EUUG

EUROPEAN UNIX® SYSTEMS USER GROUP NEWSLETTER

Volume 4, No. 1
SPRING 1984

EUUG

European UNIX† Systems User Group

Newsletter Vol 4 No 1

Spring 1984

† UNIX is a Trademark of Bell Laboratories.

Can You Afford Not To Be There?

UNIX SYSTEMS EXHIBITION 84

19 • 20 • 21 September 1984

Kelsey Kerridge Hall • Cambridge • England.

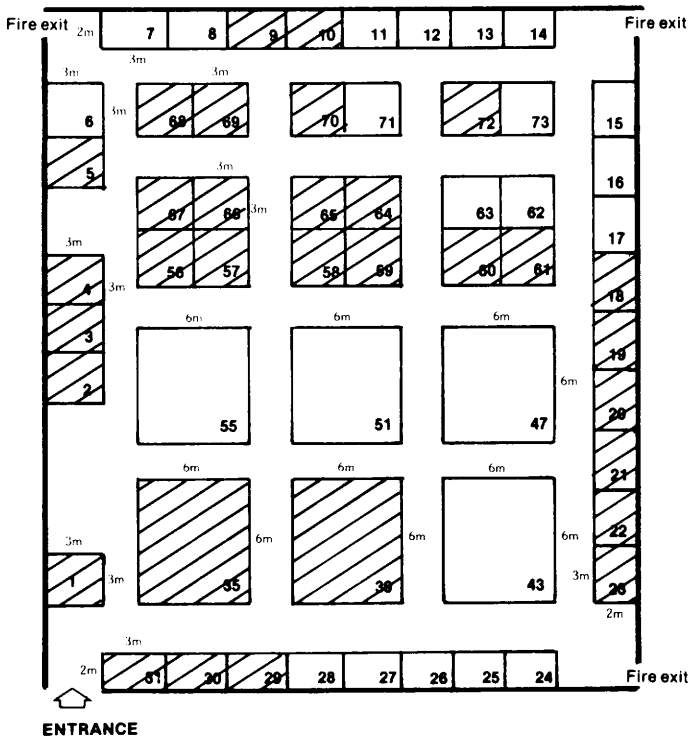
FACT
For the first time the U.K. will be hosting the European UNIX Users Group (EUUG)/USR/Group/U.K. Exhibition and Conference.

FACT
In only 3 weeks 50% of the total space has been allocated.

FACT
If you don't book space now you may find your Company left on the Computer Shelf.

FACT
Exhibitors can have all the Promotional Material they require.

FACT
Visitors can have as many free entrance tickets as they wish.



Space Allocated

PHONE, TELEX OR WRITE . . .

EXHIBITION SPACE

Mr. David Price
Network Events,
Printers Mews,
Buckingham,
MK18 1JX. U.K.
Telephone: 0280 815226
Telex: 83111

CONFERENCE INFORMATION

Mrs H. Gibbons
EUUG
Owles Hall,
Buntingford,
Herts,
SG9 9PL. U.K.
Telephone: 0763 73039

FACT
This is another show organised by Network Events.

EUUG
USR/GROUP

aab

Network EVENTS

You Need To Be There

Australasian Unix-systems Users' Group
1984 Winter meeting

The 1984 winter meeting of the AUUG will be held at Melbourne University on Monday August 27 and Tuesday August 28.

Keynote speaker for the conference will be Rob Pike, from AT&T Bell Laboratories, co-author, with Brian Kernighan, of the book *The Unix Programming Environment*.

This announcement is to provide early information for prospective attendees and speakers. It also constitutes a first, and only, call for papers for the conference.

Papers on any subject related to the Unix system, or any unix-like systems will be considered for the meeting. Prospective authors should send a detailed abstract (approximately a page) of their talk to Robert Elz, at the address below, before July 1.

Manufacturers desiring to display equipment should also contact the above as soon as possible to reserve space. Available space will be allocated on a first come first served basis, and those who request early will obtain prime positions. Manufacturers should forward details of space, power, and any other special requirements with their application. There will be a charge to manufacturers using this service. The charge will be based on the facilities allocated and will be set individually for each display.

A final announcement will be mailed to users receiving this mailing, and others who express interest, early in July. That will be due to be returned by registrants seeking early registration discounts, or reserved accommodation in University Colleges by the first of August. Further details will be provided in that announcement.

Further information is available from

Robert Elz,
Department of Computer Science,
University of Melbourne,
Parkville,
Victoria. 3052.

Phone: (03) 341 5225

SUN address: kre:munnari

European UNIX System User Group Meeting
Faculty of Science, University of Nijmegen, 16/18 April 1984

Peter Collinson
Secretary

Introduction

This was a very good conference, and if you missed it, then you missed a large number of very good talks, the biggest exhibition at a EUUG conference to date and, of course, some Dutch beer.

This report is being done from my incomprehensible notes and mostly from the abstracts which were submitted before the conference began‡. I must confess to missing some of the sessions; still, that is always going to happen. I also feel that this report is not up to the usual standard because I am doing it too long after the event and this makes it difficult to precis the talks. So, where I am in doubt I have kept quiet. If have got any names wrong, then I am sorry.

However, the intention is to publish a full conference proceedings in the near future. Meanwhile, this can serve as a summary of what happened.

Day 1 - 16th April 1984

Emrys Jones officially opened the conference and chaired the first session.

Item 1: 10.00am **Michael J. Kelly, AT&T/Teletype Corp**

An Intelligent Windowing Graphics Terminal for the UNIX system

Abstract

An important feature of the UNIX System is per-user multiprogramming; that is, each user may control several concurrently executing processes. However, this feature breaks down at the user interface. UNIX systems rely on "dumb" or semi-smart terminals as the primary user interface, and these are not able to maintain several concurrent interfaces. The solution found in BSD, job control, still does not adequately solve the problem of maintaining display contexts for concurrently executing processes.

This talk was about AT&T 5620, the Teletype Corp's version of the Blit terminal. It is a workstation with a high resolution green screen, a mouse and has a reasonably powerful machine to drive it. The machine does not run UNIX because the device is a terminal and not a working CPU in its own right. The processor is a WE 320001 CPU with 64K bytes of ROM, 256K bytes of RAM expandable to 1 Megabyte. The terminal also has an RS232 interface.

The main power of the terminal derives from an interface to UNIX System V, which allows several windows to be placed on the screen. Each window has its own control software running in the terminal and also a user level protocol handler running in the host.

Q. Is the protocol specified and available?

A. Not yet

Q. Are there any cross development tools for the 32000 CPU?

A. Yes.

Q. What is the availability in Europe?

A. Olivetti will distribute it.

Q. Will the software run on any UNIX system other than System V?

‡ Thanks to Jaap Akkerhuis for finding them in a machine readable form and sending them to me.

A. No comment.

Item 2: 10.30am **P.Freund, Hewlett Packard**

A Layered Implementation of the UNIX Kernel on the HP9000 Series 500

Abstract

An implementation of the UNIX operating system kernel has been layered on top of an existing operating system kernel for the HP9000 series 500 computer. The mapping of UNIX functional requirements onto the capabilities of the underlying OS are presented in this paper, including the changes and extensions necessary to support UNIX semantic and performance requirements. The paper covers in retrospect the advantages and disadvantages of a layered approach.

This talk discussed HP's approach to the implementation of UNIX System III (with those familiar Berkeley enhancements - i.e. **vi**). The hardware is based on a single chip 32-bit CPU with a stack based architecture. A configuration can consist of several multiple CPU's. HP wanted to support operating systems other than UNIX, and have developed an internal kernel called SUN (no relation to those other folks, folks) onto which UNIX has been layered.

The SUN kernel has memory management, process management, a file system supporting multiple directory formats, device drivers, some I/O primitives, a real-time clock and interprocess message passing. It does not have a human interface because is it designed to support operating systems and not humans. Also, there is no means to load programs.

The talk then described the ins and outs of the implementation which I hope will be reproduced elsewhere. I was left with the feeling that the system worked, but perhaps someone out there in UNIX-land might like to give a slightly less biased report.

☞ Coffee (and a peek at the exhibition) ☜

Session 2 was chaired by Adrian Freed, Ircam, France. I missed the next person's name, sorry.

Item 3: 11.31am **Donald St..?, Amdahl Corp**

Future directions for UNIX at Amdahl

Abstract

This talk will cover future plans for UNIX on Amdahl.

The talk started by summarising the history of UNIX running on Amdahl machines. The system is called UTS and runs in a virtual machine under the VM operating system, the latest release is UTS 2.2 and UTS 2.3 will be released shortly. The current release has 'good V7 compatibility'. The idea of marketing UNIX is to take advantage of an internal product which had been developed for in-house engineering use, to increase their reputation as a software vendor rather than their being known as simply a hardware supplier, and to make inroads into the academic community. Future plans are to: continue leadership in the large system UNIX market and to maintain compatibility with the Bell Labs product at current release levels. I.e. they are working hard on System 5, which is running in-house and at some installations of early customers.

Q. Pricing?

A. An AT&T license is required, plus $1500 per month.

Q. European support?

A. Yes.

Item 4: 11.53am **Larry L. Crume, AT&T Bell Labs**

UNIX* System V Release 2.0 and Future Directions

Abstract

Enhancements to UNIX System V Release 1.0 will be reviewed. These enhancements which are to be released in April 1984 include feature updates and improvements in the following areas: C Compilation System; Job Control/Virtual Terminals; Shell; Commands; Cron Facility; Curses/Terminflo Package; Standard Disk and Tape Names; Accounting Package; Performance; and New Documentation.

Future Directions: Future enhancements to the UNIX System will focus on the user interface. The paging system under development at AT&T Bell Laboratories shows one dimension of the user interface for systems programmers. The architecture must be general enough to support both paging and swapping kernels and many different memory management units.

The work on command syntax and error message handling provides for a consistent user interface and to easily determine and recover for errors.

Work on unbundling and repackaging the UNIX System will provide for a consistent user view of the UNIX System.

To fill that in a little. This year's system from AT&T is called System V, release 2. There are several new applications programs which come with the system and some enhancements to the operating system.

The biggest development objective is to maintain upwards compatibility while improving performance. This does not necessarily mean that the increased speed for increased size trade-offs will be done, but AT&T are looking at some of the UCB enhancements.

Job control has been introduced. This has been done in a totally different way from the UCB implementation because it was not desirable to introduce the necessary new signals. AT&T are committed to support and not extend the current set of system calls. Job control is done by having a number of layers which define virtual terminals. On login, the user talks to a control layer and has the ability to start a number of shells in other layers. Input and output from these virtual terminals can be controlled independently.

The new C compiler will have long variable names. Also, there will be a C cross compiler for the M68000.

Larry then talked about the current ideas of unbundling UNIX. The pieces will consist of a basic set of utilities plus the kernel and yes, your favourite command is bound to be missing. There will then be several add-on pieces, such as the C language tools, the various workbenches, and other utility sets.

On the list of things to be looked at in future releases include: record and file locking, this will initially be the /usr/group standard; bad block handling; and file system integrity, such as protection from unexpected halts, ordered writes to discs, timely flushing of buffers and improved detection of corrupt file systems. A big thing on the list is a paging kernel in order to provide a large address space. Here, the main aim is to not affect users who don't wish to have paging in their machine. To this end, an architecture for memory management has been developed, with the idea that the paging should be easily configurable in or out. This is not on the current release, because it wasn't good enough.

This was an interesting talk, really, packed with a lot of stuff which I feel there is no room to put here. AT&T seem to be very responsible with the developments which they propose, it is perhaps possible that they are trying to generate too many new systems and are not leaving things to settle enough. My other main criticism is that most of the things coming out are fairly mundane and ordinary, the interesting leading edge of technology stuff is staying under wraps until it becomes safe and boring. I suspect that as an academic user, I would like to have Edition 8.

