkable in that only one person on each team ever actually uses a *foot* to kick the *ball*, and he spends most of his time sitting on a bench, waiting for the ball to be placed in front of the goal posts. Maybe we should rename soccer *handball*, since only one person on each team is allowed to handle the ball?

The Dutch for baseball is "honkbal".

Wednesday Morning

Come Wednesday and it is time for the official conference to begin in earnest. The Symphony Hall is impressive, and I found the lack of central aisles (one had to enter from the sides) adequately compensated for by the comfy chairs and leg room.

There was a professional audio-visual set-up, three large screens at the back of the stage, one for overheads (hardly used), one for slides, and the central one with a projection of the speaker's head and shoulders. Hello Mum. Slides were definitely the order of the day, and some poor soul slaved away night and day making slides from the overheads which everyone brought; he deserves some thanks.

Some computer-graphics films were shown on the multi-media equipment as the UNIX community filtered in for the start of the meeting (why do these things always start so early?). Someone told me they were from the last SIGRAPH. Econo-Mars Earthtours looked preferable to British Caledonian.

First business of the day was some announcements. There were two competitions being run during the conference, the First USENIX Computer GO Tournament, run by Peter Langston, and the UNIX Trivia Quiz, run by Rob Pike, 84 questions along the lines of "What was the telephone extension of the inventor of multiplex files?".

The next meeting will be held in Dallas (there will be a simultaneous conference and trade-show held there by /usr/group, and future meetings will be in Portland (USENIX only), with Atlanta, Boston and San Francisco as possibilities after that. USENIX is also sponsoring limited attendance (approx. 120) workshops, on topics such as distributed systems and computer graphics; participants will have to send in a position paper first.

The USENIX Newsletter *;login:* now has Brian Redman as technical editor, and USENIX is still looking at the possibilities of a journal and electronic publication.

After that, someone from the EUUG spoke for five minutes about what was happening in Europe and announced the forthcoming EUUG meeting in September in Cambridge. One of his slides appeared to be in Dutch, but it was just back to front and upside down.

There then followed the keynote address, *An Architecture History of the UNIX System* by Stuart Feldman (the man who brought you *make*). This was a real treat. It didn't appear in the proceedings, as that would have spoiled the fun. As it was there was lots of fun, not all of it expected. The talk was basically a pictorial trip through the last thousand years of Western architecture, detailing the rise and decline, with side references to the history of the UNIX system. The added fun came as Stuart's slides caused havoc with the slide-projector, having the audacity not all to be of the same thickness (many were borrowed from libraries and museums). At one point, as one refused to be ejected from the projector, there were cries to pass the remaining slides round the audience instead; coffee was taken early as the projector was dismantled.

Michael Tilson then gave a well-reasoned talk about the need for standards in UNIX, and the coming confrontation between UNIX and Godzilla. He gave a history of UNIX, including a slide depicting the real UNIX family tree, it looked like a plate of spaghetti.

Hot in pursuit came some talks about electronic mail and news. The most interesting was given by Robert Elz and described ACSNET, the Australian alternative to UUCP. There are some published papers on this, and it is quite interesting, all routing being taken care of by the network, explicit routes don't need to be given. There was a lot of speculation as to how this would scale up to a network the size of the current UUCP network.

Other talks in this session were *Broadcasting of Netnews and Network Mail via Satellite*, *The Berkeley Internet Domain Server*, *MMDF II: A Technical Review* and *DRAGONMAIL: A Prototype Conversation-Based Mail System*.

I didn't see all of these, having to go to a meeting over lunch, where I had a banana-split King Kong would have had trouble with.

Wednesday Afternoon

Apart from the effort needed to digest the world's largest banana split, the afternoon was spent listening to talks about distributed things.

The first session was about networks and distributed systems. In one talk about converting the BBN TCP/IP software for 4.1BSD to 4.2BSD, the claim was made that the throughput was comparable. Someone from the audience pointed out that the particular Ethernet controller being used was in fact the limiting factor in performance in both cases.

Other talks included how to synchronise the clocks of the different machines on a local area network, the *LOCUS Distributed UNIX System*, and *Project Athena*.

This last is interesting, as it gives a sense of scale of investment being made in the U.S. by computer manufacturers. Project Athena is a joint venture by MIT, DEC and IBM, which will install approximately 1600 VAX-based personal computers at MIT for an experiment in undergraduate teaching; about 2500 machines will be installed in all. 4.2BSD is the current base software. You can imagine the problems of networking, backup, maintenance and just plain scale. The results, whatever they are, will be interesting.

The second afternoon session (after the delights of fishing cans of strange liquids out of enormous buckets of ice; what is "root beer", it tasted of toothpaste?) was about distributed filesystems. Peter Weinberger talked about the *Edition 8 Network Filesystem*.

Peter was fun to watch, he bounced around so much, dashing over to try to read his own slides, that the audio-visual projectionist had to work up a sweat. The talk didn't differ a lot in content from that he gave in Amsterdam in 1982, but he'd had the time to replace the hand-written overheads by BTL-style slides which even he couldn't read.

The filesystem gives you network independence by using asymmetric virtual circuits, the client reads the server's files, the server handles permission translations and time differences. Remote objects are filesystems, not disks; pairs of machines negotiate a connection. Don't expect much from the network, as that's what you get.

The implementation uses user-level servers, and switches at the inode-level in the kernel to test if this is a remote file or not (doing it at the user library-level is described in the abstract as a "maintenance nightmare").

The other two talks in this session were also quite interesting, a system called IBIS from Purdue which is based on 4.2BSD and uses the *host:filename* syntax to access remote files, and the *Livermore Interactive Network Communication System (LINCS)*. This runs in a *very* heterogeneous environment, and includes TCP/IP connections from VAXs running VMS/EUNICE to "real" UNIX machines.

The last part of the afternoon was taken up by a panel session on distributed file systems. Not a lot interesting was said. Bill Joy was in "Paris mode".

Wednesday Evening

Back to nature, a bus to a ski-lodge in the mountains, and the official conference reception. Sunshine, trees, snow, rushing streams, all to be experienced as you stand beside a bus with burned-out transmission.

But it was fun. Cable-car from the ski-lodge (Snowbird) to the top of a mountain (11,000ft), slither around in T-shirt and gutties. Back down for food and wine (altitude affects both your breathing and also the ease with which alcohol affects your system). Crossed-legs bus ride back to the hotel, then on to the DEC hospitality suite, it was open every night, all the time. This time people in uniforms came to break it up. What fun.

Thursday Morning

Nooo problem! Breakfast is though. "What you got?" "Oh! We have ... etc." "Got any muesli?" "What's *that?*" Kelloggs wins the day.

My enthusiasm for cornflakes is matched by my enthusiasm for talks about C compilers and other programming language issues. There was a talk about Modula-2 in the system-programming environment which didn't look too bad. However, many people's vocabulary doesn't extend beyond the third letter of the alphabet.

Coffee break. Well, they call it coffee; there's more strength in the C-compiler's type checking. There are also strange sticky pastries, guaranteed to necessitate the application of a Swiss Army knife to the moustache.

A number of talks about programming environments and windowing systems. Rob Pike described a development which we can't get of something he talked about before which we still haven't got. Pity, it's probably what I want.

A team got together over lunch to discuss questions such as "Which university stole UNIX by telephone?". I like pastrami. They disagreed over some of the finer points, and two different entries went in to the "Pastrami Trivia Quiz".

Thursday Afternoon

Into the kernel. Multiprocessor UNIX, what you need to do to make it run on a 3B20, the system can be configured to run in uni-processor mode (no performance penalty) or multi-processor mode (70% throughput increase).

Richard Miller, of porting UNIX to the 7/32 fame, talked about adding demand paging to System V. Only ps(1) need be changed, the performance is equivalent to the swapping system. Uses a working-set page replacement policy as opposed to the 4.?BSD global clock algorithm. It wasn't done on a Vax originally, though.

After more fishing around in buckets of ice for cans of refreshment, the talks resumed. Tom Killian gave his talk about processes as files (Andrew Hume gave a very brief presentation of this in Nijmegen). It really is neat the way it fits in with "the UNIX way of doing things". Debugging and ptrace just fall out naturally. Implementation goes through Peter Weinberger's Edition 8 filesystem, where there is a *file system type* for each of these weird things.

Thursday Evening

Time to hit the hospitality suites again. Silicon Graphics have their flight simulator in a hotel room, but the room is simulating the Black Hole of Calcutta. Still, they did have some *Anchor Steam Beer*, positively tasty compared to the competition. Have something to eat instead, visit the GO competition (won by a ported IBM-PC program), visit the Human Computing Resources Coffee and Liqueurs, drop in on the DEC suite on the way home, finally to bed. It must be the jet-lag.

Friday Morning

Slightly more of a problem. Cornflakes, weak coffee are beginning to take their toll.

Performance analysis and comparisons is what we have first. Someone at AIM collected a lot of benchmark data, then worked out how to display it; what it means is just the same as any other UNIX benchmark you've ever seen. Not a lot.

Two talks, one from a mannerist comparing SV to 4.2BSD, and one from a rococoist about how to speed up 4.2BSD. A big fight was expected, involving a somewhat partisan audience, but it never happened. Everyone was amazingly appreciative of both sides of the coin, or had also been to the DEC hospitality suite. Both systems have their good and bad points, you pays your money, you get the same performance provided you matched the system to the use.

Before the coffee break, Rob Pike announced the results of the Trivia Quiz. He was surprised about how much people out there actually knew about the dark history of UNIX. However, the winning entry from a team only managed 60 correct answers out of 84. The second placed entry managed 56 correct, and received the best prize, a DECtape labelled "11/20 8K sys with RP11 units July 5/72".

The talks in the remainder of the morning were missed due to a combination of worrying about an export licence for the tape I'd just received, and the need to take an early lunch with some people who had to leave the conference early.

Friday Afternoon

One of the interesting things about the talks throughout the conference was the number which dealt with forecoming technology, more saying "what are we going to do with this when it arrives?" rather than "here's my solution to last years problem". And two things which are coming are optical disks and the need to distribute software as automatically as possible where possible. I wouldn't say I agreed with the possible solutions offered to these two particular examples, but at least some people are looking a bit further than yesterday.

That was more or less the conference, and a good one it was too. There were some other talks and panel sessions, but (and this is my only complaint about the conference organisation) they were held in another place, about 7 minutes walk away and in parallel with the other talks. This made it impossible to see whatever was best at the time. Some looked like they might be interesting, the future of BSD, the UUCP/USENET projects (doomed to failure in my humble opinion) and the ANSI C standard draft.

Throughout the conference Rob Pike was taking pictures of people for the face-server, which would be digitised into high-resolution form, then digital faces of any size are generated on demand; some commonly used ones, such as the 48x48x1 faces for mail announcement are precomputed.

There was strong competition for the "hairy knees of the conference" award, Salt Lake City being a warm place. However, after a day spent skiing in his shorts, the twin pillars of flame owned by the runner-up from the Nijmegen conference, Andrew Hume, made all other entrants pale in comparison.

Friday Evening

There was a "wizards" party up in a cabin in the mountains. Fun trekking through the snow and trees to somewhere where the door was a window, the fridge was a snowdrift at the backdoor and the toilet a gopher (or similar beast) hole.

It was great, especially trying to get down in the dark through a snow covered forest to a car after the snowdrift was empty.

Saturday Morning/Afternoon/Sunday

Somewhere in here I negotiated two U.S. airports without mishap (pity British Caledonian can't learn to do that too).

What's the collective name for a bunch of U.S. kids? A brace. No wonder they all need psychoanalysis in later life, having to go through puberty with scaffolding on their faces.

Somewhere in here Saturday became Sunday very quickly. No, "Reuben, Reuben" is not about sandwiches.

"I climbed on the back of a giant anchovy, and flew off through a gap in the clouds...."

Trivia Quiz Answers

Below are the answers to last the trivia quiz that appeared in the last issue of *;login:*. The syntax for the answers is:

left to right precedence
a | b is a or b
a & b is a and b (any order)
a ! b is a and not b

1. The source code motel: your source code checks in, but it never checks out. What is it?
   sccs
2. Who wrote the first UNIX screen editor?
   irons
3. Using TSO is like kicking a [what?] down the beach?
   dead whale
4. What is the filename created by the original *dsw*(1)?
   core
5. Which edition of UNIX first had pipes?
   third | 3
6. What is -=O=-?
   empire
7. Which Stephen R. Bourne wrote the shell?
   software | 1138 | regis
8. Adam Buchsbaum's original login was sjb. Who is sjb?
   sol & buchsbaum
9. What was the original processor in the Teletype DMD-5620?
   mac & 32
10. What was the telephone extension of the author of *mpx*(2)?
    7775
11. Which machine resulted in the naming of the "NUXI problem"?
    series 1 | series one
12. What customs threat is dangerous only when dropped from an airplane?
    belle | chess machine
13. Who wrote the Bourne shell?
    bourne
14. What operator in the Mashey shell was replaced by "here documents"?
    pump
15. What names appear on the title page of the 3.0 manual?
    dolotta & petrucelli & olsson
16. Sort the following into chronological order: a) PWB 1.2, b) V7, c) Whirlwind, e) System V, f) 4.2BSD, g) MERT.
    cagbef | c a g b e f
17. The CRAY-2 will be so fast it [what?] in 6 seconds?
    infinite | np-complete | p=np
18. How many lights are on the front panel of the original 11/70?
    52

19. What does FUBAR mean?
    failed unibus address register
20. What does "joff" stand for?
    jerq obscure feature finder
21. What is "Blit" an acronym of?
    nothing
22. Who was rabbit ! bimmler?
    rob
23. Into how many pieces did Ken Thompson's deer disintegrate?
    three | 3
24. What name is most common at USENIX conferences?
    joy | pike
25. What is the US patent number for the setuid bit?
    4135240
26. What is the patent number that appears in UNIX documentation?
    2089603
27. Who satisified the patent office of the viability of the setuid bit patent?
    faulkner
28. How many UNIX systems existed when the Second Edition manual was printed?
    10 | ten
29. Which Bell Labs location is HL?
    short hills
30. Who mailed out the Sixth Edition tapes?
    biren | irma
31. Which university stole UNIX by phone?
    waterloo
32. Who received the first rubber chicken award?
    mumaugh
33. Name a feature of C not in Kernighan and Ritchie.
    enum | structure assignment | void
34. What company did cbosg ! ccf work for?
    weco | western
35. What does Bnews do?
    suck | gulp buckets
36. Who said "SEX, DRUGS and UNIX"?
    tilson
37. What law firm distributed Empire?
    dpw | davis & polk & wardwell
38. What computer was requested by Ken Thompson, but refused by management?
    pdp-10 | pdp10
39. Who is the most obsessed private pilot in USENIX?
    goble | ghg
40. What operating system runs on the 3B-20D?
    dmert | unix/rtr
41. Who wrote *find*(1)?
    haight

