AUUGN

Australian Unix systems User Group Newsletter

Volume 9 - Number 6

December 1988

Registered by Australia Post Publication No. NBG6524

Warren Toomey 21/2/89

The Australian UNIX* systems User Group Newsletter

Volume 9 Number 6

December 1988

CONTENTS

* UNIX is a registered trademark of AT&T in the USA and other countries.

AUUG General Information

Memberships and Subscriptions

Membership, Change of Address, and Subscription forms can be found at the end of this issue.

All correspondence concerning membership of the AUUG should be addressed to:-

The AUUG Membership Secretary,
P.O. Box 366,
Kensington, N.S.W. 2033.
AUSTRALIA

General Correspondence

All other correspondence for the AUUG should be addressed to:-

The AUUG Secretary,
P.O. Box 366,
Kensington, N.S.W. 2033.
AUSTRALIA

AUUG Executive

President	**Greg Rose** *greg@softway.sw.oz* Softway Pty. Ltd., New South Wales	Secretary	**Tim Roper** *timr@labtam.oz* Labtam Limited, Victoria
Treasurer	**Michael Tuke** *no net address* Victoria		
Committee Members	**Frank Crawford** *frank@teti.qhtours.oz* Q.H. Tours, New South Wales		**Richard Burridge** *richb@sunaus.aus.oz* Sun Microsystems Austrlia New South Wales
	Chris Maltby *chris@softway.sw.oz* Softway Pty. Ltd., New South Wales		**Tim Segall** *tim@hpausla.aso.hp.oz* Hewlett Packard Australia, Victoria

Next AUUG Meeting

Regional Meetings will be held during February 1989, and the major Conference and Exhibition, AUUG89 will be held at the Sydney Hilton Hotel from Tuesday 8th to Friday 11th August 1989. Further details are provided in this issue.

Editorial

Welcome to the Newsletter.

The year 1988 was important for the UNIX community as I think it was the year that UNIX was taken first taken seriously by the general computing community, who ever that may be. For example, it was very rare to be able open a Computer Magazine and not find an article which at least mentioned UNIX. Usually it was about the battles between OSF and AT&T or the product line of major company such as IBM or DEC. DEC are even talking about making VAX/VMS - POSIX (Unix?) compatible.

I am not so sure UNIX as we now know it, will survive this intense interest. My hope is that these forces will produce something that provides us with that *something* that we each find in UNIX. I leave it for others to tell us what the *something* is.

The Newsletter will continue in 1989 and will need your continued support by the way of subscription via Membership and articles. Here is the intended copy deadlines for Newsletter during the next year.

Number 1 - Friday February 17th 1989
Number 2 - Friday April 14th 1989
Number 3 - Friday June 16th 1989
Number 4 - Friday July 14th 1989 - AUUG89 Conference issue
Number 5 - Friday October 20th 1989
Number 6 - Friday December 15th 1989

Also next year, expect a new cover format featuring the new AUUG logo and utilising the new coloured spine to which appeared on the AUUG88 issue to display issue details. The colours for each volume or years issue will be the colours of rainbow.

In THIS ISSUE, you will find preminary details of the Regional Meetings that will be held in Februry 1989. You should try and attend as I am sure you have much to gain from them. An interesting book review on "Godel, Echer, Bach: An Eternal Goldern Braid", and article on porting the Berkeley Filesystem to UNIX System V. There is some excerpts collected from the Net-news by David Horsfall, a well known *aus* news personality. Reprints from the latest issues of Newsletters of the European (EUUG) and U.S. (USENIX) User Groups are also included.

The AUUG is announcing two exciting offers for members ONLY. The first is the Inguugral AUUG Software Distribution which contains X windows and other useful public domain software. The second is that the Nutshell Handbooks are available from AUUG to members at a discount. Details can be found within. Two new very good reasons to join the User Group if you are not already. Membership forms can be found and photocopied from the back of this issue.

I hope you enjoy this issue and please feel free to contribute an article soon. I have not recieved a letter to the Editor for a long time - how about it ?

Thanks to those of you that send me articles and contributed to the production of issue - it is very much appreciated.

REMEMBER, if the mailing label that comes with this issue is highlighted, it is time to renew your AUUG membership.

AUUGN Correspondence

All correspondence reguarding the AUUGN should be addressed to:-

John Carey
AUUGN Editor
Webster Computer Corporation
1270 Ferntree Gully Road
Scoresby, Victoria 3179
AUSTRALIA

ACSnet: *john@wcc.oz*

Phone: +61 3 764 1100

Contributions

The Newsletter is published approximately every two months. The deadline for contributions for the next issue is Friday the 17th of February 1989.

Contributions should be sent to the Editor at the above address.

I prefer documents sent to me by via electronic mail and formatted using *troff -mm* and my footer macros, troff using any of the standard macro and preprocessor packages (-ms, -me, -mm, pic, tbl, eqn) as well TeX, and LaTeX will be accepted.

Hardcopy submissions should be on A4 with 35 mm left at the top and bottom so that the AUUGN footers can be pasted on to the page. Small page numbers printed in the footer area would help.

Advertising

Advertisements for the AUUG are welcome. They must be submitted on an A4 page. No partial page advertisements will be accepted. The current rate is AUD$ 200 dollars per page.

Mailing Lists

For the purchase of the AUUGN mailing list, please contact Tim Roper.

Back Issues

Various back issues of the AUUGN are available on request from the Editor.

Acknowledgement

This Newsletter was produced with the kind assistance and equipment provided by Webster Computer Corporation.

Disclaimer

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

Adelaide UNIX Users Group

The Adelaide UNIX Users Group has been meeting on a formal basis for 12 months. Meetings are held on the third Wednesday of each month. To date, all meetings have been held at the University of Adelaide. However, it was recently decided to change the meeting time from noon to 6pm. This has necessitated a change of venue, and, as from April, meetings will be held at the offices of Olivetti Australia.

In addition to disseminating information about new products and network status, time is allocated at each meeting for the raising of specific UNIX related problems and for a brief (15-20 minute) presentation on an area of interest. Listed below is a sampling of recent talks.

D. Jarvis	"The UNIX Literature"
K. Maciunas	"Security"
R. Lamacraft	"UNIX on Micros"
W. Hosking	"Office Automation"
P. Cheney	"Commercial Applications of UNIX"
J. Jarvis	"troff/ditroff"

The mailing list currently numbers 34, with a healthy representation (40%) from commercial enterprises. For further information, contact Dennis Jarvis (dhj@aegir.dmt.oz) on (08) 268 0156.

Dennis Jarvis,
Secretary, AdUUG.

Dennis Jarvis, CSIRO, PO Box 4, Woodville, S.A. 5011, Australia.

PHONE: +61 8 268 0156

UUCP: {decvax,pesnta,vax135}!mulga!aegir.dmt.oz!dhj
ARPA: dhj%aegir.dmt.oz!dhj@seismo.arpa
CSNET: dhj@aegir.dmt.oz

WAUG
Western Australian UNIX systems Group
PO Box 877, WEST PERTH 6005

Western Australian Unix systems Group

The Western Australian UNIX systems Group (**WAUG**) was formed in late 1984, but floundered until after the 1986 AUUG meeting in Perth. Spurred on by the AUUG publicity and greater commercial interest and acceptability of UNIX systems, the group reformed and has grown to over 70 members, including 16 corporate members.

