EUUG

EUROPEAN UNIX® SYSTEMS USER GROUP NEWSLETTER

Volume 5, No. 1
SPRING 1985

EUUG
European UNIX† Systems User Group

Newsletter Vol 5 No 1

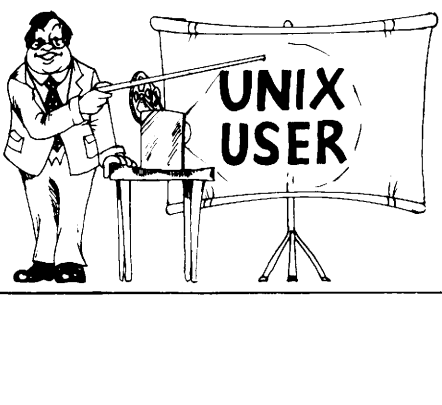
Spring 1985

† UNIX is a Trademark of Bell Laboratories.

Required: Course Instructors for UNIX, C, Pascal and Software Engineering

We are a leading company in the field of continuing education and run a wide range of short intensive courses for engineers, programmers and managers. Our clients include some of the largest international corporations as well as many small high technology companies. To cope with demand, we need more course instructors, particularly for **UNIX, C, Pascal** and **Software Engineering** and also for **Ada, Multiprocessor systems** and **networks.**

You would only be required for a few weeks scattered throughout each year. It is flexible and can be arranged to fit in with your other commitments. If you are interested in this challenging, well-paid work, please send details of your background to:



The Microcomputer Unit
235 High Holborn
London WC1V 7DN

or telephone: (01) 405 3020 for further information.

European UNIX System User Group Meeting
Palais des Congrès, Paris 1st — 4th April 1985

Peter Collinson
Secretary

Introduction

A report of a conference headed 'April 1st' may lead you to be somewhat suspicious. But, yes, there really was an EUUG conference starting on that date. It was held in the Palais de Congrès in the centre of Paris.

There were three days of conference meetings and Network Events Ltd. organised a three day exhibition which started on the second day of the sessions. Allowing a free day to look at the exhibition was a successful idea, and was a reaction to the problem at Cambridge where attendees at the conference had to choose to miss a session to go to the exhibition.

The programme chair for the meeting was David Tilbrook, in positively his last appearance in the rôle because he returns to North America later this year. I would like to take this opportunity to thank him for everything that he has done for EUUG in the last two years. I suspect that a high proportion of the speakers who have allowed their bodies to be subjected to the punishment known as 'long distance air—travel' have come to Europe because of arm twisting (or the threat of constant abuse in the mail) from David. It also is true to say that most of the speakers have enjoyed the conferences and hopefully returned to their homes telling their collegues of the good time which they had. This makes it easier to get other people to contemplate a trip to Europe in the future. Anyway, David expended a considerable amount of nervous energy and electronic mail handling time in setting this meeting up. The result was, I think, a good set of talks.

There were also a number of changes in the overall format of the conference. First, speakers were given longer slots than at previous conferences. This meant that there were only two speakers per session rather than the three which were squeezed in before. It had been felt that the short 20 minute talks in previous conferences did not give speakers enough time to get to grips with their subject. Actually, it also seems that 35 minutes is not enough time for some speakers. I suspect that people are just used to talking for an hour and have difficulty in pacing themselves for a shorter period. However, the longer talks were an improvement.

Second, this was the first officially multi-lingual conference; there was simultaneous translation into French, German and English. The original intention was to provide translation between French and English, but the German interpreter came as part of the package. I have never experienced simultaneous translation before and I think it worked.

One French speaker was a bit confused when the audience burst into gales of laughter because the English interpreter had a coughing fit at about 90 decibels. The laughter continued when the interpreter asked "did I translate something funny?" One problem was that the conference was mostly in English. Linguistic chauvinism meant that many people did not use the headphones during completely English speaking sessions and were then totally unprepared for the inevitable question in French from the floor (I did this too). Still, the translation enabled non-English language speakers to talk at an EUUG conference, and this **must** be a good thing.

The third change to the normal course of events was to give about two minutes to the session chairs so that they could say something about their local UNIX user group.

Many of the speakers supplied papers which were printed in a book entitled *Papers presented at the EUUG Spring Meeting, 1985* which was handed out in the Registration Packs. Another innovation — proceedings were printed *before* the meeting! The book is available from the EUUG office for the very reasonable price of £5.

Oh well, enough of this drivel; what follows are the abstracts for the talks plus some comment from me. Thanks again to David for getting the abstracts and sending them to me. This was no small job; getting abstracts from speakers is hard enough, getting the Papers even worse, but David maintains that getting mail through my machine is the hardest thing which he was obliged to do for the conference.

Day 1 - Monday April 1st, 1985

As usual this wasn't Day 1 for me, I had already managed to clock up two nights in Paris because the cheapest air fare requires a stay over a Saturday night and it seemed reasonable to pay a short visit to my brother who lives there. What he didn't tell me was that we were invited to a party. This would have been fine but we didn't get back to the flat until 3am. I will never be quite sure which 3am it was, since France is an hour later than the UK anyway and the powers-that-be decided to change to daylight saving time at 1am on the Sunday morning.

Day 2 of my visit, the Sunday, was taken up with Committee meetings of different varieties, a late meal and fairly early to bed to try and get over the lack of sleep and the two hour time change. I was in bed by midnight local time but the body kept saying "this is much too early" and refused to allow me to sleep. Still... conferences do seem to be about lack of sleep.

After a rather disorganized pre-organised breakfast of session chairs and speakers, I managed to make it into the Salle Bleue in the Palais, grab my headset and find a chair at the front. After the last conference in Paris spent sitting on hard benches, these chairs were sumptuous and comfortable — just the thing for sleeping in ... good job I had to take notes.

9.43am Session chair: **Michel Gien, CNET PAA/TIM**

Michel officially opened the conference and introduced the Chairman of AFUU, the French UNIX user group.

9.44am **AFUU**
Jean-Louis Berand, Chairman, AFUU

The French group has 250 members. The group has grown from 8 members to its present number in three years. From the outside, It does seem that the French scene is being driven along much more by industry and software houses than the gleaming towers of academia. The group is involved with a commercial publishing company (CXP) to produce a UNIX software catalogue.

9.49am **Conference Organisation and Logistics**
David Tilbrook, Imperial Software Technology Ltd.

David introduced the program chairs and some of the speakers who were not speaking until the last day.

10.07am **VLSI Assist in Building a Multi-Processor UNIX System**
Ian Johnstone, Sequent Computer Systems

Abstract

Multiprocessors have been of interest to computer scientists and designers since the first computers. Three factors have limited the commercial success of multiprocessor systems; entry cost, range of performance, and ease of application. Recent advances have removed these limitations, making possible a new class of multiprocessor systems based on VLSI components.

The requirements for constructing an efficient MP system are detailed, including: low level mutual exclusion, interrupt distribution, inter-processor signalling, process dispatching, caching, and

system configuration. A solution to these problems is described and evaluated.

Comment

I think that this talk was interesting because it represents the start of the new era of applying specialised VLSI chips to solve computing problems.

The main goal of the work was to provide a low cost, high performance multi-processing system. The base model of the system which was adopted was a fairly normal one, with several CPU's attached to a system bus supporting memory and I/O devices. In Sequent's system, a special control bus parallels the system bus and is used by a VLSI device called a *System link and interrupt control* (SLIC). There are thus two busses in the system, a high performance bus for data and a specialised bus for control functions. This keeps each bus simple with lower cost and higher reliability.

The SLIC chip is a 6000 gate custom CMOS array.

☞ Coffee ☜

11.15am Session chair: **Peter Collinson, University of Kent**

Well, my 'news from' was information about the UK part of USENET which I have fallen into administration more through misfortune than design. The UK network is now around 55 map entries and is growing in leaps and bounds — new sites are turning up every day and our mapping administration system deals with at least one map update every two days. The network is really a star based on **ukc**, this is not good and we are busy re-organising the topology.

This information did not raise a single laugh; which was to be expected because networks are intrinsically boring, they serve to enable people to communicate and should work with no problems. Pity they don't.

11.23am **Concurrent processing in Ada and UNIX**
Henk Hesselink, Delft University of Technology

Abstract

Over the past years interest in programming languages supporting concurrent processing has grown. Some of these languages are extensions of existing ones, e.g. concurrent Pascal, others were designed to support concurrent processing from the start. An interesting (if only because of its backing ...) example of the second group is Ada which is likely to become a major language in the future.

At Delft University of Technology a group is working on DAS, the Delft Ada Subset, which implements most of Ada except for its concurrent processing features. It therefore seemed like an interesting exercise to compare the support for concurrent processing in UNIX and Ada and to see to what extent this part of Ada could be mapped onto UNIX.

It may seem a little odd to compare a programming language with an operating system, however UNIX in combination with a programming language (especially C obviously) offers a similar kind of programming environment to Ada. Ada has simply integrated into itself what in UNIX are separate system calls, in the process ensuring that such facilities are standardised. The comparison is therefore between Ada on the one hand and UNIX plus a programming language on the other. Because the programming language was C the combination is referred to as **cunix**. The version of UNIX discussed in this paper is Version 7, so as not to cloud the issue with features present in some of the more current versions of UNIX but not in others. Mention is made of the effect of some of these features on the UNIX view of concurrent processing.

11.45am **MIRANDA: an advanced functional programming system**
David Turner, University of Kent

Abstract

There has been an upsurge of interest in functional programming in the last few years, and a great deal of progress has been made in the design of functional languages. The spread of these new ideas has probably been delayed by a lack of widely available high quality implementations. The MIRANDA system has been developed to make available to the UNIX community a modern functional programming language, which it is hoped will become a *de-facto* standard. The language draws its features from the earlier languages SASL, KRC and ML, and is embedded in an interactive system providing the tools needed to build large applications. The talk will give an overview of the main ideas in the MIRANDA programming language, and will discuss the way in which the nature of UNIX has influenced the design of the MIRANDA system and facilitated its development.

Comment

Looking further forward than the next release of UNIX is something which we should perhaps do more often. I have been keen to get David along to an EUUG conference for some time. He enjoyed himself, "functional programming conferences are usually so dull affairs", he told me after the conference. David's talk was an introduction to the subject.

Functional languages are not a new idea but a new style of language. Conventional programming languages such as FORTRAN or C are (1) based on assignment statements and state changes and (2) are sequential, there is explicit flow of control. There is evidence that (1) and (2) have harmful consequences and have lead to the current software crisis.

Functional languages are descriptive rather than imperative. There is no assignment statement. There is no explicit flow of control. Such languages are FP, pure LISP and SASL.

The main features are (1) conciseness, programs are shorter and easier to develop; (2) transparency, the programs have good mathematical properties; and (3) parallel executability, it ought to be possible to execute in concurrent processing environments. It is this last point which has the hardware and VLSI designers jumping up and down because hardware behaves in this way.

MIRANDA is a modern functional language implemented under UNIX. The system provides a good programming language and an interactive environment for programmers. The main properties are (1) the language is purely functional; (2) it allows infinite data structures by delaying evaluation; (3) it has a set abstraction and (4) it has polymorphic strong typing.

David gave a few short examples, like the *quicksort* algorithm in around 20 characters.

He went onto talk about the MIRANDA programming environment. The program works in interactive 'desk calculator' mode with a set of equations being input in any order and a result requested. The compiler works in conjunction with a screen editor which is currently **vi** but this is user selectable. There is an on-line reference manual which is menu driven. The system allows full access to the UNIX shell (which is also nearly a functional language) and there is a useful library of built-in functions.

The current implementation is not fast, but faster implementations are coming. Because of speed limitations some possible uses for the system are: teaching, research, providing a vehicle for formal specifications and a system for fast prototyping.

☞ Lunch ☜

Everyone who had looked at my previous reports had supposed that this one would be about wine; well, the presence of good wine needs no special mention in France. Instead, I think that this conference was about lunches; which were excellent.

2.35pm Session chair: **Jean Wood, Digital Equipment Corporation**

Jean's 'news from' piece was about Decus Europe whose next meeting is 16th - 20th September at Cannes. Slides of Cannes, beaches, boats and some parts of some bodies followed.

2.38pm **The influence of the UNIX operating system on the development of two Video Games**
Peter Langston, Bell Communications Research

Abstract

The Lucasfilm Games Group was established to explore ways of applying the technological and methodological expertise developed at Lucasfilm (principally for film production) in a new entertainment medium — video games. One of the major tools on which that expertise is based is the UNIX operating system. It is an interesting coincidence that this operating system's early development was heavily influenced by games and the interests of game designers.

This paper describes the development of the two video games "Ballblazer" and "Rescue on Fractalus!" and the ways in which UNIX software aided and influenced their design.

The work described here was done at Lucasfilm Ltd., San Rafael, Calif.

Comment

This was a video tape of what had been produced with a very large piece of software running on a UNIX system. Very large pieces of software are often written in LISP, and this was no exception. The programs run on an Atari games machine and were very small pieces of software in comparison. The Atari machine is a 6502 processor chip plus a four channel sound chip. The graphics were really good for the games and I particularily liked the way the sound (some tunes and other noises) had been constructed to enhance the games.

3.14pm **3-D Computer Graphics, the UNIX Way**
Tom Duff, AT&T Bell Laboratories

Abstract

The UNIX text processing tools are successful because they are simple, yet powerful, single-purpose operators which consume and produce documents in a simple, universal representation. Pipes and a programmable shell allow the simple operations of the individual commands to be combined in powerful ways. The data structures and compositing operators described in 'Compositing Digital Images' (by Thomas Porter & Tom Duff, in Siggraph '84 Proceedings) can be the foundation a similarly powerful image synthesis and processing environment.