☞ Lunch (and a bottle of beer) ☜

This session was chaired by Bjorn Eriksen.

Item 5: 2.05pm **Bill Murphy, AT&T International**

UNIX Licensing

Abstract

What you can and cannot do for your $43,000, $68,000 etc.

Well, all Bill did was show his face and get off. The idea was that people could tackle him outside the main hall. Which I suppose they did.

Item 6: 2.08pm **Robert Ragan-Kelley, Pyramid Technology Corp.**

OSx: Towards a single UNIX system for super-minis

Abstract

OSx is a dual-port of 4.2BSD and System 5 onto the pyramid 90x computer, a high-end super mini. OSx is designed to be fully compatible with both 4.2 and System 5 in a fashion that neither suffers performance penalities from the coexistence of the other. This paper discusses some of the details of this design, both internal to the kernel and at the user interface level, along with some of the problems we faced in its implementation.

The idea is to implement 'universes', one called **btl** and the other **ucb**. These two names are used as commands to switch between the two universes. It is also possible to have a command line containing commands from one universe in the other, by prepending the alien command by the appropriate universe name.

The implementation started from 4.2BSD and emulates System V. The main reasons for starting from 4.2BSD are the demand paging, the fast file system, flexible file name lengths and the larger block size. Also, the 4.2BSD networking would be difficult to implement on a System V.

The main problems are: the differences in the directory structure, System V FIFO's (named pipes), signal handling, System V IPC and worst, the differences in terminal handling and **ioctl**'s.

Item 7: 2.40pm **Eric Allman, Britton-Lee, Inc.**

The special advantages and difficulties with databases in UNIX (1)

Abstract

Many applications maintain some long term state in the form of a database. In many cases ad hoc algorithms are sufficient (e.g., sequential scans of the password file are adequate on most systems), but often more sophisticated algorithms must be considered (e.g. mailboxes must be locked while mail is being delivered; the dictionary is too large to be practically searched sequentially).

Although ad hoc approaches are acceptable for small applications, larger applications often find it convenient to utilize a full-blown database system. Such systems may include such features as efficient access methods, logical independence from data structure, aggregation, protection, integrity constraints, multi-file capability, concurrency control, crash resilience, audit trails, and transaction control.

The structure of a database system incorporating most of these features is examined. Interfaces, data models, cost/performance tradeoffs, and the special advantages and difficulties UNIX offers to database systems are discussed.

This, the first of two talks, gave a general introduction to database terminology and operations. It seems to me that it would be silly to attempt to summarise his talk here, we'll wait for the paper which will no doubt be considerably more comprehensible than anything I can write.

☞ Coffee ☜

Item 8: 3.42pm **Eric Allman, Britton-Lee, Inc.**

The special advantages and difficulties with databases in UNIX (2)

The second part of the talk was concerned with the facilities which are provided by data base systems and the trade-offs inherent in such systems.

Item 9: 4.01pm **P.B. Pynsent, University of East Anglia**

The Norwich Renal Unit Programme

Abstract

In Europe 120 people per million of the population suffer from chronic renal disease and of those 80% depend on an artificial kidney machine for survival. We have developed a UNIX based computer system which not only provides access to a patient database but also controls kidney machines during the haemodialysis of patients.

The objective of the Norwich Renal Unit project is to improve patient care using computer technology. First we have provided facilities for computer controlled kidney machines to optimise dialysis therapy to the individual patient and secondly we have provided an easy to use patient database to aid the physician in his assessment of patients. The UNIX operating system has proven an ideal environment satisfying both the multitasking and data processing requirements of our project.

I really liked this talk. It is so rare at UNIX gatherings to see people who are doing something real with computers. I think that I mean that the majority of talks at EUUG conferences are really 'Computer Science' and I would like to see more applications oriented presentations.

Item 10: 4.31pm **Andrew Hume, AT&T Bell Labs Computer Research**

Eric: an experimental information manipulation system

Abstract

Eric is a testbed for a model of how the user interacts with a computer system. The major components are the filing system, multi-tasking, the use of forms as the only means of data input and a user interface dependent on a bit mapped graphics display.

The abstract does not do justice to the talk, but my notes are even worse because I spent most of the time concentrating on what was being said.

☞ End of Day 1 (and onto the Hotel Erica for free drinks) ☜

Day 2 - 17th April

Well, I managed to make it out of bed to chair the first session of the day.

Item 11: 9.39am **Kirk McKusick, University of California, Berkeley**

The Dynamic profiling system

Kirk's talk centred on the new profiler **gprof** which comes with 4.2BSD, and described how he had been using it to improve kernel performance. The original UNIX profiler presents a flat profile of program performance with the emphasis of the amount of time spent in a particular routine. **Gprof** does better than this by tracing the path through the code and giving statistics for: how often routines are called; from where; and in turn, which routines were called by the routine under consideration. (Having used it to analyse program performance, it's really good).

Kirk then described a scientific investigation into UNIX kernel performance. He found that **namei**, the main directory search routine in the kernel took one quarter of the system time in 4.2BSD. He described various attempts make this go faster, pointing out that some obvious solutions appeared to improve some aspects of system peformance (i.e. **ls** appeared to be better) while profiling showed that overall system performance had not really improved.

'Make it work and then make it go faster' is an old Bell Labs axiom, perhaps we should add: 'then show it really does go faster under all circumstances'.

Item 12: 10.05am **A. Burns, University of Bradford**

A Comparison of the UNIX and APSE Approaches to Software

Abstract

An important aspect of the Ada project is the attempt to design and implement a standard support environment for the development and maintenance of Ada programs. A number of Ada Programming Support Environment (APSE) projects exist; many are using UNIX as a basis for the work and as the starting point for design. There are however many important differences between the use of software tools under UNIX and that envisaged for an APSE. This paper is concerned with the use of software tools and their interfacing. Comparisons between a UNIX and APSE approach are given.

This was another, much needed, introductory talk.

Item 13: 10.36am **Bill Weir, STC IDEC Ltd**

C Unit Test Harness

Abstract

Module testing is an important and often neglected area of software testing. Traditionally, it has tended to be a largely undocumented operation in which the programmer pokes data interactively at a module until he believes it is working satisfactorily. Studies have shown that tests conducted in this manner are seldom adequate in their coverage of program paths. It is hoped that, by automating much of the process, and by relieving the programmer of the drudgery of creating driver modules, collecting results, etc., more extensive testing at a module level will be encouraged, with consequent reductions in testing and debugging costs at a later stage of the software cycle.

A talk full of interesting ideas. Bill presented a system where the testing of routines can be formalised and automated. I felt that I need something like this, if only to aid regression testing. The main problem was the specification of tests.

Q. Are there any tools to help in building tests?

A. No, not at present.

Q. What is the availability of this?

A. Sorry, don't know.

☞ Coffee ☜

The session chairman was Teus Hagen.

Item 14: 11.26am **M.A. Rathwell, University of Bradford**

Distributed Decision Making under UNIX

Abstract

Distributed Decision Making attempts to meet the need for supporting tasks which involve cooperation and conflict between differentiated organisational units. A DDM system provides a mechanism for linking several decision support systems in an organisation, so enabling groups which are not necessarily linked in a hierarchical manner to cooperate with one another. This is particularly significant in planning operations which require information from different people dispersed throughout an organisation. It can enable independent decision makers to semi-automate their work, explanations for decisions can be made widely available, and the resolution of conflict between nodes with different interests and perspectives can be supported.

Item 15: 11.50am **A.R. Pell, University of Reading**

An Interactive Information Retrieval System for UNIX

Abstract

Many problems exist in office and information systems for which an appropriate solution is an information retrieval system. The aim of the first part of this project has been to build an easy-to-use interactive system. This has involved analysing the concepts and information structures needed, as well as considering the user interface. The emphasis throughout has been to construct the system in such a way that it is easy for a novice user to handle. Building on the established system, the second part of the project will involve experimenting with differing input devices such as touch screens, mouse input, trackballs, voice, etc.

Item 16: 12.15am **Theo de Ridder, IHBO "de Maere**

Automatic Generation of Syntax Directed Screen Editors

Abstract

From a new effective and automatic error-recovery scheme for LALR(1)-parsers a program generator is developed that produces a syntax directed screen editor for any language specification written in LEX and YACC.

☞ Lunch ☜

Session chair was Jim McKie.

Item 17: 2.00pm **J. R. Nicol, University of Lancaster**

CRS - A Powerful Primitive For Resource Sharing in UNIX

Abstract

This abstract focuses on a resource sharing system which we call 'CRS' (Connect Remote Shell). CRS is layered on top of the UNIX operating system and provides a powerful set of network services. The environment in which CRS was developed formerly consisted of a number of PDP-11/44 mini-computers, each running UNIX. A user's computing activities were typically centered on one of these machines. Circumstances changed when we obtained a Cambridge Ring local area network, since we were then presented with the possibility of sharing the available computing resources.

Item 18: 2.37pm **Brian E. Redman, Bell Communications Research**

Honey Danber - The UUCP of the Future

Abstract

In 1978, Mike Lesk was considering a mechanism to aid in the administration of software on the growing number of computers running UNIX at Bell Labs. He envisioned a system that would automatically synchronise several machines with updates from a single source. At the time, no networking software existed upon which to build such a system, so he invented the UNIX to UNIX Copy program (**uucp**) to transfer files among machines. Little did he know that **uucp** would become the foremost file transfer and remote execution facility for untold years to come. That which was created as a temporary measure to get data from one UNIX system to another has endured through time as one of the most beleaguered, yet most critically required UNIX utilities.

Brian's talk centred on a recent re-write of the **uucp** suite of programs which will be available on System 5, release 2. The re-write was undetaken by an ad-hoc committee and the programs are a product of 'software engineering' and not just hacked together. The result is a much more flexible, faster and better controlled **uucp**.

☞ Coffee ☜

Session chair was Keld Simonson.

Item 19: 3.45pm **P Tintel, Bell Telephone Manufacturing Company**

EURONIX: A UNIX based system using European natural languages

Abstract

The UNIX system has gained enormous popularity. It is used in many places for software development, and it is beginning to be used in office environments. However, the average office worker is no computer specialist and he has other demands concerning the system then software developers.

The UNIX user interface as it is today has certain drawbacks for office applications and more specific for office applications in Europe. The manuals are not very readable for someone not familiar with UNIX; all communication with the user is performed in English (or American); and UNIX is not capable of working with the different European characters in a uniform way.

In the Euronix project, focus is on the second and third problems: EURONIX will communicate with the user in the user's natural language and will be able to handle in a uniform way the special European characters.

The implementation is to alter UNIX from working in ASCII to working in the TELETEX character set which can cope with all the European 'funny letters'. In addition, all messages from programs pass through a string data base in the user's natural language.

Item 20: 4.16pm **C. Roberts, Inf. Techn Task Force, EEC**

Standardisation of the national character sets

Abstract

A review of the European Standard policy for national character sets will be given. The way it fits in the current EEC programs will be discussed: the general options of setting standards in character sets, some examples of standards of character sets, as well as the implementation policy (some keyboard experiments).

Item 21: 4.45pm **Emrys Jones, Chairman EUUG**

EUUG Annual General meeting

Abstract

This is the official general meeting to present the new constitution.

This meeting was held as a bootstrap device to get the new constitution off the ground. The preliminary constitution was passed and the new structure formally inaugurated.

☞ And so to Dinner at the Erica ☜

The idea of having a conference dinner was a good one. However, the red wine was awful. There were some after dinner speeches to make it taste better. Theo de Ridder made a speech which I have acquired in machine readable form. I am indebted to Jaap Akkerhuis for twisting his arm, leg or some other piece of anatomy for it.

Item 22: 9.45pm? **Theo de Ridder**

After dinner with Theo

Dear delegates.