A major activity of the group are monthly meetings. Invited speakers address the group on topics including new hardware, software packages and technical dissertations. After the meeting, we gather for refreshments, and an opportunity to informally discuss any points of interest. Formal business is kept to a minimum.

Meetings are held on the third Wednesday of each month, at 6pm. The (nominal) venue is "University House" at the University of Western Australia, although this often varies to take advantage of corporate sponsorship and facilities provided by the speakers.

The group also produces a periodic Newsletter, **YAUN** (Yet Another UNIX Newsletter), containing members contributions and extracts from various UNIX Newsletters and extensive network news services. **YAUN** provides members with some of the latest news and information available.

For further information contact the Secretary, Skipton Ryper on (09) 222 1438, or Glenn Huxtable (glenn@wacsvax.uwa.oz) on (09) 380 2878.

Glenn Huxtable,
Membership Secretary, **WAUG**

AUUG Institutional Members

ACUS / UNISYS
Australian National University
Australian Nuclear Science & Technology Organisation
Australian Telescope Computer Group (CSIRO)
Autodesk Australia Pty Ltd
CSIRO DIT
DBA Computer Systems Pty Ltd
Department of Industry Technology and Commerce
Dept of Lands - Central Mapping Authority
Flinders University, Discipline of Computer Science
Fujitsu Australia Limited
Great Barrier Reef Marine Park Authority
Harris & Sutherland Pty Ltd
Honeywell Information Systems
Ipec Transport Group
Lands Department (Queensland)
Monash University Computer Science
Olympic Amusements Pty Ltd
Prentice Computer Centre
Prime Computer Research & Development
Pyramid Technology Australia
Q.H. Tours Limited
Qld State Govt Computer Centre
Racecourse Totalizators Pty Ltd
Reark Resources
Sigma Data Corporation Pty Ltd
South Australian Institute of Technology
Sun Microsystems Australia
Swinburne Institute of Technology
University of Adelaide
University of New England
University of New South Wales
University of Sydney
University of Technology Sydney - Computing Services Division
University of Wollongong
Webster Computer Corporation

AUUG Summer 1989 Meetings

Timothy Roper

Secretary

timr@labtam.oz

As reported in the last issue of AUUGN, AUUG will be holding informal, regional, technical meetings early in 1989. We had hoped to publish full details of guest speakers, venues and dates in this issue of AUUGN but unfortunately they are not quite finalised. There will be electronic and paper mailouts in late January or early February giving all those details.

At the moment we can confirm that the meetings will all be in mid to late February. They are definitely being planned in at least the following cities, hosted by the organisations/people listed.

Perth	Uni of WA	Bernd Felsche	bhf@bhfamy.acp.oz
Geelong	Deakin Uni	Craig Warren	ccw@kokab.cc.deakin.oz
Hobart	Uni of Tasmania	Steven Bittinger	steveb@diemen.cc.utas.oz
Brisbane	Uni of Qld/CiTR	Sarah Barry	
Townsville	James Cook Uni	Ian Hunter	ian@sheila.cs.jcu.oz
Darwin	Cass Group	Ken Frame	

Meetings in other areas may still be possible (eg. NSW/ACT and SA) if someone volunteers to organise them soon enough. Anyone interested in doing so should contact the President, Greg Rose.

"I am Lying"

(Epimenides Or Liar Paradox)

Jack Dikian

Q.H. Tours Ltd.
141 Walker St. North Sydney NSW 2060
jack@teti.qhtours.oz

A Review of Douglas R. Hofstadter's "Godel, Escher, Bach: An Eternal Golden Braid". Penguin Books 1980, ISBN 0-14-005579-1.

What can Kurt Godel a mathematician, M.C. Escher a graphic artist and Johann Sebastian Bach the famous composer have in common. Intending to write an essay about Godel's theorem, Hofstadter gradually expanded to include Bach and Escher until finally he realized that the works of these men were in essence "shadows cast in different directions by some central theme". The book tries to reconstruct this central theme.

A single musical theme can be played against itself (by say having 'copies' of the theme played by various participating voices) to create what is known as a canon. Songs such as "Frere Jacques" or "Row, Row, Row Your Boat" are examples of canons. "Frere Jacque" for example, enters in the first voice and after a fixed time delay, a 'copy' of it enters, in not necessarily the same key. After the same fixed time delay in the second voice, the third voice enters carrying the theme, and so on. In order for the theme to work (as a canon theme), each of the notes must be able to serve as a dual (or triple, or quadruple) role: The notes must be part of a melody as well as part of a harmonization of the same melody. When there are three canonical voices, for instance, each note of the theme must act in two distinct harmonic ways, as well as melodically. Thus, each note in a canon has more than one musical meaning. The human brain will (well!) automatically figure out the appropriate meaning, by referring to context.

However, 'copies' of the theme can be staggered not only in time, but also in pitch. The first voice might sing the theme starting on C, and the second voice overlapping the first voice, might sing the identical theme starting five notes higher, on G. It is also possible that speeds of the different voices might not be equal. That is, the second voice might sing twice as quickly, or twice as slow, as the first voice. The former is called diminution, the latter augmentation (since the theme seems to shrink or to expand). To complicate matters, it is also possible to invert the theme which means the second voice jumps down wherever the first theme jumps up, by exactly the same number of semitones. Bach was especially fond of inversions, and they show up often in his work. Finally, the most esoteric of 'copies' is the retrograde copy. This is where the theme is played backwards in time. It should be noted at this point that all the copies mentioned preserve all information in the original theme. The original theme is fully recoverable from any of the copies. The book is in fact full of "information preserving transformations" or isomorphisms.

A fugue is like a canon, in that it is based on the theme which gets played in different voices and different keys, speeds, directions etc. However, the notion of fugue is much less rigid than that of canon, and consequently it allows for more emotional and artistic expressions.

Hofstader introduces us to the notion of "Strange Loops" for the first time by giving us an example of one of Bach's musicical offerings to the king called "Canon per Tonos". This theme has three voices with the lower of the voices singing in C minor. This is the key of the canon as a whole. What makes this theme different however, is that when it concludes, or, rather, seems to conclude, it is no longer in the key of C minor, but now is in D minor. Bach has somehow changed keys right under the listener's ear. It is also so constructed that the "ending" ties smoothly onto the beginning again only this time one key higher. After exactly six such modulations, the original key of C minor is restored. Hofstadter uses this as an example to illustrate the concept which he calls "strange loops". Here he says "The strange loop phenomenon occurs whenever, by moving upwards, or downwards through the level of some

hierarchical system, we unexpectedly find ourselves right back where we started..". Strange loops occur over and over again in this book.

For many years, mathematicians were among the first admirers of Escher's drawings. Hofstadter explains how one of the most powerful visual realizations of the notion of strange loops are presented in the work of Escher. Many of the drawings have their origins in paradox, illusions, or double meaning. However, strange loops are the most recurrent themes in Escher's work. Escher's 1961 lithograph "Waterfall" is compared to Bach's "Canon per Tonos". The Waterfall is an illustration of a six-step endlessly falling loop. The concept of "Loop Tightness" is introduced by comparing it to levels in the strange loop. For example, both Bach's "Canon per Tonos" and Escher's "Waterfall" are six-step strange loops. As the loop is "tightened", the example of the "Drawing Hands" is illustrated. Here, each of the two hands draws the other: a two-step strange loop. Finally, the tightest of all strange loops is realized in "Print Gallery". This is a picture of a picture which contains itself. Hofstadter asks; Or is it a picture of a gallery which contains itself? Or of a town which contains itself?