Recently, I have extended the Porter/Duff model to handle anti-aliased full 3-D images, and am developing a collection of 3-D rendering tools based on it. Each of the tools is a simple, single-function operator that consumes and produces pictures in this common format. For example, the programs that render fractal terrain and quadric surfaces use unrelated (even incompatible) algorithms. Nevertheless, since the programs produce output in a common format which the 3-D compositing program also reads and writes, we can render scenes which contain both sorts of object. I will describe the data representation and its 3-D compositing operation. A short 16mm film will illustrate how this scheme allows simple special-purpose rendering programs acting in concert to produce rich images.

Comment

A stunning film produced by some small pieces of code. Tom's main message was to underline the fundamental tools notion of UNIX. A quote which I liked: "The good thing about UNIX is not the code but the thought those people had; to use small pieces of code to generate big things in big ways." Interesting how different people's work in totally different areas have such large parallels, David Turner's message of 'small is beautiful' is fundamentally the same as Tom's.

☞ Coffee ☜

4.17pm Session chair: **Daniel Karrenberg, University of Dortmund**

Daniel spoke a few words in un-announced German to test the interpreters. He then talked in English about the German UNIX† user group. The German group has 151 members. They have one meeting a year, the first one was in Frankfurt in February this year. They also produce a newsletter, the intention is to generate more than four a year.

4.21pm **Image Synthesis with UNIX**
Etienne Beeker, INA (Institut National de l'Audiovisuel)

Abstract

Realistic representations of three-dimensional scenes is a very attractive goal, object of research for several laboratories for the last few years. Our team, at INA (the Institut National de l'Audiovisuel), has been writing such programs for two years, with the aim of creating special effects for video productions.

Two kinds of programs are now operational:

- Synthesis programs: These programs take a data base of geometrical objects as input. The data base consists of polygons and/or bicubic patches, and parameters such as colours, light sources, position of the virtual point of view, transparencies, etc. After the elimination of hidden areas and calculation of each pixel intensity with an appropriate illumination function, these programs produce as output an image in full colour, with shading, reflections, etc. One of them, using a scan-line algorithm is completely operational. Other programs based on ray tracing and particle systems are still under development.

  These programs run under UNIX on a Perkin Elmer 3210. They require a lot of CPU though, and we expect a more powerful machine, like a VAX-785 or a Gould 3297.

- Design programs: These are more conventional CAD-like graphic programs. They allow the design of skeleton of objects in a wire frame, either with polygons or with mathematical surfaces like B-splines or bicubic patches. Geometrical smoothing of shapes is also implemented. An animation program interpolates object movement and the point of view. Movement control can be done quickly because of the wire frame display.

  These programs are running on a French 68000 based super-micro called SM90, with UNIX. Graphic output is done on a bitmap display. Coordinates are digitized with a tablet. These programs don't need much computation time but are highly interactive. That's why run them on a machine used as a personal workstation. The data base is then transferred to a more powerful machine for further calculations.

All our programs are written in C. The Perkin Elmer is running Edition VII (a V7 with some Berkeley enhancements), the SM90 is running UNIX V7.

Video production slides and animations produced with these programs will be shown, as well as a demonstration of the design programs running on a SM90.

Comment

The abstract sums up the talk well. It doesn't say that the system is being used for real to produced short 'jingles' for French TV.

† UNIX ist Warenzeichen der AT&T International

4.52pm UNIX at IRCAM

David Wessel, Robert Gross, IRCAM

Abstract

Since 1976 IRCAM, (Institut de Recherche et Coordination Acoustique/Musique) has supported research on fast real-time digital signal processing, room and instrument acoustics, psycho-acoustics, compositional algorithms, as well as different methods of sound synthesis including the singing voice and physical modeling. The results of this research have been applied by invited composers in many contemporary music pieces with real-time digital electronics and computer generated tapes.

IRCAM has developed a series of high speed digital signal processors culminating with the 4X machine recently used by Pierre Boulez in his work *Repons*. General purpose computing and program development is done on a group of VAX, SUN, Plessey, and Valid computers networked together. Work with artificial intelligence and expert systems has affected a majority of the current research projects.

Two years ago UNIX became the underlying foundation for all the research and musical production at IRCAM. UNIX must not only drive the 4X and support standard program development including numerical computation, and compiler design but also must do musical sample computation, storage, and real-time playback or record operations. This latter problem is not easy to deal with in UNIX and has been solved by using the advantages of UNIX files to simulate a hierarchical sound-file system based on the UNIX model. We are currently using the CARL (Computer Audio Research Laboratory) sound-file system developed at UCSD.

This talk will explore some of the current development projects at IRCAM including the 4X software design, Acoustical research, Chant/Formes project, applications on the array processor, and the development of music oriented software for the Apple Macintosh using the SUMACC's (Stanford University) environment. We will also discuss our experience with the CARL software and our attempts to utilize the 4.2BSD file system to support the I/O throughput required by the DA/AD conversion.

Comment

IRCAM is funded by the French Government through the Minister of Culture and the Minister of Research. They also have sponsors in Switzerland and the US. They have about 50 members of staff. Their work is entirely musical and their aim is to aid composers to learn about the new technology.

Their computing environment is based on a 4.2BSD system running on a VAX-780. The machine has an array processor and D/A and A/D conversion systems. The machine is connected via Ethernet to some workstations which are used for graphical input. They deal in large volumes of data, for instance, two seconds of 16 channel 32 Kb/sec digital data for the converters occupies two cylinders on an RM05.

There followed some tape recordings of some of the stuff which is being done. One was a totally synthesised singing human voice, it was somewhat eerie. Another was the output from a program which modelled the physics of a violin. The original program sounded like a beginner because they hadn't programmed the bowing action properly. A later example of synthesised jazz violin was terrific.

The session finished with a video tape of a system which connects a flute player with an automatic accompanist. First, the piece was played straight and the accompanist followed it. Then, the player tried to put the program off; he made changes of timing and played 'badly' by missing notes or playing wrong ones. The accompanist followed. I suddenly had this vision of a small box which buskers on the Tube take with them to accompany their efforts.

All impressive stuff and this was followed on the last night of the sessions by a concert at IRCAM itself.

☞ End of Day 1 (??) ☜

No, it wasn't. The formal meeting were immediately followed by a small session called 'Birds of a feather' sessions. Actually, they were short presentations.

5.35pm **The SM90**
??

Sorry, I didn't get this person's name and I grovel most humbly. I think that he was from IRCAM.

This was a brief description of a UNIX system running on a multiprocessor machine — the SM90. The usual pattern of a shared system bus is repeated here, but every CPU has its own local bus supporting memory and I/O devices. The global memory is used for system tables and local memory in each processor contains user processes and system code. The selection of which processor is to be used by a process is made by the user.

5.58pm **The Network File system**
Brian Novak, Pyramid

Pyramid have adopted Sun's network file system for their machines. This talk was brief introduction to its use.

The design goals of the NFS were: transparent remote file access, a simple implementation, the use of a general purpose protocol and the implementation of a fast file system.

The system does not attempt to be a distributed UNIX system, nor was it designed to provide access to devices on the network or file locking.

The NFS uses a stateless protocol which makes easy recovery from temporary interruptions. Local file caching gives the system good performance. It is possible to connect file systems running on architecturally different machines because the protocol has a canonical form of data representation.

To my mind, the hardest thing to get right on these systems is the area of file security and access. The NFS insists that a user has a unique **uid** across the network and I think this is entirely unrealistic. The system prevents super-user access across the network, but this is just one obvious thing to do. In my experience, potential system busters look first at the people who are able to become super-user. Still.. no doubt the system works.

6.15pm **UUCP matters**
Various

How this got onto the agenda, I am not sure. When the moment came, there was no-one to speak to it, so I got lumbered. I did not think that much came of the meeting, but we did manage to air the problem of funding mail on the European network. There are no real solutions, unfortunately we are all running software which was intended to run without accounting or controls.

☞ End of Day 1 (it really was, this time) ☜

And onto a wine reception paid for by UNIX Europe, for which many thanks. Unfortunately, I had to leave and go to another UUCP meeting, this time with backbone administrators. That is best glossed over — except that it seemed to be held in a sauna because we were unable to succeed in opening the window in the room and the temperature soared.

Day 2 — Tuesday, 2nd April

9.36am Session chair: **Peter Langston, Bell Communications Research**

Bell Communications Research is an R&D department which is funded by the local telephone companies in the US. It should not be confused with Bell Laboratories, which is the research arm of AT&T. Peter talked about the trivia at the last USENIX meeting. It was interesting that the program placed second in the Go tournament was one which contained no strategy but just did random moves. The next USENIX meeting is in Portland, Oregon starting on June 12th.

9.43am **VIDMAN:**
Didier Galmiche, Laboratoire d'Informatique Theòrique et Programmation

Abstract

VIDMAN is new software running on a 4.2BSD UNIX System and enables the creation of a visual manual. First of all, we should say that this product was necessary to enable the 'broadcasting' of our research works. The best encouragement we have received is the unanimous support from LITP's members.

Our main idea was to build up a manual that could improve at the same time as its contents. As a matter of fact, continuous updating is quite impossible with a manual written on sheets of paper; whereas it becomes almost natural with VIDMAN.

We think that this is a useful tool which is easy to use and to a new way of communication for UNIX users. Moreover, VIDMAN can be used to describe any sort of information, either about lectures, new languages and theories or description of systems.

Thanks to judiciously chosen controls it offers unusual freedom in document styling and consultation. **Nroff** style commands are used for information formatting creating paginated text having a user defined style. For inspection of the information, **Emacs** style key definitions allow access to the desired information. Three types of pages can appear on the screen: menu pages (index pages with lists of the stored items), text pages (information pages corresponding to each item) and mixed pages containing the two last types in any proportion.

VIDMAN can prepare output for all terminal types and allows the utilization of the full resolution of each terminal, thanks to the termcap database.

Finally, we must say that VIDMAN is written in the C programming language with use of some libraries of programs such as **curses** and **termlib**, the use of **lex** (Lexical analyser generator) being essential. This means that VIDMAN is specific to UNIX.

Comment

Another system done for a French TV channel. The system is aimed at efficient manipulation of an on-line manual. The data is tree structured where the leaves are display data and the nodes are either shell commands or point to leaves.

The system is in use at LITP and IRCAM.

Q. Can you store an update history?

A. Yes, this should be possible but isn't currently implemented .. that's a good idea .. we had better do that.

Q. Is it available?

A. Yes, in a limited form.

10.10am **System Aspects of Low-Cost Bitmapped Displays**

Dave Rosenthal, Information Technology Center, Carnegie Mellon

Abstract

The design of low-cost bitmapped displays is reviewed from the perspective of the implementors of a window manager for UNIX. The interactions between RasterOp hardware, the multi-process structure of UNIX, and the functions of the window manager are discussed in the form of a check-list of features for hardware designers.

Comment

Dave started off by talking about what is happening at CMU. The Information Technology Centre (ITC) at Carnegie Mellon has the aim of generating a workstation based environment. Every student who comes into CMU will be required to buy a workstation which will have 1 megabyte of memory, perform at 1 Mip, have a bit-mapped display, and a mouse. All the machines on the campus will be linked by a high bandwidth local area network and the workstations will have access to a network file system which will encompass all the machines on the campus. The workstations are to run 4.2BSD and the machine currently in use is the Sun workstation. This may change because the system is being funded by IBM. ITC currently has about 30 people and has a lot of hardware.

The network is very large, currently every research and teaching room have Ethernet or Pronet connections and there are about 500 taps into the net. All the campus machines (6 DEC-20's, about 70 VAX's, some HP machines and of course some IBM machines) use the DARPA IP protocols. There are plans to have at least 8K taps onto the network by 1987. It is proving expensive to wire the campus, the projected cost for wiring every room is in the region of $8 to $10 million.

The network file system provides each workstation with global file access to a large UNIX tree-structured namespace. Currently, there are two file systems in production, one with 50 clients on four servers and one with 80 clients on six servers. A total rebuild of the system is taking place, this is based on the results of the current versions.

The user interface to the system on the workstations uses bit-mapped graphics. ITC have developed a portable way of addressing the bitmap display giving window access from any machine in the network, not just the workstations. The interface allows easy use of UNIX from menus, icons etc. Users have uniform access to high quality multifont text. The interface is currently running on a network of about 100 Suns, the software drives both the Sun1 and the Sun2 hardware and avoids the built-in Sun RasterOp hardware. The software requires no kernel modifications. ITC seeks to encourage development of educational applications with a uniform, easy-to-learn interfaces.

The window manager is a user level server process, unlike Sun's. Clients do remote procedure calls over TCP/IP stream connections, so clients can access a window on your workstation from anywhere on the network. Output is via lines, multifont text and RasterOps. The window manager process tracks the mouse, implements menus and de-multiplexes keyboard input.

After all that, Dave managed to get onto talk about his paper. I quote the concluding paragraph: "We have set out a number of points worthy of consideration in the design of low-cost bit-map displays intended to support multi-process interaction. Although RasterOp hardware may appear attractive, it needs careful design if its potential is to be fully realised, particularly for passing characters. Assistance with cursor drawing and sharing of the colour map may be more cost effective uses for limited hardware resources."

Q. Can we get the software?

A. Yes, the window manager, base editor and other software is available on a tape to selected (mostly academic) sites. The tape also contains some public domain screen fonts derived from the TEX fonts. The address to contact is:

Distribution Co-ordinator,
Information Technology Centre,
Carnegie Mellon University,
Schenley Park,
Pittsburgh, PA 15213.

☞ Coffee ☜

11.20am Session chair: **Jim McKie, Centrum voor Wiskunde en Informatica**

Jim introduced Johann Helsingius who talked about the Finnish UNIX User group or FUUG. Johann started in Finnish to confuse the interpreters and later refused to translate what he had said — perhaps the conference competition should have asked for some guess about the content of his initial sentence.