In the past scientists used a very sensible language to express and exchange their ideas. The importance of that old dead Latin was that you had to be conscious of its fixed syntax and semantics in order to use it. At this moment I am permitted to make a speech in English, without much knowledge of its syntactic or phonetic structure. I dare to do so because in a more interactive situation no-one ever interrupts me with error messages whatever my mistakes.

Looking around in an UNIX environment the simularity between a programming language and a natural one is remarkable. There is not any formal syntactic or semantic description available and still programs are made and ported all over the world. In case of a programming error the system is kind enough not to complain about its exact type or place, it just gives you a wink that something went wrong. Isn't it a nice paradox that such an honourable human reaction is considered as typically user-unfriendly?

Let me go on with an anthropomorphic view on UNIX. I am aware that most of you do have a rather intimate relation with it. Using the statistical argument that for 90% you are male and will prefer the opposite sex, I conclude UNIX is female! Well gentlemen, what about your behaviour over the last decade? In any case you exploited her. Sometimes you even raped, suppressed or sold her. And some individuals had the courage to publish the insultant proposal to bring her in the public domain! In spite of certain parallel liberation movements in society UNIX was not able to free herself from historical bounds into an independent respectable creature.

There is a more philosophic and fundamental aspect of the human condition than being male or female, and that is being mortal. So, UNIX is not eternal. She must be in one of the binary states, alive or dead, or else she is instable in illness. It is hard to prove where she is. Her growth and continuing changes indicate liveliness. The exponential increase however reveals a disease like cancer. And finally the cult of these meetings, the existence of gurus, and the need for myths synthesize the declaration of a posthumous holiness.

After dinner it is fun to tell each other fairy tales. Maybe you need a tutorial example. I hope it will be simple to apply the following to your own business.

Once upon a time there was a princess called V6. She was considered small and beautiful by her people. Many a handsome academic prince came along to ask for her hand. But her mother, Ma Bell, locked her up to prevent disintegration of the empire. And so she became old and ugly in grim electronic towers. Her only pleasure was looking out of a window into the silicon valley and watching the play of the big cat AT&T with a lot of little licenced mice like ... you.

Day 3 - 18th April

Well, unlike yesterday, prolonged late night discussion kept me in bed to miss the first session. The session chairman was Joachim Wolff.

Item 23: 9.30am **Joe Carfagno, Central Services Organisation**

Using UNIX on a large software project

Abstract

This talk is about the uses of the UNIX operating system in a large software project. Many different implementations and releases of the UNIX system, including the UNIX (1100) system, are used in a variety of ways from software development, testing, project management, site support, and others. This talk will show the versality, flexibility, and portability of the UNIX system.

Item 24: 10.05am **Ludo Vemmekens, Amdahl Nederland B.V.**

Large Systems UNIX: Opportunity for Innovation

Abstract

In bringing UNIX to the large mainframe world, there is a merging of two standards: the UNIX standard of openness, ease of use, ease of development, and the IBM large operating system standard of high reliability, security, performance, and support. The combined standards are a new challenge to the operating system products world.

This paper will discuss in detail the technical issues in making UNIX a viable mainframe operating system. These include reliability, production operations management, communications, memory management, full duplex support, database, security, support, compilers, applications, coexistence with MVS, VM, and UNIX.

☞ Coffee ☜

I got to the hall just in time to see Andrew Hume, session chairman and contestant in the 'hairy knees of the conference competition' make a small opening speech deploring the current fashion of being rude about UCB. More power to his knees.

Item 25: 11.18am **David Tilbrook, Imperial Software Technology**

The 5 Pitfalls of Interactive Graphics

Abstract

The NEWSWHOLE system was created almost 10 years ago. A video tape of it has been widely distributed and used to teach interactive graphics. One way it has been used is to examine the way in which it avoids the 5 pitfalls of interactive graphics: Boredom, Confusion, Discomfort, Frustration and Panic. This presentation will discuss those pitfalls and the design strategy used to avoid them.

The abstract does not point out the thing of which David is most proud. The system worked on a high resolution display and a different cursor shape was used to indicate different operations. The one that springs to mind is the Buddha which meant 'please be patient, I am computing'. This idea is a goody, it's one of those things which when you see you wonder why everyone doesn't use it, but of course, you have to have the idea first.

Item 26: 11.50am **Brian E. Redman, Central Services Organisation**

Behind every Binary License is the UNIX Heritage

Abstract

Lately there seems to be some pessimism about the future of the UNIX system. Many who have watched its development from the earliest days feel that the system appears to grow corrupt and is no longer a model of innovation in operating system design.

Unix was originally designed by a talented fraternity with a clear and common vision for a better environment. Ever since the system has been redesigned by a diversity of people with different goals the tend to be less clear. UNIX has evolved from a simple, elegant model into one that is certainly complex and often seems convoluted. It no longer constitutes a statement of smallness, but appears to be growing unrestricted. ...

This was certainly the funniest talk of the conference, we hope to get a reprint. Contrary to popular opinion, Brian was never in the running for the hairiest knees of the conference award.

☞ Lunch ☜

The afternoon session was chaired by Mike Banahan

Item 27: 2.02pm **Andrew Hume, AT&T Bell Labs Computer Research**

Processes considered as files

Abstract

Tom Killian has implemented images of running processes as full and legitimate elements in the file system. The image for process nnnnn is accessed as '/proc/nnnnn'. Read(2) and write(2) work normally except that some system data is write protected. Ioctl's include stopping and starting a process, masking out signals that cause a stop, and returning a file descriptor for the text file (for the symbol table).

This is a neat idea costing 4K bytes in the kernel.

Item 28: 2.08pm **Daniel Karrenberg, University of Dortmund**

University of Dortmund's Spooling system

The University of Dortmund have several machines running several versions of UNIX but have few printers. They also have no Local Area Network, such as an ethernet. The talk described the spooling system which has been set up to cope with these problems.

Item 29: 2.15pm **Andy Greener, Imperial Software Technology**

EUUG Benchmarks: some results

Abstract

The EUUG Bench mark tests have been run on a variety of VAXs running UNIX-5, BSD4.1, BSD4.2. Results are presented and discussed.

The tests were on VAXes and should be shown elsewhere in the newsletter.

Item 30: 2.29pm **Johan P. Moelaert, Twente University of Technology**

A Semaphore Implementation in UNIX

Abstract

For certain purposes the UNIX system lacks strength in its possibilities of interprocess synchronisation. Several processes waiting for one event require an abundancy of process control when implemented with signals, and this is not very reliable. Mutual exclusivity may be implemented using 'open' and - if this fails - 'create', but this is

neither very elegant nor completely foolproof. The pipe mechanism offers a good instrument for synchronisation, but can only be used between processes that have hierarchical relationships. For these reasons we decided to implement Dijkstra's semaphores in the UNIX system.

This was an ingenious solution which added some system calls to do the semaphore user interface. The system uses the in-memory inode table to store state of the semaphore. The talk was the only one to actually put some C up on the screen, and for that reason alone, scored some points.

Item 31: 2.42pm **Neil Mayhew, Bleasdale Computer Systems**

Experiences With Implementing IBM Bisync IN UNIX

Abstract

Bleasdale UNIX micros are used in a variety of situations commercial, scientific and academic, and there is now increasing demand for distributed processing in the form of connecting micros to existing mainframe and database facilities.

Bleasdale's first step towards meeting this demand was to promote file transfer facilities, including RJE (Remote Job Entry), using IBM 3780 Bisync.

Item 32: 2.55pm **Theo de Ridder, IHBO "de Maere**

MABENCH: a portable benchmark machine

Abstract

In the computer science education (HIO) department of our institution we developed a complete synthetic benchmark package BENCH. In BENCH it is possible to specify arbitrary workloads with any number of parallel processes. Scriptfiles can be made by editing or by running application oriented scriptfile generators. In the output all the characteristic performance values (throughput, response time, service time) are given in table format.

The MABENCH system is a small portable machine which can be connected to the system under test via normal terminal connections. The small processor (6809) then sends several scripts to the test machine while performing measurements.

Item 33: 3.10pm **Nick Nei, University of Glasgow**

Measuring the Disk I/O on a VAX

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.

It seems that the measurements show no really discernable statistical model which is applicable to disc performance.

☞ Coffee ☜

Item 34: 3.30pm **Panel Session chaired by David Tilbrook, IST**

Is there a future for UNIX?

During this session, I had to go to catch a plane. Richard Hellier from the University of Kent kindly agreed to take notes and produce a report. So here it is:

The panel was:

L.L.Crume, AT&T
R.Raglen-Kelly, Pyramid
K.McKusick, UCB
H.J.Thomassen, U of Nijmegen CS Dept.
M.Banahan, The Instruction Set Ltd.
A.Freed, IRCAM.

Each of the panel members made a short statement on the subject:

Does UNIX have a future, and, if so, which way is it going?

Larry Crume expects further developments in networking tools and support and much greater use of windowing software at every level; Many novice interfaces would be required in future. He also mentioned the unbundling of UNIX and the simplification of the licencing process.

Roger Raglen-Kelly was concerned about people hacking UNIX for the sake of UNIX itself. He thought that UNIX was already too big and that something should be done to arrest the growth. What he wanted to see: was better internal support for multiprocessing and true networking; functional and object-oriented programming languages & shells.

Kirk McKusick reminded everyone that the days of formal releases of code from Berkeley were over and that they were returning to being a research institute.

Adrian Freed stressed the importance of separating UNIX from UNIX-based applications; With the preponderance of commercial and industrial users of UNIX, he felt that only a minority of UNIX users cared what the underlying system was. All they are interested in is their Spreadsheet, Data-Base or whatever.

H.J.Thomassen followed up this line, later amplified by Teus Hagen, and pointed out that the Academic Community was moving in a different direction from the business community.

Questions and comment were taken from the audience.

Emrys Jones introduced a point of information, namely that Computer Magazines & Journals, as a group, were now outselling "girlie" magazines. He asked,

Did the panel feel that the "hobbyist" section of the user community would ever have much impact on the UNIX community as a whole?

Due to the fragmentation of the enthusiast sector, no-one anticipated any significant developments from this area.

Teus Hagen pointed out that although business users may be moving one way, any individual user will still be free to follow his own path.

Another speaker emphasised that UNIX should be viewed as a way of doing things rather than as a static entity.

Following a prediction of researchers moving away from UNIX towards, say, expert and knowledge based systems, Eric Allman noted that all this new code would have to be developed somewhere, and probably on UNIX systems.

Kirk McKusick spoke of possible further developments of the UNIX kernel to support concurrent running for tightly-coupled multiprocessor systems.

Replying to a question on the conformity of AT&T products to /usr/group standards, Larry Crume said that AT&T were part of /usr/group and so their code would be 100% compatible with those recommendations.

Several speakers expressed concern that the UNIX tradition of distributing source code, even of the kernel, would not be upheld by the commercial users and software-houses. Larry Crume affirmed that AT&T would continue to supply source code with System V and its successors. Many attendees wanted some form of pressurisation on business users to distribute sources of their products. Before taking up the legal aspects of this topic, Mike Banahan observed that few business users care about source code; All they want is a tool for a job, he felt.

The next question was:

How can software developers protect their investment unless they restrict source?

Eric Allman, speaking for Britton-Lee, pointed out that small firms simply could not afford even one lawsuit to test a copyright case. He quoted 'one week to liquidation' if his firm ever became involved in such an action. Only the giants, like AT&T, could afford such a case. Britton-Lee will sell the source, but at a high price.

There was then a brief discussion of the impact of APSE technology on the UNIX community. Several speakers feel that even if the project fails to achieve its goals, like Multics, it will contribute much to the industry.

In response to the alleged unchecked growth of the UNIX kernel, Kirk McKusick observed that size must be traded off against functionality, i.e. that a kernel with a given set of services must be of at least a certain size.

The final topic was the future of Inter-Process Communication, instigated by Dave Tilbrook. Larry Crume felt that, in the current absence of any formalism for describing IPC we still don't know which way to go. None of the panel members would be drawn on this one.