Intuition senses that there is something mathematical involved in both Bach's and Escher's works. In fact, just as the Bach and Escher loops appeal to the very simple and ancient intuitions - a musical scale, a stair case - so the discovery, by Godel of starage loops in mathematical systems. Godel's discovery (in its absolutely barest form) involves the translation of an ancient paradox in philosophy into mathematical terms.

The statement, "This statement is false", is usually considered a fair demonstration of the Epimenides Paradox. If you tentatively think it is true, then it immediately backfires on you and makes you think it is false, but once you decide it is false, a similar backfiring returns you to the idea that it must be true. The Epimenides paradox is a one-step strange loop, like Escher's Print Gallery. But how does it have to do with mathematics? That is what Godel discovered.

Godel's idea was to use mathematical reasoning in explaining mathematical reasoning itself. This notion of making mathematics "introspective" proved to be enormously powerful. Godel's greatest work resulted in "Godel's Incompleteness Theorem". Godel's theorem appears as proposition VI is his 1931 paper "On Formally Undecidable Propositions in Principia Mathematica and Related Systems I" It states:

> To ever w-consistent recursive class k of formulae
> there corresponds recursive class-sign r, such that
> neither v Gen r nor Neg(v Gen r) belongs to Flg(k)
> (where v is the free variable of r).

This was actually presented in German, and you may feel that it might as well be in German anyway. Another attempt; A paraphrase in "English":-

> All consistent axiomatic formulations of number
> theory include undecidable propositions.

The proof of Godel's incompleteness theorem hinges upon the writing of a self-referential mathamatical statement, in the same way as the Epimenidies paradox is a self-referential statement of language. But where it is very simple to talk about language in language, it is not at all easy to see how a statement about numbers can talk about itself. The difficulity arises when we consider that mathematical statements (Number theoretical) are about properties of whole numbers. Whole numbers are not statements, nor are their properties. A statement of number theory is not about a statement of number theory; it just is a statement of number theory.

Godel had the insight that a statement of number theory could be about a statement of number theory, if only numbers could somehow stand for statements. The idea of a code was therefore emerging. In the Godel code, usually called "Godel-Numbering", numbers are made to stand for symbols and sequence of symbols. Each statment of number theory, being a sequence of specialized symbols, acquires a Godel number, something like a telephone number, by which it can be referred to. This coding scheme allows statements of number theory to be understood on

two different levels. As a statement of number theory, and also as statements about statements of number theory.

Godel finally transplanted the Epimenidies paradox to the statement "This statement of number theory does not have any proof" or more correctly, "This statement of number theory does not have any proof in the system of Principia Mathematica" whereis the Epimenides statement creates a paradox since it is neither true or false, the Godel sentence is unprovable inside P.M but true.

Hofstadter procedes to place Godel's work in its historic background. Russell's, Grelling's, Whitehead's and Hilberts works are touched on.

I know of no better explaination than this book presents of what Godel achieved and of the implications of his revolutionay discovery. That discovery concerns in particular recursion, self-reference and endless regress, and Hofstadter finds the three themes mirrored in the art of Escher and the music of Bach.

Each chapter is preceeded by a kind of prelude which early in the book takes the form of a "Dialogue" between Achilles and the Tortoise (from Zeno of Elea). Almost all concepts are presented twice. Firstly, metaphorically in the Dialogue, yielding visual images. These then serve, during the reading of the following chapter, intuitive background for a more serious and abstract presentations of the same concept. Each Dialogue is patterned on a composition by Bach. For example, if the composition has three voices, so does the the corresponding conversation. If the composition has a theme that is turned upside down or played backwards, so does the particular conversation. Each dialogue states, in a comic way (Lots of word play), the themes that will be further explored in the chapter that follows.

The book is packed with examples of strange loops, loops that exemplify the self-reference that is one of the book's central themes. For example consider the following impossibility:-

The following sentence is false
The preceeding sentence is true.

Taken together, the sentences have the same effect as the original Epimenides paradox (or reminiscent of Drawing Hands), yet separately, they are harmless. Nevertheless, we who see the "picture" or statement as a whole can escape the paradox by "Jumping" out of the "system" to view it from a meta- level, just as we can escape the traditional paradox of logic by jumping into a meta language.

Hofstadter introduces modern mathematical logic, computability theory, Feynman diagrams for particals that travel backward in time, Fermat's last theorm, Turing machines, computer chess, computer music, expert system on the simulation of Natural languages, molecular biology and many other fascinating topics.

The book's discussion of artificial intelligence is also stimulating. Does the human brain obey formal rules of logic? He believes no computer will ever do all a human brain can do until it somehow reproduces that hardware.

Formal systems are discussed and the reader is urged to work out a puzzle to gain familiarity with formal systems in general. Fundamental notions such as "theorem", "rule", "inference" etc are introduced.

The idea of recursion is presented in various contexts. These include musical patterns, linguistic patterns, mathematical functions, computer algorithims etc. The proceeding Dialogue ("Little Harmonic Labyrinth") to the section on recursion is about stories within stories. The frame or outer story is left open, instead of finishing as expected, thus leaving the reader dangling without resulation.

A discussion of how meaning is split among coded messages is found with examples including strands of DNA and undeciphered inscriptions on ancient tablets. The relationship of intelligence to "absolute" meaning is postulated. In the proceeding Dialogue, Achilles and the Tortoise try to resolve the question, "which contains more information - a record, or the phonograph which plays it".

A chapter discusses the idea's of Zen Buddhism. Hofstadter suggests that in a way, Zen ideas bear a metaphorical resemblance to some contemporary ideas in the philosophy of mathematics. Godel's theorem is visited.

Brains and Thought is the title of chapter XI. Here, the question "How can thoughts be supported by the hardware of the brain" is asked. An overview of the large and small scale structure of the brain is given. The relationship between neural activity and concept is discussed.

The Church-Turing thesis, which relates mental activity to computation is presented in several ways. These are analysed in view of their implications for simulating human thought mecanically. A discussion of the famous "Turing test" opens a chapter on Artificial Intelligence. A Dialogue is found where the idea of how we unconsciously organize thoughts so that we can imagine hypothetical variants on the real world all the time makes the subject matter.

The book closes with a wild dialogue which is simullaneously patterned after bach's six-part Ricecar and the story of how Bach came to write his musical offering. In his dialogue, the computer pioneers Turning and Babbage improvise at the keyboard of a flexible computer called a "smart-stupid" which can be as smart or as stupid as the programmer wants. Turning produces on his computer screen a simulation of Babbage. Babbage however, is seen looking at the screen of his own "smart-stupid", on which he has conjured up a simulation of Turning. Each man insists he is the real and the other is no more than a proposed program. An effort is made to resolve the debate by playing the Turning game, which was proposed by Turning as a way to disinguish a human being from a computer program. At this point Hofstader himself walks into the scene and convinces Turing, Baggage and all the others that they are creatures of his own imagination.