FUUG has 80 members but expect the membership to grow because they have not followed up contacts made at an initial meeting. The proceedings of the meetings are in Finnish. There are about 20 UUCP sites in Finland. The main problems are concerned with character sets, the characters '{', '}', '[', ']' and '\' are used for special characters; text written in Finnish looks like C to the **file** program.

11.27am **UNIX networking via X.25**
Radek Linhart, Hewlett-Packard GmbH

Abstract

X.25 packet switched networks provide a far more reliable and cheaper data transfer method than asynchronous links via telephone lines. This talk will give a brief overview of the typical features provided by national X.25 networks. In addition, different alternatives of network access will be discussed.

Finally, a Hewlett-Packard company internal implementation will be introduced. A flexible, easily configurable approach with a high degree of security has been taken.

Comment

This talk was a good overview of the communication problems associated with running the UUCP network. Radek wished to connect the European Hewlett-Packard sites into USENET which is used for a considerable amount of internal company mail and information.

He came to the conclusion that, for European sites, X.25 connections were 8 times cheaper, 8 times faster and 100,000 times more reliable than telephone connections. It is also considerably more expensive for US sites to get X.25 connections than it is for European sites.

The system they have designed uses standard UUCP protocols (and Piet Beertema's f-proto) and has a program which allows high level programming of X.25 PAD parameters.

11.55am **ACSNET — The Australian Alternative to UUCP**
Piers Dick-Lauder, University of Sydney, Dept. of Computer Science

Abstract

ACSNET is a network with goals to serve a function similar to that currently served by the UUCP network. Routing is implicit, and addressing absolute, with domains. The network daemons attempt to make use of full available bandwidth on whatever communication medium is used for the connection. Messages consist merely of binary information to be transmitted to a handler at the remote site. That handler then treats the message as mail, news, files, or anything else. Intermedi-

ate nodes need not consider the type of the message, nor its contents.

Comment

Piers started by talking about the Australian UNIX User Group - AUUG. It has around 250 members and around 40 network sites. The next meetings are in Brisbane on the 26/27th August and in Perth on February 1986. Piers then talked about his paper.

ACSNET is the Australian alternative to the UUCP network and has obviously benefited from being designed with the notion of wide area networking rather than this function being grafted on in an *ad-hoc* way at a later date. The basic requirement was to have the ability to send a message from one point to another, a message is not simply mail but can be any sequence of bytes used for any purpose. The network is constructed from a set of nodes connected by any medium capable of connecting machines — phone lines, Ethernet, X.25, twisted pairs, etc. The system runs better over 8-bit data paths but this is not mandatory. Any link may fail during transmission and care is taken to restart the transmission where it left off rather than restarting from scratch.

Addressing is domain based, which is a better model than the routing address used by UUCP. Messages may be broadcast to all machines on the net or in a sub-domain. Users can also explicitly route messages using 'bang notation', this is not strictly necessary but has proved useful for testing and loopback messages.

OK folks, let's all throw UUCP away....

☞ Lunch ☜

2.30pm Session chair: **Teus Hagen, Ace**

Teus talked about the Netherlands UNIX User group, NLUUG. They held a meeting in March which 150 attended, the content was a 'technical sales talk'. Old hands thought that it was dreadful, but the audience liked it and asked for another. The next meeting is in the last week in November and will be a single day with parallel sessions consisting of technical talks, talks from distributors, and an exhibition. The meeting will be in the Congres Center in Utrecht. The Dutch group are also involved in producing the UNIX Products Catalogue, which will become an EUUG publication; and of course, are also prime movers of EUNET.

2.39pm **Addressing in MMDFII**
Steve Kille, University College, London

Abstract

The Multi-channel Memorandum Distribution Facility (MMDF) is a powerful and flexible Message Handling System for the UNIX operating system. It is a message transport system designed to handle large numbers of messages in a robust and efficient manner. MMDF's modular design allows flexible choice of user interfaces, and direct connection to a number of different message transfer protocols.

This paper is intended as a sequel to the paper presented by Doug Kingston at the 1984 Usenix meeting in Salt Lake City. A brief technical overview of MMDF is given, and then two aspects are considered in more detail:

(i) Address handling. The table-driven approach taken by MMDF to handling a structured address space is described. The extension of this approach to use with distributed nameservers is considered.

(ii) Message reformatting. A number of systems using similar, but distinct, message and address formats have emerged. The approach taken by MMDF to allow interworking is described.

A comparison with other systems, in particular **sendmail**, is made.

Comment

MMDF was initially developed in 1979 by Dave Crocker to supply a mail system for CSNET. A production system was running in 1980. The new version was started in 1982 to take advantage of some of the new ideas from **sendmail** and also to take account of various alterations in protocols, specifically ARPA 822 and the peculiar UK protocols.

The goals (or perhaps features) of MMDFII are: robust and efficient message relaying; the support of multiple message transfer protocols; the ability to deal with several user interfaces; authentication of messages; and authorisation control on basis of sender and/or recipient. A big feature is the submission time address checking which allows clean user interface design and minimises address transformation.

Steve then delved deep into the addressing mechanism, which is best left to his paper.

3.07pm **What is an International UNIX System?** **Duncan Missimer, Hewlett Packard Company**

Abstract

UNIX system users overseas are getting tired of talking to UNIX systems in whatever language it is that UNIX systems speak. Some people have responded by hand-crafting variants that speak the local language and properly handle exotic character sets. Others have created sub-environments that support different languages but do not offer the full functionality of the UNIX system. The authors believe that it is possible to create a fully international, fully functional UNIX system which can assume more than one linguistic identity at run time.

This paper presents a model for language and country custom independent software, which we use to outline some of the ways in which a vanilla UNIX system is unsuitable for use outside the English-speaking research environment.

We define two terms — native language support (NLS), and localization - and show how these concepts can be used to design an international UNIX system by moving language or country custom dependent information out of the source code and into the file system where it belongs.

Comment

The talk really pointed to the problem areas for linguistic change for utilities which run on the UNIX system. The problem areas are:

1) The system assumes the use of 7-bit ASCII and often the eighth bit is used for specialised functions in programs.

2) Collation sequences in UNIX are based on the ASCII numeric sequence. This is not even adequate for American dictionary order, and much less for a language like Welsh where two symbols are used to represent a single character ('ll', 'ff', 'rh' etc).

3) Directionality, the assumption that languages goes from left to right is plumbed into the system.

4) Classification of characters in programs assume a seven bit character set, this is insufficient for other varieties of character set.

5) In a multi-lingual environment, standard escape sequences will most likely to be used to indicate a change of character set. Software which processes these sequences may have to dynamically alter its treatment of the characters in any of the ways discussed above when an escape sequence is found.

6) Hyphenation and spelling — currently, **spell** and **nroff/troff** hyphenation support English only.

7) The computation and display of the date and the time is very US specific.

8) The names of the days of the week and the months are in English.

9) The names of currency units and ways of subdividing currency units requires alteration.

10) There is significant world-wide variation in the representation of numbers, variation in the symbol used for the radix character, and variation in the symbol used for grouping digits, as well as the number of digits grouped.

11) Error messages, prompts and responses to prompts and mnemonic command names should all be based on the user's language.

12) Messages built up in chunks from programs may have to alter in order when translated into another language.

Well, that list was taken from the paper, I thought that it was useful to include it here so that the magnitude of the problem can be envisaged. The main solution seems to be to comb through the code removing language dependency, HP's idea is to use a termcap-like language specification file and access the file from a variable in the environment.

3.29pm **Internationalisation of UNIX**
Gary Lindgren, UNIX Europe

AT&T are also addressing the problems mentioned above and wish to provide a framework and tools for supporting local character sets, error messages in the local language, and a multi-lingual help facility.

There are plans to remove the 8th bit usage in editors.

There are plans to enhance the operating system, utilities, languages and applications software to support international requirements in the areas of date and time format, collating sequences and numeric representation.

The System V Interface description contains a preliminary specification for the support of 8 bit and 16 bit character sets. This uses the top bit to indicate which character set is in force; coupled with a special shift character to indicate changes from one character set to another. My suspicion was that the scheme was a little complicated and would not really work. This whole subject seems to be a can of worms and very difficult to tackle in any easy way — but at least some people are trying.

☞ Coffee ☜

4.17pm Session chair: **Sunil Das, City University, London**

Sunil is the chairman of UKUUG, the UK chapter of the EUUG. The group has 337 members and is the largest in Europe — for historical reasons. The main objectives of the group are to provide a UNIX information service; to provide a forum for discussion in the UK; to hold one-day technical sessions; to aid and support UKC's efforts in administering UKNET.

4.24pm **Greek Characters on UNIX**
Steve Hull, Research Centre of Crete

Abstract

A case study is presented of a project to provide full Greek alphabet capabilities on a UNIX system. The particular difficulties of the Greek language in its various forms are discussed, as well as the full ramifications of providing "full capabilities" for a variety of hardware. The necessary tradeoffs are discussed, their limitations and advantages evaluated, and alternate approaches compared. It is hoped that, in addition to illuminating the particularly European problem of providing multilingual

capabilities, we illustrate a worthwhile approach to solving UNIX problems in general.

Comment

The title of this talk was given as Ελλ*ηνικοœ Χαρακτήρεs στο* ΙΟΥΝΗΞ†. This was a really interesting talk which illustrated the complexities of the written Greek language and proposed some solutions to the problem. Again, I cannot do the paper justice here.

4.57pm **The SINDEX Chinese language project**
Paul Thompson, SINDEX

Abstract

The problem of Chinese text input derives from the large number of Chinese characters (10 to 20 thousand) and the way they are mapped on to a small number of monosyllables (approximately 1200). The solution is (1) to avoid direct input of characters (i.e., the requirement to specify uniquely the character intended as we are accustomed to doing in western languages); (2) to input syllabic units phonetically; and (3) to use the linguistic constraints on syllabic combinations to convert the phonetic input into Chinese characters.

This solution has been implemented on UNIX where the script conversion program has been written in C and the UNIX tools have made it possible to manipulate the relatively large linguistic data-base efficiently.

This talk will briefly discuss the problems and other solutions and will concentrate on the the use of linguistic constraints in our solution.

Comment

The talk started with a short piece of video-tape from a UK TV programme showing the system in operation. People involved in the mechanical reproduction of Chinese usually ask about the speed of operation of the equipment being used and not the speed of the typist. Speed is not a problem with Paul's system. He has utilised a phonetic representation of Chinese which is taught in schools before writing in ideograms is taught. The system recognises the phonetics and makes a best guess about what character is to be output. It turns out that a high success rate is achieved by looking at the combinations of syllables and making judgements about them from their order.

5.20pm **EUUG Annual Meeting**
Teus Hagen, EUUG Director

Abstract

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

Comment

Teus had several matters of report. The first was to note that Emrys Jones has resigned as Director of EUUG after three years toil. The Executive Committee has decided to award him an honourary membership. Teus was elected Director of the EUUG at a Governing Committee meeting.

EUUG now has member groups in the UK, Eire (is just thinking of breaking away from the UK group), Norway, Sweden, Finland, Denmark, Netherlands, Germany, Belgium (is just breaking away from the Dutch group), France, Switzerland (looking at EUUG to see whether it is a good thing to join), Austria, Italy and Greece. Some countries in the Arabic world are thinking of joining.

The national groups all elect two representatives to the Governing Committee who in turn have control over the actions of the Executive Committee (who pretend to do the real work). The current Executive Committee is: a Director, Teus Hagen; a vice-director, Michel Gien; a secretary, Peter Collinson; a newsletter editor, Jim McKie; a financial manager, Mike Banahan; and a member without any special responsibility, Keld Simonsen. The business manager is Helen Gibbons.

† ΙΟΥΝΗΞ is not a trademark of AT&T Bell Labs.

The constitution had been voted on by postal ballot. There were 150 replies, 147 of these were 'for', 2 'against' and 1 abstained. The constitution has thus been accepted by a fairly resounding majority,

Teus spoke of the various issues which were causing discussions in the various committee meetings and also amongst the several committee members in bars (or perhaps one should say café's) in Paris.

I suppose that the main topic of concern is the issue of what sort of conference should be held. Is it important to have large externally organised exhibitions? When the exhibitions were organised by ourselves, the exhibitions used to subside conferences to a great extent and now they raise very little revenue. They seem to cause the conferences to be held in expensive locations which in turn cause conference attendences to drop. Should the conferences be shorter? Longer? Should the conferences contain more sales promotion talks? Should the conferences include tutorials? Should we set up standard SIG's to provide smaller groups for meetings at conferences? We should raise some of these issues for discussion on the network. This, of course, does not get to everyone — perhaps people should submit their views for publication in the newsletter.

The newsletter is another issue for discussion. The fundamental problem is lack of copy and general input from the membership. There seems to be some contradiction because in general, members say that this is a useful thing for EUUG to do. However, there have been newsletters which have consisted totally of contributions from me. This is not healthy and eventually I will run out of asinine jokes.

The next conference will be in Copenhagen from September 10th to the 13th. There will be technical sessions from the 11th to the 13th, industrial sessions on the 10th and 11th and an exhibition running from the 10th to the 12th. One of the guest speakers at the technical sessions will be Brian Kernighan.

☞ End of Day2 ☜

I think that there was a short 'Birds of Feather' session which I managed to miss (sorry). I realised that I had been inside for some time and had missed the fine Parisian weather. I started a movement to walk down to the Seine for the conference dinner which was held in a boat called a 'Bateau Mouche' travelling up and down the river. A number of passes were made past the Statue of Liberty, which had shrunk a bit. After a large quantity of alcohol, I found myself walking back to the hotel and managed to get involved in some late night discussions about this and that. And so to bed.