Eric Allman observed that IPC enables software developers to write code that runs in user space and then move it into the kernel if memory allows.

There was a general view that there must be a logical separation between particular applications and the IPC mechanisms they employ. IPC enables us to *communicate* rather than *convert* software. That is, when some new service becomes available we converse with that rather than its predecessor, and to keep the applications small.

This spawned a nostalgic side-discussion about the good-old-days of Version 6; Eric Allman observed that much of the "elegance" of that system arose from its implementation on a particular machine architecture, i.e. PDP 11, and that many of the techniques employed were simply the only ways to proceed on such a machine.

As a postscript, there were a few questions about the purpose, or lack of it, of having future EUUG meetings.

Tutorial sessions

A number of tutorial sessions were run is parallel to the main meeting. A summary of each session is given here. The summary is derived from the meeting programme as I cannot be in two places at once (sometimes, I can't even be in one place at once).

Day 1 - April 16th

Item 1: 2.00pm **Mike Banahan, The Instruction Set**

Advanced editing: ed, ex, vi, etc.

The common aspects of line-oriented, screen-oriented and more advanced editors will be given. How to write your own editing macros, how to handle key stroke definitions and how do you program in your favourite editor's very own command language.

Item 2: 3.45pm **Bill Murphy, AT&T International**

Licensing, workbenches and UNIX System V 2.0

An overview of all the UNIX packages from AT&T Int. are given. Prices, availability and future developments to be expected. Most of the topics about licensing will be addressed.

Day 2 - April 17th

Item 3: 9.30am **Mike Banahan, The Instruction Set**

Advanced Shell programming

Just using one command with some arguments is easy. How you can make use of all the features of the UNIX command interpreter is some more difficult. It is addressed how you can make your command life easy - no flags, just combinations of existing commands. Yes, you can program in Shell.

Item 4: 11.15am **Jim McKie, Centrum voor Wiskunde en Informatica**

UNIX 4.2 BSD

What 4.2BSD gives you, doesn't give you, gives you too little of, and, of course, what it gives you too much of.

Day 3 - April 18th

Item 5: 9.30am **Nigel Martin, University College London**

The UNIX market

An overview of the market situation concerning machines with UNIX will be given. How do they differ and what kind of choice should be taken? The direction of this market. Slides of the rivals will be shown.

Item 6: 11.15am **Teus Hagen, Centrum voor Wiskunde en Informatica**

UNIX networking with UUCP

How do you connect your machine to the outside world. Maintenance, structure, software and connection costs for an UUCP link. Statistics, software layers and tools are also addressed.

Item 7: 2.00pm **Jaap Akkerhuis, Centrum voor Wiskunde en Informatica**

Advanced Text processing

Text processing is not like word processing. To layout your text nicely, you should know more about the possibilities of n/troff, eqn, tbl.

Endpiece

Well, that's it for another EUUG conference. The papers which constute the proceedings will be published separately.

Thanks are due many people who worked very hard to make the event a success. Hendrik-Jan Thomassen and George Rolf headed up a team of tireless workers. David Tilbrook did the programme organisation aided and abetted by Teus Hagen. The EUUG secretariat, Helen and Debbie, did their usual good job.

And the winner of the 'hairy knees of the conference' award? Well, it was a close run thing. But since there were only two competitors, and both were Australian, the winner must be Richard Grevis.

EUUG - Nijmegen, 1984

Harold Cross

As UniForum was described as a zoo (more like a circus), EUUG brought to mind a peaceful gathering in the park (perhaps reading poetry aloud). There were approximately 300 attendees at the meeting. The technical content was high and the commercial influence was low-key. The conference was held at the University of Nijmegen. The auditorium was a lecture hall in design fitted with high-tech audio/visual apparatus. There was a vendor exhibition held in the promenade outside the auditorium and in a couple modest sized rooms. Although the wares were uninteresting for the most part, there were machines to play with and give-a-ways of buttons, tee-shirts and plastic bags. The lack of a powerful exhibition in no way detracted from the meeting, it enhanced it. There were a couple of interesting terminals. The best was the M2150 from Microcolour Graphics. It has a 4096 color palette (16 at a time) with hardware (power of two) zoom, area fill, pan and a cross-hair cursor. The graphics resolution was 640x384. There is an independent text plane (80/132 x 80). The price was around $US3000.

The meeting began the evening before the first day of talks as people formed SIGs in the local hotel bars. The European Unix community is not unlike its North American counterpart. It is populated by interesting and knowledgeable personalities (some more than others of course). Perhaps because of my perspective (but more likely because of the character of the meeting) the conference was excellent, technically rich and socially invigorating.

I listened to three quarters of the talks, attended a few more than that and missed a few entirely. I will describe some of the talks.

Emyrs Jones began the meeting by not giving the opening remarks in Dutch (apparently his general custom was to open in the native tongue).

Mike Kelly from Teletype talked about the 5620. (He also gave a short sales pitch on the 3B series of computers which I found irritating.) One of its features is that it doesn't require you to come up with another version of Unix for your workstation (you just have to run a version that supports the 5620 on your host). It runs on standard system V from AT&T Technologies. It even runs on a VAX, "which is important today, but probably won't be tomorrow.", Mike says. A megabyte of memory makes a more usable system. He described layers, xt and demux functionally. Although he couldn't make a commitment, he wouldn't be surprised if there was a Berkeley port before long. They are looking at color, peripherals (disks), and compatibility with other Unix systems. A questioner from the audience (from AT&T-BL) suggested a terminal ought to come with the capability to download it. Another asked why Mike was talking about peripherals. Next they'll put Unix on it. Kelly replied that they certainly won't put Unix on it, it's not a suitable architecture. Someone from England with a Canadian accent bitched about non-interaction of the layers and the lack of ability to have multiple processes in a layer. Kelly promised IPC within the 5620 by the next release and RS422 support within a year. Olivetti will be the exclusive distributors of the 3B line and terminals.

After the coffee break, Adrian Freed introduced a couple of talks on the future of Unix at Amdahl and AT&T-BL. He advocated (I think) a merger of System V and Berkeley versions and voiced a plea for the availability of source.

The speaker from Amdahl talked about their product. There was a 13 MIP model and a 20 MIP attached processor model. Lacking was full duplex ASCII support. Their next product would be an implementation of System V. It would also have paging and vfork(). He mentioned that after the third port to new versions of Unix, they used the Bell (sic) source and implemented their changes with ifdefs.

Next Larry Crume spoke of where AT&T-BL Unix was headed in the future. They will continue to provide source, they will continue to provide portability. He said humbly that AT&T-BL has learned, although it took them a while to learn, from Berkeley. He had the right perspective in that Berkeley was a research organization and their output was to be looked at carefully. Some of their work would be incorporated, some of it served as models which would be reimplemented,

other things were not necessarily relevant or useful. He mentioned that job control was changed slightly from 4bsd (cough) so user programs don't have to worry about it. With respect to other features, "We will move forward to implement those in an evolutionary fashion." He was concerned over the growth of commands' sizes. Future Unix - A base, the kernel with a minimal set of drivers and utilities (libraries). There would be add-ons such as the C language and other utilities (grep, sort, awk ?). He listed the basic commands. Paging kept cropping up and was eventually addressed. It isn't available yet because AT&T-BL hasn't been satisfied with the performance of their implementation. He hopes that a satisfactory implementation will come out in third quarter of '84 (he hopes many things will be available then). It will provide a large address space and isolate the memory management architecture. There will be no application program changes and it will not hurt users who don't need paging. He also expected record and file locking a la the Unix standards committee.

Lunch was served. I believe it was some sort of chicken along with a meatball soup and fries. Fortunately Dutch Heineken does not taste like swamp water.

Bill Murphy spoke very briefly offering to give people the answers they need. (Is that like giving people the answers they want?)

After the tea break Eric Allman continued with his analysis of databases. I didn't take any notes having spent the entire time trying to get the writing shelf for my seat/desk in place.

The next talk was about an interesting application of Unix. It was to provide a database for kidney dialysis patients and to control the dialysis machines dynamically (e.g., to remove water at a variable rate over time.) Using the computer, new techniques were implemented which otherwise were impractical. The obvious fear from some of us in the audience was the inevitable "Out of blood, core dumped" message. But the machines were very sophisticated and provided many fail-safes. Artificial kidneys were passed around. The speaker observed that touch screens were a good gimmick but not really useful for his application.

Andrew Hume talked about integration of various user services. For instance given a mail service and a calendar service, could one send a calendar as mail? Surprisingly often, this is not possible. He talked about his work with multiply typed objects and a bitmapped/mouse interface which overcame these deficiences.

At some point during the talks a notice appeared on the chalkboard. "This year's competition. NOT annoy Rob Pike with new switches for cat. BUT suggest an extension to C which will benefit the community??" The list of suggestions grew throughout the day and was anonymously erased the next morning.

We were on our own for dinner (thank goodness) and a few of us ate in a hotel restaurant. It was fantastic and not very expensive. I did not attend the bar SIGs that evening but dutifully retired to my room (10 minutes away in nearby Groosbeck) to finish some transparencies.

Tuesday morning began with a talk on dynamic profiling from Kirk McKusick. He covered the dos and don'ts of tuning as well.

Next was a discussion of the APSE and unix approaches to software tools. The problems of a large software project were discussed (>100 people, >50K lines, 30+ years lifetime). They were listed as:

1) divergence of intended program behavior
2) cost estimates exceeded
3) late delivery
4) logical errors/unreliable
5) high maintenance costs (> 70%)
6) duplication of effort

Next the outcome of the software crises was described. Better languages, better methodologies, and support environments. An APSE is a database comprising an information repository for the entire life cycle. Also, communication interfaces (user, system and tool interfaces) and a toolset - integrated set of tools for entire life cycle. The same kernel APSE (KAPSE) is implemented on

various operating systems to make the environment look the same. Why a database?

1) tools need not know information representation
2) information can be added without affecting other tools
3) flexible attributes and relationships can be constructed
4) transactions
5) version control
6) integration

As always, there is the conflict between flexibility and structure.

The next talk was about a C Unit Test Harness. It performed automatic test driver generation using a test spec file and did execution logging. The spec must have predicted output. The test then is easily repeatable and can be used with a test coverage monitor (profiler).

After the morning break the following session included a talk about automatic generation of syntax directed screen editors. A bottle of fine Dutch beer was introduced. The beer represented students - full of energy. The bottle itself was the Dutch bureaucracy. The bottle was shaken and opened, the students' energy was impressive. The syntax directed editors are generated from the language definitions as represented in the scanner and parser. A generic editor module is linked to the lex and yacc outputs. The editor incorporates a cursor synched parser, as the cursor is positioned, the grammar is parsed. State information appears just below the cursor. There is a window for output. There is an interactive option which caused the editor to enter a dialogue. Some applications were discussed. A cobol editor ("Cobol is a nice language because it is mostly redundant"). A 'sh' editor. It was noted that the shell was not a language, the frustrated efforts to generate a BNF for it were discussed. Its inconsistency was described along with an interesting bug. (Try "<<'ls'" on your machine. We did on most of those in the vendor exhibit. The shell either died, with or without a core dump, or hung interminably. In one case a micro was unable to be rebooted after this experiment was performed.) The editor handled errors as best it could. If only one token was possible it was inserted. If more tokens could be used it suggested one. Otherwise it refused the token with an error message.

We were again subjected to lunch (something yellowish served over rice).

A simplistic network implementation was described. It is very much like the 'net' command which first appeared in Unix 3.0 and suffers many of the same deficiencies. Such a limited approach can however be quite useful, although perhaps not completely satisfactory.

Brian Redman spoke at length about the next version of uucp. Europeans are concerned mostly about phone costs and therefore clamor for more efficient use of the phone line and direct X.25 connections.