Porting the Berkeley Fast File System to System V

Stephen Prince
sp@labtam.OZ

Labtam Information Systems
Braeside, Victoria 3195

ABSTRACT

This paper describes a prototype implementation of the Berkeley Fast File System (FFS) within AT&T's System V Release 3 version of UNIX.†

The specific goals of this work were, firstly to implement the FFS within the framework of the System V File System Switch (FSS) architecture. The FSS abstraction provides an object-oriented interface in System V for file system objects and their associated operations. The second major goal was to ensure binary transparency of existing user applications. This was achieved by ensuring System V file system semantics within the new file system. A hybrid system containing the original System V and new file system was constructed and each architecture was further examined to understand performance implications of disk and file system layout and how that layout affects the file system implementation.

This paper discusses the implementation of Berkeley file system abstractions within the kernel, including some interesting design issues and how they were resolved, and point out some of the shortcomings of the FSS implementation. Also considered, are the needs of providing support for distributed file systems, and the techniques used to validate the new file system semantics. Finally, we give some performance comparisons between the original System V file system, the new file system and an original 4.3 BSD file system.

1. INTRODUCTION

Several new features have been added to System V with Release 3. One of these is the File System Switch (FSS), a feature that was designed to permit the UNIX operating system to support several different types of file systems simultaneously. A new file system type is implemented by writing a set of kernel-level file system management functions. Once implemented, the new type is incorporated into an existing system and accessed by programs that issue the standard UNIX system calls, such as *read(2)* and *write(2)*.

This paper describes the implementation of a file system type. The new file system type chosen for this exercise was a reimplementation of the 4.3BSD Fast File System (FFS).

The System V Release 3 file system has been reimplemented from Release 2 using the FSS. This new implementation is often referred to as **S5**. Throughout this paper, the 4.3BSD file system implementation is referred to as **B5**. As discussed in the following section, this name was chosen to reflect our primary goal, to maintain S5 file system semantics within Berkeley's fast file system.

This paper assumes a knowledge of the UNIX System V internals as discussed by [Bach86].

2. METHODOLOGY

A number of references, [Lankfo84] and [Peacoc88] point out that a S5 file system (S5FS) suffers from poor performance with respect to the 4.3BSD file system. Pre-System V UNIXes attempted to rectify this problem by increasing the maximum number of system buffers. It was hoped that this would reduce the disk interactions by increasing the likelihood that desired data would be retained in memory [Feder84]. However, using the System Activity Reporter (SAR) package we observed that when using a large number of buffers, it only compounds the problem when the system flushes those buffers back to

† UNIX is a trademark of Bell Laboratories.

disk during the "sync". Sar was used to monitor the outstanding requests for the block device, in particular, the queue lengths, average queue idle times and average service times, which takes into account seek time, rotational latency and transfer times.

The performance deterioration, large queue lengths and service times were the result of disk blocks being continually modified, causing the free list to become scrambled, resulting in files being fragmented and dispersed over the file system. This demands additional seek time and rotational latency to read the file. A combination of other factors also contribute to the large service times; a small block size and limited read ahead meant that the next sequential data was not within the same cylinder, forcing unnecessary seeks.

Considering these factors and that faster CPU's demand considerably more file system bandwidth, [Smith85] it was concluded the S5FS had become inadequate for the needs of our systems. Hence the relative merits of a BSD file system was considered, either as a replacement for, or as an alternative to the existing system. Keeping in mind the semantic differences, we decided the final system should comply with the existing S5FS semantics, so as to maintain upward compatibility for our existing application software base which dates back to System V Release 2. This meant the S5FS semantics must be faithfully implemented. For example, a 16 byte directory entry consisting of 14 character file names with 16 bit inode numbers, named pipes and file/record locking to name a few.

To increase the raw speed component of the file system performance without unnecessarily wasting disk space, the big block/fragment scheme [McKusi83] was seen as a prime requirement. The fragment allocation scheme is needed with large block sizes unless it's being used for a special file system. While the fragment allocation is needed with large block sizes, the scheme gains no performance, in fact it looses due to the CPU overhead required to manage the fragment scheme, copy fragments and extra disk i/o to maintain file system consistency [Elz88].

The final BSD file system feature considered was the cylinder group scheme. If contrasted in light of the layout policies, the result can be a major bonus for directory reader programs. The layout policies main objective is to keep inodes close to the directories which reference them and the inodes that relate to directory files being within the same block.

Our initial target was to provide the new file system as a user type file system rather than as one of the main file systems (ie "/" or "/usr"). In leaving the main file systems as S5, the existing bootstrap programs could remain unmodified and yet still bootstrap the new system. Hence, we had to initially support the two file systems within the same kernel.

3. IMPLEMENTATION

The focal point of this work was the System V Release 3 FSS. The FSS structure is a mechanism designed to permit the introduction of new file system implementations. It contains the functions that define the file system specific abstractions. Operations carried out below the FSS are with a file system implementation and only apply to that single file system. These operations are bound to a mount structure during mount time.

3.1. Kernel

It had been our hope that the hybrid file system could be implemented within the defined framework of the FSS interface. However, during development, a number of areas in the design of the FSS were discovered that simply prevented B5FS from being just another file system type.

In terms of ''coding'', much of the work related to fixing name space clashes. In particular, macro and parameter definitions specific to the S5FS are located along side file system independent definitions.

Since our main requirement was to maintain S5FS semantic compatibility, some code was taken from the existing S5FS implementation, for example all pathname translation and system call code was reused. However, a number of FSS abstractions still had to be written.

Some file system operations couldn't use the BSD code directly. These cases were due to the file system independent section of the FSS making incorrect assumptions about the file system dependent section. One example is the pathname translation routine, *namei*. The file system independent side is

designed to translate pathnames on a per component basis, consequently it assumes the file system dependent *namei* will do the same. This assumption destroyed any chance of making use of the 4.3BSD name and directory offset caches.

Memory management in Labtam's version of System V Release 3 is based on demand paging, which uses a page size of 1024 bytes for ns32032 based systems and 4096 bytes for ns32332 and Intel 80386 based systems. When a file system is mounted, the file system block size is stored in the mount structure for use by the page fault handler. This value is used when setting up the page tables for the executable file; the disk block descriptor is marked with the logical block number that contains the page. When a page fault occurs, the fault handler uses the logical block number to load the page from the disk file. However, a basic assumption in the paging system algorithms would break with the new file system. The assumption being that a file system block size is not larger than a page size, which is used in determining the number of blocks per page. This problem appears to be generic to System V, since the relevant code fragments originated from the 3B2 source.

An inode cache is not used in B5FS, since it is implemented within the file system independent side. The B5FS only makes use of a file system dependent inode freelist. The file system independent inode cache makes use of a freelist and set of hash queues, similar to the technique used by the buffer cache. On the last reference to an inode, (in *iput*) the inode remains on its hash queue with its *inumber* filled in until it is reused by *iget* or until *iget* allocates it from the freelist, in which case it is moved from its current hash queue to the new one.

The BSD IPC is via a method known as sockets, whereas System V has a number of implementations, including shared memory, semaphores and named pipes (also referred to as FIFO's). The only System V IPC mechanism which impacted on this port, is that of the named pipes, which traditionally uses the file system for data storage. The majority of S5 FIFO code was reused, while the remainder amounted to code fragments added to *rdwri* routine to handle the process blocking or unblocking and the manipulation of the FIFO read and write pointers. The FIFO implementation redefines the inode's indirect block addresses to be the data pointer's and counters, while the direct addresses hold the addresses of the data blocks making up the circular buffer.