Day 3 — Wednesday, 3rd April

Up betimes and after a leisurely breakfast into a Governing Committee meeting. The first session of the morning had been cancelled because Mike Banahan had to go home for family reasons and Mike O'Dell was forced to cancel at the last moment because of some mess up with air tickets. There was some plan for Mike O'Dell to give his talk over the telephone but I am convinced that this was formulated as an April Fool's joke by a certain big Canadian. I do hope that we will manage to get Mike to come to another conference. Anyway, the first session was cancelled. I think that something to do with UNIX Europe Ltd went on in the Salle Bleue; but as usual I found it difficult to be in more than one place at once. So there is no blow-by-blow commentary of the events.

11.15am Session chair: **Bjorn Eriksen, ENEA DATA Sweden**

The morning started (or continued depending on your point of view) with a few quick 'news from' sessions.

11.17am **Indre By-Terminalen, Kobernaven Universitet**
Keld Simonsen

Keld was supposed to be chairing the cancelled session, so he got his chance to talk about the Danish Group. The Group was started in November 1983 and now has 91 members. The members are the usual mix of vendors, commercial users and academics. They put a small leaflet into most places where UNIX is sold in Denmark; which is a good idea, I think. The group produces two newsletters a year and has also published a UUCP installation guide. The network in Denmark has 12 to 15 mail-only sites and 3 to 5 sites taking news.

The group is naturally interested in international action on character sets and are involved in starting a group discussing 'internationalisation'† of UNIX.

Denmark is the host of the next EUUG conference.

11.23am **The Norwegian Group**
Tor Jörgen Lande

The Norwegian group was started in May 1984. UNIX has not spread very far in Norway. They intend to hold one meeting per year and produce an annual newsletter.

11.24am **News from Sweden**
Bjorn Eriksen, ENEA Data

Bjorn finally got on to say his bit about the Swedish group - EUUG-S. He kindly gave me his overhead projector slide, so at least this bit of the report is accurate.

March 21, 1983: The group was started when Dennis Ritchie visited Sweden. There were about 40 members to start with and there are now 97. There are 69 institutional members and 28 individual members; of these 17 are educational sites and 80 are commercial.

Oct 19, 1983: The first annual meeting, the invited speaker being Teus Hagen.

Sept 26, 1984: Second annual meeting. David Tilbrook couldn't make it so they had Mike O'Dell and Mike Banahan.

March 1985: Presentation of Sun workstations.

May 1985: Meeting cancelled. David Tilbrook was going to explain the 'Tilbrook Philosophy' but couldn't make it.

June 1985: Presentation of ULTRIX.

Sept 1985: Next Annual meeting. David Tilbrook is in Canada so they have Brian Kernighan instead.

Well, after all that, which didn't take long, onto the scheduled business of the morning.

† This was definitely the in-word of the conference.

11.27am **A Contractual Model of Software Development**
Vic Stenning, Imperial Software Technology

Abstract

The technical process of software development and indeed, development of any complex man-made system, can be viewed as repeated application of a single step. Each individual step takes some existing representation (or mathematical model) of the desired system and transforms it into some more 'concrete' representation. This results in a sequence of representations leading from some very abstract model of the desired system to an actual implementation (i.e. a representation that is usable for the real world application).

Any practical approach to system development must address not only the technical issues, but also issues of project and data management. Thus the technical process outlined above must be incorporated into some broader project organisation. One possible approach is to employ a so-called 'contractual' model of development, whereby contracts are let internally within the project for the performance of the individual transformation steps and any other work which needs to be done.

Individual contracts can be fulfilled by the use of appropriate technical development and project management methods. Data management methods must be employed both within individual contracts and at the level of the contract hierarchy for a complete project. An integrated project support environment can support the chosen methods within the framework of the contractual model.

Comment

Vic related a tale from a conference on Systems Design and Implementation where Freedman was discussing the analysis of problems within projects. He said, "Problems which were actually encountered during the design and implementation of systems ... only one problem was common to every system" and then moved onto something else. Of course, everyone was dying to know what the single problem was. At the end of the talk, someone managed to pluck up courage to find out. The answer was "Oh, basically, people did not know what they were trying to do."

12.03pm **Automatic Generation of Make Dependencies**
Kim Walden, SYSLAB & ENEA DATA

Abstract

It is standard practice on UNIX to use the **Make** program to keep a system of interrelated modules up to date. When text files contain include statements referring to additional text files, the task of manually keeping track of all **Make** dependencies implied by such references soon becomes unmanageable. Therefore, the dependencies are often generated automatically from extracted include statements.

However, the problem of producing the correct set of implied dependencies is a bit more complicated than it may appear at first sight, and all automatic methods we have seen in use over the past years have been erroneous. Errors in the generated **Make** dependencies often lead to acceptance of systems with obsolete parts, which may be very difficult to detect, particularly since the number of dependencies rapidly increases to hundreds, or thousands, even for systems of moderate size. Therefore, the problem was analyzed and an algorithm presented in the article *Automatic Generation of Make Dependencies* by Kim Walden, Software Practice and Experience, vol. 14, no. 6, pp. 575-585 (June 1984). The algorithm was demonstrated on a simple example, which may be used for testing dependency generating tools.

Since the time of publication, a number of **Make** dependency generating programs have been released over USENET, but none of them handles the test example correctly. Obviously, there is a need to draw further attention to the problem, and show that if certain fundamental points raised in the article are not observed, dependency generating tools will indeed produce incorrect results in

very common practical situations.

Comment

Basically, don't use **#include** in complex ways if you want **make** to work correctly.

☞ Lunch ☜

2.30pm Session chair: **Helen Gibbons, EUUG**

Helen showed a picture of Owles Hall and talked a little about the operation of the office which she runs. Afterwards, she claimed to have told her one joke. I will not repeat it, in case she wants to use it again.

2.35pm **TEX must eventually replace nroff/troff**
Timothy Murphy, School of Mathematics, Trinity College Dublin

Abstract

TEX must eventually replace **nroff/troff** as the standard UNIX text-formatter. For the output of TEX is an order of magnitude superior to that of **troff**. Thus

(1) TEX reads a whole paragraph before deciding where to break the lines.

(2) In TEX the space between 2 letters depends on both, in troff only on the first;

(3) The spacing around mathematical symbols in TEX depends on the "function" of that symbol: binary operator, relation, left parenthesis, etc;

(4) The size of matching parentheses in a mathematical formula is automatically adjusted in TEX to the height of the expression they enclose.

But if TEX is to be integrated into UNIX it must undergo major surgery.

This paper discusses the advantages of TEX and the problems or properly integrating it into a UNIX environment.

3.11pm **Computer Aids to Rewriting and Copy Editing of English Text**
L. L. Cherry, AT&T Bell Laboratories

Abstract

Writing is generally thought of as a three stage process: planning, producing a first draft, and revising and editing that draft. The UNIX Writer's Workbench† Software is a set of programs to help writers of English with the last stage — revising and editing. It includes programs that analyze the writing style of the text, provide the writer with other views of the text that may identify the parts that need revision, and help in the copy editing process. In this paper I will describe the programs, discuss how they have been used, and how they might be extended to help writers who are writing in English as a second language.

Comment

The Writer's Workbench consists of two main programs. The first **proofr** is a shell script which runs five other programs: **spellwwb**, an augmented **spell** program with the ability to look in a user's private dictionary; **punct** is a punctuation checker; **diction** prints misused and wordy phrases from a dictionary of 450 phrases and in turn runs **suggest** which gives alternatives to the phrases; finally, **gram** is run, this program is a rudimentary grammar checker.

The second part of **wwb** is **prose**, which compares the values from a table of statistics computed from the text with a set of standards and produces a two or three page description of the text

† Writer's Workbench is a trademark of AT&T Bell Laboratories

in terms of a set of stylistic features. This program consists of two other programs: **parts** and **style**.

☞ Coffee ☜

Michel Gien managed to arrange for the technicians in the hall to tape the final session on cassette for me — for which many thanks. This means that you can have most of it in full gory detail. Actually, I think that it reads very well.

Teus Hagen started the session by thanking the technicians in the hall, the interpreters, and the local conference organisers especially Michel Gien and the AFUU. He also thanked Network Events for organising the exhibition.

He introduced David Tilbrook who was to chair the session. David got a very large ovation because he was wearing a suit, tie, and shoes. He was presented with a Swiss Army knife with 124 tools as a memento of his time in Europe and in recognition of the work which he has done for the EUUG. This replaced his own knife with only seven tools. Teus fooled David into using this miniscule piece of hardware to extract the new upgraded model from the enormous box in which the new knife had been packed. Teus failed to read out what all the 124 tools were.

4.19pm **I2U**
Luigi Cerefolini

The Italian UNIX User group has sensibly compressed the two U's and made something which is much easier to say. An extra G at the end is optional.

The group was started in July 1984 and has 49 members. They have had several meetings of the Executive Committee and the result was two meetings in the last months of 1984 including an exhibition. They are preparing a newsletter, number one is in the press. They would like to join EUNET but they have a problem with the national telephone company which does not provide any form of auto-dialling modem.

"We are not discouraged by this and we have two projects to solve the problem. The first is to train people to dial numbers during the night while they are asleep. There are some good results with this but the problem is that people tend to dial the wrong number.

The other project is to have a dedicated network in Italy. We will use completely autonomous lines and are thinking of a new type of cable. The cable will be made of spaghetti. This is much better than other cables like fibre-optics, TV cable or telephone paths because it is a *de-facto* standard. It is supported by industry, the spaghetti industry and is available on world-wide basis, which is very important. Last, but not least, the other important thing is that it is edible."

Luigi reminded us that the next Spring meeting of the EUUG will be in Florence.

David then announced "I have just received a bulletin from the Department of Industry in the United States. All wheat shipments to the Soviet Union are being held back to ensure that there is no technological advantage which the Russians can gain by having spaghetti manufacturing materials."

4.24pm **CUUG & LUUGACAS**
David Tilbrook

David went onto give a couple of 'news from' items, they don't translate well onto the printed page because they relied heavily on the use of overhead projector slides.

The CUUG, the Canadian UNIX User Group was formed instantly at this meeting by David, who elected himself onto the Governing Committee so that we can pay for him to come to EUUG conferences.

The LUUGACAS, which stands for the London UNIX User Group and Curry Appreciation Society, meets every last Thursday in the month (excluding December) in the Lyric Tavern†, Great Windmill Street, London W1. David claimed that this was world's most active UNIX group. The

† Check on the venue with Jim Oldroyd or Mike Banahan of the Instruction Set, they are thinking of moving it.

group was formed about 18 months ago and members from the group figure very heavily in this session because two of the debates were set up there.

4.30pm **The final session - debates**
Various artistes

I think that one of the good conference fixtures which David has instituted is the 'Final session'. He started by explaining the history of these.

"We had panel discussions at the two previous conferences. At Nijmegen, we were blessed with the presence of Larry Crume, Kirk McKusick, Hendrik-Jan Thomassen, Adrian Freed and Mike Banahan. We had Larry Crume representing the System V view; two places away from him was Kirk McKusick, and in the middle was Robert Ragen-Kelly toggling his environment switch. The panel was relatively successful. We also had Andrew Hume and Eric Allman in the audience, which meant that the audience drowned out the panel.

We tried this again in Cambridge and it wasn't quite so successful. This can best be explained by the fact that there was a question from the audience asking: 'which would the panel choose — 4.2 or System V?' Five people on the panel said 4.2. Tom Killian said V8, at which point, four people changed their minds and said V8 — if we can get it. But it did show that there wasn't enough difference. Mike O'Dell was in the audience, he participated at lot but it didn't get the spirit which we wanted. There was no blood letting. So, we set up debates for this conference. The debates, I hope, will be amusing and informative. When we suggested this, David Turner said that the debates would be likely to shed more heat than light. I hope so."

There was a slightly formal format where the speaker for the motion was allowed three minutes to state their case; the speaker against the motion spoke for 4 minutes; and this was followed by a one minute rebuttal by the first speaker. Five minutes of discussion from the floor was then allowed. Jim McKie acted as the 'Sergeant at Arms', reading out the title of each debate and biographies of the speakers who had not been introduced before. As I did not manage to get hold of the biographical notes for the rest of the speakers at the conference, I will minimise space by omitting them here also (this document is too long already).

1. **Declarative languages must eventually replace imperative languages**
For: David Turner, UKC
Against: Tom Duff, AT&T Bell Laboratories

For: David Turner
"Mr Chairman, Ladies and Gentlemen — and C hackers. David Tilbrook told me to cut the technical arguments and go straight for personal abuse. I want to move the motion that declarative languages must eventually replace imperative languages, and preferably sooner rather than later. Existing programming languages, the ones in common use anyway, are based on the same basic model of computation. They have the assignment statement as their basic step, they are sequential and are based on side effects and so on. This is a model of computation that emerged in the late 50's. Fortran is really the paradigm, or perhaps Fortran and Cobol, and all modern languages can be considered dialects of Fortran.

One has honest dialects of Fortran such as C; and dishonest dialects of Fortran such as Pascal or Ada — which are pretending to be something else. I submit that there is something fundamentally wrong with this whole model of computation, and something fundamentally wrong in the way that we produce software at the moment.

There are two main symptoms of this. First of all, software is too expensive to produce and secondly, once produced, it is usually wrong. It usually has errors, or bugs, as we call them.

Let me take the correctness problem first. Most non-trivial programs have errors, that's a situation which we have gotten used to. Many large programs upon which we depend from day to day are riddled with errors; this is really a terrible situation that we ought not to be willing to put up with. We try to eliminate errors after the event by a process called: 'debugging'. Now, I want you to compare that with the way which other engineering disciplines proceed. It is not the case that they design aeroplanes by debugging them. They don't put an aeroplane up in the sky, watch it crash in a nose-dive, and say 'Oh dear, I think that the wings must have been the wrong shape, let's try again'. They have mathematical models of what makes aeroplanes fly and they do a lot of calculations first. They know to within a very fine degree of accuracy how that aeroplane will behave before they build it. We need to do the same with software. We need to have proper mathematical basis for programming.