Sometime after tea there was a general meeting of the EUUG. A draft constitution was voted in by show of hands and an initial executive committee was approved (I abstained). EUUG has unique problems due to geographic considerations. The idea of free student attendances based on personal recommendations was discussed (scholarships of a sort). The prospect of tutorials to subsidize the technical meetings was put forth. (By the way, EUUG does not pay the lecturer in a tutorial.) And in order to bring down the fees, unbundling of the registration was discussed. For instance, don't include lunch. It's interesting to note that EUUG is somewhat like USENIX was before Boston. They are about to struggle with the same problems of dealing with vendors and keeping the meeting down to a manageable size and keeping it technically rich and informal. I trust they can learn a few things from USENIX's tribulations.

Two talks followed emphasising the anguish Europeans have about character sets. The problem is that 128 is certainly not enough, 256 can be made to do. Unfortunately many programs believe characters are only 7 bits. The first talk was technical, the second was delivered by an EEC bureaucrat to the wrong audience. Both talks had a message; if you think there is a rift between Bell and Berkeley, try getting the Europeans to agree on character set standards.

There was a EUUG sponsored dinner banquet at which several members masqueraded as scheduler states (runin, runrun, etc). There was a speech by Theo de Riddler in which he compared

Unix to a woman, prostituted, abused, etc. Then it was compared to a famous deceased religious figure whom many worship. As a gesture to the past (glorious in its modesty), Fifth Edition manuals were presented to Bill Murphy and Jim Kennedy of AT&T International.

At the bar afterwards there were many interesting discussions with various people. Several of us got into an extended discussion of the future of EUUG (and USENIX) relating to its size and purpose. There was discussion of uucp and ifdefing different algorithms for time and space tradeoffs. There was discussion of a source code stockroom administered by USENIX. Someone from AT&T-BL was very optimistic that all sorts of add-ons would be released soon. There was much interest in V8, the advice was to write letters. There were arguments about the proper placement of network routing algorithms. On the one hand it should be a separate command (mail 'route ist'!dt). On the other it was argued that it belongs in uucp (or mail). Along similar lines there was discussion of the feasibility of a dynamic routing database. How do you deal with an anarchic environment. Nothing will work for long. However any attempt is better than the current situation. There was no tequila and no Amaretto so we were stuck with beer and Grand Marnier. After the management kicked us out from the bar, the diehards reconvened in someone's room. The meeting lasted until after 5 am. I however dutifully retired so that I might prepare for a talk.

On Wednesday morning Joe Carfagno spoke about large systems. A very good presentation describing overall management of a two million line application. The talk covered development, customer interface, management tools, testing, etc. A lot of material covered smoothly and clearly. Interesting aspects of the project described are that it was successful, on time, reliable and efficient. Apparently a key factor was the rich Unix environment.

After coffee Andrew Hume opened his session by briefly admonishing the current practice of criticizing Berkeley merely because it's fashionable to do so. There are plenty of valid criticisms to be made without resorting to aspersions as social rhetoric.

David Tilbrook talked about the five pitfalls of interactive graphics. Boredom, Confusion, Discomfort, Frustration and Panic. He showed a film about "Newshole" (an interactive news copy layout aid he had designed) to make his points.

The next talk was a replay of the "Behind every Binary License" rhetoric from UniForum '84.

After another astounding lunch there were a series of five minute talks on various topics. Andrew Hume talked on /proc in V8. Karrenberg from Dortmund discussed a spooling system implemented in a similar way and at the same time as Berkeley. Greener from Imperial Software Technology gave some EUUG benchmark results. System V is fairly close to 4.2BSD. Speeded up comets (15%) DO EXIST. Tummers from Twente (Holland) described a simple implementation of Dijkstra's semaphores. Mayhew from Bleasdale gave a rudely long talk on implementing IBM bisync on Unix without the vpm. Ridder from Holland described an inexpensive and portable way to benchmark Unix (or most other systems). It consisted of a 68000 master computer controlling a small network of 6809's feeding terminals. Nick Nei from Glasgow gave a fast and furious talk on modeling disk requests. Apart from an unexpected peak at 50Hz, there was no real insight into what is going on.

Then after tea there was a panel discussion of the future of Unix. Each panel member gave a brief statement:

Larry Crume - Need Unix to drive complex micros and loadable device drivers. Predicted people don't want to be given software, that it has to be packaged. Noted that the good old days are gone.

Robert Ragen-Kelly - Concerned that Unix will be made to do things it wasn't intended to do. Urged that Unix be kept small.

Kirk McKusick - Looking forward to receiving his degree. Said that Berkeley will move to function as a research facility. Predicted less formal releases.

Adrian Freed - Ought to be source distributions of everything.

Mike Banahan - People at one time were interested in Unix itself. Now they're interested in what the system can do. Be careful of relying on a single piece of software.

H. J. Thomassen - Unix will go the way of any commercial success. Away from universities towards businesses. Hopes that Unix will provide good administrative packages. Growth of community is fragmented to points where a meeting is meaningless.

Discussion then was involved with who is still interested in Unix in and of itself. The hobbyist market perhaps? The state of Unix was compared to that of the automotive industry in the 40's. A good motor was available, what was needed was bodies and luxuries. Lots of discussion about pros and cons of the availability of source. Discussion of development environments. What about creating them for large teams by stepwise construction from working smaller environments? I've yet to see a really good panel discussion among intelligent level headed participants. This was no exception.

I conclude that the meeting was a grand success. It held my interest throughout and I came home with some ideas and many more contacts. It's too bad Europe is so far away, but I suspect the Europeans like it where it is.

The Future of UNIX

John Lions
University of New South Wales

*[A talk to the Australian UNIX User's Group, Sydney, February 20, 1984.]**

I have been asked to speak on the future of the UNIX system. This will be a personal view of course, and in order that I may not immediately be proved wrong by the revelation of commercial decisions already taken elsewhere, I want to talk about the UNIX system as it might be ten years from now. Predicting ten years ahead seem much easier than predicting ten months - ten years is beyond the planning horizons of all but the largest or most confident companies in this business. And it is possible to get ten year plans very, very wrong, as anyone in this state's Electricity Commission today can tell you. No doubt companies such as IBM attempt to plan ten years ahead - but I would be prepared to wager a large amount of money that their ten year plan for 1974 made no mention of the UNIX system.

1974 - ten years ago - was the year that UNIX went international. The July, 1974 issue of the Communications of the ACM carried to the world at large the paper by Dennis Ritchie & Ken Thompson entitled "The UNIX Timesharing System". This had originally been delivered on October 16, 1973, at the Fourth ACM Symposium on Operating System Principles held at the IBM Thomas J. Watson Research Center, at Yorktown Heights, New York. So IBM had certainly heard of UNIX by 1974.

Ritchie and Thompson's paper has been revised and reprinted on several occasions. Whereas the 1974 version begins: "There have been three versions of UNIX.", the 1978 version states: "There have been four versions of the UNIX time-sharing system.". Note the subtle change in wording: not only had a new portable version of UNIX been developed that ran on the Interdata 8/32, but sometime around 1976, the word UNIX had become aggrandised to become a trademark of Bell Laboratories. As the lawyers will tell you, a trademark is an adjective, not a noun, that must be applied to something that can be made and sold. So will there still be UNIX programmers ten years from now?

In 1974, the announcement of yet another operating system was not overly exciting, whereas ten years earlier, introduction of a new operating system was a major media event.. I would venture to predict that again by 1994, any announcement by a credible authority of a major new operating system will once again be newsworthy, because it will be so rare an event.

The 1974 paper was important to us at the University of New South Wales because it coincided with the decision by the university to replace its existing computer by a CDC Cyber mainframe and brace of DEC's PDP11/40 computers. Serendipity. UNIX became available to us just when we were about to take some bold new steps, and before our user community had developed new addictions to some of the more mediocre software alternatives that, like the poor, always seem to be with us. UNIX was important to us because it allowed us to exploit our own facilities, especially in Electrical Engineering, more efficiently. It also allowed us to deploy our available manpower much more effectively (but that is another story ... I digress).

If we are going to look a long way forwards, then we should first prepare by looking a long way back. History is not bunk. If we do not look back then we will not judge correctly the directions for change, nor accurately asses their force. This need was reinforced strongly for me recently: I had asked groups of third year students in Computer Science to prepare proposals for a computer network development that might serve our university until the year 1990. I was dismayed to find that they projected that the number of computer terminals on campus would only double or triple in that period. Whereas in the last decade, the number of terminals has gone from about 20 to about 500 - an increase by a factor of 25. Perhaps the students may be right in one sense - the

* Originally appeared in *The Australian UNIX User Group Newsletter*, Volume 5, Number 2, March 1984. Reprinted here with permission.

number of terminals for student use may not grow by more than three times - but this will only be because the students themselves will have their personal computers, a possibility that most of last year's students overlooked entirely.

The Past

How far should one go back in time? Forty years is too far: in 1944 Colossus was working for the British against the Germans, but nobody knew. In 1944 in Philadelphia, the ENIAC was being constructed with all of 20 ten-digit storage registers. And the EDVAC, which was to have 2000 words of memory, had been proposed.

Ten years later, in 1954, much was changed, but computers were still modest in their capabilities, and were still very much the exclusive territory of those computing aboriginals, the numerical analysts. The economics of computer programming was already a matter for concern: one of the first automatic algebraic programming systems was introduced by Laning and Zierler for the MIT Whirlwind computer, which had 1024 16 bit words of memory. Backus and company had started work on the first Fortran compiler for the IBM 704. The cost of memory systems still dominated both programming and computer architectural design. Prospects for operating systems - good or bad - hardly existed.

It didn't take too long for things to improve however, and by 1959, the idea of timesharing was suggested formally as a way of improving the productivity of computer programmers, for the first time by Christopher Strachey.

A further five year jump forward in time brings us to 1964, just 20 years ago: several important events happened in that year but one event was of undoubted importance to nearly everybody: the announcement by IBM on April 7, of its System 360 line of computers. (We now know that Edsger Dijkstra regards this as one of the blackest days in the history of computing.) But, for better or worse, in spite of gross defects in architectural design that have now been untidily plastered over, the design continues to this day as the basis of a commercially successful line of computers that has been widely imitated. (We also know that Dijkstra regards the copying of the /360 design by the Russians as one of the greatest coups of the cold war.)

There are several aspects of the IBM/360 that are worthy of comment here:

1. for the first time, a single computer architecture was being presented for a set of computers whose raw power varied by a factor of fifty to one.
2. by using main memory inefficiently it created an insatiable market for memory devices, and a guaranteed source of income for memory manufacturers. This market has remained incredibly active and competitive until today, with benefits for all, and will remain so for the forseeable future.
3. it sanctified the eight bit character, distinguished it forever by the name 'byte'. It legitimised the concept of character addressable memory as an alternative to word addressable memory that had been in vogue since the time of Charles Babbage.
4. it introduced the moving head disk drive as a commercial reality, and with it, the era of low cost random access mass storage.
5. it legitimised the concept of a multiprogramming operating system.

For their sins, IBM did not deliver multiprogramming support until 1967, then only in the form of a totally gauche system called MFT II. But by announcing it in 1964, they saved the bacon of several manufacturers such as Burroughs who were encountering sales resistance in selling their already working multi-programming systems to sceptical customers. Once again, in 1984, IBM seems to have found the need to come to the aid of the party ... but does A.T.&T. really need their help?

In their 1964 announcement IBM overlooked the attractions and usefulness of both virtual memory (as it came to be called) and time-sharing. They thus missed participating in a major development: the world's most ambitious timesharing system, Multics, a joint project involving MIT's Project MAC, the General Electric Company, and Bell Laboratories.

If we look forward another five years, to 1969, now ten years after Strachey, we find that , while Bell Laboratories had withdrawn officially from the Multics project, a small group of researchers at Murray Hill were burning the midnight oil, trying to build a modest replacement - which was to become the system that we know as UNIX.