42. In what year did Bell Labs organization charts become proprietary?
    83
43. What is the UNIX epoch in Cleveland?
    1969 & dec & 31 & 19:00
44. What language preceded C?
    nb
45. What language preceded B?
    bon | fortran
46. What letter is mispunched by *bcd*(6)?
    r
47. What terminal does the Blit emulate?
    jerq
48. What does "trb" stand for (it's Andy Tannenbaum's login)?
    tribble
49. allegra ! honey is no what?
    lady
50. What is the one-line description in vs.c?
    screw works interface
51. What is the TU10 tape boot for the PDP-11/70 starting at location 100000 (in octal)?
    012700 172526 010040 012740 060003 105710 012376 005007
52. What company owns the trademark on Writer's Workbench^{tm} Software?
    at & t communications
53. Who designed Belle?
    condon | jhc
54. Who coined the name "UNIX"?
    kernighan | bwk
55. What manual page mentioned Urdu?
    typo
56. What politician is mentioned in the UNIX documentation?
    nixon
57. What program was *compat*(1) written to support?
    zork | adventure
58. Who is "mctesq"?
    michael & toy & esquire
59. What was "ubl"?
    rogue | under bell labs
60. Who bought the first commercial UNIX license?
    rand
61. Who bought the first UNIX license?
    columbia
62. Who signed the Sixth Edition licenses?
    shahpazian
63. What color is the front console on the PDP-11/45 (exactly)?
    puce
64. How many different meanings does UNIX assign to '.'?
    lots | many | countless | myriad | thousands

65. Who said, "Smooth rotation butters no parsnips"?
    john & tukey
66. What was the original name for *cd*(1)?
    ch ! dir
67. Which was the first edition of the manual to be typeset?
    4 | four
68. Which was the first edition of UNIX to have standard error/diagnostic output?
    5 | five
69. Who ran the first UNIX Support Group?
    maranzano
70. Whose Ph.D. thesis concerned UNIX paging?
    ozalp & babaoglu
71. Who (other than the obvious) designed the original UNIX file system?
    canaday
72. Who wrote the PWB shell?
    mashey
73. Who invented *uucp*?
    lesk
74. Who thought of PWB?
    evan ivie
75. What does *grep* stand for?
    global regular expression print | g/re/p | g/regular expression/p
76. What hardware device does "dsw" refer to?
    console & 7
77. What was the old name of the "sys" directory?
    ken
78. What was the old name of the "dev" directory?
    dmr
79. Who has written many random number generators, but never one that worked?
    ken | thompson
80. Where was the first UNIX system outside 127?
    patent
81. What was the first UNIX network?
    spider
82. What was the original syntax for "ls -l | pr -h"?
    ls -l>"pr -h"> | <"ls -l"<pr -h
83. Why is there a comment in the shell source "/* Must not be a register variable */"?
    registers & longjmp
84. What is it you're not expected to understand?
    6 | 5 & process

;login:

The USENIX Association Newsletter

Volume 9 Number 5 **November 1984**

CONTENTS

* Reprinted here with permission.

After Newcastle, What?
A Report on the Distributed UNIX Meeting

Newport, RI, September 1984

Veigh S. Meer

Bell Communications Research
Morristown, NJ 07960

ABSTRACT

Consider the Cartesian product of: (kernel|library|shell) × (remote files|remote procedures) × (heterogeneous|homogeneous hardware) × (use what's there|redo everything) × (works now|being built). That covers this conference pretty well: how to handle a larger user community by piecing together conventional UNIX service on smallish machines. The easiest thing to do is remote files via a new C library. It's about time somebody started thinking about doing something new. When Multics was as old as UNIX, it had been dead for ten years.

Introduction

About 80 UNIX hackers met at the Viking Hotel, Newport, RI Sept. 12-14, 1984 to talk about distributed UNIX systems. Among major projects represented were 4.2BSD (Leffler and McKusick), Newcastle (Lindsay Marshall), Athena (Dave Mankins & others), BB&N (Gurwitz), and SUN (Bob Lyons). Nobody spoke from CMU-ITC or Bell Laboratories research. This is an opinionated report which reflects only the views of the author.

This conference description will be divided into two parts. The first is simply a chronological narration of the talks, since there is no printed proceedings. This is followed by a summary of the state of the art, combined with my personal view of the future.

Narration

Doug Comer, from Purdue, described their TILDE project, a broadcast net allowing terminals to broadcast service requests (for whole processes) which servers then bid for. They imagine a basically conventional environment in which the distribution serves for load-balancing, thinking in terms of user files migrating around with a time constant in days, and for increasing community size. They have not faced up to authentication and dissimilar hardware, preferring to do something useful now. TILDE has similarities to a lot of dumb terminals connecting to standard UNIX hosts, with a self-adapting switch in between that evens out the user distribution on machines without a system administrator intervening.

Nancy Hall, of Wisconsin, followed with a talk on Crystal. A PRONET ring (as is TILDE), Crystal has an unusually large number of diskless CPU nodes (40 11/750s, of which 4 have disks). The idea is that a few machines (mostly 3 780s) support remote files for a lot of processes running on the CPUs. The operating system is Charlotte, not UNIX, but has similar properties and goals. Basically this is an attempt to increase the throughput of the three 780s by offloading CPU work to the 750s. It seemed the 750s had arrived for unspecified other reasons and Crystal was an attempt to make use of them. I think you can get the same benefits more cheaply by offloading to 68K boards (which can just be dedicated to particular host CPUs since they're so cheap) and the architecture here has far too little disk and far too much CPU for typical use. Maybe they have lots of number-crunchers at Wisconsin.

Keith Lantz, of Ridge and Stanford, followed with a description of the Ridge operating system (ROS), based on the Stanford V-system. He was the purest of the pure, planning network-wide virtual memory, global userids and authentication, etc. implemented on a message-based operating system. His demand for no execution cost in using the network prompted an interesting discussion among Leffler, Gurwitz, Marshall and Lantz about efficiencies of implementations; the general word was you have to put it in the kernel to make it fast. Lantz's goals are so large (absolute network transparency, IPC with no programming cost, etc.) that neither 4.2BSD nor LOCUS is distributed by his standards. More seriously, I think his goals demand centralized administration, and I see a lot of pressure for distributed systems that do NOT have central control. We got UNIX partly because people did not want to submit to big computer center bureaucracies when computers got cheap enough for each group to have one; now I see the same phenomenon with workstations. Anyway the goal of ROS is one big operating system using message-passing down below so the parts can be scattered around. Entire campuses or even more can be a single computer system. If it works, the Justice Department will get them.

Joe Requa, of Livermore, next showed that the rich are like the rest of us, except they have to connect Crays while we connect Suns. They have 5 Crays (4 Cray-I, 1 Cray-XMP), 3 7600s, and only 5 VAXes. Two NSC Hyperchannels connect everything. Most of the talk was at the low level of what system primitives would be provided; they are providing a complex protocol for message handling which is intended to support UNIX tasks. This is hard because UNIX processes are expensive, none of the standard device drivers are appropriate for their network, and they have heterogeneous hardware. Livermore is intending to run full ISO; maybe you need Crays to keep up with that. What with ISO, datagrams, machine-independent reliable flow control, etc. this project may pour more cycles into networking than anybody else. It does not help that they are unwilling to change UNIX because they want to buy only binary licenses; Rob Kolstad was among many who marveled at the thought of buying more Crays than VAXes but saving on software licenses.

John Forecast, of DEC, described the job of making 4.2BSD talk to DECNET. His talk convinced everybody that it was a bad idea. DECNET is large and has many complex options (e.g. optional connections) not known to UNIX. Making the link without changing DECNET is both complex and limiting. However, the job was done in impressively little time (6 man months, no previous UNIX experience).

Peter Reiher, of UCLA, next described how Locus will keep track of names. This was a fairly routine description of a plan for name servers keeping track of mounted file systems, each server being very reliable, and each workstation caching its local needs.

Dave Mankins, from MIT, now talked about naming in Athena. There was a long discussion about their convention (/@name for a remote name) which seems to get people overly excited. More interesting is that their file opens return connections, not files, and can actually be pipe-type connections to processes. Athena achieves 30Kbytes/second transfer rate over the network, and can also do remote procedure call. The basic protocol is datagrams. Athena was unusual at this meeting for its scale, and seemed fairly far along towards accomplishment.

There followed the most outspoken speaker of the meeting. Lindsay Marshall talked on the Newcastle connection. In two months he replaced the C library with remote file system routines, running on the Cambridge ring. They use /.. to signal remote names. His message was that remote file systems are easy. He did not claim it was exciting or new; but Newcastle publicized it and sold it. Personally, the earliest library rewrite I know of is Priscilla Lu's R-library at Bell Labs in 1976; anyone know an earlier remote library? Marshall is now working on a triply redundant file system with majority voting for ultra-reliability.

Bill Appelbe of UCSD described the reverse of the Livermore situation: teaching 500 students Pascal on 55 IBM PCs. Each station has only a floppy, and the Omninet is slow. On the other hand, all the students have to run is Pascal, and it is more important to restrict them (for security and resource conservation) than to get work done. Most of the discussion was security; students can not even unmount their floppies without logging out. But the equipment limits here were so serious that much of the design is seriously distorted from the viewpoint of a programmer or researcher.

Chris Kent at Purdue described a disk block cache. Disk caching is harder than file caching because of locking problems; the talk and discussion covered write-through, multiple caches, and similar techniques. He proposed a broadcast protocol. This work is just beginning. Bob Lenk (HP) suggested reasonably enough that simultaneous writing was so rare that canceling caching in such cases would cause no overall performance penalty.

Dan Franklin from Interactive was also doing disk block caching. Again, the goal is remote file access with full support of upward movement (*chdir* ...), many networks, and kernel processes. Franklin has protocol details worked out to handle write-through and avoid races.

The next few talks were short talks given in an extension session; being shorter, they get shorter summaries.

Mike Dean, of BB&N, discussed the Cronus system, 9 hosts on an Ethernet providing an object-based operating system on top of some host. This project conforms to standard goals (remote file system, remote procedure call, reliability, authentication, message-based) but my bet is that the combination of flexibility, data abstraction, and user-space code will make for poor performance.

Dave Korn of AT&T-Bell Labs had put UNIX on top of Apollo Domain. He added //x to /.. and /@x as names for the global file system.

Wayne Hathaway of NASA-Ames envisages an even larger and more expensive world than Livermore, as part of a national resource for computational fluid dynamics. Think about a Cray-2 with 500 MB of main memory, 200 GB of mass storage, 25 Silicon Graphics IRIS workstations as terminals, and Hyperchannels connecting them. Then go back to your ADM3a and cry. (Or think about swapping 500 MB or backing up 200 GB and smirk). Anyway this was not a distributed operating system, or even a distributed file system. The users have to know where everything is. This project was an example of "pride in plagiarism" (speaker's words): they took lots of System V, 4.2BSD sockets, Newcastle, etc. Or maybe it was mostly pride in hardware.

There followed the fastest contrast in the meeting: Larry Phillips of the University of Toronto is trying to teach 6000 engineers on 2 VAXes. They use NSC 32016 microprocessors (previously named 16032) on a Hubnet and limit the VAXes to fileservers. Quick description of the software is Newcastle in the kernel.

Divyakant Agrawal of SUNY Stony Brook is building a name server for a distributed relational data base system. A quick description of this software was Newcastle for relations.

Now we return to the full-length talks. Mike Lesk of Bell Communications Research spoke next, arguing that multi-processors should make one job faster, not just run more jobs. Furthermore he does not like CSP-type languages for programming. So he proposed rewriting *troff* in a concurrent production systems language, e.g. OPS-V. This permits dividing either the workspace, or the rule space, or doing multiple productions at once. The audience was generally skeptical. Martin Levy has distributed *make*, if you want a particular program that works now. This talk, like many others, was premature (not enough work yet).

Bob Lyon, of SUN, described the SUN language (XDR) for representing data on a network. XDR is supposed to deal with byte order problems, pointers, etc. Each data type must come with a routine to describe it. SUN has written routines for typical C types; plus the IEEE floating point standard is used. Some suggested there might be international standards coming, but I doubt it. An important data type missing, according to Dave Korn, was the bitmap. The cost of remote procedure call between dissimilar machines with binary arguments looks like it will be high, no matter what techniques are used.

Mike Hawley, of Lucasfilm, compared the SUN and Blit user interfaces for graphic programming. He is building an audio library for sound effects (laser guns, stomach punches, etc.) and has programmed extensively on both the SUN and Blit. Hawley strongly preferred the Blit, saying Rob Pike got it right (e.g. a rectangle is 2 points, not 4 ints), and ported the Blit stuff to the SUN in a month. Sam Leffler had tried a sample job twice and it took 1/2 the code in the Blit version. This was the only evaluated comparison in the entire conference.

Jeff Langer, of AT&T Bell Labs, described another remote procedure call system based on messages. They provide the messages via virtual circuits, however, separating the RPC from the network protocol; they intend to run on any of X.25, Ethernet, RS232, etc. The RPC itself is fast (a few hundred microseconds per transaction) but the underlying network they are now using is expensive. Again, multiple layers to separate protocols neatly costs performance. The overall design of the message-based RPC resembled DMOS. This talk, and the next, both reference the Nelson/Birrell PARC work.

Bob Souza, of Athena, described their remote procedure call. It is intended to have 2000-3000 stations across the MIT campus, as part of a curriculum development experiment. They support both blocking and non-blocking calls, with non-blocking calls intended for operations such as writing on a printer when you are never really sure the characters got onto the paper anyway. The RPC is multi-language; Athena supports C, Lisp, Fortran and (possibly) Pascal. A datagram protocol based on UDP is used and arguments are passed by value. There was no performance data yet.

Jehan-Francois Paris, of UCSD, described a proposed fault-tolerant file system, based on multiple hosts. Replication is a file system attribute; consistency is checked on open and majority vote decides discrepancies. This was implemented in the user level library (i.e. replicated Newcastle).

Tim Hoel described UNIX on the Cray-II, a new machine "so fast that it can run an infinite loop in 6 seconds." They are porting System V, and trying to maintain very fast file transfers; this includes track I/O, non-blocking read and write, and enormous files, covering several 600 MB drives. Even though the machine costs $15M, the average customer spends more on disk than on CPU. They do file striping (putting consecutive tracks of a file on different disks for speed), use sector or track allocation depending on file size. Little of this talk dealt with distribution; too much of the audience seemed to think that multiple machines was what you did when you could not afford one machine big enough to handle all your customers.