I submit that no language containing an assignment command can have reasonable inference properties. Declarative languages do have reasonable inference properties. They are also much more compact. So, if we switched to declarative languages, not only would we have a proper formal basis to make possible formal development of programs and formal verification of programs; but also programs would be much shorter and much quicker to produce. Finally, because they are not sequential and not based on side effects they have the capacity to take advantage of new kinds of very highly parallel hardware that VLSI makes possible. I urge you to support the motion."

Against: Tom Duff
"Arthur C. Clarke once said that if a scientist tells you that something is true then he's probably right, if he tells you something is impossible he is very likely to be wrong. I feel like that I am in a bad position on that ground. But, there are a number of things to say.

Niklaus Wirth once said (or Nicholas Worth as we call him in America) if housebuilders built houses like computer programmers build programs then the first woodpecker that had come along would have destroyed civilisation. Which is roughly a paraphrase of one of David's points. Wirth obviously never saw a carpenter build a house.

Secondly, I have a friend who worked at Boeing for many years and he said that the average Boeing 707 has something like three tons of shims in it. So, the argument from other engineering methods doesn't hold up at all.

As for formal verification, it is fine for mathematicians and fine for people who are working on *very* small problems. There exist no significant systems that have ever been verified, done in any sort of language, let alone the sort of nonsense that my worthy opponent advocates. It requires Ph.D mathematicians to understand this sort of stuff, I got out of mathematics at an early age because it's just too tough for my feeble brain. The general class of people that you have writing computer programs are not up to it. I include myself; but, of course, not the audience.

The fact that formal verification methods cannot handle languages with assignment is not the fault of the languages, it's the fault of the methods.

Languages without assignment don't match current hardware very well at all and if you think that you are going to change that situation then I think that you're dreaming. We have had Von Neumann machines, machines with big memories and little CPU's since before I was born and I think that the chance of that situation changing at all is negligible at best.

There are plenty of problems which the assignment statement is precisely the right model to think about the problems. For example, a case from interactive computer graphics: take a frame buffer which is just a large memory and you want to draw pictures in it using a mouse or a digitising tablet. The model you have there is taking something that's shaped like a paintbrush and plopping it down in the frame buffer. It's precisely an assignment model which describes that sort of problem. There doesn't seem to be a good way of describing it otherwise.

The applicative models don't describe important features of a lot of algorithms. An example which came up the other day was quicksort where there are two important properties to the algorithm. One is the notion of recursive sub-division which the applicative model handles just fine; the second is the notion that you can do the whole thing in place and there is no place for that sort of idea in applicative languages.

A text editor is something that you are going to have a lot of trouble describing if you have no notion of a variable. Where are you going to write the file out? The idea is that you read in the file, you make some changes and you *put it back where it came from.* The idea just doesn't exist in the applicative model.

Portability is another issue."

Tom was cut off by time but managed to slip in, "that's about all I have to say. And besides, Turner is a foreigner and can't possibly know what he is talking about."

Rebuttal: David Turner
"I can't possibly pick up all of the points. But first, let me say, that of course we don't know how to program everything in the functional style. As it happens, we do have a working interactive text editor, there's obviously no time here to tell you how it works, but you can solve that problem. There are other problems that we don't yet know how to solve but that of course is why functional programming is fun. It's a frontier, there are still new things to find out.†

Formal verification doesn't work, we are told, because it only works on toy examples. Two points about that: of course it doesn't work at the moment because we are using languages with the wrong fundamental properties. If you switch to languages with the right properties, like functional languages, formal verification becomes very much easier. Of course, like everything else, formal verification hugely benefits from the intervention of computers. I don't advocate that people do fully formal verifications by hand, that's crazy. I advocate that we build computer systems to do verifications for us, computer systems steered by a human being. I believe that is possible on the basis of functional languages.

If it's the case that the hardware we have now doesn't match functional languages, let's build new hardware."

Comments from the floor:

Mark (Seiden?)
"I'd like both speakers to comment on the ease of debugging large programs written using their respective styles."

Tom Duff
"There are no programs written in Turner's style. He cannot debug them."

David Turner
"Not true, not true. There are no large programs, that's because programs become brilliantly small when written in my style."

Tom Duff
"They'll get small in APL too, it's got nothing to do with functional style, it's got to do with big operators."

David Tilbrook
"Fault, it's fifteen love."

Eric Allman
"I can't help but notice that in your functional language, you've hidden that fact that you actually do have state by putting things into huge vectors. You then run through them and essentially pretend that what you really have is all the values of the variable stuffed into one, in a convenient little way. It seems to me that there still is state in your language."

David Turner
"Yes, I think that's a legitimate comment. What happens is that instead of having a sequence of states in time; we represent it as a sequence of values in a data structure; so it looks like space instead of time. That is exactly the trick which the physicists use to describe the world. That's exactly what Newton did when he invented the equations that we use to describe dynamics, you treat time as a dimension of space. That's really what functional programming is advocating. That you handle state change by having a data structure that represents the successive states and the whole thing can be conceived of statically instead of dynamically and that makes it more amenable to reasoning. But your comment is legitimate."

† Tom Duff: "that wasn't the argument."

David Tilbrook did say at this point that debaters do not necessarily hold the views that they are stating. Your roving investigative journalist can now reveal that Tom and David did have some difficulty in finding something to disagree about.

Ah well, onto the next debate.

2. Version 6 was the last real UNIX
For: Dave Lukes, The Instruction Set
Against: Mark Seiden, Lucasfilm & IBM

Dave was dragooned into standing in Mike Banahan's socks when Mike was unexpectedly called back to London.

For: Dave Lukes "What's good about V6? Those of you who remember V6 will remember an amazing thing which is the shell. The shell did not have nice structured programming constructs; it had **goto**. The good thing about **goto** was that it was a separate program. People are talking about unbundling UNIX nowadays. V6 was great, if you didn't like having a high-powered shell, you removed the **goto** program.

People argue about what are the bad points with the things which came after V6. Here's my favourite bad point. The program line

```
wc -b9600
```

What's this number 9600? It's something to do with baud rates. That's the kind of thing that happens when you start extending a nice clean simple system. That was 4.1BSD, there are worse things that have happened since. I won't say that 4.1BSD is the worst offender.

V6 **pr**: look at the **pr** program and count the number of flags. Most people give more flags than filenames to **pr**, all the time. There's something wrong there. It's got worse, **pr** has had more flags with every single release.

The shell, remember that? A nice simple little thing. Your give it a file maybe and some arguments. Now, the V7 shell has the **-ceiknrstuvx** flags. What do these arguments do? Well, they're all really interesting you know. Like one of them does things like, I think **v** prints out its input. Ever heard of **tee**? There's a lot of useless flags in the world. I don't know what the rest of them do, but I am sure that they're quite interesting as well.

V6? It's simple, it's cheap, it's extremely nasty but it works. The shell is 20 pages of C in Version 6. Beat that anybody else. What else can you say?

That's V6, the greatest UNIX that ever lived, simply because you could carry it around in your briefcase."

Against: Mark Seiden
Mark's talk was punctuated by pictures from the history of transport. He started with early stationary engines, passing through vintage cars, steam trains and ending with British Rail's Advance Passenger Train. Naturally, the pictures cannot translate to this document so a certain amount of imagination is required, but we sat and laughed a lot at the pictures.

"The proposition is 'Was Version 6 the last real UNIX'. I think that if you look at it carefully, you'll realise that it wasn't a very real UNIX to begin with and even if it was real, other later systems are much more real. But, first some history to help put us in focus.

In 1969, Ken Thompson first wrote the game of Space Travel for a little used PDP-7 at Bell Labs in Murray Hill. Within the year, due to Ken and Dennis Ritchie, the first UNIX system was written in PDP-7 assembler. Even by then it was an obsolete machine. The first version for PDP-11 was in assembler as well.

Some four years later, the system had evolved considerably to Version 6. It was now capable of real work, although it did have many limitations. Sizable pieces of code were still in assembler and the code still ran only on PDP-11's. It couldn't support heavy peripherals, mass storage devices, because of 16-bit block addresses. The maximum size of a file system was 32 mega-bytes. It made no claim to satisfy real time requirements or single user requirements. It certainly didn't satisfy the needs of the European UNIX community, so people started hacking it in a heavy way.

So, let me ask you. Could you do now without the Bourne shell, the back-quote operator, **pcc**, **lint**, **make**? They weren't in Version 6. So, Version 7 was the first real UNIX. It had those things and more importantly, it was portable. What does portability mean? It means independence from a single manufacturer and can be translated directly into personal freedom for UNIX programmers. It means you can work almost anywhere.

One problem with Version 7 was the file system. A power failure, or a test kernel scribbling all over the system buffers could cause great unhappiness. If you remember **icheck**, **dcheck** and **ncheck**; and trying to apply them in the right order at two in the morning, you'll know what Version 7 was like. **Fsck** came along just after Version 7.

The system still didn't support virtual memory and you needed a workhorse like Berkeley UNIX to do that. Berkeley UNIX versions added support for virtual memory, a full screen editor **vi**, auto-configuration and then later, flexible usable networking. A faster file system on 4.2. Can you do without **fsck**, history, job control, symbolic links, Ethernet? Perhaps.

Perhaps we should be asking: is UNIX real, even today. It still needs **fsck** every time you boot, you still have to get out of the mail reading program when new mail has arrived, that is ridiculous. The world has changed in 10 years and reality has to reflect those changes. Now, ordinary people use computers every day, we have semi-conductor memories, 5 (and a quarter) inch discs with 500 megabytes on them, optical discs soon to come and wide choice of networking technologies. There may not be a real UNIX even today, I think. Perhaps in fact, the last real UNIX might be **Multics**."

Rebuttal: Dave Lukes
"Well, I'll knock these things down one by one.

First, he says that V6 is not portable. I used to carry it around everywhere, I had a listing in my briefcase, and what's more I could get a sandwich and an apple in there with it — and the manuals. I admit that I didn't eat much lunch, Tilbrook had a bigger briefcase.

Next assertion, everything is too small the guy says. 2^{24} bytes? When was the last time you made a 2^{24} byte file? Next, 2^{24}. Is that big enough? No way, discs are bigger than that now. Also, if you have a 2^{24} byte file, there is no standard UNIX utility which can manipulate that object. You can buy in data-bases which can do it but there is no standard UNIX utility which can do anything with a file that big. So, what do you need it for?

They say it doesn't have **awk**. Well, people don't really miss **awk** because all they ever get out of it is p&P6. That's what lives at location zero on the PDP-11, it's the bit of code that you get from printing out a null pointer.

Comments from the floor and other diversions:

Mark Seiden
"Dave you are wrong, there is a utility which will operate on a file that large and you mentioned it at the beginning of your talk — **wc** — at least on a VAX."

Eric Allman
"**rm**?"

David Tilbrook
"Eric Allman, the master of small programs..."

Eric Allman
"I'll get you for that.

Now, you see, you're wrong. The problem with Version 7 is not the code itself but the documentation. With 6250 tape drives you could still put the system into your briefcase. The problem is that they documented everything in Version 7 which they hadn't in Version 6. If you wanted a different option out of the program, then you went into the program and changed it, if the option that you wanted wasn't already there."

Unknown speaker
"The biggest mistake with the later UNIX versions which was not mentioned was that it opened up room for shitty PC-DOS. Right now, there is no UNIX system around which is really small enough and uncomplicated enough that it can stand up in the very low end of the personal computer market."

Mark Seiden
"But the reason for that, it seems to me, is a marketing one. That is the enormous market penetration of IBM. Application developers choose to write their application to run in the PC-DOS environment. But, to many of their credit, they write them in C so they will also run in the UNIX environment if they see a business plan for it, for example, **multi-plan**. I don't think that there is a technical argument."

Radek Linhart
"I have heard from somebody that there is a UNIX which fits into ROM's and you can carry it in your briefcase."

Mark Seiden
"I have here an optical disc on which is 4.2, all its documentation and all historical versions of UNIX before that. It fits very nicely into my briefcase."

3. **Vi is a piece of wombat do**
For: Nigel Martin, The Instruction Set
Against: Tim Snape, Chameleon Software

I'd like to insert a little from Nigel's biography because it contained a great line: — 'he was promoted from a lecturer in Computer systems at University College, London to a junior operator as a result of using UNIX.'

For: Nigel Martin
"Let us begin at the beginning of time when we all got used to using **ed**, we all enjoyed **ed** and it had a number of facilities. One of the biggest downfalls of **ed** was that you had a lot of confusion associated with using it and you wanted to be able to view larger amounts of your file. Therefore, along came an editor called **vi**. Its name: 'visual'. Everyone was extremely pleased about the arrival of **vi**. Hopefully, as the name implied, 'visual' would allow you to see what was going on with your edits and allow you to see a substantial portion of your file. Unfortunately, this appears not to be the case. **Vi** shows you a picture and allows you to see a number of characters. Regrettably though, there is an extensive amount of user confusion still present.

Invariably, you will sit at your keyboard and you will type a number of characters. Those characters will sometimes, if you are very fortunate, have an effect. They may, if you're even luckier, be inserted into the file at exactly the right place which you wanted. This experience is extremely rare.

It is far more likely that when typing in your characters, you will type a number. For instance, '1'. Nothing happens, you look distressed. Is it because the machine is slow? Is it because **vi** is slow? No. It's because you have just typed a '1' at the wrong moment. So, you type a '2'. What happens? Nothing. You are equally distressed, you look around in absolute bewilderment. So you type another number: '3'. Still nothing happens. You realise that **vi** clearly does not like numbers, so you try and go for some letters. You hit a 'd'. What happens? Nothing. You then hit a second 'd' and hey presto! The picture changes — 123 lines disappear from your file!