From a system administrator's perspective, a subtle difference between the 4.3BSD and System V file systems is the later contains 'state'. The 'state' information stored in the superblock allows automatic checking of possibly damaged file systems during a system reboot after the system had crashed. When a file system is mounted, the *state* is set to a 'magic number' indicating the file system is active. Similarly, when unmounted the *state* is reset to a different 'magic number' representing the file system is okay.

One of the main obstacles with supporting multiple file system types, which is not addressed by the FSS, can be found in the buffer resource code. Since System V traditionally uses fixed sizes for the underlying buffers, the sizes must be at least the size of the largest file system request. For the 4.3BSD file system this is 8192 bytes.

However, uniformly large buffer sizes have a number of disadvantages. Firstly, the full capabilities of the buffer resource are not utilized. For example, if the above buffer size is used for S5FS or 1K fragment requests, the buffer will consist of 87.5% wasted memory. Secondly, a large buffer size requires that for a given size memory cache, we are only able to hold fewer number of buffers. This has an unpleasant performance side effect on the S5FS; benchmarks within Labtam proved S5FS performance deteriorates rapidly when file sizes larger than the number of buffers are used. Fewer buffers reduce the S5FS effectiveness, since some blocks are retained that hold only a small amount of useful data.

In order to use larger buffers without undue waste, a more memory efficient buffer scheme was required. In VAX‡ and Tahoe type BSD systems, this problem is handled using dynamic buffer sizes and by remapping memory pages between buffers. Unfortunately this is only suited to machines with a page size that is no larger than the smallest buffer size, which is not the case for the Labtam systems. Our new buffer resource manager accomplished this goal by pre-allocating a number of buffer classes, similar to the technique used by the *STREAMS* implementation, these classes being fragments of the

‡ VAX is a trademark of Digital Equipment Corporation.

maximum buffer size.

Buffer cache semantics ensure that a disk block be mapped by at most one buffer at any point in time. With the introduction of variable buffer sizes, the existing cache algorithms could no longer guarantee consistency. This problem was alleviated with more stringent tests in the cache searching routines, so that buffers are now checked if they overlap the requested block. Any overlapping buffers have to be invalidated and if they had been marked dirty, only be invalidated after being written back to disk.

One final problem remained with the existing buffer cache implementation. The individual routines provide an interface which assumed logical block requests, which are in reality S5FS block numbers. This was modified to provide an interface for both logical and physical block requests.

3.2. Utilities

As well as the kernel code, a number of user file system utilities had to be ported, **fsck(8)** and **mkfs(8)** being the main requirements, while **newfs(8)** and **tunefs(8)** were ported for completeness. Apart from the differences in file system semantics, the *fsck* changes also required the whole file system checking implementation be re-worked. Since System V provides automatic checking, the B5 fsck needed to "look and feel" like the existing S5 fsck. This basically meant the options had to be brought into line with the System V mould. On top of this, we added a heuristic version of fsck, who's sole job it is to determine the file system type and "fire-up" the correct version of fsck with the arguments consisting of the given "options" and name of the special device to be checked.

Some existing utilities (**df(1)**, **du(1)**, **ls(1)** and others) were found to be problem areas, since they had not been made totally independent of the underlying file system type when the FSS was implemented. The problems related to hard-coded knowledge of the file system superblock.

Porting the **newfs** utility proved an interesting exercise in its own right. The BSD version uses disk geometry and file system parameters stored in ascii databases (**diskpart(8)**). Rather than porting *disktab* we decided to change **newfs** to use our existing disk driver *ioctl(2)* interface which return a disk label structure, containing disk geometry information and partition layouts. The disk label is initially written after the drive is formatted. This is similar to the implementation adopted in the recent 4.3BSD Tahoe release [Bostic88].

3.3. Strategy

Since the basic operation of the kernel is to interpret an existing file system and multiplex user requests, which consists of allocating and freeing inodes and data blocks. The initial step required a disk partition be formatted with a new file system before any kernel operations could be attempted.

This initial file system had to be built using an existing S5 system. Creation of a new file system only requires access to the raw disk interface, which canonically is a randomly addressable array of sectors [Thomps78]. Hence, the user utilities to create a new file system (**mkfs**) and check the consistency of an existing file system (**fsck**) had to be ported.

To minimize sources of problems with the new system, the initial kernel used the existing buffer resource routines. However, the system buffer size (**SBUFSIZE**) had to be changed to 8k, so that the largest request could still be accommodated.

The first prototype kernel was then booted and each of the 26 FSS operations carefully debugged and tested. With the system operational, a few simple benchmarks quickly convinced us the project should be continued. Work then began on designing a new set of buffer resource routines.

In terms of time spent, the total design and development time for B5FS and the buffer cache code rewrite consumed approximately equal amounts. The first working prototype was demonstrated within 2 person months of starting this project.

3.4. The Hard Issues

Clearly the file system is a vital portion of any system, and any bugs in its implementation would have serious effects on the end user. For this reason it was decided the verification of the B5FS implementation include a number of independent processes. Two of the processes being a "Code Review" and a

"File System Profiler Suite".

The purpose of the code review was to identify and report on differences between the B5FS implementation and a set of standards; the System V Interface Definition (SVID), System V Release 3 Programmer's Reference Manual and the System V Release 3 implementation (ie source code). The review was to be done in complete isolation from those people who were involved in the original implementation. Although semantic differences are the most serious, we were also interested in any other differences. These differences being classified into three groups:

- differences between B5FS and S5 semantics that could be resolved by reference to SVID or System V documentation
- semantic differences that could not be resolved by the documentation, and
- internal differences between B5FS and S5 but not effecting semantics.

The second verification process was the design of a file system profiler. The purpose of which, was to detail the actual behaviour of a file system implementation rather than become a verification suite for the SVID (or any other standard). The profiler consisted of a small battery of programs designed to specifically exercise the error and boundary conditions of each file system component. If the profiler is used for verification purposes, the profiler output need only be redirected to a file and then compared against a similar file containing the known output from the standard in question.

4. SUPPORTING DISTRIBUTED SERVICES

The SUN NFS was originally ported to System V Release 2 as a joint effort by Lachman Associates Incorporated and The Instruction Set [Sandbe88]. A summary of the porting problems can be found in [Rosen86]. However, under System V Release 3 the majority of these problems are located in the file system independent section.

Much of the B5FS is unaffected by the NFS port. However, NFS requires an inode generation number be kept for each file on disk. This is used by the NFS server to detect accesses to files which no longer exist [Rosen86]. The structure of B5FS inodes (which are the same as 4.3BSD inodes) consist of 108 bytes packed into 128 bytes, thus allowing 4 bytes of the free space to hold the generation number. This also required a trivial change to the file system dependent **iput** routine to maintain and increment the generation number. The file system consistency check utility required a change to preserve this inode generation number when it must zero out the on disk inode.

As part of the FSS implementation, a new system call was added (*getdents(3)*) which returns directory information in a manner independent of the disk directory format. The B5 implementation used a copy of the S5 prototype along with a few minor changes. However, the original *s5getdents* contained a design flaw which assumed the return path would always be going to a user buffer. The RFS implementation used a "kludge" to get around the problem with a "hook" in the *copyout* routine to redirect it. NFS was unable to make use of this "hook" and required *b5getdents* be changed to use u.u_segflg (the segment flag) in deciding where to copy the data.