Neil Wiedenhoffer, of Denelcor, described UNIX on the HEP. The HEP is a multiprocessor machine that can run up to 16 processors and up to 64 interleaved instruction streams ("we can't run an infinite loop as fast as the Cray, but we can run 50 of them at once"). The basic primitive will be *fork*() + create new stream. Each instruction stream is a 2-3 Mips stream. The software is based on System III, plus the Bourne shell and the 4.2BSD file system. The LAN is a switch. The software has some basic primitives for synchronizing by "wait for write" type operations. Debugging is difficult; they are working on a new debugger named "The Force" (described as "like duct tape, it has a dark side and a light side and it holds the universe together").

Finally, there was a summary session, featuring Lindsay Marshall, Doug Comer, Al Nemeth and Rob Gurwitz. It gave a good perspective on the meeting; a little coverage of what has been done followed by a lot of discussion of what needed to be done. UNIX has been made to work on distributed machines, but functionality has not been improved, Marshall pointed out. Jim Gettys emphasized that distribution was about control of resources; you have to achieve both local service and access to a community to get happy users of distributed systems. Nobody knows how to debug distributed systems, and there is little progress so far on fault tolerance. Nemeth wanted more information on performance; speed is needed. The summary session sort of tailed off, with discussions of control of resources and administration, but reaching no real conclusion. So far we really do not know much about doing authentication, connecting dissimilar CPUs, controlling and allocating resources, and how to do all of this while leaving enough cycles for the users to accomplish something. There's lots of opportunity left in the field.

The conference organizers should be complemented on the attendance. The audience was vocal and knowledgeable, and nobody was in a three-piece suit. Publicity exclusively by netnews and invitation seems to work. Discussions were very technical and productive; almost as much fun as hacking. One sour note for future planners to remember: although Newport was conveniently located and provided good weather, the Viking Hotel provided a noisy conference room, appeared to be built of parts that flunked the Best Western acceptance test, and charged at least one attendee who made reservations separately substantially LESS than the "special conference rate."

Where We Are

Those interested in this conference should also be aware of the CMU ITC project. It involves a plan for a standard workstation with a campus-wide Ethernet and cached files. Dave Rosenthal would be a good contact. Meanwhile, at Bell Labs Dennis Ritchie has replaced all the character handling and communications routines in the kernel, and on top of that Peter Weinberger has built a network-independent remote file system, using the Datakit switch as the first exemplar. Remote files are implemented in the kernel; you always execute on the original machine, and ordinary file system name syntax (e.g. /n/machinename/...) is used to name files.

Anyway, what of the actual conference? It is clearly possible to build distributed systems by changing any of the library, the kernel, or the shell. The library is smallest and easiest; hence Newcastle. This gives you the ability to access remote files. Next easiest is remote processes, but the UNIX processes are so expensive that this is not a good way to break up big jobs. Remote procedure call is now achievable on identical machines, but is still plagued with problems of data description on heterogeneous hardware.

In all cases, performance is a problem. The standard solution seems to be code in the kernel. What happens when everything is in the kernel I do not know; maybe we should go back to a RISC-type minimal kernel and shell, giving faster processes and command execution, and reducing the need to do kernel code.

There is a basic philosophical difference between combining existing systems, retaining local autonomy, and designing a grand university-wide plan. A clear contrast, for example, is between MIT-Athena and CMU-ITC. University politics, funding, and so forth may decide which road is taken; but there are many hard questions on authentication, resource management, and degree of system integration that are affected. It is hard not to think that the future belongs with the heterogeneous systems, using some kind of "open systems software architecture" that defines remote procedure call protocol or remote file protocol. Many people can then connect their system. I would emphasize making the protocol as simple as possible, at the expense of power; consider the ease with which people have changed the C library vs. the difficulties of dealing with the kernel or the shell. Simplicity would also, of course, help with the serious and completely unsolved problem of debugging.

What do we need in the future? I doubt either the NASA & Livermore experience (multiple Crays) or the UCSD or Toronto configuration (hordes of students and almost no hardware) is typical of the future. Suppose the standard 5M workstation (1 Mips CPU, 1 Mpixel display, 1 Mbit network, 1 Mbyte main memory, and 1 mouse) becomes routinely available as an individual terminal. In that case distribution for load-sharing is not going to be interesting; remote files will be interesting. More important, however, is that putting several CPUs in the station will be fairly cheap. What should they do? Well, they could handle communications, so maybe efficient protocols will become less important. They could also handle caching, a favorite technique to improve performance. But most important, we need a way to use multiple processors, in local environments, to improve the performance of individual jobs.

One of my old managers at BTL used to show a viewgraph that went "1959 BESYS; 1964 Multics; 1969 UNIX." The trouble was he had nothing for 1974 or 1979. UNIX was designed as a uniprocessor operating system; C was designed as a single-thread procedural language; and I think we need some research that starts from other premises. Otherwise something else is going to be the operating system of the future.

European UNIX System User Group Meeting

Cambridge University, 19/21 September 1984

Peter Collinson
Secretary

Introduction

I hesitate to start this report with the magic words 'this was another good conference', but it was. There were three components: technical sessions in the Chemistry Lecture theatre; industry sessions held in the University Arms Hotel and organised by /usr/group/UK; and a large exhibition organised by Network Events Ltd., a company specialising in (yes, you guessed it) running exhibitions. I did manage to get to the exhibition, but didn't make the industry sessions, I will endeavour to get someone to write a report of what happened there.

This report is again helped vastly by the abstracts which were supplied by speakers before the event, for which many thanks. We hope that the papers submitted for the conference will be printed soon. It is also hoped that the Proceedings for Paris will be printed before the conference, so speakers can talk about other things and confuse us all.

Day 1 - 19th September

Well, this wasn't Day 1 for me, I had spent most of the day before in committee meetings of various sorts. This meant considerable quantities of local ale the previous night. Nothing daunted, I managed to make the really early breakfast and start the long trek down to the lecture theatre. It didn't rain - I had my lucky umbrella.

9.45am **Opening of the conference**
Richard Stibbs, Cambridge University

Richard had mostly domestic matters to talk on, the most important being where to find the best beer in Cambridge. He violently denied the rumour that Greene King had sponsored the conference, and said that 'Abbot Ale' was not a trademark of that well known footnote.

After that David Tilbrook had a few minutes to introduce the people who were chairing sessions. He had brought a cutting from Computer Weekly of 13th September. This was in their gossip column and reads (all the spaces are theirs):

> **Peculiar**
>
> ANYONE wanting to attend one of the scintillating technical sessions at the forthcoming European UNIX systems user group meeting in Cambridge has to apply to (wait for it): (qtlon,ukc)! ucl-cs! nigel qtlon! ist! dt.
>
> Is it any wonder that the rest of the world thinks that Unix people are a bit peculiar. Unix? Yukk! (sic) Geddit?!

The item was signed *Chad* Recent attempts to send mail to their quoted net address: Yukk! (sic) Geddit?! has resulted in a profound silence.

9.50am **UNIX Europe**
Vanni Papi, UNIX Europe Ltd.

This was really an opportunity for Mr. Papi to announce AT&T's hospitality suite. More seriously, he briefly described the state of the art in UNIX Europe. It is still quite a small organisation of only 12 people. They are responsible for all UNIX licensing in Europe and will also be running training sessions.

9.55am **Communication Product Announcement**
Joanne Miller, AT&T Technologies

Abstract

This talk will describe the current set of COMMKIT‡ Networking Software products for UNIX System V. A directional statement of the future of UNIX networking will be covered that relates to the network architecture, the operating system and the COMMKIT line.

Comment

The major thing of interest to me was the announcement of the availability of the new *uucp* 'Honey Danber' which Brian Redman talked about at the conference in Nijmegen. It seems that it will never be on a System V tape — it will always be a separate product.

The talk boiled down to a statement of intent which said: AT&T will use agreed international and de-facto standards, will document the protocols and make this documentation available so that other systems can converse with AT&T products. They will also supply technical support for customers.

10.28am **Development of the MicroVAX 1**
Robert Short, DECWEST Engineering

Abstract

Last year Digital announced the first of a new family of computers called MicroVAX. MicroVAX is a proper subset of the VAX architecture intended for VLSI implementation. The first member of this family, the MicroVAX I, was developed at DEC's engineering site in Bellevue, Washington, making it the first major DEC hardware product designed and developed outside of New England. The MicroVAX I processor includes a custom MOS VLSI datapath chip, a cache and translation buffer. This talk describes the MicroVAX I hardware development, including:

- The advantages of subsetting the full VAX architecture to allow a small machine to be designed.
- The goals of the hardware design team.
- The microcode/hardware tradeoffs necessary to provide good performance while still meeting the cost/size/power goal.
- A fairly detailed overview of the hardware architecture of the machine.

Comment

The aim of the exercise was to make a VAX on a chip with the same cost as the M68000. The time scale for the project was 18 months, which is a short development period for a project of this sort. The time scale meant that the machine was based on the Q-bus which so that a totally new bus was not required. It was also decided to simplify matters by removing some of the 'fancy' but little used instructions from the VAX architecture. The machine can run non-privileged VAX native mode programs which are able to run under VAX/VMS on other machines in the VAX range.

The removal of PDP-11 compatibility mode, the decimal instruction set and some of the string instructions meant that there was considerably less micro-code than a full VAX. Instruction decode is

‡ COMMKIT is a trademark of AT&T Technologies.

easier because addressing is simplified by the removal of the PDP-11 instructions, the decode looks at a single byte at a time. All virtual addresses are 32 bits.

The following decisions were made in order to get a 32-bit processor on a 16-bit bus. The code block size is a longword and block mode on the Q-bus allows the processor to read 2 words at a time. The CPU microcode does not know about the 16-bit bus and does not wait for writes to complete down the bus. There is an instruction pre-fetch FIFO to speed things up a bit.

Well, DEC had one of these in the exhibition running ULTRIX. It apparently is faster than the VAX-11/730 but slower than the VAX-11/750.

☛ Coffee ☚

11.20am **Large Systems experience - System 370**
J. Carl Hsu, AT&T Bell Laboratories

Abstract

This talk will discuss will discuss the problems encountered and the experience gained during the development of the UNIX system for the IBM System 370.

Comment

Indian Hill is the largest computer centre in Bell Labs. Disc storage is 10^{12} bytes! The organisation is a technology leader in computing and networking applications, providing standard UNIX systems to other AT&T sites and also generating add-on packages to the standard system.

With the IBM machines, the objectives were to put UNIX on the IBM 370 in order to provide the full power of large systems and provide a working system for the computing centre environment. The implementation was a joint AT&T/IBM effort and uses an IBM supervisor for low level functions. The software is standard UNIX and is mostly in C. The system will support 180 users.

P and V semaphores were used instead of sleep/wakeup because the IBM architecture allows processes to be arbitrarily blocked in the kernel.

11.48am **The portability dream**
Neil Urquhart, Sphinx Ltd.

Abstract

The range of microcomputers offering UNIX has grown and with it the requirement to transport existing software between operating systems. The difficulties in porting software is seen able to be classified as four categories: UNIX development; machine implementation; programmer techniques; and software dependency.

The development of the UNIX operating system, from Version 6 through to System V, is overviewed, with special note being made of their idiosyncracies. Discussion of the contribution of the Berkeley implementation is included together with its influence on the new release of operating particular features which influence the transportability of application software or programs.

Examples of idiosyncratic code are presented to illustrate differences between machines, UNIX implementations and programming styles. C is claimed to be a portable language; it is discussed, with examples as to how much it encourages programmers to deviate from the 'straight and portable'. Some of the UNIX tools which are known to avoid portability difficulties are illustrated together with their limitations. Software dependency is discussed and the difficulties it can cause are illustrated.

12.07pm **Multi-processing on UNIX**
Steven J. Buroff, AT&T Bell Laboratories

Abstract

This talk will describe the development of a UNIX operating system that runs on multiprocessor configurations. The system currently runs on AT&T 3B20 computers, but the ensuing discussion applies equally well to other machine architectures that support a multiprocessor environment. Existing user level C programs can run without recompilation on uniprocessor and multiprocessor models of the 3B20 computer, unless the program contains system dependent code (e.g., *ps*(1)).

The talk will cover the critical region solution that was used (semaphores), dead-locks and resource starvations, scheduling, the protection scheme used for device drivers.

Comment

The abstract says it all. The current system is a dual 3B20 system and runs 1.6-1.9 times a single processor system. The speed improvement is typically 1.7 with a mixed workload.

☛ Lunch ☚

2.31pm **Interactive three dimensional molecular graphics under UNIX**
C.Huang, Computer Graphics Laboratory, University of California

Abstract

The Computer Graphics Laboratory at UCSF was established in 1976 for research on the structures and interactions of proteins, DNA, drugs and other molecules of importance in biomedicine.

MIDAS (Molecular Interactive Display and Simulation) is a large interactive molecular modelling graphics package developed at UCSF under UNIX, originally on a PDP-11/70 with an Evans and Sutherland Picture System 2 in black and white. It now runs on a VAX-11/750, and provides a flexible tool for the study of small and large molecules and their interactions, taking full advantage of available interactive three dimensional color display capabilities on both the Evans and Sutherland Picture System 2 and Multi Picture System and eventually on a PS300. Bond rotation, interactive monitoring of several distances and 'docking' with real time representations of molecular surfaces is well supported.

Among its more innovative features is an unusually coherent hierarchical database for storage of macromolecules which minimizes both storage space requirements and access time. The 'tool building' philosophy encouraged by UNIX has resulted in a well organized and maintainable program that is well suited to reimplementation on UNIX -based graphical workstations such as the Silicon Graphics IRIS. The lecture will be illustrated with color slides and a movie. Supported by US National Institutes of Health research grant RR1081.

Comment

I was particularly impressed by the real-time nature of the pictures which could be generated.

3.08pm **Does Darth Vader still program in Fortran? (or what do they really do at Lucasfilm)**
Sam Leffler, Lucasfilm

Abstract

The original question posed by Bill Reeves at the USENIX meeting held in Boston, Massachusetts was 'Does Darth Vader program in C?'; as one can see by the title the answer was clearly 'No, he programs in Fortran.' This presentation will attempt to update the audience on this noteworthy subject, as well as items of similar importance. High quality audio-visual material will be prominent in the presentation in an attempt to distract the audience from noticing that this talk is almost completely content-free.

For those that don't believe the previous paragraph, the presentation will concentrate on describing what Lucasfilm does, how they use computer technology, and, most importantly, how UNIX is an integral part of everyday work, from the mundane to the exotic. Slides and 16mm film will be shown

of the most recent work from the computer graphics project.

Comment

Sam's answer to the question:

> Does Darth Vader still program in Fortran?

was

> No, the Ewoks do it for him.

Lucasfilm is split into several sections: Administration; Industrial Light and Magic, which is responsible for special effects; Skywalker Development Co., responsible for building a 'ranch' which will be used for filming; Sprocket Systems which does post production work on film; and the Computer Research and Development, where Sam works.