But before leaving the /360, there is a personal postscript: in 1966, IBM shipped Gene Amdahl, chief architect of the /360, around North America on a good-will fence-mending tour (they figured that if they couldn't ship the software, then perhaps some liveware would keep the troops calm). He said many things but one statement that I remember particularly, because I strongly disagreed at the time, was that the way to build a powerful system was to use only one CPU, and to invest all your efforts into making this as fast as possible.

Back once more to 1974: this was the time of proprietary operating systems - when each manufacturer sought to convince the marketplace that not only was its hardware better in every way than that of its competitors, but that its operating system, and also its Fortran compiler, was the best there was. UNIX, in the role of yet another orphan operating system, and almost naked without its Fortran compiler, had a hard road to hoe. The recent history of UNIX does not bear repetition in detail now: it survived and prospered because it did a lot of things right, and it did hardly any things wrong, and it did some things not at all.

The Present

UNIX has often been criticised because of features that it does not have. In most cases, where a prized feature of some other system is missing (take index sequential files as an example), one can be reasonably sure that:

1. Dennis and Ken did consider it as a possibility;
2. either they decided that it wasn't such a good idea after all (consistency and economy were important goals); or
3. they have not been satisfied by any of the proposed ways for implementing the concept (e.g. inter-process message passing).

In 1979, five years ago, the Seventh Edition of UNIX was released by Bell Laboratories. This was the last release to be prepared for external distribution under the supervision of Ken Thompson and Dennis Ritchie in the Computing Science Research Center of Bell Labs. Since then major organisations have been developed both within Bell Labs and also at UC Berkeley to do the same task. In 1979, Dennis Ritchie cam to Australia and explained how many of the things in UNIX had developed the way they have. In that year also, I went into print saying that I believed "the UNIX system is a phenomenon whose full influence has not yet been experienced". I certainly do not feel any need to retract that statement now.

Today, in 1984, there exists an ecological niche for a standard, machine independent style of operating system that can be applied to a wide range of machine sizes. This niche has existed for some time, but few perceived it. It has always been there just as long as there have been talented engineers around with new ideas on how to build better and cheaper computers. CDC, SDS, DEC, Interdata, Prime all started that way - and to a greater and less extent, because there was no other way, they have all faced and then stumbled over the software hurdle. New entrants in today's race, such as Pyramid and ELXSI, don't have to jump the same hurdle, provided they can push past all the other competitors trying to leave via the gate marked UNIX. Why should UNIX be the one to fill this niche? Because it is the first system to get its internal structures approximately correct, but nevertheless to conceal these effectively so that they can evolve. Also, it has gained wide-spread acceptance just at the time that the need to fill the niche has become really pressing. Cynics might say that there is another reason: that during its formative years, it totally lacked the quote-unquote support of any major computer manufacturer. Imagine what might have happened if DEC had decided to jump on the UNIX bandwagon in the mid-1970's. They might have been able to smother it with kindness!

It really is trite to suggest that UNIX will become a de facto standard - the domino effect has already started. Soon, once we have outgrown the present generation of personal computers, which

are still a little too modest and toylike in their capacity, UNIX will outpace CP/M and MS/DOS and such like, and become the de facto standard for personal computers. Since Cray computers have already announced their intention to join the fold, we can now see one software architecture spanning a range of computer systems, whose powers vary by a factor of at least one thousand - a significant achievement by any standards, and one that IBM would gladly claim, if it were its own.

The Future

The future of UNIX as a widely used system is now secure, and it will remain so for several years. There are several questions we may ask:

1. will UNIX survive as long as ten years?
2. if so, will it be subject to serious challenge by 1994?
3. if so, where will this challenge come from? will it come from without or within?
4. how will UNIX change in the next ten years? will it survive intact or only in parts?
5. will it be able to adapt fast enough?
6. will it be able to cleanse itself of accretions contributed by legions of well-meaning system hackers?

I am not sure in what order I should attempt to answer these questions:

Technology

Throughout the history of computing, the dominant force for change has been the evolution of hardware technology, mostly in the USA. VLSI, the latest dominant technology is almost unique in that, if you want to make something faster or more reliable or more efficient or more economical, then you should make it smaller! Everything points in the same direction. There are not the tradeoffs of speed v. efficiency, or safety v. reliability, say, that confront the automobile designer. This fortuitous circumstance has created opportunities for innovators to cash in, leading to phenomena like Silicon Valley - the cancer of modern capitalism - and the rapid developments that we now regard, not as surprising, but as inevitable and expected.

Microchips are already incredibly cheap - take a walk around your local hardware store and find how few low technology items you can buy today for less than $10. But, as hardware technology innovation is not slowing, the next decade should bring yet another hundredfold or more improvement in CPU and related device price/performance. I am reminded of the story about the customer in the Rolls Royce dealership who asked the salesman how many horsepower did the engine have? And in measured tones, the salesman replied "Enough sir, enough". While most of us will have to stick to our Mazdas and Toyotas on the roads, I see no reason why, by 1994, we should not all have the equivalent of a Rolls Royce personal work station.

By 1994, even modest personal computers will have a few megabytes of main storage, and several MIPS of processor power. New ways of soaking up these resources will have to be found: program swapping to and from secondary storage, once the hallmark of all timesharing systems, has already almost passed into history. The UNIX system block buffer cache has long ago expanded way beyond the modest recommended 15 blocks, or rule of thumb 'two per user', for Sixth Edition UNIX. But there must be a cache size (say one megabyte on a personal computer) beyond which it is not worth going. Large bitmaps for controlling displays on CRT screens will consume much storage in future, but one megabyte per screen should be ample for most purposes. That probably still leaves a few megabytes to go, so there will still be a problem. If it is going to adapt to this, UNIX will have to learn not to throw away information, just because no one seems to be using it anymore. For example, program texts (even without the sticky bit) will stay around in main memory even when no one is executing them. Thus there will have to be changes in the way UNIX assigns and manages resources - it will have to learn to be much more laid back - but there is no real difficulty here!

Organisation

Changes in hardware technology have always been accompanied by changes in computer architecture and organisation. But while technology has contributed improvements in price performance of the order of 10 to power 8 over the last 40 years, improved organisation has contributed, charitably, at most 10 squared. Register sets have been expanded, replicated, generalised, specialised, hidden and revealed; instruction sets have been enriched and expanded (but now it is fashionable to contract them again); pipeline techniques have been used to reduce effective instruction execution times; and memory accessing and addressing schemes have been greatly improved.

But today, in 1984, things seem ready to change: economies of scale in manufacture favour the small CPU more than the large one, and Grosch's law has been repealed. New approaches to computer organisation have become feasible and attractive. There seem to be many people, with virtual shoeboxes full of microprocessors, wondering what would be the best way to string these together to create super-fast, high capacity calculators. But as the designers of Illiac IV will verify, it is much easier to propose than to deliver. Such systems pose new problems for the operating system designer. The problem of course depends on the style of organisation chosen.

For so-called tightly coupled systems, where several processors share access to a common memory, significant changes (equivalent to the addition of sets of Dijkstra's P & V operations) need to be made to the UNIX kernel. These can be done; and have been done effectively. Thus there is no theoretical difficulty in using UNIX in the traditional multiprocessor configuration.

But multiport memory is expensive, and not suitable for more than a handful of processors at a time. There is an alternative system model that is frequently proposed: a set of processors each with its own private memory, but able to communicate with the others via a suitable network e.g. an Ethernet channel. This model is easy to describe and seems to be conceptually tidy, especially at the hardware level. However, it suffers greatly from the problems inherent in using messages for communication between sequential processes. There are so many things that can go wrong when messages are passed around among groups of potentially unreliable devices that the associated protocols are heavy handed, and such systems have, in my limited experience proved to be constipated, and a disappointment to their designers and advocates. Many people seem determined to build and use such systems. Surprisingly, many of these are using UNIX as the basis of their software development even though, as has often been observed, UNIX is lacking in inter-process communication facilities. My attitude here is best summed up by saying that I have become converted to Amdahl's viewpoint: no matter how attractive the multiprocessor solution appears performancewise, a single processor solution will be better.

Networks

But this is not to say that networks will all wither and fade away. Far from it. UNIX itself has already spawned a new phenomenon - the loosely-ravelled, lightly coupled network exemplified in Australia by the SUN network, and in North America by USENET. The distinguishing features of such networks are:

1. limited bandwidth connections;
2. mixed processor and software types;
3. owned and controlled by groups with vastly different management styles and viewpoints;
4. cooperation is for data transfer, and not in general, for load sharing.

Networks such as the SUN allow remote login and hence a limited amount of load sharing. personally, I find this only of interest, and hence start to migrate around the network, when the system I happen to be using has become depressingly slow (rare) or because it has ceased to function. (This was true when I was preparing this, which happened to coincide with an overdose of preventative maintenance for the machines in our laboratory. I never did locate a peaceful machine that day. But before the day was finished, I had generated an explosion of copies of all the files of current interest to all four machines that I use. And then, with the pain still fresh in my memory, I did the same again next day. Now, sometime soon, before I forget, I will have to go around and tidy up the pieces.)

File Sharing

While I seriously believe that load-sharing will not be a major consideration in the networks of the future - most of us don't really know how to keep the equivalent of three or four VAXes busy - I do believe that file sharing will be of prime importance. Attempts have already been made with varying degrees of success. But if I ask you to consider whether, given that the networks of the future will consist not of dozens but thousands of machines, whether a simple, apparently hierarchical scheme will be sufficient. As an example, suppose someone has removed my copy of the program 'adventure', but I know that somewhere out there, someone else is sure to have a copy. Should I be allowed to execute (as presumably I may under the Newcastle Connection) the command

```
cp /../*/bin/adventure /bin
```

and hope for a reasonable result?

As long as communication channels are less than perfect and of less than infinite capacity, then people are going to want to make local, accessible copies of distant data objects. We will want to make, and keep temporarily, local copies of distant files. Eventually, getting rid of these files - the garbage collection problem - will become serious. There are programs, for example 'uudiff' that can determine whether two apparently similar files on different machines are identical. When such a pair is detected, then we can most probably happily abandon one of the copies. But what if they are only almost identical - which one should we keep? or should we keep both? who decides?

There is another technology-derived dimension to this whole problem. In a few years, information will no doubt be stored more economically on-line than as black marks on white paper. Ten years from now, it will undoubtedly be possible economically for any person to keep on-line, if he or she wishes, a file copy of every major document or program he or she ever writes. I suggest that if you attempt to do this, then you will have a real database problem on your front door-step. Does anyone sell real personal database systems yet?

Perhaps you regard problems such as these as minor. I suggest that ten years from now they will have reached plague proportions. The solution may have to be drastic: a complete reevaluation of files, and how to manage them. If you reread Dennis Ritchie's 1979 paper, you will find that, in a real sense, that is where Ken Thompson and Dennis, with some help from Rudd Canaday, all came in - fifteen years ago.

Bytes or Bits

There is another aspect of UNIX systems to which I wish to draw your attention: the internal structure of UNIX files. As you know, there is almost none - a UNIX file is just a sequence of characters. Officially, any character set will do, but if you look more closely, it is fairly clear in the present implementation in several places that these had better be eight bit ASCII characters. UNIX programmers are only half joking when they refer to their CRT terminals as glass teletypes. UNIX may have thrown over the yoke of the Hollerith card - we should all be grateful that 80 is not a magic constant anywhere in UNIX - but it is tied to at least one aspect of mid-century electromechanical hardware. So far this has not been irksome, but I think it soon will become so once the new generation of bit-mapped terminals really arrives. The latest version of 'troff' represents characters internally as 32 bit quantities, because there is size and font information , as well as a value, to be accommodated. In the future we may also want to represent colour and texture, for example. What will be the best character size then?