The Lachman NFS implementation was also changed to make better use of our new version of **bio.c**. This cut down the amount of data copying, since the original implementation used a temporary 8192 byte kernel buffer and then used this to scatter the data into the kernel buffer pool.

5. PERFORMANCE

No performance improvements to file systems are complete without benchmark results. The predominant access method for any UNIX file system is that of sequential reads and writes. For this reason, we needed some measure of throughput. The tests used the *fstime* portion of the **MUSBUS 5.2** benchmarking suite [McDone87]. A quiescent system was chosen to eliminate any other contention for either the CPU or the disk arm. Each test was run with 8 iterations and the results averaged.

Since our main goal was to approximate the Berkeley file system performance, we were particularly interested in comparing B5FS to a real 4.3BSD file system. The results for a Vax 11/750 with 4 MByte of memory, using a UDA controller with an RA81 drive are shown in table 1. As shown by the results in table 2, we see that the 4.3BSD results compare favourably with a Labtam equivalent system. The

NS32032 Labtam system, which benchmarks at approximately that of a VAX 11/750 in CPU speed, uses a Fujitsu Eagle drive on an Interphase 2190 controller.

	File Size (KBytes)			
	62	125	250	500
Write	121.7	153.9	173.8	183.9
Read	115.7	133.3	149.3	155.5
Copy	60.0	70.8	77.4	76.5

Table 1: 4.3BSD fstime benchmark results in Kbytes/sec

	File Size (KBytes)			
	62	125	250	500
Write	114.2	101.7	133.6	160.7
Read	193.8	199.3	210.1	217.4
Copy	66.4	70.9	85.3	95.8

Table 2: B5FS fstime benchmark results for an NS32032 system

While these results were encouraging, a quick look at the system with **sar** revealed everything was not quite as good as was first thought. The ratio of CPU usage to "waiting for I/O" had made a significant increase. This was to be expected, since the BSD file system requires more CPU than does System V.

For these reasons, it was decided the development be moved to a different CPU where the I/O time would become the predominant factor. All remaining tests used a Toshiba asynchronous SCSI drive on an NS32332 Labtam system. Since the motivation for this project was to improve the performance of the System V file system, tables 3 and 4 summarize the *fstime* measured throughput results for a 1K S5FS and 8K/1K B5FS respectively.

	File Size (KBytes)			
	62	125	250	500
Write	457.7	253.7	123.5	100.0
Read	933.3	682.6	105.2	105.1
Copy	307.9	53.2	49.8	46.0

Table 3: S5FS throughput benchmark results in Kbytes/sec

	File Size (KBytes)			
	62	125	250	500
Write	160.2	171.4	254.6	337.1
Read	376.9	362.5	383.2	371.1
Copy	82.7	98.4	113.9	122.8

Table 4: B5FS throughput benchmark results in Kbytes/sec

Several comments need to be made about the results.

The times measured for small files (in particular the write times), are the most sensitive to the disk block cache (ie the size of the disk block cache). The rates are measured against elapsed time, so there is room for variance. However, the absolute times are usually long enough to make this effect insignificant.

While B5FS exhibits a drop in bandwidth (for cached files) over its S5FS counterpart, it does provide an steady increase in bandwidth across the range of sizes. The overall effect on response time is hard to gauge, since no one benchmark will fully characterize the response times on a multiuser system.

However, the non-cached file sizes typify the disk activity associated with library archivers and database packages, while cached files would be the type of workload associated with compilers (which generate alot of small files).

The difference in cached file performance has been attributed to a number of factors, but mainly it is the cost of the complex block and fragment allocation schemes. This fact can be illustrated by considering that the figures collected above used a **stdio(3S)** buffer size (*BUFSIZ*) of only 1024 bytes. If this BUFSIZ is increased to that of a full block (8192 bytes) it minimizes any fragment allocation from being attempted. The results shown in table 5 were obtained using a modified BUFSIZ.

	File Size (KBytes)			
	62	125	250	500
Write	1400.0	497.0	719.0	491.1
Read	666.7	574.0	563.6	652.6
Copy	360.6	218.3	184.1	199.6

Table 5: B5FS throughput using a modified BUFSIZ

6. WHERE IT'S AT and GOING

B5FS in a production environment became a reality in July 1988, when packaged in a form suitable for binary distribution was installed on one of the central R&D machines. A few problems were found while installing the system, and promptly fixed, but since that time the system has been very stable. The new file system is currently running on a number of Labtam systems within the department, ranging from National Semiconductor 32332 to Intel 80386 CPU configurations.

A major problem of supporting different file systems, is they can have conflicting needs for disk buffers. Since buffer class requirements can be at each end of the spectrum. As part of the on going development, this issue will be resolved using a scheme which dynamically adjusts the number of buffers in each class, using a method which borrows buffers from other classes. This technique is not new, it is basically a variant of the scheme used by the buddy system [Koch87].

We also hope to profile the new kernel in the near future, in order to gain a better understanding of the performance. Also, there are plans to prototype a non-S5 root file system in the near future. This will require file system heuristics in the bootstrap program.

Finally, while doing the original 4.3BSD FFS port, all BSD system call and related functionality was not entirely removed. It is our aim to re-enable some of this functionality and provide a user compatibility library giving access to the functions (for example: symbolic links and the disk quota system).

7. CONCLUSION

We have presented the experiences of porting a new file system to the System V Release 3 FSS. The FSS interface is now better understood and we have shed some light on some of its weaknesses and suggested one possible solution. A hybrid system of the two file system types has been used to show the differences in performance a user can expect. Much work still needs to be done on B5 and the FSS.

ACKNOWLEDGEMENTS

Credit where credit is due; I would like to thank Michael Podhorodecki, Tim Roper and Russell McDonell for being there to discuss ideas with and yet still keeping sight of the original directions of the B5FS implementation. Early discussions with Robert Elz were very timely in putting us back on the right track. A big thank you must go to Steven Bodnar for the effort he put into the "Code Review" process which must have been frustrating not being able to discuss problems with the original implementors who were sitting next to him. Finally, I wish to thank Erick Delios for his help in obtaining some benchmarking results.

References

Bach86.
Maurice J. Bach, *The Design of the UNIX Operating System*, Prentice-hall, New Jersey, 1986.

Lankfo84.
Jeffrey P. Lankford, "UNIX System V and 4BSD Performance," *USENIX Summer Conference Proceedings*, Salt Lake City, 1984.

Peacoc88.
Dr. J. Kent Peacock, "The Counterpoint Fast File System," *USENIX Winter Conference Proceedings*, Dallas, Texas, February 1988.

Feder84.
J. Feder, "The Evolution of UNIX System Performance," *Bell Laboratories Technical Journal*, vol. 63, no. 8 Part 2, October 1984.

Smith85.
Alan Jay Smith, "Disk Cache - Miss Ratio Analysis and Design Considerations," *TOCS*, vol. 3, no. 3, ACM, August 1985.

McKusi83.
Marshall Kirk McKusick, William N. Joy, Samuel J. Leffer, and Robert S. Fabry, "A Fast File System for UNIX," in *Berkeley 4.3 BSD SMM*, University of California, Berkeley, CA 94720, July 27, 1983.

Elz88.Robert Elz, *Private communication*, Feb. 4, 1988.