Sam talked about the computer based system which is used to fuse together several images in a process called 'blue-screen matting'. This is where separate films of different objects are taken on a blue background and the images are joined together eliminating the blue. Previous systems used a special machine where the original images are projected and a new negative made. Lucasfilm now have a system where the images are joined using a computer system.

Another area of interest at Lucasfilm is the use of computers to synthesise images. Sam showed a number of astounding slides, the best of which (or at least the one which sticks in my mind) was a picture called '1984'. This showed four pool balls on a table being struck by a cue ball. The pool balls had the obvious numbers. The four balls had all moved and the picture showed their motion blur. Sam pointed out that the high-lights in the balls (pin-points of light) contained the reflections of the room around the pool table. The picture was really great, and very hard to describe.

Sam talked about 'Andre and Wally Bee', a 2-minute animated film generated totally by computer. He was going to show the film, but the projector broke down, so we had an early tea. The film was shown when the projector bulb had been replaced, it has everything: three dimensionality, motion blur, forests generated automatically, to name just a few features.† Great stuff.

☞ Tea followed by the film ☜

At this point, I decided to visit the exhibition and skip the remaining talks for the afternoon. Everyone at the conference had to make the decision about what talks to miss so they could get to the exhibition. The program organisers had thought that two hours would be sufficient time for most people to go in the lunch break. However, they hadn't reckoned on the eons which it took to have lunch in St. Catherines.

So, I left and made my way along the road to the exhibition hall. On my way past the RAC sign saying UNIX SYSTEMS 84, I began to feel that EUUG had come a long way since those 20-odd people met in Scotland those several years ago. Still...

The exhibition was large, at least by previous standards. Most exhibitors were in blue booths, obviously supplied by the organisers, Network Events. It felt very professional, and that was no bad thing. Certainly, the exhibitors who I talked to felt that it was OK, and since there were 2500 visitors, it must be adjudged a success.

One of the things which caught my eye was the SUN system with the wide screen. This was fun to play with and is the way I want to talk to computers.

At the low end of the market, Torch's Unicorn sitting behind a BBC computer is still the cheapest UNIX system you can buy. However, it really is VERY SLOW. The salesman told me that Torch is bringing out a system with a M68000 running at twice the speed of the current version. So, if you're thinking of buying a Unicorn, wait. Of course, the other problem is that the system is UNIX System III 'with the usual Berkeley enhancements'. The visible bell in *vi* was fun on the colour screen, giving bands of different colours.

† And you didn't need to worry about −v, either.

Possibly, the smallest machine, in physical terms, was the Spirit from UNIQIX Ltd. This looked an interesting product.

Anyway, I can't do real justice to all the exhibitors, so I'll stop. Let's hope that most of them come to Paris, so that perhaps there will be more time to look at them.

Meanwhile, back in the Chemistry lecture theatre:

4.10pm **UNIX user interfaces for applications**
Stephen Travis Pope, BST - Basic Software Technology Dept.

Abstract

All computer applications need some man-machine interface method. UNIX is especially well suited as a base for applications because programmers have access to system utilities and can substitute their own front-ends for or on top of the 'shell' program. Developing new user interfaces that are tailored to particular applications requires, however, that the designer repose several basic questions about the desired application, its users and computer I/O in general.

Computer configurations involving bitmap terminals and mice can also be used for very-high-level interfaces and programs can be built to take full advantage of these features. So-called 'window systems' as seen on the Xerox Alto, Lisp machines and the Apple Lisa computers can also be implemented in multi-user UNIX environments with excellent results.

Several current projects being undertaken at the BST dept. of PCS GmbH will serve to demonstrate special, window and/or menu based user interfaces for applications in program development, databank management systems, networking and computer music. Topics of this part of the discussion also include the hardware for the current PCS multi-processor implementation of the *wsh* window shell.

4.35pm **Winnie: a new multiple window screen editor**
Patrick Amar, United Software Artists Inc.

Abstract

We shall describe a new full screen editor called Winnie. Winnie belongs to the Emacs family. It has two major enhancements over Emacs: a more flexible windowing discipline and a multi-language extension facility. As opposed to Emacs, we have strived to build a small and efficient implementation. Winnie uses no more than 35 Kbytes of storage for code (compared to 121 Kbytes for Emacs on VAX) and loads the computer much less than Emacs.

Winnie can handle arbitrarily many windows of any size and placed anywhere on the screen. Windows can hide each other as sheets of paper on a table, or as screens in the Xerox Star station or the Apple Lisa machine. For the present time we handle only alphanumeric screens, but a graphic screen version is forthcoming.

5.05pm **Questions and answers**
Vanni Papi, UNIX Europe

As I had left, I asked Sean Leviseur from UKC to write some notes. So, this bit is from him.

Q. When will virtual memory support be available for System V?

A. Virtual memory will be available in the last quarter of 84 for System V.2.

Q. What will be in System V.3?

A. Chiefly file locking and support for virtual memory and paging.

Q. Will Edition 8 be generally released?

A. No.

Q. How will the presence of UNIX Europe affect pricing?

A. European prices will be the same as in the States.

Q. What price will the upgrade to System V.3 be?

A. This has not yet been decided.

Q. Will Edition 8 be released to educational institutions?

A. It will only be released under strict supervision to six American universities.

Q. When will the UNIX manuals be stabilised?

A. With the next release. They are currently being restructured, only one set is currently available. It would be useful if people could comment on the current System V manuals.

Q. Will System V be unbundled?

A. It is intended to unbundle more of UNIX but not until next year.

Q. Will the information flow from the States improve with the existence of UNIX Europe?

A. We will have more people from the States, so hopefully we should get information quicker.

Q. What about European availability of the Blit and its software?

A. The Blit is sold by Olivetti in Europe, although it is still available from AT&T's old distributors in Europe. The software is available from UNIX Europe, as is the source code.

☛ End of Day 1 ☚

And some of us went onto AT&T's hospitality suite to obtain the odd alcoholic beverage; and then onto the Reception for all conference attendees in the University Combination room, where even more of the you-know-what was consumed.

Day 2 - 20th September

The lucky umbrella didn't work this morning, the heavens opened and tried to wash Cambridge into the River Cam.

9.33am **MMUs & the UNIX kernel**
Robert Jung, Root Computer Ltd.

Abstract

UNIX system performance depends on many factors, one of which is the memory management unit (or MMU). Porting the UNIX kernel to Motorola 68000 and 68010 based systems has given Root a unique insight into the implementation and performance of MMU designs. This talk describes the kernel's interaction with the MMU, observations and recommendations of MMU design and implementation, and a brief discussion of the memory-management schemes Root has come across.

Comment

Root has done many ports of the UNISOFT UNIX system. They have also used a number of MMU's but the talk centred on comparing the Stanford MMU with the Motorola 68451 MMU. The Stanford MMU is also called the SUN MMU, and has nothing to do with the company of the same name.

The Standford MMU is better for memory allocation, switching processes and accessing the user area of the current process. But it is worse at accessing other processes, which is used for swapping. The Stanford MMU is cheaper and runs faster than the M68451.

10.01am **Paging in the UNIX system**
Steven J. Buroff, AT&T Bell Laboratories

Abstract

Two research derivatives of the UNIX system have supported paging for several years: Reiser 32V, and BSD. Work is under way at AT&T Bell Labs to bring together the features of both of the systems [and others] to form a demand paged kernel for UNIX System V. This talk will discuss three areas of this work: requirements, architecture, and implementation.

Comment

The requirements are: there should be no user program changes for either binary or source; the system must not hurt users who don't require paging, this means that if you want to use a paging system because it is faster, then you can do so; and the system should provide the capability of large address spaces if that is wanted.

The idea was to generate a general model of memory management in the UNIX kernel and to abstract all the code which deals with it into one generalised set of routines. Different memory allocation methods can then be used because a clean internal interface has been designed. The main primitive in the design is a *Region*, which is an area of memory. It can be shared or private and is manipulated by a set of well defined operations. These operations are: create, delete, attach, detach, grow, load and copy. The system allows copy on write by adroit use of the page descriptors.

The current implementation is for a paging kernel which appears to work. There didn't seem to be a swapping implementation as yet.

10.28am **Productizing (zic) UNIX**
Armando Stettner, Digital Equipment Corp.

Abstract

DEC is now offering a UNIX product. This talk will discuss some of the problems that were encountered when creating that product.

Comment

At the time of the decision to supply and support UNIX, it was decided to base the system on 4.1BSD. The system was fast and flexible and had support for many peripherals, so there would be much less work for DEC to do in preparing it to be a product. Also, at that time more VAX's ran 4.1BSD than any other system.† The intention was to switch to 4.2BSD when it came out.

The goals of the product were that it should be as least as reliable and predictable as 4.1/4.2BSD. Also, it should be compatible with those systems. DEC didn't want to introduce yet another flavour of UNIX. DEC have not altered anything in the program execution environment and have only altered one user program (*tar* now follows symbolic links).

There were several questions:

Q. Can you install user written device drivers?

A. Yes, the system is supplied in a configurable binary with important files being supplied in 'uncompiled' form.

Q. Does ULTRIX have the device drivers which control non-DEC devices.

A. Yes, anything on the 4.2 distribution is also on ULTRIX. But, it will obviously be harder for DEC software support to provide advice and bug fixes on the 'foreign' peripherals.

Q. What does the management think about that?

A. ULTRIX must fulfill the expectation of what UNIX does. So, the foreign device drivers must be supplied because they are part of UNIX.

† At some point in the conference, I was told by a DEC person that 25% of VAXes in the UK are running UNIX.

Q. Can ULTRIX support the new DEC device clusters?

A. Not at present.

Q. Does DEC intend to move to System V?

A. No.

☛ Coffee ☚

11.17am **A secure high-speed transaction protocol**
Sape J. Mullender, Centrum voor Wiskunde & Informatica, Amsterdam

Abstract

Most computer networks use a byte stream protocol for communication between processes, which suffer from two important drawbacks: the addressing mechanisms provided often are process-dependent or location-dependent, and communication is slow. While carrying out research into distributed operating systems at the Vrije Universiteit and the Centre for Mathematics and Computer Science, we have developed a transaction-oriented transport protocol, aimed for high-speed, with an addressing mechanism that is not only more general, but provides a protection mechanism as well. The basic mechanism for communicating between processes is the transaction: a client process sends a request to a server process, which carries out the request and returns a reply. Protection is provided by using ports, chosen from a sparse address space, for addressing services. These ports serve as a 'capability' for communicating with the service. Through its simplicity, the transaction protocol can achieve high transmission rates (more than 200 Kbytes/sec process-to-process, eventually).

The protection mechanism will be described, and the mechanisms for realising high transmission speeds.

11.48am **Connecting UNIX systems using a token ring**
Robbert van Renesse, Vrije Universiteit, Amsterdam

Abstract

As part of the research on distributed operating systems being done at the Vrije Universiteit, we have implemented a set of network-oriented programs for use on several UNIX machines connected by a high-speed token ring. With these tools it is possible to transfer files between machines, log in to remote machines, and implement multimachine shell scripts. The transaction protocols discussed in another paper at this EUUG meeting are used to implement two basic services: a 'shell server' and a data transfer service. Other services are easily implemented as shell scripts that use these services. A file transfer program, for instance, executes the command *to* < *file1* on one machine, and *from* > *file2* on the other machine. More examples of these facilities and their implementation and performance are discussed in the paper.

Comment

Some real process-to-process throughput figures were mentioned: VAX/VAX - 25Kbytes/second and PDP-11/PDP-11 - 10Kbytes/second.

12.08am **A project development environment for UNIX**
Malcolm Crowe, STRG Paisley College of Technology

Abstract

A software environment for project development should include tools for all phases of the development process. Many such tools already exist, or are under development, in the UNIX system.

However, apart from archiving systems such as SCCS, little support exists in UNIX for quality and project management. A key activity of quality management is concerned with the identification and control of all items produced during software development.

Until recently configuration management has been an essentially manual activity, possibly performed by a member of the project team. In this paper, we describe facilities which allow project management to implement a configuration management plan on a per-project basis.

The basis of the system is an enhancement to the UNIX file system (not affecting the kernel), which does not alter the user interface for naive users. The resulting environment retains all the UNIX tools, while allowing for their use to be restricted in various ways on 'controlled' objects.

The system is available to other participants in the Alvey Programme.

Comment

This was an interesting idea which uses the C library to alter the nature of the file system. Files can now have: long file names (even on V7); a project defined set of attributes; multiple versions; and controlled access. The file system also supports the notion of projects and project hierarchies. The file attributes are defined by name and act rather like shell environment variables.

☛ Lunch ☚

Jim McKie who was chairing this session got some biographies, official and unofficial, which he used to introduce the next three speakers.

Mike Karels

Official: University of Notre Dame (Indiana), B.S Micro-biology 1978; worked in the Molecular Biology Department of the University of California, Berkeley 1978-1983, doing bacterial genetics and kernel hacking in UNIX V7 - 2.9BSD; in August 1983, joined the Computer Systems Research Group.

Unofficial: In 1982, the Paris EUUG conference was blessed with the attendence of Bill Joy. His performance at that conference gave us the phrase 'Paris mode'. Sam Leffler was brought over to the Bonn conference to apologise, and did such a good job that he also attended the Dublin conference. Due to his sterling performance in Eire, we required two UCB people, Eric Allman and Kirk McKusick to compensate at the next conference. This conference is please to have Mike Karels and we have yet to decide who will apologise for him.

Tom Killian

Official and unofficial: Tom Killian began his career as a high-energy experimental physicist but was unable to convince his collegues of the value of Computer Science. Since 1983, he has been with the Computer Science Research Center at Bell Laboratories, where he has successfully dealt with painful childhood memories of MVS and SCOPE.

Greg Chesson

Jazz drummer: CC Riders 1967-9, Woody Herman Orchestra 1969-70; B.S Math, Union College, New York 1972; M.S. Computer Science, University of Illinois 1975; Ph.D. Computer Science, University of Illinois 1977; Member of the technical staff at Bell Labs, 1977-1983; Chief scientist at Silicon Graphics 1983 onwards. At Bell Labs, Greg was responsible for V7 multiplexed files; line disciplines, character drivers and boot programs in V7; packet drivers; circuit simulation, PLA, board layout software in UCPS; design and simulation of Datakit protocols; and the design and implementation of the B-machine 10mips processor for Bell. At Silicon Graphics, Greg has done the XNS network software for 4.2BSD, System V and VMS.

Unofficial: Greg Chesson, known as Bambi to his friends, has made contributions to the UNIX world, none of which should be mentioned prior to his presentation to ensure a reasonable reception. He is well known for his ability to find tone-deaf band leaders who let him exercise his other talents which are normally executed with the tact and diplomacy for which he is well known. His hardware skills are second to few. It is said that there is yet to be a machine which Steve Johnson couldn't overload or Greg couldn't break. Greg enjoys sports that don't involve standing; and the only skill about which he shows any modesty is his extraordinary imitation of a synchronised swimmer using Datakit.