It is argued that the facilities of **vi** allowing you to prefix a command with a number are extremely important. It is unfortunate though that this is not extended to every command; consider, for example, the tilde command.

One of the rules of UNIX is that UNIX is built out of a number of tools. Therefore, we would have hoped that an important and useful UNIX tool was the screen editor. We all know that one of the most fundamental rules of a tool is that: silence is golden. If you walk into a room of people using **vi**, you will realise that **vi** clearly is not a tool. There is an extensive amount of sound. The sound comes from people screaming, from **vi** ringing the bell and from many, many, other sources.

It is evident therefore that **vi** cannot possibly be considered to be a tool.

Finally, in order to make a very very trivial edit to your file, perhaps as a result of a small typing mistake, you will have to type numerous keys to shift you in and out of command and edit mode. This is most unfortunate. Clearly, if you want to move the cursor up, the same key should allow you to move the cursor up at all times. You should not have to go through the process of negotiating your way round this supposedly useable tool. I therefore urge you to consider that **vi** is not only unsuitable, that it certainly is not a tool, and it certainly is not a screen editor."

Against: Tim Snape
"I don't know if you know this, but in fact, 'Wombat do' is very high in nitrates. Certain small South American countries have based their economies on such products, so IBM take note.

I would like to say that I use **vi** and I like it. I will itemise the points which I find attractive.

The most important benefit of **vi** is that it is standard. You can go from one UNIX machine to another and you will still have the same screen editor on it. You will be able to edit your files using a screen editor.

Another benefit is that it uses **termcap**. This means that you can customise **vi** to work on even more bizarre and weird terminals as they become available.

The heritage of **vi**: Nigel briefly mentioned that **ed**, well, in fact, **vi**, is based to a small or lesser extent on **ed**, **sed** and **ex**. It's a family of text editors. While I was researching this talk I tried to discover the parentage of **vi**, it was rather difficult.

What it means, is that once you have trained your first-year undergraduates to use **ed**, something which can be done fairly rapidly. They can move up and use **vi** in a fairly easy way. (Fingers crossed).

The lack of *wysiwyg*ness, the problems that Nigel has been having with **vi** are in fact not bugs at all; but features. What **vi** is attempting to do is to minimise the amount of character I/O which is taking place over your terminal line. So, you can use **vi** over PAD's and 1200 baud lines. You cannot do this with some of the other visual editors.

Vi has a lot of functionality, a lot of power. It's been said that real programmers are not really worried about the cosmetics, they are more concerned with power. Multi-modal editors do offer a lot of power. Of the things which **vi** has, which I like, is the syntax directed extensions; the ability to tab around functions and the ability to tab round curly brackets. I find that very useful. Also, the undo facility and the ability to pass text through UNIX filters are both very nice features.

That's not the point, the real point that we are discussing today is: should we be using single mode editors or should we be using multi-mode editors? Single mode editors are attractive to secretaries, typists, English students, and people like that. Multi-mode editors are very attractive to people who like to have a powerful command structure. It has been said that multi-mode editors increase the thinking time for users. I think that this is probably a good thing for some C programmers.

Vi, like UNIX, like C, has been criticised for its lack of user friendliness, its been criticised for its complexity, but these aren't real arguments. Like C and UNIX, **vi** is a tool which grows on you, the more you see other people using it, the more you like it and the more you learn.

Finally, Nigel, if you do have problems using **vi**, perhaps you would like me to run a course for you."

Rebuttal: Nigel Martin
"In the introduction to Mr. Snape here, my learnèd collegue on the other side, he informed us that he was interested in producing tools and utilities to aid programmer productivity. It is unfortunate that the very tool or thing he describes does not quite come up to this.

Vi is standard. Yes, unfortunately it is. We know there are many things wrong with many standards, how many people here are happy with any standards they know of?

With regard to the use of termcap, perhaps, Tim, you would like me to supply you with a free copy of a termcap driven **ed**.

As regards the family information, it is most amusing that you are unable to trace the parentage of **vi**, perhaps that is something to do with its quality.

With respect to character I/O, I would suggest to you, ladies and gentlemen, that **vi**'s primary objective in life is to try to melt any form of communications facility. Every character you type results in you having to use the undo command to put back all the mistakes you have just made.

Finally, I am sure that many programmers invariably think that they are undergoing a secretarial task. Therefore they would like to relax, concentrate on their program and not have to pay too much attention to trying to negotiate their way round these things which are obstructing us in our way. The one thing that UNIX was trying to do when it was originally designed in '69, was to allow people to get on with their job, that of programming, rather than having to fight with the system. So, I would urge you that **vi** is not an appropriate tool to do this."

Comments from the floor

Erik Fair
"Mr. Martin, I am curious, since you are a detractor of **vi**; which editor do you prefer to use?"

Nigel Martin
"It is most unfortunate, **that** is not the subject of this debate."

Eric Fair
"Oh dear, that doesn't help at all. How can I detract from your personal heritage if I don't know what editor you use?"

Nigel Martin
"I am sure, Sir, that this is of no importance to the discussion in hand."

John Haxby
"I'd quite like to know why **vi** was written in the first place. We had a perfectly good multi-mode editors with none of the features of **vi** long before **vi** even appeared in the UK."

Nigel Martin
"Mr. Chairman, I object to the word 'feature' being used in the same sentence as **vi** without a *not*."

Dave Rosenthal
"It's a while since I used **vi** but I've just heard a load of people saying things about the undo command. As somebody who uses an editor with a real undo command, like **Emacs**. There isn't an undo command in **vi** because if you undo it, it undoes itself."

There were some more comments in this vein, nobody was really brave enough to stand up and say anything much in favour. Eric Allman did make a slight stand, but I don't think that he scored many points! I guess, **vi** lost here.

Well, there were only three debates in the official order of events and the next one was a surprise to us all. Jim McKie (wearing what might be described as 'hot pants' — or am I showing my age) read out the resolution.

4. **Jim McKie has nice legs**
For: Jean Wood, DEC
Against: Louise Sommers, IST

Both speakers in this debate used some wonderful slides to complement their talks. The slides showed giraffes, rhinos, hippos, chimpanzees and other fauna. Oh, the first slide was a picture of Jim showing the main topic for the debate — his legs — 'where did you get that?' asked Jim.

For: Jean Wood
"We have to consider this from a serious point of view. Both functionally and aesthetically, those are great legs. Of course, their main function is to connect the top half of the body with the ground, and you can see they do that exceptionally well. They're straight. They're strong. They have knees in the right places. I guess in the language of computerese we could call those 'very robust legs'. On the other hand, they're also very user friendly. They have the correct quantity and quality of hair. The knees have kneecap worthiness.

It's very difficult for me to look at this and speak unemotionally. To put things into perspective, I thought that we could look at some similarities from nature and compare them with Jim's legs. *Here were the various beasts.*

To be honest, I have had very little opportunity to look at Jim's legs, except at conferences, like the rest of you. I talked with an expert, his wife, she's a nurse and she spends lots of time doing things like bed-bathing men and she has seen lots of legs. Her comments were: 'Jim's legs are nice and straight. With a nice pair of legs, the less attractive appendages, like chins, don't matter'."

Against: Louise Sommers
"I am speaking against the motion. My honourable opponent has described Jim's legs in glowing terms. But I feel that she has picked the wrong comparisons. *More pictures of fauna.*

My opponent also quoted Gilly McKie in saying that 'Jim's legs were nice and straight', but she didn't give you the whole quote. When do they ever? What Gilly actually said, and I wrote it down at the time, was that 'Jim has nice straight legs' (she said this in the most beautiful Scottish accent, which I can't do) 'but they're not very muscley, you know. If he's going to wear a kilt, he'd have to wear very thick socks as well.'

Being a expert on men's legs (and bottoms I might add), I did a little survey while I was here at the conference and I looked at all the men's legs. I want to tell you what nice legs really are, and they are David Tilbrook's. Would you please show your legs, David."

At which point, Mr. Universe stood up, ripped off his very smart trousers and demonstrated (or perhaps one should say 'flaunted') his legs. Of course, Louise is somewhat biased since the large Canadian and Louise are married. She went onto say "Now, these are nice legs, well shaped, firm, strong and believe me when I say *sexy*."

There was one further comment from the floor, an unknown speaker.
"I've heard that Jim's legs are bug-ridden, is that true?"

Jim McKie
"No, my legs are written in a functional programming language."

☞ End of day 3 ☜

The evening event was a concert of electronic music at IRCAM. I must confess to have decided to miss this event, largely on the grounds that I **KNOW** that I don't like that sort of stuff. I asked Jean Wood to write me a short piece on the concert and hopefully she will.

The exhibition was open the next day and I spent most of the time there, apart from about three hours sitting outside a café in the sun, isn't Paris wonderful? Anyway, I digress. I suppose that the most noticeable thing at the exhibition was the emergence of the colour display, there seemed to be a colour display on almost every stand.

The other thing which caught my eye was really nothing to do with UNIX, it was the French PTT terminal which they are putting into every home and office in certain parts of Paris. This gives access to an electronic phone directory, electronic mail and telex facilities. My brother has one of these in his office and was enthusing about what he could do on it.

I spent a lot of time playing with a colour Sun workstation on the Gould stand, this is fun, but of course pricy.

Endpiece

Well, another conference gone. We will be learning some lessons from this one, just like we have learned from previous conferences. I have always thought that every conference is an experiment in doing conferences. I traditionally give private awards in these reports. We didn't have a silly competition at this conference, there was no publically available blackboard to write things on and we couldn't really come up with something silly enough. So, the private awards.

The 'best freebee from an exhibition stand' award goes to the Instruction Set for their wine labelled 'Chateau NUXI' which is the name of their UNIX system (not the Chateau, the NUXI). I took the bottle back to my brother's flat with the thought that he could always use it for cooking if it was vile — but it wasn't, we drank it. I suppose that you could say that I have no wine discrimination facility.

The 'bore of the conference' award must go to the French customs official who confiscated all the EUUG T-shirts (which had not already been given out) because they were made in Turkey. We could get no convincing explanation of why Turkey's T-shirts should cause such offence.

The 'event of the conference award' goes to a deserving failure. At attempt was made on the world record for mismatching shoes. The record, previously held by the British Army in the Crimea (who took delivery of several hundred left boots) has stood for more than a hundred years and involves getting as many people as possible in one place wearing mis-matched footwear. It's unfortunate that there were just not enough people involved in this event.

The 'coming out of the closet award' goes to Nigel Martin, who managed with great difficulty to disclose in a suitable small circle of close friends that 'he is a **vi** user'. I always thought that Nigel was a bit that way inclined, perhaps it is the way he walks.

The (more serious) award for the fixer for the conference should go to Helen Gibbons. She dealt with a great many problems in an uncomplaining way. Not many people were aware of the problems at the time, because by the time conference attendees had got there, the problems had simply gone away.

Oh well, I must stop. This document breaks size records. Thanks to all who made the conference work.

The Influence of the UNIX [1] Operating System on the Development of Two Video Games.

Peter S. Langston

Bell Communications Research[2]

ABSTRACT

The Lucasfilm Games Group was established to explore ways of applying the technological and methodological expertise developed at Lucasfilm (principally for film production) in a new entertainment medium video games. One of the major tools on which that expertise is based is the UNIX operating system. It is an interesting coincidence that this operating system's early development was heavily influenced by games and the interests of game designers.[3]

This paper describes the development of the two video games "*BALLBLAZER*"™ and "Rescue on Fractalus!"™ and the ways in which UNIX software aided and influenced their design.

HISTORY

"Months ago in a deserted warehouse in Marin County, George Lucas met with a lone programmer..." so begins one of the many descriptions of the formation of the Lucasfilm Games Group. Needless to say, some journalistic license is evident in this and other accounts of this group's formation but when the purple prose is stripped away what remains is the important part of the group's charter to look at the booming industry of video games and see in what ways the "magic" of Lucasfilm could be brought to bear. That "magic" consisted of two parts: an uncompromising attention to detail (leading to a sense of involvement and realism even in the most unrealistic of situations) and the use of high-tech tools to make possible the creation of formerly impossible sequences and images. An important part of those tools was the UNIX operating system introduced within Lucasfilm by the Lucasfilm Computer Division, of which the Games Group was a part.

The first project we (the games group) undertook was a survey of the games industry to learn how games development was being carried out. We were amazed. Home video games were being produced in one of two ways. Either a single programmer would spend a year or two of his spare time creating the game concept, graphics, sound, play mechanics, and writing the game program in assembler (typically in his basement on a system with too little memory, too few floppy disks, no reasonable way to make a backup, and few if any debugging tools). Or a team consisting of a programmer and one or two helpers would be allotted 2 to 3 months to carry a game from concept to debugged implementation. Programmers in either case often had no prior experience in computer programming and were plagued by flakey equipment and a lack of many aids that we considered basic: high-level languages, support tools, libraries of software, input/output redirection, hierarchical file systems, large storage devices, etc.

When we looked at arcade ("coin-op") video games the situation was different, but no better. The typical development cycle appeared to be: 1) come up with a game concept and get approval

[1] See the last page for a list of trademarks.

[2] The work described here was done at Lucasfilm Ltd., San Rafael, Calif.

[3] c.f. "The influence of games on the development of the UNIX operating system" (unpublished).

for it 2) design the hardware 3) build the hardware 4) test the hardware 5) write the software 6) try the game out 7) decide whether to continue 8) debug and polish. Although steps 3, 4, and 5 could be overlapped it still took 6 to 18 months to get to step 6. At that point it was too late to make major changes and dropping the project would entail a big loss. As a result many uninteresting coin-op games were produced. We learned that the reason general purpose hardware was not used was that special purpose hardware is less expensive to manufacture than general purpose equipment and they wanted to sell 10,000 to 100,000 of these games. Thus a manufacturing savings of $1,000 on each unit could mean a saving of $10,000,000 to $100,000,000 overall. This combination of optimism and greed apparently blinded the industry to any thoughts of rational planning.