Bostic88.
Keith Bostic, "V1.61 (4.3BSD-tahoe release)," *USENET, Newsgroup comp.bugs.4bsd.ucb-fixes*, p. 3, University of California at Berkeley, Jun. 15, 1988. Article <8806152300.AA28766@okeeffe.Berkeley.EDU>

Thomps78.
K. Thompson, "UNIX Implementation," *Bell System Technical Journal*, vol. 57, no. 6, Part 2, July-August 1978.

Sandbe88.
Russel Sandberg, "The Sun Network File System: Design, Implementation & Experience," *AUUGN*, vol. 8, no. 5, Oct 1988.

Rosen86.
Mordecai B. Rosen and Michael J. Wilde, "NFS Portability," *USENIX Summer Conference Proceedings*, ATLANTA, Georgia, 1986.

McDone87.
Ken J. McDonell, "Taking Performance Evaluation Out Of The "Stone" Age," *USENIX Summer Conference Proceedings*, Phoenix, Arizona, June 1987.

Koch87.
Philip D. L. Koch, "Disk File Allocation Based on the Buddy System," *TOCS*, vol. 5, no. 4, ACM, Nov 1987.

OFF THE NET

Editors Note - I am sure some of you will find parts of this article offensive but other parts interesting. Please inform the author David Horsfall and myself whether you feel whether this is worth continuing in its current form, changed or not all - John Carey, AUUGN Editor

Hello, good evening and welcome to the Uri Geller Institute of Advanced Spoon-Bending! Oops - wrong sketch - I'll come in again...

Hello there, and welcome to a new section of AUUGN, tentatively titled "Off the Net". The story behind it is that I was approached at the recent AUUG conference with a view to contributing articles gleaned from USENET, seeing as I'm paid to read it. Unfortunately, I was not able to make the next issue, due to sudden commitments (like, being told my services were required in Melbourne etc etc). Anyway, here it is.

It is basically excerpts from various newsgroups, which I thought our readers may be interested in. I should point out that I have a weird sense of humour, and things that I find funny, others may not. If you don't like it - tough. See figure 1. I'm sure that John Carey will appreciate having many more people contributing articles, so if you don't like mine, then write yer own.

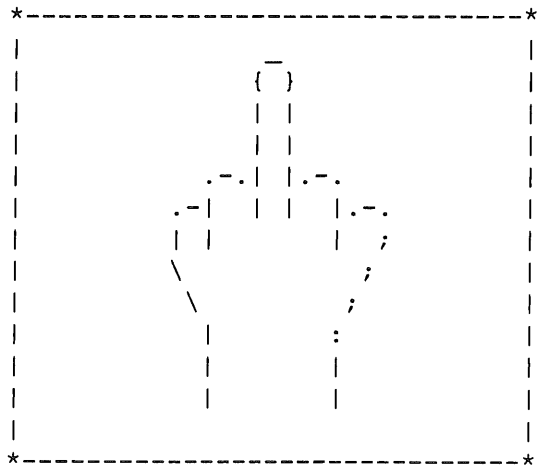


Figure 1.

Now, to business. Here are my impressions of the recent AUUG dinner. No - I'm not too proud to include my own postings, and I'm sure John will appreciate hearing from other attendees:

From: dave@stcns3.stc.oz (Dave Horsfall)
Newsgroups: aus.auug
Subject: AUUG dinners

The AUUG committee are going to have to do some hard thinking in the years to come, otherwise we could run out of places to eat! Having attended almost all AUUG meetings since the 70's, it is my considered opinion that they are getting grosser. If this keeps up, we could find ourselves *persona non grata* in the better eating places.

Perhaps the balloon stunts were inspired by the STC-sponsored dinner last year, but they certainly seem to have started a trend.

For those who missed out, the highlights (as I hazily recall) were:

The "Pyramid Technology" slide that shone back-to-front (not accidental) for about 1/4 hour, and the half-hour it took 3 Pyr. people to right it;

The (now traditional) graceful flight of balloons carrying assorted payloads up to the ceiling, and the shooting down of same with satay skewers fired from (I think) blowpipes made from ballpoint shells. Some of the skewers are still embedded in the ceiling tiles... A special mention must go to the catapult cunningly constructed from a table place holder;

The design, construction, test flight and post-flight modification of sundry paper aeroplanes;

And the now-infamous refacing of the Pyramid poster converted (with the aid of several strategically-placed napkins) from "Know Unix Think Pyramid" to ".no. Unix ..in. Pyramid". We couldn't reach the other one - it was too high up.

I shudder to think about the next dinner...

Here's a little something I found in comp.risks, and reposted to aus.jokes:

From: dave@stcns3.stc.oz (Dave Horsfall)
Newsgroups: aus.jokes
Subject: Dial PIG for Police

Found this in comp.risks (7.48) and couldn't resist posting it here:

| The Chicago Police Department & Fire Department can be reached through
| the main centrex number for the City of Chicago Offices: 312 - PIG -
| 4000. To reach individual police officers, etc, just dial PIG and the
| desired 4 digit extension.

Now, wasn't that interesting? Let's get back to the weird side of things again, with this one (mine, of course) from aus.auug:

From: dave@stcns3.stc.oz (Dave Horsfall)
Newsgroups: aus.auug
Subject: The network at AUUGM

Just out of curiosity... How many attendees noticed there was an Ethernet connection between some of the machines at AUUGM? I believe it was Blue Sky's idea. Not quite in the same league as the Connect-a-thon (or whatever it's called at USENET), but still, it was marvellous to see the cooperation between competitors, n'est-ce pas? Reminds me, must check the brakes on my car again... Those pyramids, you know...

Hmmm... I am reliably informed (thanks Kev!) that IBM treats TCP/IP in much the same way it treats SNA. You know, a definite master-slave relationship? It blasts out whenever it feels like it... Just as well they weren't connected, I guess.

OK - idea time! Let's get this rolling for next year! All vendors are invited to take part in... hmmm, what do we call it? Oz-connect? Unix-connect? Connection-between-eunuchs-oops? With a little bit of luck (we won't be home - sorry) we'll have a few NFS demo's as well. Oh - all right - you RFS people can join in too :-)

And hopefully, we can educate the people wielding (as opposed to people-wielding - you know, man-eating shark vs. man eating shark... Ahhh... skip it) vacuum-cleaners to not come quite so close to the co-ax. Then we won't see the forlorn study of a certain personage, clutching a BNC connector in one mitt, and a somewhat shredded co-ax cable in the other, wondering how to insert one into t'other... "Use pliers!" Christo said... I think I should've stayed to watch!

[OK, Mr Carey. Is that enough now? Yeah, I thought it was. Hey, there's an idea! How about _posting_ AUUGN in aus.auug? Hmmm, consider the savings - no printing. Hmmm, consider the losses - no advertising. Oh well - just an idea. Hey! (yeah - me again). Has anyone on the AUUG committee considered this? It's been a while since I was last on, so I've lost touch. One more problem - I might go recursive! (If that's lost on various readers, I mean human readers, not news-readers (skip it), as of now you'll start seeing the highlights of USENET appearing in AUUGN, by explicit invitation of certain personages, none other than Greig (sic) Rose and John Carey! Hope youse can all stand my weird sense of humour... Blame them, not me! Perhaps a disclaimer would be in order? "AUUG [officials] do hereby disclaim and disavow all knowledge and actions of the rampant Python-loving AI-program about to follow...". Well, you asked for me, you got me...) Brackets matched? Yep...]