2.36pm **Life after 4.2: measuring and improving the performance of 4.2BSD**
Mike Karels, CSRG, CSD, EECS, University of California, Berkeley

Abstract

The 4.2 Berkeley Software Distribution of UNIX for the VAX includes a number of new features and facilities that substantially increase its utility. However, it has several problems that can severely affect the overall performance of the system. These problems were identified with kernel profiling and system tracing during day to day use. Once potential problem areas had been identified benchmark programs were devised to highlight the bottlenecks. These benchmarks verified that the problems existed and provided a metric against which to validate proposed solutions. This paper examines the performance problems encountered and describes modifications that have been made to the system since the initial distribution. It also describes other work underway or planned at Berkeley.

Comment

Mike started with some general observations on 4.2. First of all, it seemed slower than 4.1, and the system throughput was down by about 20%. The many new servers added considerable system overhead. The new fast file system altered the workload characteristics, so that processes which were previously disc bound were now processor bound.

The system was measured in order to get some idea of what was happening. The results showed that the micro operations were about the same speed as 4.1, although *pipe* and *exec* were a bit slower. The measurements also showed that name translation was about 40% of the system call overhead. Symbolic links add a measurable overhead.

There is some mileage in improving some user level programs. For instance, programs accessing the password file can be made to go faster. The standard I/O library has been altered to use optimal buffering.

In the kernel, the name lookup routine has been made 60% faster by use of caching. This results in an 8% improvement in the total system time. The dz and dh drivers use the silo for slow transfers, which loads the system. The code has been altered to use a single interrupt for the intermittent transfers which constitute most terminal input. The clock interrupt routine has been made to work faster. The arguments to the *exec* system call used to be read into the kernel a character at a time, the new system uses the much faster block copy code to read in strings. The context switch code has been made to run faster. Pipe performance has been improved by supplying more buffering. There have been several other minor alterations.

The system will be available to the public before December 1995[?].

New things which are being worked on a UCB include: a network file system which connects systems using a single tree using a remote mount system call; there is work being out into protocol layers; and a reliable remote procedure call mechanism.

At the end of his talk, Mike made a presentation of a game of 'Battlecars' to Armando Stettner. Apparently, Armando is famed for his accident prone driving and has received many car (or perhaps I should say automobile) related presents at US USENIX conferences.

3.00pm **Processes as files**
Tom Killian, AT&T Bell Laboratories

Abstract

We describe a new file system, /proc, each member of which, /proc/nnnnn, corresponds to the address space of the running process whose pid is nnnnn. Access to these files is restricted, via the normal file protection mechanism, to the process owner. Lseek(2), read(2), and write(2), allow inspection and modification of the process' image. Other services are available via ioctl(2), including stop/go on demand, selective interception of signals, and the ability to obtain an open file descriptor for the process' text file. The technical problems related to the implementation of /proc on a VAX under the 8th Edition of the UNIX operating system have mostly to do with the paging system. Security issues are also considered.

The window-based interactive debugger *pi*, developed by T. A. Cargill, is the first major user of /proc. It can control multiple processes dynamically and asynchronously. Thanks to the network file system, /n, these processes may be running on several different machines. We also describe an efficient, almost portable *ps*(1).

Comment

This is such a reasonable addition to the file system name space, it took a genius to think of it. *Pi* is based on the Blit terminal.

3.31pm **Multicast ring protocols for real time games and other useful pursuits**
Greg Chesson, Silicon Graphics Inc

Abstract

Many great advances in computer science have been motivated by things that some (though not all) would deem as unimportant. This talk is about the solution to the less than pressing problem of being able to 'fly' multiple flight simulators in formation. Only the future will tell if this solution is a great advance, but it is brought to you by the implementor of multiplexed files and this conference's shoo-in for the hairy knees contest.

A demonstration (on video tape) will be shown.

Comment

This is fairly verbatim from Greg's initial slides:

WHAT	do we want to do? To use a Ethernet as a token ring.
WHY	do we want to do it? To synchronise real time processes.
MOTIVATION	to transform a real-time flight simulator on UNIX into a dogfight program for several machines running several programs controlled by several 'pilots'.

Features of the flight simulator include the ability to choose one of several aircraft - Cessna 180, 747, F15, F18 and F16. Weapons on the aircraft include Sidewinder, rocket and cannon. Pilots of the Cessna spent most of their time circling the airfield and picking off other people as they take off. The output is in 3 colours with a 1K resolution screen, input is by mouse - no joystick yet.

The problem is how to get one program running on one machine update all the other programs on the position of their aircraft. The mechanism used is to broadcast datagrams rather than having a 'star' network of virtual circuits.

This talk was certainly one of the highlights of the conference. The video tape spend most of the conference winging its way in a non-simulated aircraft across the Atlantic. It reached Cambridge by

Friday evening, and was worth waiting for.

☛ Tea ☚

4.31pm **UNIX IPC: where it's been, why it left, where it might be going**
Mike O'Dell, Group L Corporation

Abstract

Few topics in the UNIX community have provoked as much discussion and as many implementations as the issue of providing 'good' interprocess communication (IPC) for UNIX. The problem, of course, is not with pipes, for everyone agrees they are wonderful. The problem of interest is a scheme whereby unrelated processes can rendezvous and communicate. Indeed, it is a safe guess that any IPC scheme which has appeared in print has been implemented at least once in some UNIX system somewhere.

If there have been many implementations of IPC for UNIX, why is it that none of them ever seem to catch hold and be adopted widely as a general mechanism? Why indeed!

This talk will attempt to provide some historical background by reviewing some of the more influential IPC implementations, and discuss the author's view of the question posed above. Finally, the author will propose a model for unifying two important stream IPC facilities, and make some speculations as to how this unification might point the way toward a 'natural embedding' of IPC in UNIX.

Comment

This was one of the best talks of the conference. I feel that I cannot do any justice to it from my notes - you will all have to wait until the proceedings are published. Suffice it to say that the talk gave a really comprehensive review of the many different schemes for IPC with an idea of the problems and advantages of each.

☛ End of Day 2 ☚

Well, via AT&T's hospitality room to the conference dinner, which was a pretty splendid affair. After the dinner, The Instruction Set did a side splitting selection of sketches. The EUUG committee were asked to sit in the front, we were bearing up for custard pies or something of that ilk, but nothing really nasty happened - we just laughed a lot. I also met Mr, Mrs and Miss Tis who were very nice.

Day 3 - 21th September

9.33am **Implementation of OSI protocols under UNIX in the EIES network**
J. Loveluck, Bull

Abstract

The Esprit Information Exchange System (EIES) is an infrastructure to support collaborative R and D projects in information technology within the European Study Program in Information Technology (ESPRIT) launched in 1984 by the European Economic Commission. The work is being carried out by a consortium of 6 industrial partners.

The EIES will provide the electronic mail, teleconferencing, document handling and transfer, file transfer, remote login services which are necessary for cooperative R and D work.

The project aims at a maximum connectivity of potential users through the use of Open Systems Interconnection ISO services and protocols, and starts with an implementation under UNIX.

After a short description of the objectives of the project, the paper describes the detailed architectural choices made for the interconnection of local area networks and wide area networks; the addressing scheme is discussed, and some considerations given to the management aspects of the network. Finally the main choices made for the implementation under UNIX are described.

10.12am **The Instructional workbench: a CAI system and more**
Thomas B. Reddington, AT&T Bell Laboratories

Abstract

Any computer-assisted instruction (CAI) system must allow an author to easily create courses for producing on-line dialogue with a student. Instructional Workbench (IWB), the CAI system implemented for the UNIX operating system, is particularly efficient for building a friendly human-machine interface and, therefore is a strong alternative to other CAI systems and programming languages.

Two important features of IWB that will be discussed are the design of the authoring language and the authoring system that enables novices to 'mass-produce' on-line dialogue programs (courses) without requiring a detailed knowledge of the authoring language.

A production system served as the model for the authoring language of IWB, called TOPIC language. This design has proved to be particularly useful for defining the logic inherent in complex human-machine interactions and for easing the work required to modify that logic. Features which are 'built-in' to the TOPIC language are:

- I/O from files
- keystroke verification of user input using regular expressions
- terminal independent screen management capability.

The design of the TOPIC language has been so successful that parts of IWB that interact directly with a user are written in the TOPIC language.

The design of the IWB authoring system was driven by the needs of the intended authors: subject matter experts rather than programmers. The authoring system is template-based and allows authors to produce on-line dialogue programs quickly by filling in the 'holes' of the templates. The templates, which are data driven programs coded in the TOPIC language, allow an author to build dialogues by specifying the data particular to an interaction. In CAI common types of templates are true/false questions, multiple choice questions, and tests. In addition the authoring system can be extended by the addition of templates written by a programmer with some familiarity with the IWB TOPIC language. This robustness has enabled the authoring system to be useful in areas outside of CAI such as on-line help systems.

The presentation will also discuss other features, such as computer-managed instruction, that make IWB more than just a CAI system.

Comment

This is brand new and is not yet a product. I think that it looks very good.

10.34am **What is happening at Pyramid**
Robert Reglan-Kelly, Pyramid

I am afraid I missed this talk.

☛ Coffee ☚

11.22am **UNIX in Australia, 1984**
John Lions, University of New South Wales

John started by talking about the history of UNIX in Australia. He then spoke of some of the work which Tim Long has done in making the file system faster. This involves keeping the inode list of free disc blocks in ascending order; varying the look-ahead from between 1 and 32 blocks depending on experience; and finally coalescing requests for I/O into one large request. John then spoke about the scheduling system which Sydney University uses to get 100 students working on a VAX. The mechanism is based on the user having a share of the resources in the machine; the system is called MUSH and is where UCB got their disc quota system from.

The Australian alternative to the *uucp* network is called ACSNET. This was originally called the Sydney UNIX Network, or SUN; but unfortunately this name has been stolen by another company in the UNIX world. The name is still retained for the software. ACSNET is a message passing service involving multistage transfers but with no explicit routing. We are promised more on the system when Piers Lauder comes to Paris.

John noted that there was a general movement from DEC hardware in Australia; he mentioned that Melbourne was very happy with the Pyramid.

John finished which a few slides: one was of a wallaby reading the Kernighan/Pike Book - and very fetching it was too.

11.55am **UKUUCP and other EUUG software**
Lee McLoughlin, Westfield College, University of London

I have the benefit of Lee's overhead projector slides for this - Ta.

UKUUCP is a single *uucp* combining the best features of the dozen or so *uucp* versions found in the UK. It includes work done by Mike Bayliss, Tony Luck, Jeff Smith, and Lee himself. UKUUCP includes hooks for dial-in/call-out on the same line, on V7 and 4.2BSD and has loads of bug fixes. Its performance is enhanced by 40-fold by improving the scanning of queues.

The code should port easily to other systems. It is currently running on VAXes, PDP 11s, LSI 11/23, various 68000s and the High Level Hardware Orion. The code is running in the following UNIX systems: V7, 4.1/4.2BSD, System III and System V.2. The code has failed on GEC 63's and Perqs.

The main claim to fame is that it can transfer over very simple connections, such as PADs, GECs, PRIMEs, and other systems which provide only tty facilities for networks.

UKUUCP needs York Box support (this is underway); X.25 support as in System III/V; and many other things.

The code is available on the EUUG tape. You require a V7 license (or better).

12.11 **EUUG business meeting**
Emrys Jones, EUUG

The main business was to report that there had been one written submission about the proposed constitution with syntactic alterations. The constitution will be sent to all members for a postal ballot in the near future.

Actually, there is also one matter of report which Emrys did not mention but which I thought I would slip in here. At the Committee meeting in Nijmegen, it was decided to award an honorary membership to Alan Mason in recognition of his work in laying the foundations of the EUUG.

2.30 **Speech input and UNIX**
R. M. Johnstone, University of Glasgow

Abstract

This work investigates the performance of speech input in certain application areas.

The preliminary work involved selecting a task which seems particularly well-suited to use of speech input, and building a speech system around the task (using a UNIX host), following principles derived from previous work in the area). Low-level recogniser functions are controlled by a set of simple C programs. Vocabularies are stored as UNIX text files, for re-use in the next speech session. The facilities of *emacs/ mlisp* were of considerable use in implementing and maintaining dialogues for particular tasks.

Initial experiments involved the experimental manipulation of task variables (e.g. operator experience, system training procedure, vocabulary composition), and certain recogniser parameters. This type

of work aims to produce guidelines for applying speech recognition, which will enable a system to be optimally tuned.

Our current project looks at how well speech recognition can replace keyboard input. Once again, our answer will depend on a large number of variables, including user cooperativeness and experience, and characteristics of the particular task. Candidate tasks include Pascal programming and *nroff* command insertion.

2.57pm **Measuring disc I/O on the VAX**
Nick Nei, University of Glasgow

Abstract

This paper describes a project under way at Glasgow University to gather statistics about disk performance on a VAX running Berkeley UNIX 4.1. These results will be used to construct a stochastic model for the behaviour of the disk subsystem. We hope that by modifying the parameters on the model and studying the results we can discover new ways of improving the disk and file system performance.

Comment

Nick was rushed when he talked about this at Nijmegen and was given more time to present the results again. The main idea was to measure disc performance in order to find useful measures of spread and central tendency of disc traffic, i.e. the mean arrival of requests for disc transfers and the pattern of requests. Armed with the mathematical model, it should be possible to predict future trends and generate reconfiguration forecasts.

The measurements taken were at the start of the *strategy* routine, which is the point where UNIX says 'OK here's something to do on the disc' to the relevant disc driver.

The measurements showed a generally exponential distribution but there are two interesting peaks: one at a few micro seconds and a one at 20ms. Why these peaks exist is still being investigated.

3.20pm **sndawk - a signal processing language**
Dan Timis, Institut de Recherche et Coordination Acoustique/Musique, Paris

Abstract

Signal processing and sound synthesis often use pipelines of programs performing specific treatments as filtering, changing the sampling rate or the gain etc. on binary floating point samples. Users who want to make their own algorithm will spend sometimes, for a very simple thing, time and energy to write, to compile and to test a program in C or Fortran.

Sndawk (sound *awk*) provides an interpretative signal processing programming language simple to learn and to use. Inspired from the well known pattern scanning and processing language *awk*, it respects much of its syntax as it respects much of the syntax of C.

This talk will discuss the structure and use of *sndawk*.

☛ Tea ☚

4.30pm **Pontifications, Accusations, Prognostications & Mystifications**
Chaired by David Tilbrook, Imperial Software Technology

Abstract

This session will be an open forum for discussion of UNIX and its future. Each panelist will make a 3 minute presentation on their views and prejudices, after which the floor will be open for questions,

comments and discussion.