Media interest in Lucasfilm has always been high and so it was not long before we found ourselves being quoted in the press, pompously telling the games industry How It Should Be Done. In the midst of all this media attention our group realized that since none of us had ever taken a video game through all the steps of production we might be setting ourselves up for some embarrassing surprises. "**Lucasfilm Computer Scientists Eat Words**" and "**Ivory Tower Talk is Cheap**" screamed the banner headlines in our nightmares.

To avoid such possibilities we decided to design and implement one or two "throw-away" games as a combination of rite-of-passage and reality check. These games would also serve as a way of identifying the areas that would profit most from the creation of some software tools. It's a credit both to the members of the Games Group and to the ideas that we championed that the two "throw-away" games produced have gotten such enthusiastic reviews[4] - ranging from "George Lucas does it again!" to "setting a new standard in the industry".

Of course the games never did get "thrown-away" and may even appear in stores in the near future.

THE HARDWARE

The Atari 800 computer contains a 6502 microprocessor chip, 32K to 64K of RAM, and several special purpose chips designed by Atari. The "ANTIC" chip handles DMA, some interrupts, vertical and horizontal scrolling, and the vertical line counter. The "GTIA" chip handles the graphics generation, graphics objects, collision detection, and miscellaneous switches and triggers. The "POKEY" chip handles the keyboard, the serial communications port, timers, hardware random number generation, reading potentiometers, and the generation of sound. The "PIA" chip handles the joystick controllers and interrupt lines.

An Atari 800 computer, complete with keyboard, power supply, and cabling to use a conventional television set as the screen, costs about $100 in discount stores today and an Atari 5200 Super Game System (which lacks a keyboard and comes with joysticks but is otherwise nearly identical) costs about $70. The 6502 microprocessor chip used in the Atari machines (also used in the Apple II and other home computers) costs about $1.50 (retail) in single unit quantities. All in all, the Atari 800 and the 5200 are *small* computers. We found that writing programs for a small computer often requires using tools that are very large. Such a discovery rules out using the Atari machines to run the tools; however a hybrid that allows use of a large machine for editing and assembly and a slave Atari machine for testing is an ideal arrangement.

THE TOOLS

Several pieces of software were created by our group specifically for the game development projects. Among these were:

1. a cross assembler

[4] OMNI magazine named one of the games in its "Top Ten Video Games of the Year" (tied for second) even though the game has never been released.

2. a library of macros for the cross assembler
3. a pair of programs that download an Atari 800 home computer over a serial line from a DEC VAX 11/750
4. a similar pair of programs that download an Atari 800 floppy disk over a serial line
5. a simulation of flight dynamics and graphics on an Evans & Sutherland Picture System I
6. a program that turns an Atari 800 into a drum rhythm machine
7. a music scoring system using *pic* and *troff*
8. a storyboard editor that runs on a Sun Workstation
9. a 6502 disassembler

It was not surprising to us (although it was to people in the industry) that the majority of our software was written in high-level languages. The cross assembler and its macro library were written in Lisp. The Atari end of the download programs and the drum machine program were written in our Lisp-like cross assembler. The rest of the tool software was written in C.

Lisp was chosen for the cross assembler because it only took two weeks to implement a cross assembler that allowed both assembler macro definitions and inclusion of arbitrary Lisp expressions in the assembly task. This allowed us to extend or reconfigure the assembler as we discovered unforeseen needs and deficiencies.

The macro library had two important effects. First, it allowed us to write in a pseudo-structured assembler that included "*if then else*", "*for*", "*while*", and other familiar constructs. Second, it acted as a first step toward a subroutine library of commonly used routines. We found that many of the routines that we included in the initial set never were used at all, while others that we added as we went along were used frequently.

The download programs made it easy to maintain the source on larger machines running UNIX where we could assemble the code and debug syntactic problems using many different tools before sending the executables to the small Atari machines for debugging. Unfortunately debugging on the Atari was not always convenient so we designed a dual-ported memory card that would allow us to debug the running Atari program using software running on the larger machines. We discarded the possibility of simulating the Atari on the larger machines because there are an incredible number of important undocumented eccentricities in the Atari hardware.

The simulation on the E & S Picture System allowed us to start fine tuning the flight dynamics and make some general decisions about ways to achieve the graphic results we wanted while the Atari graphics rendering code was still being designed.

The drum machine program provided a workbench on which we could experiment with rhythmic patterns to be used in the games. We were able to include the features that make commercial microprocessor-based rhythm units so popular and add new features to suit our particular composition styles. Further, the program had the advantage that we were hearing the exact sounds that we would be able to include in the games.

Although a music scoring system using *pic* is unavoidably unwieldy it still allowed use of familiar text editors, the Sun Workstation graphics screen, and the Imagen printer for manipulation and distribution of musical ideas and final compositions.

A computerized storyboard editor seemed like a particularly good idea because the pixel sizes and graphics modes of the Atari graphics system are quite eccentric. Unfortunately the Atari graphics are so eccentric that even the Sun Workstation's fairly high resolution screen was unable to capture all the "nuances". A further problem was the need to display the colors that appear on the Atari screen; because of artifacting effects[5] this would have been difficult to simulate even if we had had color graphics on the Sun Workstations. As a result we found the existing, primitive graphics editors available for the Atari more useful than the more sophisticated but inappropriate editor we had written for the Sun Workstation.

[5] The term "artifacting" refers to the interaction of colors in adjacent pixels on the screen.

In our survey of game software we found one or two programs with unusually sophisticated graphics and arranged to speak with their authors. We were disappointed to discover that few game programmers are willing to discuss the technical details of their work or share insights they may have gained.[6] With a little work we were able to write a disassembler that made the difficult task of deciphering a 6502 executable feasible.

Our intention had been to learn the steps necessary to produce a video game. As such we did not want to try any radical, new approaches; instead we wanted to develop a list of ways to improve on the state-of-the-art. On the other hand, some aspects of the "state-of-the-art" were just too primitive for us to bear. As a result we compromised by making only those improvements that were quick to implement and required minimal long-range planning. The need for an improved macro assembler was obvious and it was relatively quick to implement whereas the dual-ported memory cartridge turned out to be a larger project than we had expected and its construction was postponed.

UNIX AIDS

The ways in which the UNIX system aided the games design process can be grouped into two broad categories, software available on UNIX systems, and capabilities of the UNIX operating system itself. Available software included program languages, editors, communications programs, and other miscellaneous tools. The capabilities of the system included a hierarchical file system that made file sharing convenient, input/output redirection, and a general "permissiveness" that kept the system from getting in our way when we needed to do something strange.

Languages

Having languages like C, awk, the shell, and Portable Standard Lisp (PSL) readily available allowed us to choose a language to fit the requirements of any particular task. Some programs had to execute a specific, well-defined task as quickly as possible and were written in C (e.g. the download programs). Other programs were more experimental and needed flexibility in the way they manipulated symbolic entities and were written in Lisp (e.g. the cross assembler). Still other programs had to be written quickly and would only be used once or twice; these were written as shell command files or as awk scripts. Had we been using the typical games industry systems we would have written all these programs in assembler (or perhaps Fortran if we were lucky) and we would still be debugging some of them today.

Editors

Although every class of Computer Science graduates seems to produce another editor-to-end-all-editors, none of these editors seems to have made it into the video games industry. We made much use of *emacs*, *vi*, and *ed*, again fitting the choice of editor to the requirements of the task at hand with few, if any, religious debates over the virtues of modelessness or the sins of meta-cokebottle-shift.

Communications

Our group put in many 22 hour days, sometimes spending 16 hours at work and then going home to say "remember me?" to the family and then log in for another few hours of work. After a few months of this it became hard to predict when any particular person would be in the office or even awake. The UNIX mail system allowed us to keep in touch with each other and share ideas and status reports even when schedules didn't overlap. The ability to access files and programs through dial-in lines allowed occasional family visits and the speedy incorporation of middle-of-the-night inspirations.

[6] Not wishing to follow this clandestine approach we went so far as to write an article called *Ten Tips from the Programming Pros*, for the Spring 1984 issue of Atari Connection Magazine in which we described many of the techniques we used to write our games.

Miscellaneous Tools

We used everything from *adb* to *yacc* at one time or another in this project. We got recalcitrant programs to work with *adb*, we kept track of important deadlines using *calendar*, we found the changes you make by falling asleep *on* your keyboard with *diff*, we kept track of immense numbers of interdependent modules with *make*, we printed cartridge dumps for the copyright office with *od*, we scored music with *pic*, we made global name changes with *sed*, we prepared press releases using *spell*, *tbl*, and *troff*, we borrowed tools from other sites with *uucp*, we disassembled other designer's programs with *yacc*, etc., etc. Because we had access to these tools we were able to participate in parts of the production that normally are either left to specialists who have no other connection with the project or are left undone entirely. As a result the games display an unusual coherence, with a common thread of design style unifying all the parts.

We also found that other parts of the Computer Division had created tools useful to us. The Pixar (graphics processor) project had written a program to load PROMs that we found useful. The Digital Audio project had written a program (LUDS, pronounced "Lewds") to aid in circuit design and create schematics, wire lists, etc. for their design of the ASP digital sound processor. We used LUDS for our dual-ported memory design.

Finally, we used programs written for, or as a result of, other game projects. These ranged from programs that helped maintain large numbers of related source files to programs written to manipulate the masses of data produced by games like "Empire".

Although none of these programs were written with our specific needs in mind they, like much of the software associated with UNIX systems, were written to make as few limiting assumptions as possible and were perfectly adequate for our needs.

File Sharing

Crucial to a team approach to software development is the ability to break programs up into separate modules and share access to them among the entire team. The UNIX operating system made this easy although a little more concurrency control in the various editors would have avoided one or two minidisasters. Often three people would be working on related parts of a game, passing files back and forth for advice or criticism while a fourth was compiling and testing the results of the others' changes.

Input/Output Redirection

Many of the tools that we used could not have existed on a system where the author of the software has to know what kind of device the data is coming from and what kind of device it will go to. Because we had to know less about the intended uses of a tool at the time we wrote it we were often able to use an existing tool to solve a problem instead of having to create a new one.

UNIX INFLUENCES

The ways in which the UNIX system influenced our games development are subtle, as influences often are. While some of the members of the games group had not been exposed to the UNIX operating system before joining, the majority of people in the group were long-time UNIX aficionados and their enthusiasm for the system was adopted by all.

Many of the influences we felt were philosophical and although no one of them is exclusively associated with UNIX software, the group, taken as a whole, implies UNIX influence strongly. It would be impossible to consider the influence of the UNIX system without including the influence of people like Brian Kernighan and John Mashey who have been the eloquent spokesmen for many of the ideas and philosophies that permeate the system.

The concept often called "fail fast" seemed particularly relevant to games development. Much of the decision making in game design comes down to making a yes-or-no decision about a potential approach. Such choices vary from "is this idea interesting enough?" to "can it be done at all?" We found it immensely effective to try solving the hardest parts first in deciding any of these questions, allowing us to discard doomed approaches quickly and devote our energies to potentially successful ones.

We always tried to make individual *tools* that performed a single general task well and then use combinations of such tools to perform complex tasks rather than making a new program to solve each specific complex task. As a result we had to write relatively few tools and ended up using each in many contexts, often in concert with tools written by others.

The programs we wrote generated little or no gratuitous output. Thus, when someone wanted to make that program a part of some larger construct, there was little or no effort spent trying to get rid of useless verbosity. Further, when diagnostics were called for they really stood out rather than being lost in a sequence of "program starting...", "Pass 1 completed", "pass 2 loading...", and so forth.

We found simplicity to be an even better policy than honesty. Every time we were tempted to elaborate on some program the result was an unending job of fine tuning the elaborations probably difficult because they never really belonged in the first place. When we stayed simple it was easy to see what needed to be done and easy to tell when we were finished and could go on to the next task.

UNIX ENVIRONMENT

The UNIX environment was perfectly suited to our task. When we needed a program it was either already there or it was easy to write. Perhaps that's because the UNIX system was designed by (game) programmers to make the job of program development as easy as possible. In any case it made *our* job easy. We didn't have to fight with programs that almost did what we needed but had made some limiting assumption. Nor did we have to trick the operating system into letting us do something that it thought we shouldn't do.

We were heavily influenced by the software, philosophies, and concepts associated with the UNIX operating system. It could be argued that the time was right for these developments and that the UNIX system was just the vehicle for ideas that would have appeared anyway; but it's clear that without these our games projects would have ended very differently. We would have spent more time chasing down blind alleys and would have examined fewer potentially profitable approaches. We would not have been able to participate in as many areas of the production as we did, thereby leaving many crucial design decisions to the same people who crank out the standard, bland products that flood the market. Worst of all, we probably would have had to throw away the "throw-away" games not only because we would not have had time to complete them but they would not have been as interesting as they are.

THE GAMES

In a paraphrase of the reviewer from *Video Game Update* "These games simply must be experienced to be appreciated."

CREDITS

The work described in this paper was done by the members of the Lucasfilm Games Group, a part of the Lucasfilm Computer Division which, in turn, was a part of Lucasfilm Ltd. Although I headed and often spoke for the group it would be a mistake to think that I contributed more than any of the others in the group. Every member performed an irreplaceable function in the development of these games, and without any one of these people the games would not have been as they are, indeed, they might not have been at all.