It seems to me the only dependable magic constant is the value one. A machine with a bit-addressable memory is going to be attractive both for dealing with arbitrary communication links and for managing bit-mapped graphic displays. Sonner or later, some silicon packer is going to do the obvious thing and produce a microprocessor with 32 bit addresses, and addressing down to the individual bit. The Burroughs 1700 system showed how it could be done, and if Burroughs sales and marketing had not been so completely inept, such systems might have been commonplace already.

If machines that deal with variable length bit strings as their stock in trade do become common - and I believe there are strong reasons to investigate them further - then this could be one development that will rock the present UNIX design to its foundations. I am sure of one thing: in the new born-again UNIX file system, files will achieve their ultimate evolution as just 'a sequence of bits'.

The User Interface

In the coming decade, some features of the UNIX system are going to survive better than others: some will emerge almost unscathed; others will have become almost unrecognizable. The user interface is one of these.

Many new users of the UNIX system are unhappy with the user interface. A reasonable response from an old hand to a new chum making such a complaint is to suggest that he might like to change it (after all the user interface is not set in concrete). By the time the user is capable of remodelling the interface he usually doesn't worry any more. I am told that this is not what is usually meant by a user friendly interface!

The Bourne shell represents a landmark in control language design: it should not be criticised because, for instance, it does not maintain and edit scripts of commands entered previously. Such features are useful, desirable, and should exist. But they do not belong in the shell program, because they are more generally useful than just that. They belong in yet another layer (should we call it the veneer?) of the user interface.

Such arguments aside, the standard UNIX interface is about to be transformed by the arrival of the new generation of terminals with bit-mapped displays. For reasons that can't be discussed here, the so-called Blit terminals have not yet escaped outside Bell Labs, but within the Labs they are already proving a potent force for change. Keep watching for further developments. The standard UNIX user interface is not going to be what it used to be.

What else?

There are many other aspects of the UNIX system that can be singled out for individual comment besides the file system or the user command interpreter and its accompanying conventions.

For many people, UNIX is the set of commands in Section One of the UNIX Programmer's Manual. Many of these, thanks to the efforts of Brian Kernighan and Bill Plauger, have already been born again as "software tools". Commands such as *grep*, *awk*, and *diff* are based on the results of serious original research, and represent important contributions to computing technology. I would venture a bet that by 1994, the Oxford English Dictionary will include an entry for 'grep': a synonym for search sequentially for a particular pattern.

The number of programs that are candidates for inclusion in Section One may be expected to grow enormously - one or two decent Fortran compilers may even be amongst them.

For some users, the important parts of UNIX are defined in Section Two of the Programmer's Manual, not in Section One. This defines the interface between user programs and the kernel, and represents the place where hackers like to hack. Standardisation of UNIX means above all, standardisation of this interface. There is a real chance that the current standardisation effort may succeed. There are both good and bad aspects to this.

Finally, there is the matter of the UNIX philosophy - will it survive? Many newcomers to the scene hardly seem to know that it even exists. Perhaps, like the chivalry of the knights and heralds of centuries past, it will be quietly assigned to the dusty pages of history. So let me remind you that there is a UNIX philosophy, and that it it precious, subtle, tantalising, and fragile.

I was appalled recently when the representative of one computer manufacturer, newly jumping on the UNIX bandwagon, announced that "of course they would be rewriting all the entries in the programmer's manual to conform to the company's corporate standards." I always thought that the erudition, charm, humour and conciseness of the UPM was one of UNIX's secret weapons: a real breath of fresh air compared to the competition.

One of the greatest threats to the original UNIX tradition is what I will, unfairly no doubt, refer to as the Berkeley UNIX tradition. Rob Pike tried to demonstrate this in his recent talk to UNIFORUM entitled "Cat -v considered harmful'. The original UNIX tradition is not to overload commands with armsfull of options, so that each can do a multitude of tasks badly, but to find a modest set of tools that can be usefully combined to do many tasks well. That is why history files etc. do not belong as part of the shell - they are potentially much more useful if they can be provided as a separate facility that may be used elsewhere as well.

A similar argument may be used to suggest that record locking should not be part of the UNIX file system. Since UNIX files can reasonably be as small as one likes, they can be as small as anything worth locking individually. And if you can then show that directory searches will really become too expensive, then there are certainly ways to speed up directory searches. The important thing is to identify the real problem.

Likewise many simple database enquiries can already be solved satisfactorily by a straightforward application of 'grep' to a sequential file. But remember, it is the application that is straightforward, not the 'grep' program itself.

In summary, UNIX will change in the next ten years, both for better and for worse. In its new-found maturity, it will surely miss the careful nurturing of Dennis and Ken: perhaps their greatest contribution has been the many attractive, but inessential features that they left out of their design. It seems inevitable that with so many well-meaning but undisciplined people now out to add their own improvements, the average quality of the system will decline, but not too disastrously, I hope.

Hardware and other developments are possible in the next ten years that will invalidate, or challenge, many of the basic assumptions on which most systems rest today. Challenges from new system designs are also inevitable so long as there are people with bright ideas and enthusiasm. It is rumoured that Ken Thompson is once again thinking about a new system design: if he does produce it, then I am sure we should all sit up and take notice.

On the design of the UNIX operating system

Peter Collinson

Computing Laboratory
University of Kent,
Canterbury, Kent, UK
ukc!pc

ABSTRACT

An extremely inaccurate view of the history of the design of the UNIX operating system is presented.

Why is UNIX successful?

The computer world seems to have gone 'UNIX mad', and it is hard to understand why. One good reason is the portability of the system but there must be more to it than that. Most people who use the UNIX system seem to like it even though it is full of idiosyncrasies, is terse to the point of unhelpfulness and consists of a very large number of totally forgettable commands. I think that the success of the system is summed up by the following paragraph.

> *The UNIX system is successful because the minimum number of keystrokes achieve the maximum effort. In addition, the system says very little to explain errors and relies on the intelligence of the user to deduce reasons for failure.*

The statement describes UNIX V6, which we all know is the parent of the UNIX systems running today. History tells us that the guys who designed it did their own typing into the machine. It seems to me that because of this, the main reason that UNIX enjoys/suffers from terse input and output is not through any intellectual design decisions made at some early stage but because the UNIX designers were just bad typists working on slow peripherals.

Let us examine the evidence.

Ancient history

First, there are all those incomprehensible and forgettable two letter commands: **ls**, **cp**, **mv**, **pr**, etc, etc. What delight it was to type three letter commands: **cat**, **cdb**, and **dsw**. It is interesting to note that if you were to use the text formatters, it was assumed that you could type. So, to use **nroff** you had to make five painful keystrokes, admittedly only four different letters were used. Of course, all the commands were in lower case because the designers couldn't find the shift key on the keyboard, never mind the shift-lock.

Secondly, look at the main means of the input of text, another two letter command, **ed**. Every command in the editor is a single letter, which are just about mnemonic. Every error message is a single letter, '?', which is just about meaningless‡. However, only a very few commands are needed to do basic editing operations. If the user wants to do something complicated, then something complicated must be typed in. This complicated thing is often very short and incomprehensible to anyone other than the author. Is there anything more minimal than regular expressions in **ed**? Learning to use **ed** is a continuous voyage of discovery, users find that less and less typing is required as time goes on. What joy it is to finally find out how \(is used. **Ed** is written in C and is available for hacking, and the alterations made to it by many hackers in many places were thought to be required to make the editor more powerful. That is, you could do more in less keystrokes. So, **ed** is aimed at bad typists, and it developed to support bad typists.

C is another example of design for bad typists; *integer*? - no, *int*; *structure*? - no, *struct*;

‡ OK folks, I didn't forget about the immortal TMP error message and the fact that '??' means something special.

begin/end? - no, '{ }'; ':='? - no '=', after all it's easy to type '==' ; long variable names? - yes, but only 8 characters were significant, so it wasn't worth typing any more; and the list could go on and on. C is so terse that it is enigmatic, and was obviously designed for coding by bad typists who are actually typing the code themselves. Most other languages seem to be designed for people who are paid by the number of inches of code they write on paper for other people to input into the machine.

Apart from the famous patented setuid bit, the notion of the use and exploitation of software tools is the main feature of the UNIX system which has become important and perhaps even academically respectable. But, did those guys really sit down over a cup of coffee one day and say "hey, I've had this really neat idea, let's code a bunch of simple programs onto this experimental file system, provide some way of joining the output of one program to the input of another - and yes, a good name would be UNIX". Heck no, one of them said to the other "I'm really up to here with this typing, can't we use the output of that program into the input of this, it'll save me lots of time, which I can spend in playing some really neat games". So, software tools were born.

So, the evidence is very strong that bad typing played a large role in the design of the UNIX system. With its terse input and terse output, UNIX V6 was a joy to use - I am a bad typist too.

More recent history

A number of improvements were seen in UNIX V7. Notably, better typists were employed to design and implement things. The Bourne shell, **sed** and **awk** are examples of this. Those original guys had got a bit better typists too, there are a few more comments in the kernel and some longer names. C had altered and generally requires more typing. All the new things have long names: **typedef**, **unsigned** variables, and the type checking of function returns meant the input of function specifications at the start of the program.

UNIX was ported to a number of machines using the very wordy portable C compiler. However, ports of the system to machines with long names were not successful and the enduring port was to the VAX architecture, because VAX is short enough for everyone to type. And then came the Computer Systems Research Group of the University of California, Berkeley.

It is very fashionable to be rude about UCB systems these days. This rudeness all stems from typing envy, because those people at UCB can really type. It seems that applicants for posts at CSRG, UCB have to pass a typing test to get in; the test consists of typing the full title of the institution without using the delete key.

These really great typists contributed mightily to the development of UNIX. For example, they virtually invented the useful comment in the kernel code. It is suspected that the people who understood the comment *You are not expected to understand this* don't understand why comments are useful. However, UCB hackers tend to show off their skill somewhat and embarrass others by producing really big programs. Most of the rudeness about UCB system stems from the scorn poured onto these big programs, people seem to forget that other UNIX influences have generated really big programs. For instance, you can't get the original f77 V7 compiler into a PDP-11 unless it has separate I/D space.

AT&T headed towards re-organisation and the oddly named UNIX System III made its appearance. It was odd, too, because it is based on the PWB UNIX system which not many people had used. It didn't feel like UNIX because lots of things had moved and had their names changed. With System V, AT&T tell us that this is **THE** UNIX operating system and hopes it will rule the world. On initial inspection, System V is well typed and has long manuals. Error messages are now wordy and it is almost possible to say that you don't need the source to find out why something happens the way it does. In truth, the current situation is that the UNIX systems which the outside world can obtain are in the hands of people who employ others to type - or so it seems. Typists have moved into the UNIX world, and those of us who are still bad typists complain that UNIX V6 was the last real UNIX system.

Meanwhile, back in Bell Labs, those guys have conquered the typing problem by designing high quality terminals where all you need to do is point. But that's another story.....

Information About EUUG Services

The European UNIX Systems User Group (EUUG) is a non-profit making organisation formed in 1976 to exchange information about the UNIX operating system, and has grown since then to encompass most European countries.

It exists to keep UNIX users in touch with one another, and with developments in the UNIX world. Many leading authorities on the system are members of the EUUG and the aims members share are those of any user Group, namely the improvement of usage, knowledge and ultimately of UNIX itself.

The EUUG has links with other UNIX Users Groups worldwide, the aim is to have members of all groups feel they are part of a global UNIX users network where information is exchanged freely, services of sister organisations are readily available, and the groups are working in harmony toward common goals.

The EUUG has a number of services available to its members.

Secretariat

The EUUG has a professional Secretariat which acts as a central coordinator of the User Group's various activities, and which can always be approached with initial enquiries on any Group business. The Secretariat channels enquiries to the relevant source when it is unable to provide the information itself, and is responsible for the organisation of meetings and conferences and also the publication of the Newsletter.