Comment

This session is impossible to take notes on and luckily Richard Stibbs had organised a tape recorder to record all the important noises with. I had not thought of that and will certainly do it again, — thanks for the idea, Richard.

The session started with a couple of small presentations. The first was introduced by David Tilbrook.

"I don't want to embarrass people by asking: who has an illegal copy of John Lions' book?† But when I put out the advanced notice saying *The much-xeroxed John Lions*, I got an enthusiastic response. It was a remarkable book, we owe him a lot, and we would like to give him a small token of our appreciation in lieu of royalties." The present was a pint beer mug bearing the arms of John's college, engraved with *To John Lions, in appreciation from the EUUG, Cambridge, September 1984.*

David was then presented with a large pair of big red inflatable lips by Jim McKie, who said: "There is someone here who has done a lot of work for this conference and it's often been said that he has the biggest mouth in the UNIX world - so here's the biggest lips to go with it."

The proceedings then really started with each member of the panel making a three minute presentation.

Armando Stettner - DEC

"From its beginning, UNIX could never be considered a 'standard' operating system, or rather it has always been considered to be evolving. Once UNIX allocated its first inode it started to evolve. Its evolution was in the hands of a very few people; beginning with a small group in Bell Labs, Murray Hill and moving onto small groups in Universities around the world. There were other groups inside Bell Labs who were interested in turning UNIX into a tool for research and development of applications. Berkeley got into it, and did the right kind of things with 3BSD and 4BSD. Or at least in the early days between 3BSD and 4.1, they did the kind of things which were needed.

Now we're into new era, UNIX in a larger sense will no longer be driven, or certainly no longer controlled, by purely technical and research people. Thankfully, I believe, 4BSD will continue to evolve along the path that makes sense.

Perhaps unfortunately, UNIX evolution is now in the hands of the corporations, IBM, Perkin-Elmer, AT&T, (*DEC? from the audience*), DEC, yes. These people will be driven by their perceptions of their customer's perceptions of what they think they need. I can only hope that these new features and capabilities, and the functionality that the companies implement will fit into the framework and architecture of UNIX. I feel that this is part of the job of every UNIX guru and wizard who work for these companies that sells or distributes UNIX. Hopefully, the UNIX architecture will also evolve to facilitate its use on new technologies and new ways of building systems.

With the new kids on the block, the corporations, I don't know where UNIX family is going. I can only hope that it will evolve and those of us who are involved in its evolution will keep an open mind for new things and new values."

† John Lions produced a book, or rather two books, describing the workings of UNIX V6. They were aimed at teaching undergraduates about the internals of the operating system but ended up training nearly all the UNIX hackers who existed at that time. AT&T sat on the book, only allowing one copy per site. To prevent any repetition of the incident, licencees of V7 were prevented from teaching the operation of the kernel.

John Lions, University of Sidney

John started his talk with a visual demonstration of C. P. Snow's law: *every culture is composed of two subcultures.* This doesn't translate easily onto the printed page. He wanted to talk about the small group of people

"... who can carry forward the true UNIX tradition. Which is not only to take two steps forward all the time. But occasionally to take one step backwards; and not only to add a few new features to the system; but also to remove some redundant, over-grown, over-ripe features from time to time. I think that if we all took a vow to declare 1985 the Year of the Pruning Saw and spend a lot of time cutting back, removing things which aren't really needed; then UNIX may thrive. Without this, it will grow and nobody can do what Ken Thompson and Dennis Richie did for so long, namely, cut things back. If the unimpeded growth is allowed to continue, then the UNIX tree will die in the foreseeable future."

Tom Killian, AT&T

"It says on my badge that I am from AT&T but I would like to issue a disclaimer: *The opinions expressed are those of the speaker and do not necessarily reflect those of AT&T, its lawyers, cupholders, inquisitors, or anybody else for that matter.*

Some of the questions brought up recently at this conference have been to do with the question 'What is UNIX?', and I propose a number of possible answers to this.

- A trademark of AT&T Bell Laboratories.
- It ought to be a trademark of Brian Kernighan, since he came up with the name.
- Is it a kernel?
- Is it a set of standard utilities?
- People watching this conference from the outside are probably convinced that it is a cult subculture.
- It's almost certainly a state of mind.
- In some sense, it's also what runs on Dennis Ritchie's machine.

I think the UNIX philosophy is summed up in something which was written a long time ago by William of Occam, I am permitted to quote it in Latin, since this is Cambridge (*he then quoted it in English*†): 'Objects should not be multiplied unnecessarily'. This is something which has guided UNIX from the beginning. There were a very large number of decisions which were made early on by Ken Thompson and other people in his office writing things on the blackboard. By and large, the decisions were right and it's very dangerous if you go in and mess with them. Such things as the file system and the simple kind of scheduling were very crucial to making UNIX as successful as it was on the machines which were available at that time. Obviously, some these things are going to have to evolve for UNIX to remain viable. But I think that the philosophy that you have a minimal spanning set of necessary operations is very important. If we lose sight of this UNIX will be cooked."

Mike Karels, UCB

"First of all, I will try to describe what I see of where the system is going and how we are trying to get there. Then I shall pick up on what the previous speakers have said.

The model of the computing environment that we were getting looking at in Berkeley 4.1 and which formed the major driving factor in designing 4.2, was that the single machine timesharing system was probably not going to be the way of the future. We are looking at workstations connected by a network, most probably ARPANET, and hopefully to the rest of the world on long-haul nets. So there are a lot of communications facilities and networking in 4.2. To parallel that, there are interprocess

† He put it on the overhead projector in Latin and whipped it away before I had a chance to copy it.

communication facilities. [*Tape garbled*] The new directions in 4.2 come from that. Although there are communications facilities, there aren't any really distributed facilities in existence. I don't think UNIX ever will be a distributed operating system, this is tempting nature. On the other hand, UNIX can provide facilities of a distributed nature. This is something which is just getting going.

Another strong feeling that I have about UNIX is that 4.2 has gotten to the point of being a very large complicated beast, not only in terms of user facilities but also in terms of the way things are done inside the kernel. One of my long goals has been to redesign the kernel unifying things. Most of this would be invisible from the outside, but its one of the things I'de love to spend a few years on."

Teus Hagen, CWI

"Some time ago, we had languages, and we started to say that we must standardise those languages so that we could port programs between systems. So, Fortran was a big effort and after a while the Americans discovered that structured languages were a good idea, and so we had C. C has become a little standardised now.

UNIX helped a lot in standardising operations on the machine. I don't think that a lot of people involved in 'standardising operations' can make something which is portable.

4.2 was a lot of work, mainly in the area of networking and I think that's one of the first problems. The networking implementation is very young, it's one of the first so it has to be changed. The resulting system will not be UNIX any more. We have to wait until somebody pops up doing real networking. So, I think that in a little bit, UNIX can withdraw; and in the end we will have a real networked system. But I don't know what that will be."

Steve Bourne, DEC

"At Silicon Graphics, I was considering how to make our business decisions safe against the vagaries of all the different technologies and groups. There are lots of different versions of UNIX out there, and if you were a small company like we were, which version do you chose? Well, of course, what you have to look at, is who the big guys are; or at least, who is likely to be leading the charge. What you have to look at is AT&T, IBM, DEC and (Berkeley). I've put Berkeley in parentheses because they are not a large corporation, or at least not yet, but they are clearly a major force in making technological advances in UNIX.

I am not sure what you should conclude from it. I've spent some time at this conference nattering about how AT&T and IBM are facing off; and how AT&T are yet to have a well oiled marketing machine. Well, anyway so that was one question which I was interested in.

The answer, of course, isn't very interesting. We must hedge our bets. What I mean by this is that we try to chose the subset of UNIX which we use so that programs would port. There isn't much help for porting in terms of tools to look at libraries, and to look at porting difficulties; telling you whether your program is going to port from one place to another.

The question is should we accept divergence? UNIX never legislated anything, the group which made it never legislated anything. There are two more questions: can we as a group continue to make the right decisions and secondly, if we can, can the large group continue to make progress?"

General discussion

David then opened the discussion up to the floor. Sorry folks, you aren't going to get a verbatim transcription of that for two reasons: first, I don't have the time. Secondly, the tape isn't exactly hi-fi quality and some of the floor contributions are totally lost in crackles and hiss. Much like conversation on the UK telephone system these days.

Mike O'Dell kicked off the discussion with a longish statement (most of which is lost). His main thesis was that UNIX makes a number of powerful statements about what one might reasonably expect in an environment where programs are developed. It managed to do this by not writing a paper about it but by implementing it. He then moved on to say that people seem to think that putting a C compiler onto a system imparts some of the magic of UNIX. It doesn't, a bad system with a C compiler is just a bad system on which you might be able to program in C.

The discussion then moved onto the marketing and standards in UNIX. There was general assent to the idea that restraining developments by the imposition of 'standards' was a bad thing. UNIX will always alter to fit the environment in which the system is to be run, whether this is done by companies or universities. The main hope is that whatever is done to the system is done well. One voice from the audience did plead for a single standard, even if it is a bad one, he saw that the problem was that there were too many dialects. A number of voices were raised against this view, the argument seems to be between those who think UNIX is an evolutionary process and believe that 'official' standards will inhibit this; and those who want and need a stable base for the development of applications software.

The question was raised as to whether you should run System V or 4.2 on a VAX? Most people in the audience said 4.2 - perhaps that says something about the audience. David then asked whether we should all be trying to run Edition 8 rather than 4.2. John Lions said that if Rob Pike was to be believed: the file system throughput under Edition 8 is considerably greater than under 4.2; the kernel is more compact; and line disciplines are a reasonable alternative to sockets. Cries of 'not true' from Mike Karels.

John Lions then moved the conversation onto the availability of bit-mapped displays. There was a general feeling that the cost would reduce to a level where they were affordable in the same way as a standard VDU is today.

Discussion moved onto how network file systems should be implemented. Tom Killian said that the strength of method where remote files were joined to the local file system as part of the tree was that all the current binaries just work, the *namei* routine in the kernel just works harder. Mike Karels endorsed this.

There was then a lot of discussion, with not many new points. David wrapped up the conference and we finally got to see the video tape of Greg Chesson's flight simulator system.

Endpiece

We did, as usual, have space for writing silly comments on a blackboard. The idea was for people to write down the error message which they found of least use excluding the '?' and 'TMP' messages from *ed*. We got the following list, I leave it as an exercise for the reader to can check whether they are true or not.

local symbol botch — V7 *ld*.

ERK! - (V6 *passwd* & V7), we had a later spelling correction to URK!

oops — *termlib*.

Bailing out at line ... — *awk*, especially when the ... is replaced by 1.

eh? — *chess*.

Values of B will give rise to Dom — V6 *mv*. This is my favourite.

Invalid keyword "else" — C compiler.

Very funny — V6 *pr*.

argc=2 — V6 *comm*.

Clock may be set wrong — *sccs get*.

mo40 — C compiler.

Intruder alert — 4BSD *whoami*.

Jackpot — V6 *diff*

Room must be Cory or Evans — 4BSD *chfn*.

Termination code 132 — 4BSD *f77*.

ERROR
Illegal character 104 (octal) — *pcc*.

Lints little mind is fried — V7 at least. I checked this on 4.2, and the correct coding for that system is 'Lints little mind is blown'.

No Toy clock — V6 *date*.

Modify failed — *csh*.

Syntax error near line 1 — *awk* for a 1-line program.

Mere mortals mustn't use that mantra — *sendmail*.

What? — *quiz*.

Well, that's it. This file is now about 70000 bytes, which is too long. Thanks to all who made the conference work.

USENIX Conference Proceedings Available

Proceedings for the following USENIX conferences are available from the organizations listed. California residents please add applicable sales tax. Payments must be in US dollars payable on a US bank.

Salt Lake City — Summer 1984, and Toronto — Summer 1983

Copies of the proceedings of the Salt Lake City Conference are available for $25 per copy, and of the Toronto Conference for $30 per copy. Add $15 per copy for overseas postage. Send your check or purchase order to:

USENIX Association
P.O. Box 7
El Cerrito, CA 94530

Payment must be received before proceedings will be shipped.

Washington DC UniForum Conference — Winter 1984

Copies of the proceedings of the UniForum Conference are available for $30 per copy, plus $20 per copy for overseas postage. They may be ordered from:

/usr/group
4655 Old Ironsides Drive, #200
Santa Clara, CA 95054

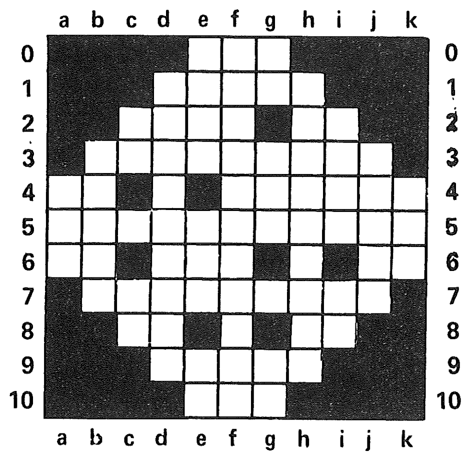
San Diego UNICOM Conference — Winter 1983

Copies of the proceedings of the San Diego UNICOM Conference are available for $25 per copy, plus $15 per copy for overseas postage. Send your check or money order to:

Software Tools Users Group
1259 El Camino Real, #242
Menlo Park, CA 94025

The First BiMonthly UNIX Crossword Puzzle

Peter Langston



across

- e0 On a PDP-11 test the N & V condition codes; if they are either both clear or both set then replace the PC with PC + (2 × offset).
- d1 What a UNIX process does when it wants a clone.
- c2 Prunus Spinosa (What does this have to do with UNIX? Well, some say it sounds like a description of 4.2 BSD.)
- h2 How does it hyphenate words?
- b3 Many people's unflattering idea of what "User Friendly" means.
- a4 UNIX's travel command.
- f4 Filthy, foul, wretched, vile, base, morally degraded, grasping (e.g. the details of TSO's workings).
- a5 "___ touch someone" (3 wds)
- a6 The more communal counterpart to UNIX RT.
- d6 Netnews caters to many appetites, but net.___-radio has no recipes.
- j6 How to convert and copy a file.
- b7 What UNIX is.
- c8 Control-M.
- h8 See b7 across.
- d9 UNIX Version in Jan. 1990 (if things keep going at the current rate).
- e10 A place where your ttys are kept.

down

- a4 High-tech device made economical by mass production for entertainment.
- b3 File that defines legal BLICN destinations (careful!).
- c2 Switch source file.
- c7 Important shell script that no user ever runs.
- d1 Largely a relic of the batch operation days.
- e0 *termcap* capability type.
- e5 The bigger Morse symbol (backwards).
- e9 Program whose "l" command mishandled DEL in version 3.0.
- f0 "Reach out and ___" bumper sticker (2 wds).
- g0 Your option to set erase and kill characters to # and @ in BSD.
- g3 Really obscure UNIX acronym (sorry).
- g9 Entertainment from a4 down.
- h1 Advantage to having a command in */bin* (2 wds).
- i2 A popular culture role model for UNIX wizards.
- i7 The way to cure disk space problems.
- j3 From */usr/goofball* print all files under */usr/goofball/k* with: "___ -print" (2 wds)
- k4 Why AT&T loves *uucp* & netnews.