The cast of characters, in alphabetical order, is:
Loren Carpenter
David Fox
Charlie Kellner
Peter Langston
David Levine
Gary Winnick

Others made contributions to our effort that should not be overlooked: Steve Arnold, Eric Benson, Steve Cantor, Ed Catmull, Terry Chostner, Mike Cross, Marty Cutler, Bob Doris, Gary Hare, Charlie Keagle, George Lucas, Lyle Mays, Pat Metheny, Lorne Peterson, Rob Poor, David Riordan, Richie Shulberg, Steven Spielberg

REVIEWS

"And the games, some enthusiasts say, are among the most innovative on the market."
-- Businessweek

"Its [*BALLBLAZER*] high-speed action and split screen ... may leave players a bit dizzy."
-- the San Francisco Chronicle

"All of which promises to do for the game industry what Lucasfilm has done for the film industry: rewrite the standards."
-- Paul Cohen in Atari Connection Magazine

"... both games include sharp, sophisticated graphics and challenging game strategy."
-- the San Francisco Examiner

"... cartridges that materially advance the state-of-the-art."
-- Arnie Katz in Electronic Games Magazine

"... the 3-D look makes a game that is chock full of sensory excitement."
-- San Jose Mercury News

"It sounds like John Coltrane!"
-- Pat Metheny

"*BALLBLAZER* ... meets the standards that Lucasfilm set in motion pictures"
-- Alan Michaels in Atari Connection Magazine

"*BALLBLAZER* ... is by far one of the most interesting two-player competition games to date."
-- Newsweek Access Magazine

"... the latest breakthroughs in the art of video game design."
-- Howard Pearlmutter, the Knoware Institute

"A breathtaking technical innovation in games design"
-- Paula Polley & Marina Hirsch in Atari Connection Magazine

"... sophisticated animation and graphics technology heretofore only seen in more advanced computer simulations."
-- the Hollywood Reporter

"We were very impressed with this game"
-- The Video Game Update

"[*BALLBLAZER* & Rescue on Fractalus!] ... use more sophisticated computer graphics and animation techniques and have more of a three-dimensional appearance than most existing games."
-- Wall Street Journal

TRADEMARKS

UNIX is a trademark of AT&T Bell Laboratories; Apple II is a trademark of Apple Computer Inc.; Atari is probably a trademark of Atari Inc.; DEC and VAX are trademarks of Digital Equipment Corporation; Picture System I is a trademark of Evans & Sutherland, Sun Workstation is a trademark of Sun Microsystems Inc.; Rescue on Fractalus, Ballblazer, Pixar, LUDS, and ASP are trademarks of Lucasfilm Ltd.

A Bit About Eighth Edition

Harold Cross

Version Eight, Eighth Edition, V8; these names refer to the flavor of UNIX which is currently in use by the Computing Science Research Laboratory of AT&T Bell Laboratories. This article is meant to familiarize the reader with V8. It does not describe it in any great detail. For additional information see the references or send me mail (bellcore!hac).

V8 is based on 4.1 BSD. Naturally 4.1 didn't spin around on 1127's disks for long before changes were being made. But it wasn't dubbed Eighth Edition until about two years ago.

> From research!dmr Mon May 30 01:45 EDT 1983
> :tcejbuS v8
>
> The Eighth Edition System is the line discipline stuff, plus PJW's
> 4K file system, plus his remote file system. I.e. we decided
> to give our state a name. Partly this was to disarm complaints
> that we were running 4BSD. Also, Doug is trying to arrange a new
> manual, so besides the considerable system changes there may be an
> actual printed 8th edition manual.

Streams

The line discipline stuff was first described publicly by **dmr** at the Winter 1981 USENIX meeting. Further coverage is found in [1]. Briefly, it is a mechanism providing a full duplex channel through which processes (user level and kernel) communicate. It is also known as a stacked line discipline. Processing modules can be pushed into (and popped) from the channel. Thus, for instance, the *init* program opens a terminal device and pushes a "tty" line discipline into a channel between it and the terminal. Likewise, when switching handlers from the "old" to "new" disciplines using the *stty* program, it first pops the old one from the channel and then pushes in the new one.

The various disciplines are kernel objects (functions). This provides an elegant (clean in design, implementation and use) mechanism that isolates many common character processing functions from device drivers in the kernel. The generality afforded is also exploited to do such things as hardware simulation or, as **rob** has done with the 5620 terminal, to place the terminal handler in another processor.

Fast File System

pjw's 4k file system is a fast file system which coexists with standard 4.1 BSD type file systems† There are two aspects which make this implementation faster than the 4.1 file system (and probably as fast as the 4.2 file system).

The block size is 4K bytes. More interesting is the fact that the "free list" is described by a bitmap. The bitmap resides in core, allowing for quickly locating free blocks and even more quickly adding blocks to the free list. A side effect of this implementation (or perhaps its impetus) is that the search for a free block (given the previous used block) can efficiently locate one on an appropriate cylinder (if there's one available). The latter aspect is probably the most significant factor in

†In fact the superblock structure is rearranged making it necessary to "fix" a 4.1 file system using fsck. But it's a simple matter to rearrange it so that this isn't necessary.

overall increased throughput.

I mentioned that the 4k file system coexists with the older type. The file system structure contains a union of the two free list implementations and the appropriate I/O routines check the file system type. Although not as gross as the 4.2 file system, this implementation also trades off conceptual simplicity for efficiency.

Network File System

pjw's conceptual pendulum swings the other way with regard to the remote file system. Here entirely new capabilities are added in a straightforward manner. The remote file system uses a hierarchical syntax where remote machines' file systems are mounted on the local file system (the convention is /n/machine/...). It's implemented in the kernel on the local machine and at the user level on the remote. The implementation is transparent to the users' programs. Locally, there is a file system switch (ala cdevsw) that causes the appropriate routines to be invoked on the different types of files (local, network, processes (see later), and faces‡). Routines that access networked file systems do so by invoking a server on the remote machine.

One of the nicest aspects of this system is its generality. A remote file system is mounted by telling the system the local mount point and giving it a stream connected to the remote file server. Thus the network file system can theoretically run over any communications path (a modem, a tty line, Ethernet, PCL, etc.). Since the server is a program utilizing nothing more system dependent than *select* (and an understanding of files), it can run under any version of UNIX. This means I can share anyone else's file systems but not vice versa.

Another new type of file is implemented in the concept of processes as files [2]. Here the directory "/proc" contains files that represent running processes. The standard file access routines in this case interact with the address space of the said processes. This is a nifty way to manipulate them. (By the way, there are 128 file descriptors in the Eighth Edition.)

V8 is more than the key kernel changes described above. Next installment I'll describe some of the wonderful utilities and applications available under V8. But the system is merely a reflection of its designers, contributors and maintainers, most of whom just seem to possess extraordinarily good taste. Remember the Seventh Edition?

References

[1] "A Stream Input-Output System", Dennis Ritchie. B.L.T.J. 63:8, October 1984, pp. 1897-1910.

[2] "Processes as Files", T. J. Killian. Proceedings of the Summer 1984 USENIX Conference, Salt Lake City, Utah.

‡When a process opens a file of this type, the appropriate representation of someone's face is retrieved. More on this next time.

HELP WANTED ......

Dear Sirs,

Recently the Department of Energy Coordination purchased a radio shack microcomputer TRS 80, Model 16B, supported by the XENIX operational system.

We urgently require, therefore, up-to-date information on new developments and software for XENIX/UNIX and the help of your members would be much appreciated.

Central de Anchicayá, Ltda., is an electricity utility dedicated to the generation and transmission of electrical energy in Colombia, South America.

Please contact me as follows:

Ingeniero JORGE ARIZMENDI MENDOZA,
Central de Anchicayá, Ltda.,
Apartado aéreo 1545, Cali, Colombia, Sur América.

MORE HELP WANTED.....

Dear Sirs,

We are a software house which has been established in Buenos Aires, Argentina, since 1974.

We are now developing software applied to the UNIX operating system and urgently require information about:

"UNIX" and "C" language; Application and base software, mainly applied to NCR TOWER 1632; and systems applied to Newspaper Industry.

Help from you or your members would be greatly appreciated.

Aldo F. Albarellos,
Manager,
Seinco S.R.L.,
Av. Belgrano 271,
2do Piso,
1092-Buenos Aires,
Argentina.

A T and T BELL LABORATORIES - SYSTEM V INTERFACE DEFINITION

This document is intended for use by anyone who must understand the source-level interfaces that are consistent across all System V environments. As such, its primary audience is the application writer who is building C language applications whose source code must be portable from one System V environment to another. In addition, a system builder should view this document as a necessary condition for supporting a System V environment which will host such applications.

The System V Interface Definition specifies the operating system components that are available to users and application programs. The components and their functionality are defined, but no specification is made of their implementation. The Definition specifies the application source code interfaces as well as the runtime behavior seen by an application program. The emphasis is on defining a common computing environment for applications and users and not on the internals of the operating system, such as the scheduler and memory manager.

Chapter Headings
(1) DEFINITION OF THE BASE SYSTEM V INTERFACE; (2) KERNEL EXTENSION; (3) OTHER PLANNED EXTENSIONS; (4) FUTURE DIRECTIONS; (5) COMPARISON TO THE 1984 /USR/GROUP STANDARD; (6) OPERATING SYSTEM (OS) SERVICES; (7) SIGNALS; (8) OTHER LIBRARY ROUTINES; (9) HEADER FILES; (10) SPECIAL DEVICE FILES; (11) SYSTEM V ERROR MESSAGE STANDARD; (12) SYSTEM V COMMAND SYNTAX STANDARD

This, together with similar publications such as the **"Bell Systems Technical (1978 and 1984)"** are available from:

Greville Edwards,
UNIX Europe Limited,
27a Carlton Drive,
Putney,
London SW15 2BS,
ENGLAND,
Tel: 01-785 6972

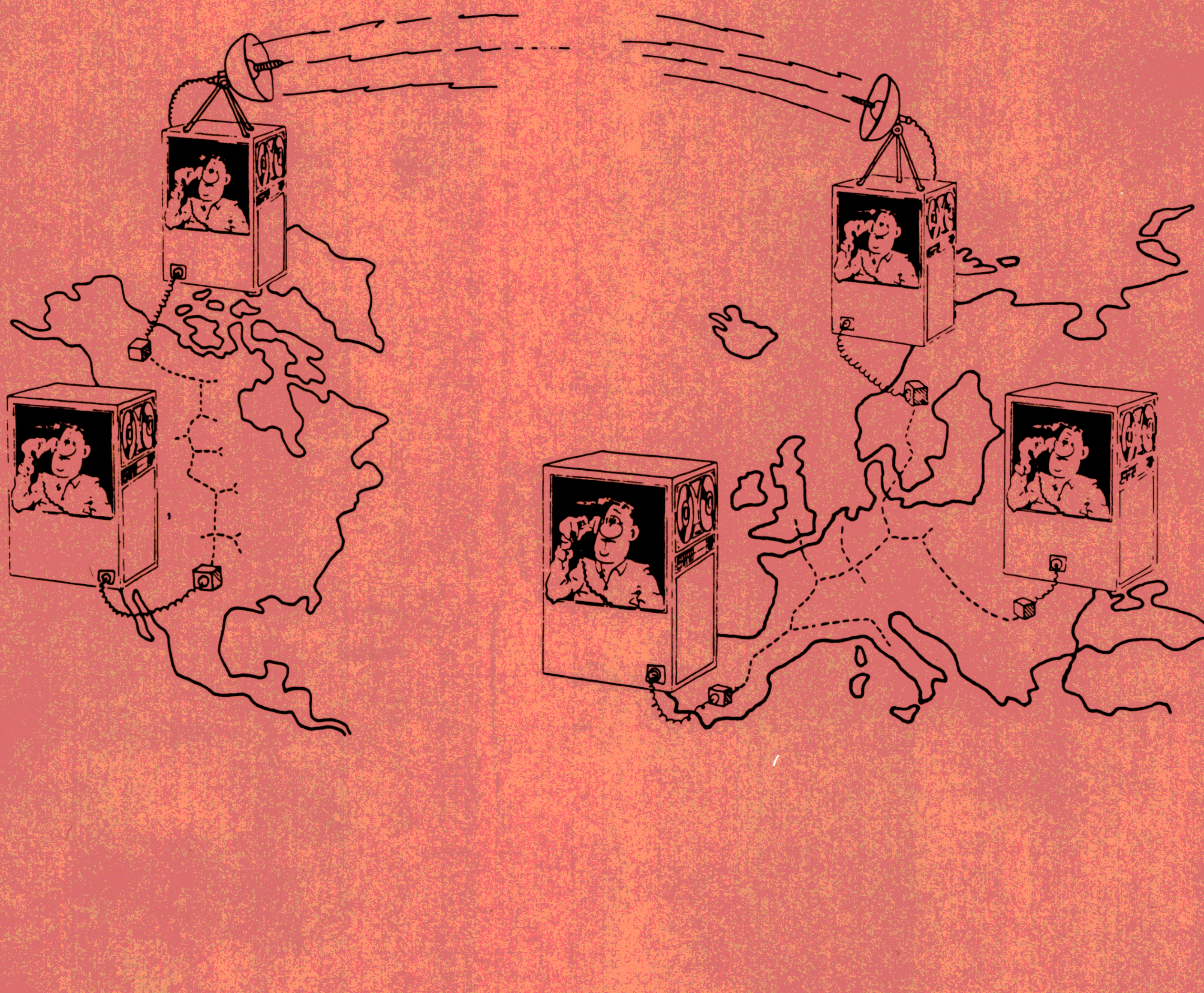
CALLING ALL .... SCHOOLS

Dear Colleagues,

I am a teacher trying to convince schools that a UNIX-supermicro is better than a bundle of PCs. I am aware that there are some schools in Germany, Great Britain and America that already use UNIX.

If there are any more out there in the wilderness, let's get in touch with each other. Here's my address:

Hans-Josef Heck,
Breslauerstr. 25,
5630 Remscheid,
West Germany.
Tel: 02191 - 34 00 64



The Secretary
European Unix® Systems User Group
Owles Hall
Buntingford, Herts.
SG9 9PL.
Tel: Royston (0763) 73039