EUUG Secretariat
Owles Hall
Buntingford
Herts SG9 9PL
England

tel: +44 763 71209

National User Groups

The EUUG is composed of affiliated National User Groups. Each National User Group keeps its members in touch with one another, and manages services such as membership lists, hardware information on sites, software availability, legal issues, etc. A representative of each National User Group sits on the Governing Committee of the EUUG. The Governing Committee also appoints a small Executive Committee to deal with the day to day running of the EUUG.

Newsletter

Since you are reading it, there should be no need to introduce this quarterly bulletin of what is going on in the UNIX world, should there?

Cooperation with USENIX

The EUUG has recently worked out an arrangement with USENIX, the equivalent UNIX Users Group organisation in North America, whereby newsletters, other publications and distribution tapes are being exchanged and available for redistribution by the respective organisations.

The EUUG has available to its members the USENIX newsletter **;login:** and proceedings from the bi-annual USENIX conferences. A yearly subscription to ;login: costs US $25,-, backcopies are available as long as stocks lasts (US $ 7.50 per copy). The proceedings from the Summer 1983 Toronto USENIX conference cost US $30.-. These publications, along with information about USENIX, can be obtained from the EUUG Secretariat.

Conferences

The EUUG holds two major conferences each year, bringing together experts from all over the world, not just Europe.

The conferences tend to be small, friendly affairs, as much given to the exploration of gastronomic and alcoholic delights offered by diverse European cities as to pushing back the frontiers of technology, etc.

EUUG Distributions

The EUUG has a number of tape distributions available which can be ordered from

EUUG Distributions
Centrum voor Wiskunde en Informatica
Kruislaan 413
1098 SJ Amsterdam
the Netherlands

In order to keep the price as low as possible and to keep the preparation time as short as possible, we only use 1/2-inch, 9-track magnetic tape at either 800 or 1600bpi as distribution media; requests for other formats or media can not be handled.

Make sure you describe fully which distribution tape(s) you wish to receive, and that you attach the proper licence agreement(s). All license agreements with AT&T International will be verified with AT&T International, so make sure that your institute address on the license agreement and on your distribution request (as well as your membership) does not conflict.

You will be billed by the CWI for the distribution; no prepayment is necessary.

Redistribution of EUUG Distributions or any material drawn from EUUG Distributions is not permitted without a written agreement from the EUUG.

Software distributions:

- EUUG D1 R6 UNIX V7 for small PDP machines
  V7 source license minimum. Price US $42.
- EUUG D2 R1 VU Pascal compiler (UNIX V7)
  V7 source license minimum. Price US $42.
- EUUG D3 R3 UUCP (not bug free), news, etc.
  UUCP: V7 source license minimum
  News etc: no license (no UUCP sources).
  Price US $63 (two tapes: $17 software, $46 news articles of last year).
- EUUG D4.[1-5] STUG tapes ($50)
  no license
  UNIX, RSX, UNIVAC, mit(2) formats
- EUUG D5 R1 benchmarks
  no license. Price US $17.
- EUUG D6 USENIX 83.1 ($75)
  V7 or V32, SIII or no license
  license dependent distr. set

EUNET - The European UNIX Network

An electronic news and mail service is available to all EUUG members. This is based on and is part of the highly successful UUCP/USENET world-wide UNIX computer network. In Europe the network is structured by having a *backbone site* in each country, which is responsible for feeding traffic to all other sites in that country, and also talks to one central European site (currently *mcvax*) which has the responsibility of maintaining links to North America.

Currently, the most common means of connecting the sites together is by using auto-diallers, modems and the telephone network. However, X.25 links are now becoming available, and some sites are already connected by this method; it is hoped to have such links to North America in the near future.

Currently (June 1984), there are about 150 connected sites in Europe (including the offshore islands). The maintenance of the network in Continental Europe is done at *mcvax* in Amsterdam, The **Netherlands**. Questions and requests for information/connection (membership of the EUUG is a prerequisite for connection) should be directed to the backbone site of your country or to *EUUG EUNET*, same address as *EUUG Distributions* above, or phone +31 20 5924127.

Backbone sites in Europe:

Country	Organisation	Phone	Electronic Address
Austria	none*		
Belgium	Erik Blockeel Vrije Universiteit Brussel Medische Informatica Laarbeeklaan 103 B-1090 Brussel	+32 2 4781520/1438	vub!erik
Denmark	Keld Simonsen University of Copenhagen Institute of Datology Sigurdsgade 41 DK-2200 Copenhagen	+45 1 836466 ext 14	diku!keld
Finland	Juha Heinaenen University of Tampere Computer Science Division P.O. Box 607 SF-33101 Tampere 10	+358 31 156180	taycs!jh
France	Humberto Caria Lucas CNAM Laboratoire d'Informatique 292 Rue Saint Martin F-75141 Paris CEDEX 03	+33 1 2712414 ext 438	vmucnam!humberto
Germany	Daniel Karrenberg Universitaet Dortmund Informatik/IRB Postfach 500500 D-4600 Dortmund 50	+49 231 7552422	unido!dfk
Greece	none*		
Italy	soon*		
Netherlands	Piet Beertema CWI Kruislaan 413 1098 SJ Amsterdam	+31 5924112	mcvax!piet
Norway	none*		

Sweden	Bjoern Eriksen ENEA DATA Svenska AB Box 232 S-182 23 Taeby	+46 8 7567220	enea!ber
Switzerland	D. Weigandt /DD CERN CH-1211 Geneva 23	+41 22 834940	cernvax!dietrich
UK	???? University of Kent Computer Laboratory Canterbury CT2 7NF	+44 227 66822	ukc!lmcl

* contact mcvax in the meantime

EUUG — NATIONAL GROUP CONTACTS

Mr. M. Rafter,
The University of
Warwick,
Computer Unit,
Gibbet Hill Road,
Coventry, Warwick,
CV4 7AL

(UK)

Mr. Teus Hagen,
Centrum voor Wiskunde
en Informatica,
Kruislaan 413,
1098 SJ Amsterdam,
The Netherlands

(NLUUG)

Mr. Bjorn Eriksen,
ENEA Svenska AB,
Box 232,
S-183 83 Taby,
Sweden

(EUUG-S)

Mr. I.R. Perry,
L.E.R.S.,
58 rue de la Glaciere,
75013 Paris,
France

(AFUU-Sec)

Mr. Keld-Jorn Simonsen,
Indre By-terminalen,
University of Copenhagen,
Studiestraede 6 o.g.,
DK-1455 Copenhagen K,
Denmark

(DKUUG)

Mr. J.M. Helsingius,
Tontunmaentie 32 AS,
02200 Espoo 20,
Finland

(EUUG-F)

Daniel Karrenberg,
Universitaet Dortmund,
Informatik IRB,
Postfach 500500,
D-4600 Dortmund 50,
Germany

(GUUG)

Mr. S. Kenyon,
Ferranti Cetec Graphics,
Bell Square,
Brucefield Ind. Est.,
Livingston,
West Lothian,
EH54 8BY

(UK)

Dr. M. Van Gelderen,
NIKHEF-K,
Kruislaan 411,
Postbus 4395,
1009 AJ Amsterdam,
The Netherlands

(NLUUG Sec)

Hans Johansson,
Logica Svenska AB,
Norra Stationsg. 79-81,
S-113 33 Stockholm,
Sweden

(EUUG-S)

Mr. Bernard,
A.F.U.U.,
c/o 152 bis, avenue
Marx Dormoy,
92120 Montrouge,
France

(AFUU)

Heikki Arppe,
VTT/ATK,
Lehtisaarentie,
Helsinki,
Finland

(EUUG-F)

Joachim Wolff,
Universitaet Dortmund,
Informatik IRB,
Postfach 500500
D-4600 Dortmund 50,
Germany

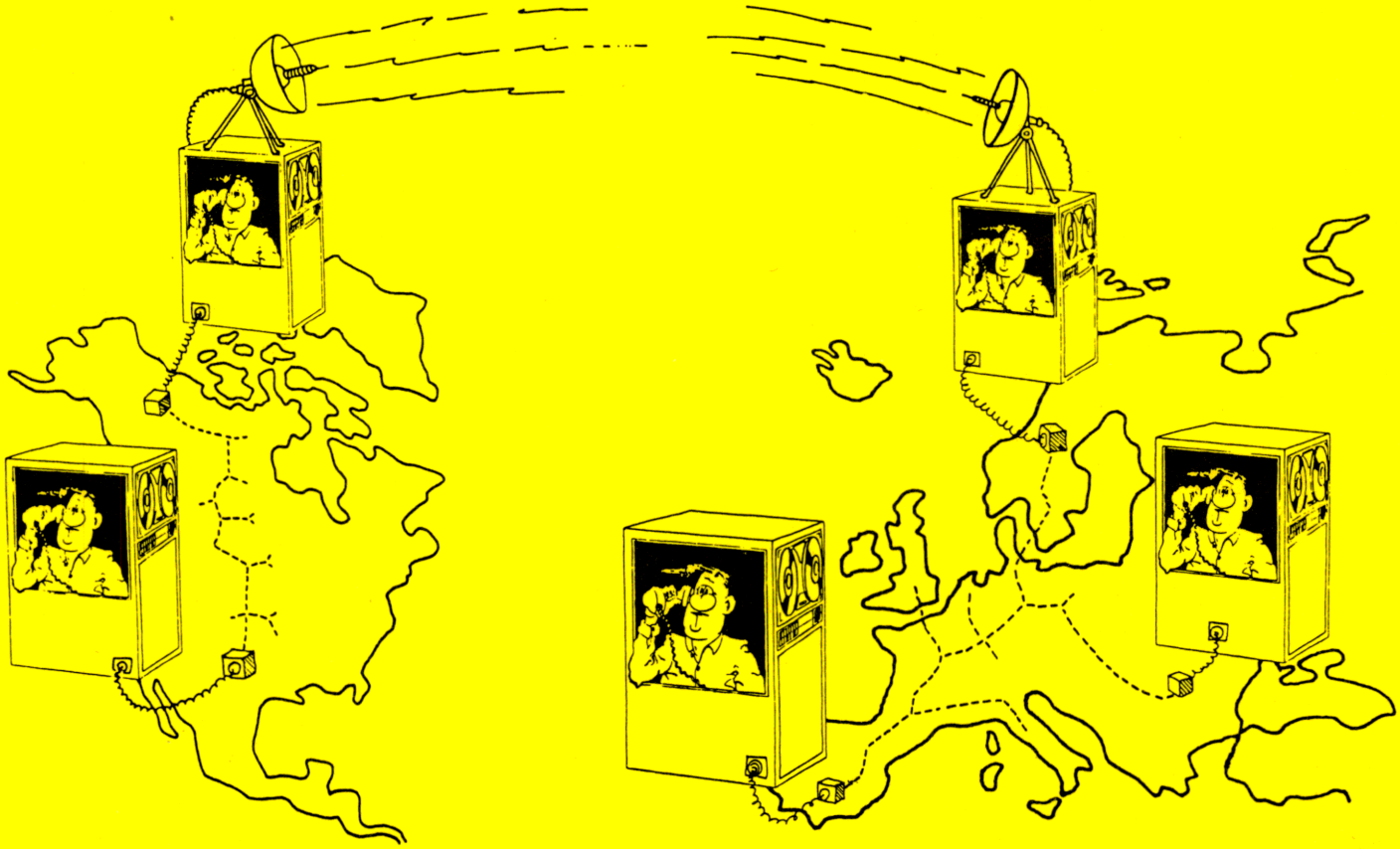
(GUUG-Sec)

Giuseppe Molinari,
Systems and Management
Piazza Solferino 7-
10121 Torino,
Italy

(EUUG-I)

Dr. Marco Mercinelli,
CSELT,
10148 Torino,
Via G. Reiss Romoli 274,
Italy

(EUUG-I)



The Secretary
European Unix® Systems User Group
Owles Hall
Buntingford, Herts.
SG9 9PL. England
Tel: Royston (0763) 73039 (UK)
or +44 763 73039 (Overseas)