Netnews

I have reproduced below some of my network mail and a few "netnews" articles that I thought may be of interest to Australian UNIX users. I have deleted some of the less meaningful data generated by various mailers and news programs. No responsibility is taken for the accuracy (or lack thereof) of anything below.

```
From gross@DCN9.ARPA Fri Oct 26 12:59:16 1984
Date: Fri, 26-Oct-84 12:59:16 AESST
Newsgroups: net.mail.msggroup
Subject: Estimate on number of Internet users
Sender: ka@hou3c.UUCP (Kenneth Almquist)
Date-Received: Wed, 31 Oct 84 09:45:22 AEST
Lines: 76
To: msggroup@brl.ARPA
```

I thought the message below might be of interest to arpanauts. It originated in the 'Human-Nets digest' (which I don't read) and was forwarded, with comments, to 'AIlist' by Ken Laws at SRI. (Law's added comments are in square brackets.) Any comments?

```
Date: Wed, 10 Oct 84 01:50:10 edt
From: bedford!bandy@mit-eddie
Subject: Net Readership
```

[Forwarded from the Human-Nets digest by Laws@SRI-AI.]

> Date: Mon, 8 Oct 84 14:28 EDT
> From: TMPLee@MIT-MULTICS.ARPA
>
> Has anyone ever made an estimate (with error bounds) of how many people have electronic mailboxes reachable via the Internet? (e.g., ARPANET, MILNET, CHAOSNET, DEC ENET, Xerox, USENET, CSNET, BITNET, and any others gatewayed that I've probably overlooked?) (included in that of course group mailboxes, even though they are a poor way of doing business.)

Gee, my big chance to make a bunch of order of magnitude calculations.... [...]

USENET/DEC ENET: 10k machines, probably on the order of 40 regular users for the unix machines and 20 for the "other" machines so that's 100k users right there.

[Rich Kulaweic (RSK@Purdue) notes 15k users on 40 Unix machines at Purdue, with turnover of several thousand per year. -- KIL]

BITNET: something like 100 machines and they're university machines in general, which implies that they're HEAVILY overloaded, 100-200 regular active users for each machine - 10k users.

[A news item in the latest CACM mentions 200 hosts at 60 sites, soon to be expanded to 200 sites worldwide. A BITNET information center is also being developed by a consortium of 500 U.S. universities, so I expect they'll all get nodes soon. -- KIL]

Chaos: about 100-300 machines, 10 users per machine (yes, oz and ee are heavily overloaded at times, but then there's all those unused vaxen on the 9th floor of ne43). 1k users for chaosnet.

I think that we can ignore csnet here (they're all either on usenet or directly on internet anyway...), so they count for zero.

ARPA/MILNET: Hmm... This one is a little tougher (I'm going to include the 'real' internet as a whole here), but as I remember, there are about 1k hosts. Now, some of the machines here are heavily used (maryland is the first example that pops to mind) and some have moderate loads (daytime - lots of free hardware at 5am!), let's say about 40 regular users per machine -- another 10k users.

I dare not give a guesstimate for Xerox.

[Murray.PA@Xerox estimates 4000 on their Grapevine system. -- KIL]

So it's something on the order of 100k users for the community. [...] Well, it could be 50k people, but these >are< order of magnitude calculations...

[Mark Crispin (MRC@Score) notes that there are 10k addressable mailboxes at Stanford, but that the number of active users is perhaps only a tenth of this. Andy's final estimate might be inflated or deflated by such a factor. -- KIL]

Now that I've stuck my neck out giving these estimates, I'm awaiting for it to be chopped off.

andy beals
bandy@{mit-mc,lll-crg}

```
From ag5@pucc-i Sat Oct 27 15:20:33 1984
Date: Sat, 27-Oct-84 15:20:33 AESST
Newsgroups: net.mail.msggroup
Subject: Re: Estimate on number of Internet users
Date-Received: Wed, 31 Oct 84 08:28:25 AEST
Organization: The Restaurant at the End of the Universe
Lines: 14
```

<<let's fuel the fire here...>>

Uh, we forgot one BIG sector of Internet users (or, perhaps maybe these folks aren't considered Internet users... let me know). IBM's internal net, VNET (which is accessible through BITNET), has over 1500 hosts worldwide... Surely this accounts for a *large* number of users...

```
----------------------------------------------------------------------
Henry C. Mensch  | User Confuser |  Purdue University User Services
{ihnp4|decvax|ucbvax|purdue|sequent|inuxc|uiucdcs}!pur-ee!pucc-i!ag5
{allegra|cbosgd|hao|harpo|seismo|intelca|masscomp}!pur-ee!pucc-i!ag5
----------------------------------------------------------------------
                  "Hit me with your laser beam!"
```

```
From schoff@cadtroy.UUCP Tue Oct 30 11:18:11 1984
Date: Tue, 30-Oct-84 11:18:11 AESST
Newsgroups: net.mail.msggroup
Subject: Re: Estimate on number of Internet users
Date-Received: Fri, 2 Nov 84 01:23:37 AEST
Organization: CADMUS [Troy Office], Troy NY
Lines: 10
```

How about Mailnet? They supposedly have 475 universities and colleges. Here at RPI the access is through a 3081K with over 200 displays and a user population of in excess of 3000. Admittedly though they have initially restricted access to faculty and staff. Their gateway is MIT-MULTICS.ARPA.

marty

cadmus!schoff@seismo.ARPA
{linus,bbncca,wivax,seismo}!cadmus!schoff

```
From vance@mtxinu.UUCP Wed Oct 31 13:44:19 1984
Date: Wed, 31-Oct-84 13:44:19 AESST
Newsgroups: net.unix-wizards,net.bugs.4bsd
Subject: 4.2 BUGLIST FROM MT XINU
Date-Received: Sun, 4 Nov 84 21:09:09 AEST
Organization: mt Xinu, Berkeley
Lines: 100
```

GENERAL INFORMATION ON THE 4.2 BUGLIST FROM MT XINU

--IMPORTANT DISCLAIMERS--

Material in this announcement and the accompanying reports has been edited and organized by MT XINU as a service to the UNIX community on a non-profit, non-commercial basis. MT XINU MAKES NO WARRANTY, EXPRESSED OR IMPLIED, ABOUT THE ACCURACY, COMPLETENESS, OR FITNESS FOR USE FOR ANY PURPOSE OF ANY MATERIAL INCLUDED IN THESE REPORTS.

MT XINU welcomes comments in writing about the contents of these reports via uucp or US mail. MT XINU cannot, however, accept telephone calls or enter into telephone conversations about this material.

Legal difficulties which have delayed the distribution of 4.2bsd buglist summaries by MT XINU have been resolved and three versions of the buglist are now available.

The current buglist has been derived from reports submitted to 4bsd-bugs@BERKELEY (not from reports submitted only to net.bugs.4bsd, for example). Reports are integrated into the buglist as they are received, so that any distributions are current to within a week or so.

Buglists now being distributed are essentially "raw". No judgment has been passed as to whether the submitted bug is real or not or whether it has been fixed. Only minimal editing has been done to produce a manageable list. Reports which are complaints (rather than bug reports) have been eliminated; obscenities and content-free flames have been eliminated; and duplicates have been combined. The resulting collection contains over 500 bugs.

Three versions of the buglist are now ready for distribution:

2-Liners:
: Two lines per bug, including a concise description, the affected module, the submittor. Approximately 55K bytes, it is being distributed to net.sources con-

currently with this announcement.

All-but-Source:

All material, except that all but the most inocuous of source material has been removed to meet AT&T license restrictions. Nearly a mega-byte, this will be distributed to net.sources in several 50K byte pieces later this week.

A paper listing or mag tape is also available, see below.

Please note that local usenet size restrictions may prevent large files from being received and/or retransmitted. MT XINU will not dump this material on the net a second time; if your site has not received material of interest to you within a reasonable time, please send for a paper or tape copy.

All-with-Source (FOR SOURCE LICENSEES ONLY):

4.2 licensees who also have a suitable AT&T source license can obtain a tape containing all the material, including proposed source fixes where such were submitted.

Once again, MT XINU has not evaluated, tested or passed judgment on proposed fixes; all we have done is organize the collection and eliminate obvious irrelevancies and duplications.

A free paper copy of the All-but-Source list can be obtained by sending mail to:

MT XINU
739 Allston Way
Berkeley CA 94710

attn: buglist

or electronic mail to:

ucbvax!mtxinu!buglist

(Be sure to include your US mail address!)

For a tape, send a check for $110 or a purchase order for $150 to cover MT XINU's costs to the address given above (California orders add sales tax). For the All-with-Source list, mail us a request for the details of license verification at either of the above addresses.

```
From minitab@uwstat.UUCP Sun Nov 11 01:40:27 1984
Date: Sun, 11-Nov-84 01:40:27 AESST
Newsgroups: net.dcom,net.lan
Subject: Re: UNIX to IBM Communication Package, UREP, and BITNET
Date-Received: Thu, 8 Nov 84 04:17:57 AEST
Organization: U of Wisconsin Statistics
Lines: 55
```

..... some text deleted -Ed......

> What exactly is BITNET? How do you get on it? How does it work?

BITNET is a quasipublic network of University computer systems and IBM labs that grew mostly by weight of its own utility from a nucleus of CUNY (City Univ. of New York), Yale, and Penn State. It now includes more than 300 nodes at more than 100 institutions in the U.S., Canada, Europe and Isreal. (Strictly speaking, only the U.S. portion is BITNET. The Canadian sites are called NETNORTH, and Europe/Isreal is EARN, the European Academic and Research Network.) BITNET grew so fast because it is easy and inexpensive to join, and communications are free for users; all expenses are born by sites as part of overhead. To join, a university must pay for the telecom equipment and lines to an existing node, and agree to let a new site attach to its own computer in the future. A grant from IBM now sponsors a network operations center at CUNY, and another IBM grant to EDUCOM (Consortium for Interuniversity Communication, Princeton NJ) helps promote the network.

The network uses the RSCS protocol, native to IBM mainframes running VM, and most nodes were originally VM sites. Now a wide variety of machines are on BITNET, emulating either RSCS or NJE nodes. (NJE is Network Job Entry, the native peer-to-peer networking protocol for IBM mainframes running MVS. All RSCS systems can talk the NJE protocol. And you thought IBM machines used SNA? That's to talk to terminals, not other computers!) About 20% of the nodes on the net today are VAX systems running the jnet RSCS emulation package, from Joiner Associates, Madison WI. Another few dozen nodes run UREP, the UNIX RSCS Emulation Program from Penn State, mentioned above. There are also Sperry Univac 1100s and CDC machines emulating NJE nodes, and a few others.

The RSCS protocol provides store-and-forward file and mail transfer, and real-time network commands and interuser messages. Other services (such as remote login and remote file access) have been built on top of these. The physical medium is usually leased lines at 9600 baud (RSCS is a member of the bisync family of protocols), but IBM machines use channel-to-channel adapters locally, and VMS systems running jnet use DECnet connections over any distance. The use of leased lines provides very fast service for the real-time messages, and files and mail are delivered promptly, though they can be queued for short periods at busy nodes like CUNYVM, the network hub.

BITNET is a part of the Internet (the "galactic network"), and users on the BITNET can be addressed from ARPA, UUCP, MAILNET, CSNET and CCNET via gateways. For example, to get information about UREP from the ARPANET, send mail to

```
        DAE%PSUVAX1.BITNET@WISCVM.ARPA
```

Steve Arnold, Joiner Associates Inc.
ARNOLD%WISCPSLB.BITNET@WISCVM.ARPA (ARPA)

```
From dmr@research.UUCP Sat Dec  8 06:45:09 1984
Date: Sat, 8-Dec-84 06:45:09 AESST
Newsgroups: net.unix-wizards
Subject: dsw: pdp7 memorabilia
Date-Received: Tue, 11 Dec 84 19:33:22 AEST
Lines: 56
```

I happened to dredge up an old notebook and found a listing of the PDP-7 version of dsw. Because several people have approached me recently about reviving a version of PDP-7 Unix as a sort of paleontological exhibit, and because the subject has been discussed here, I thought people might be interested in seeing the code. I first considered net.sources, but decided not to carry whimsy too far.

Dennis Ritchie

Notes:

1) The assembler has Knuth-style temporary labels but no literals.

2) The name of the current directory was evidently ".."

3) Formatting is faithfully reproduced.

4) "sys save" makes a core image.

```
" dsw

   lac djmp
   dac .-1
   oas cla
   cma
   tad d1
   dac t1
   sys open; dd; 0
1:
   lac d2
   sys read; dir; 8
   sna
   sys exit
   lac dir
   sna
   jmp 1b
   isz t1
   jmp 1b

wr:
   lac d1
   sys write; dir+1; 4
   lac d1
   sys write; o12; 1
   sys save
do:
   sys unlink; dir+1
   sys exit

d1: 1
d2: 2
o12: 012
t1: 0
djmp: jmp do
dd: 056056; 040040; 040040; 040040
dir: .=.+8
```

```
From: kre:munnari (Robert Elz)
Received: by munnari.OZ (4.3)
        id AA27249; Sat, 5 Jan 85 04:21:35 EST
To: peteri:elecvax
Subject: this news belongs in auugn ...

From: mulga!decvax!genrad!mit-eddie!godot!harvard!seismo!mcvax!vu44!botter!klipper!biep
Article 33 of net.news:
>From: biep@klipper.UUCP (J. A. "Biep" Durieux)
Newsgroups: net.news
Subject: Re: A Modest Proposal
Date: 2 Jan 85 05:57:13 GMT
Date-Received: 4 Jan 85 12:44:46 GMT
Organization: VU Informatica, Amsterdam
Lines: 80

In article <6719@brl-tgr.ARPA>,
        geoffs@brl-tgr.ARPA (Geoffrey Sauerborn (TANK) <geoffs>) writes:
>In article <24@epsilon.UUCP> you write:
>>In article <393@klipper.UUCP> biep writes:
>>>>> From: schu@drutx.UUCP (SchulteSM)
>>>>> Subject: Re: A Modest Proposal
>>>>>
>>>>>
>>>>>
>>>>> q
>>>>>
>>>>>
>>>>> w
>>>>> .
>>>>> dfslkfl;
>>>>>
>>>>>
>>>>> lkfdg
>>>>> ;lfdkg;lka;dlgk
>>>>>
>>>>>
>>>>>
>>>>> ZZ
>>>>> agkfjlkadfgkj
>>>>>
>>>>>
>>>>
>>>>Yes, I agree. This is definately a proposal worth considering.
>>>>
>>>
>>>No, no, no!!!
>>>I know, it sounds nice, but believe me, it just doesn't work.
>>>Someone over here brought this up on our local network, and we've
>>>been discussing it over and again, and indeed, it sounds appealing,
>>>but when you work out the details you'll find lots of inconsistencies
>>>and problems. Don't try to reinvent the wheel, there are already
>>>enough other proposals on this net which ask for our attention.
>>>
>>Well, we've actually done it here, and it works out just fine. Of course,
>>we had to make certain minor changes like "DfXlkfl;:" instead of
```

```
>>"dfslkfl;" as was indicated in the original article. You'll note that
>>otherwise a contradiction arises when "ZZ" is implemented.
>>
>>                                                    Ed Sheppard
>>                                                    Bell Communications Research
>
>
>       Alright! I'm sick of this! Why is it that every time someone
>puts a good piece into public domain, somebody has to good and change
>the source! The next thing to happen it some BO-ZO will try to use
>this changed version without taking the time to read the documentation -
>and naturally the FOOL starts flaming to the original poster!!!!
>
>
>                                              Geoff Sauerborn
```

You all don't seem to get the point. Indeed, Ed, it is possible to adapt the thing to your local network, when all machines are compatible, none is running notesfiles, all are little-endian, and at least some of them are not feeding news to decvax, mcvax or purdue (and perhaps others, I've not yet got time to find out), and some other little things. *But not all sites of USENET do!!!* And what is happening then is that everybody starts making his local patches, and we end up even worse than we started. The whole mistake is due to the starting "q", which supposes a local "edsgr w534cb67835", since otherwise indeed "ZZ" goes wrong in boundary cases.

I must say I do not yet completely understand Ed's patch, it occurs to me that this only works because of some local changes to the rot13 algorithm. That can hardly be called "portable", can it?

Geoff, I think your argument goes wrong at the word "good".

I would say: quit the idea, the gains for sf-movie-lovers do not outweigh the burden for the rest of us.

--

Biep.

{seismo|decvax|philabs}!mcvax!vu44!botter!klipper!biep

I utterly disagree with everything you are saying, but I am prepared to fight to the death for your right to say it.

--Voltaire

```
From steveg@hammer.UUCP Sat Nov 24 02:37:38 1984
Date: Sat, 24-Nov-84 02:37:38 AESST
Newsgroups: net.usenix
Subject: Re: Portland Usenix info wanted
Reply-To: steveg@hammer.UUCP (Steve Glaser)
Date-Received: Sun, 25 Nov 84 00:29:01 AEST
Organization: Tektronix, Wilsonville OR
Lines: 23
Summary:

In article <cbosgd.504> mark@cbosgd.UUCP (Mark Horton) writes:
>What are the exact dates of the Portland Usenix conference? Have
>the hotel and persons running the conference been identified yet?
>(We have a baby due June 29, which makes the timing somewhat critical.)
>
>       Mark

Conference dates are

        June 12 thru 14, 1985. (tutorials on the 11th).

Conference hotel is the Portland Marriott. Format will be fairly close
to the Salt Lake Conference. Steve Bourne is program chairman. Steve
Glaser is general chairman (I guess - they never gave me an official
title). Exhibits are being handled by Industrial Presentations West
(same folks that did Salt Lake City).

The call for papers and such is expected to go out by the January meeting.

        Steve Glaser
        tektronix!steveg                (USENET)
        steveg.tektronix@csnet-relay    (ARPA/CSNET)
        (503) 685-2562
```

```
From mcvax!vu44!ast@decvax.UUCP:mulga Mon Dec 24 22:27:34 1984
Received: by mulga.OZ (4.3)
        id AA04199; Mon, 24 Dec 84 22:27:34 EST
From: mcvax!vu44!ast@decvax.UUCP:mulga
Return-Path: <decvax!mcvax!vu44!ast>
Received: by decvax.UUCP (4.12/1.0)
        id AA15206; Mon, 24 Dec 84 03:57:05 est
Received: by mcvax.UUCP; Mon, 24 Dec 84 01:11:35 -0100 (MET)
Date: Mon, 24 Dec 84 01:11:35 -0100
Apparently-To: decvax!mulga!elecvax.oz!johnl
```

I would greatly appreciate it if you could advertise the position described below in whatever way. Is there any Australia-wide network newsgroup it could be put into? If so, could you do that for me. I would greatly appreciate it. I have only asked you to put it on the network news, to avoid having N copies be submitted, so could you send me an acknowledgement quickly. If I don't hear anything within a couple of days, I will assume you are on vacation and ask someone else. Also spread the word locally. Thanks a lot.

===

POSITION AVAILABLE FOR PROGRAMMER IN AMSTERDAM

The Computer Science Group at the Vrije Universiteit in Amsterdam, The Netherlands, has has ongoing research projects in distributed operating systems, data bases, and artificial intelligence. The distributed systems project has a vacancy for a systems programmer to help with the design and implementation of the Amoeba distributed operating system. Amoeba is a capability-based operating system running on a collection of microcomputers (currently 68010s) connected by a 10 Mbps token ring. By late 1985 we expect to have 40-50 processors operational.

The work will consist of designing, implementing, and testing parts of the system, almost assuredly followed by redesigning, reimplementing, and retesting the parts until everyone is happy with it. Happiness will be achieved when the system is not only operational, but also well-structured, portable, easy to use, and lightning fast.

We are looking for a person who would like to come to Amsterdam for, say, one or (even better) two years. Candidates should have one or more of the following:

- a degree in computer science
- a good knowledge of C and UNIX
- experience with distributed operating systems
- experience with operating systems for small computers
- experience with local area networks

The Amoeba group numbers about 5 people, depending on how one counts, so a good systems programmer can really make a big contribution.

The department currently has 4 VAX 11/750s, 3 large PDP-11s, and a substantial number of microcomputers. The VAXes and PDP-11s all run UNIX. We are planning a major expansion of the computing facilities in the coming months, possibly with as many as 50 IBM PC-ATs, although the choice has not yet been made.

The university is housed in a modern building on the southern boundary of the city, seven minutes by rail from the airport. KLM and other airlines have direct flights to just about everywhere in Europe, and also many flights to the U.S., Africa, and Asia. Train and bus connections with most major European cities are also excellent, so this position should be especially attractive to someone who is interested in seeing the world. The position has 5 to 7 weeks of paid vacation time, the exact amount not being definite yet because the civil service regulations, which cover university employees, are currently being changed. The salary will be the same as the locals get, and depends on degrees, age, and experience. Starting date is open.

The university was founded just over 100 years ago and its charter speaks of the Christian ideal of service to God and man. The candidate need not agree with the charter, but should at least aware aware that many of the faculty members do. The university has been 100% financed by the Dutch government for decades.

I will be in Australia in February for the Computer Science Conference in Melbourne, and will also visit a number of universities in Melbourne, Sydney, and Adelaide. I would like to talk to any potential candidates then. I would also like anyone interested in the position to send me a curriculum vitae and a letter describing your background and experience. Please send that as soon as possible, as I am leaving for the Southern Hemisphere on Jan. 17. My usenet address is: ...mulga!decvax!mcvax!vu44!ast. My paper mail address is:

Andrew S. Tanenbaum
Wiskundig Seminarium
Vrije Universiteit
Postbus 7161
1007 MC Amsterdam, HOLLAND

Australian Unix[†] systems User Group 1985 Summer Meeting University of Wollongong February 11, 12.

The 1985 Summer Meeting of AUUG will be held at the Department of Computing Science of the University of Wollongong, Wollongong, N.S.W. on Monday February 11 and Tuesday, February 12, 1985.

The meeting will be devoted to topics of interest to the Unix community. The keynote speaker will be Richard Miller, recently of Human Computing Resources, Toronto, Canada, who completed the first port of Unix to a machine of another architecture.

This announcement provides registration and accomodation information for prospective attendees and speakers. It also constitutes the final call for papers for the conference.

Papers on and subjects related to the Unix system, or Unix-like systems will be considered for the meeting. An abstract should arrive at the University no later than last mail on Friday, 1st February. Electronic mail is preferred. Indication of intention by phone or mail is desirable as early as possible. The meeting will also host a display of equipment and software from various manufacturers.

Further information can be obtained from the following sources:

Information	**Name**	**ACSnet address**	**Phone**
Programme	Juris Reinfelds	juris:uowcsa	270859
Meeting	Ross Nealon	ross:uowcsa	270802
Exhibition	Gary Kelly	gary:uowcsa	270813

The general address for the meeting & submission of papers and fees is

The Department of Computing Science,
University of Wollongong,
PO Box 1144 Wollongong
New South Wales, 2500

The University is located at

Northfeilds Ave, Gywnneville, Wollongong

Other contact methods are

Phone (042) 270859
ACSnet address: auugm:uowcsa

† Unix is a trademark of AT&T Bell Laboratories.

Accommodation

Accomodation is at either International House (slightly off campus residence), possibly at Coolagong (as yet incomplete on campus residence), or at any city hotel/motel. A recommended hotel is the Northbeach International. Tariffs vary, and there should be no problem finding a hotel room, but the number of places at the residence is limited, so and advance booking is a good idea. Fees are ~$24 D,B&B, ~$20 B&B at International House, and begin at $59 single, $65 twin share at the Northbeach International Hotel. (Right on the beach and reasonably luxurious.) International House is ~15 minutes casual walk or 5 minutes fast run from the University. Parking is limited, but available and free. All of the hotels are further (minimum 30 minute walk from the University) away.

Transport

Wollongong is an easy 90 minute drive from the centre of Sydney, and since parking is free and plentiful on campus, driving from Sydney is reasonable. When approaching Wollongong, take the Mt. Ousley road, and follow it to the bottom of the mountain. When you near the bottom, you will see the yellow pentagon (site of the meeting), and a sign on the right marking the campus. Take the second exit ramp (near the overhead bridge) and just follow the signs. Morning and evening trains run, taking 90-120 minutes, depending upon stops. Disembark at North Wollongong (next to International House) if you wish to walk, or at Wollongong if you wish to catch a taxi. The bus service to/from the University is intermittent, and very strange. Don't count on it. From the airport in Sydney, there is a coach service (Watts) running to and from Wollongong 4 or 5 times daily. Cost is $8.60 one way, and will drop you in town near (or even at) the hotels. Wollongong Auto Rentals (271433) are probably the cheapest car rental firm. Wollongong has an integrated taxi system - Taxi Cabs (299311).

Registration

Registration will be from 9:00 until 10:30, on Monday 11th, in the Pentagon foyer. The meeting will commence at 10:30. Coffee, tea, fruit juice etc etc will be available. The University cafeteria and bistro are available for lunch and other uncatered for meals.

Conference Dinner

The dinner will be held on campus, on Monday, 11th in the Union, beginning at 7:30pm, and will comprise a four course dinner with wines, etc. The bar will be open during the dinner for those of you that wish. Cost will be $25. Feel free to bring wives, guests and others to the dinner, as the number of places available is potentially large.

General

Registration for the conference will be $50 on site, with a discount of $20 for early registrants. In order to qualify for this discount, the attached application form must arrive at the University by the last mail on Friday, 1st February.

Australian Unix[†] systems User Group
1985 Summer Meeting
University of Wollongong
February 11, 12.

Registration Form

Name ____________________

Address ____________________

Phone ____________________

Network address ____________________

Organisation ____________________

Text for name badge ____________________

Fees

Registration		$30	__________
Late fee		$20	__________
Dinner	__________	@$25	__________
Total			__________

Please reserve accomodation for a single/twin share at International House/Coolagong (if finished)/Northbeach for the nights of ____________________. Note that the residences do not have twin share arrangements. Payment for accomodation should be made directly to the residence concerned.

Australian Unix systems User Group

Summer Meeting

Febuary 1985

Hardware Display Questionnaire.

1 **Company Name.** ______________________________

Contact. ______________________ Phone No. ______________

2 **Space Required.** ($15/m^2) ______ m^2

3 **Power Requirements.**

Number of outlets.					______	
Total heat output.					______	BTU's or kJ's
Type of connector.	______	______	______	______	______	
Current capacity.	______	______	______	______	______	Amps.
Actual current used.	______	______	______	______	______	(Startup)
	______	______	______	______	______	(Running)

4 **What Software will you display?**

____

____

____

____

____

____

____

5 **What Hardware will you display?**

____

____

____

____

____

____

____

Australian UNIX* systems User Group Newsletter
(AUUGN)

Subscription Application

I wish to subscribe to the Australian UNIX systems User Group Newsletter and enclose payment of $_________ herewith for the items indicated below.

Signed ________________________________ Date ______________________

==

☐	One years subscription (6 issues) available on microfiche or paper	$30.00
☐	Back issues of Volume 1 (6 issues) available only on microfiche	$24.00
☐	Back issues of Volume 2 (6 issues) available only on microfiche	$24.00
☐	Back issues of Volume 3 (6 issues) available only on microfiche	$24.00
☐	Back issues of Volume 4 (6 issues) available on microfiche, some paper copies	$24.00
☐	Back issues of Volume 5 (6 issues) available on microfiche or paper	$24.00
☐	Subscribers outside Australia must add an extra to cover surface mail costs	$10.00
☐	Subscribers outside Australia must add an extra to cover air mail costs	$30.00

Name ____

Mailing address ______________________________________

____

____

Telephone number (including area code) ________________________

UNIX Network address ____________________________________

	YES	NO
I agree to my name and address being made available to software/hardware vendors	☐	☐

10/84

* UNIX is a trademark of AT&T Bell Laboratories

Australian UNIX* systems User Group
(AUUG)

Membership Application

I, ______________________________ do hereby apply for ordinary($50)/student(30)** membership of the Australian UNIX systems User Group and do agree to abide by the rules of the association especially with respect to non-disclosure of confidential and restricted licensed information. I understand that the membership fee entitles me to receive the Australian UNIX systems User Group Newsletter and I enclose payment of $______ herewith.

Signed ______________________ Date ______________

==

Name ______________________

Mailing address for AUUG information ______________________

____

____

Telephone number (including area code) ______________________

UNIX Network address ______________________

	YES	NO
I agree to my name and address being made available to software/hardware vendors	☐	☐

==

<u>Student Member Certification</u>

I certify that ______________________ is a full-time

student at ______________________

Expected date of graduation ______________________

Faculty signature ______________________ Date ______________

==

Office use only 10/84

* UNIX is a trademark of AT&T Bell Laboratories
** delete one

Peter Ivanov
AUUGN Editor
School of EE and CS
University of New South Wales
PO Box 1
Kensington NSW 2033
AUSTRALIA

+61 2 697 4042