(Well, I guess all respected journals have to have their token nerd in there somewhere! You know, Norman Kemp, Gareth Powell, Bartholomew Santamaria...)

[I fully expect the above to have been excised from the printing stage, to preserve the pristine cleanliness of AUUGN...]

{What? It's still there? Blimey...}

By now, most people have read the various beat-ups in the media about the Internet worm, where hardly a sentence is correct. Here's an example of how the non-technical media Gets Things Wrong:

From: rick@SEISMO.CSS.GOV (Rick Adams)
Newsgroups: comp.protocols.tcp-ip
Subject: Re: It's in print, so it must be true...

You're leaving out stupidity and arrogance.

I once talked to a "reporter" for Computer Decisions about the TCP/IP fiber optic network we were running (this was 1983 or 84 and there wasn't much fiber around).

Among other things, I said that we were running Sun 3/160 workstations.

I also asked to see a copy of his article before publication so I could correct any technical errors he might have made.

The arrogant twit gave me a big speech about how he was a "professional" and did not tolerate "censorship" of his "work".

Well, it seems that Mr. Professional wasn't very smart and had never heard of Sun 3/160s. So he "corrected" me in the article and when it was published we ended up running a network of IBM 360s.

(I got calls from people trying to sell me IBM services for over a year after that article).

Moral: Never rely on the information in a trade magazine for any reason. You MUST verify it yourself if you are doing something that depends on it.

Those followers of aus.flame might like to see some real high-brow flaming. This lot appeared in comp.protocols.tcp-ip:

From: PADLIPSKY@A.ISI.EDU (Michael Padlipsky)
Newsgroups: comp.protocols.tcp-ip
Subject: Re: Does TCP/IP "comform" to ISO/OSI?

Marshall--

"I would certainly have atomized you by now!" you say? How scholarly. Even if Robert's Rules did apply here, I'd have a final point of personal privilege coming, so I'll ignore Merton Campbell Crockett, smile most aprreciatively at Rich Brennan, and go along with Bill Northlich. Indeed, just to demonstrate that threats of nuclear retaliation don't intimidate me, I must go one more round, even though I do believe that if you'd read my last msg more carefully you'd have felt like the atomizee rather than the atomizer. I, after all, wasn't the one who said "OSI (levels 3&4) is still quite silly at times" and "some of the actual parts are pretty lousy" while complaining about those who weren't "trying to appreciate the entire picture"; I merely called you on it.

Phil Karn's msg--which I presume is what made you "sad", not Phil himself, even making due allowances for your rhetorical practices (which do take me back to the old days when I was teaching Freshman Comp. and the book called them Propaganda Techniques)--turned out to be no surprize when I saw it. My reading of it is that he deplores the hypocrisy of those ISORMites who tout The RM in public while the protocol designers flout it in private, as I deplore it too, of course. Why else would his climactic paragraph say

"But does ISO ever revise its preconceived ideas and admit its mistakes? Of course not. It just covers them over with ever more paper and ad[vertizing] hype[rbole]."?

(Note, by the way, that it's a PT to say I "admit" I hadn't read it when I sent my last msg. Actually, I SAID I hadn't read it. Nor did I need to, since I could deal perfectly well with your response on its own demerits, given that what you'd said about there being problems with some of the constituents but "TCP extremists" [another PT] didn't understand the whole led directly to my apparently too effective restatement. [In case you're wondering, yes, sarcasm is viewed unfavorably by some observers, but I don't account it an out-and-out Propaganda Technique like, say, the mud-slinging "extremists" bit. But, then, it's been a long time since I studied such things and it would only use up too much of my norepinephrine reserves if I worried about them any more.] And while I'm parenthesizing, I should also mention that I have tried to be quite scrupulous about not accusing you of using questionable linguistic ploys _consciously_; I merely note that for whatever reason your msg to me was rife with them, and, indeed, rather hope it wasn't being done consciously.)

I don't see Phil saying We've Got the Only Answer, even if he did omit the implicit "joke" symbol after the billboard line, and I'm certain I haven't, despite your infelicitous hammer metaphor. What _I'm_ saying is that it's intellectually dishonest to claim that The RM Is The Only Answer with one side of the mouth and ignore its stated precepts with the other side of the mouth. Yet that's what the ISORMites are doing. (I still cherish hopes that you yourself are an ISORMist, by way, even if I've never gotten a [presumably jocular] death threat from an ISORMist--or even an ISORMite, come to think of it--before.)

Whatever useful stuff is being attempted at Layer [sic] 5-7 by the ISORMists, it seems quite clear to me that it's being done despite, not because of, the RM, with it's stated adherance to rigidly hierarchical, 5<->6<->7 layering/strait-jacketing. Nor, as best I can determine, are the appeals to "CASE" and "ACSE" being made just because the particular, current Session and Presentation Standards are tainted apples at best; rather, it seems to be because you _do_ need certain Session and Presentation functionality "right now" in certain Application settings. (With my usual luck, the one _Connexions_ I can't find is the one the Index tells me has your piece on "Applications in an OSI Framework" in it, so I can't offer you your own words, but I do recall thinking that it was still more evidence for my years-old "Layer [sic] 5-7" complaint when I read it.)

There is, I submit, a legitimate technical interest in whether there are _correctable_ flaws inherent in The RM, no matter who has or hasn't embraced The RM as a matter of organizational/governmental policy. Do you suppose we can discuss without propaganda the question of whether or not the L5-6-7 separation _ought_ to be viewed as mandatory, both ways (i.e., both into the Host from the net and on up, and out of the Host from the user/Application PI and on down), in which I for one have a long-abiding technical interest, and leave emotionalism for some other time? Or is my asking that just opening the door for a non-nuclear threat like (shudder) immersion in the sauce of rotten apples?

(Jocularly intended sarcasm aside, I'm really not convinced it's worth the norepinephrine depletion to me if we can't raise the level of debate a few notches. Wanna try a round or two with above-the-belt blows only, if only for the sake of novelty? Or shall we let it go--each, I daresay, quite convinced that he has "won"?)

nonetheless, cheers, map

From: mrose@TWG.COM (Marshall Rose)
Newsgroups: comp.protocols.tcp-ip
Subject: Re: Does TCP/IP "comform" to ISO/OSI?

Mike --

I am willing to give up my humorous metaphors and remarks regarding nuking you, if you are willing to talk in plain simple english in each and every message you send in the future. Frankly, I get a headache trying to read your messages! Needless to say, this does not make me particularly receptive to your arguments, which may, for all I know, might be quite reasonable and sane. Unfortunately on the average I refer to a dictionary 3 times in the course of an hour when reading just ONE of your messages.

The problem, as far as I can tell, is that you arguing against opponents who do not read this list. As surprising as it may seem, I am a moderate, and am viewed as somewhat heretical by the OSI community by even suggesting that the lessons learned in the Internet might be applicable. My favorite line which came up in a conversation back in April at an OSI tutorial:

> "The Internet was an interesting experiment, but X.25 is the only real networking solution".

Yes, I thought it was funny too. Even more funny considering the person who gave the tutorial deeply believed this. I was tempted to refer everyone to your book, particularly later on when the speaker said one didn't need something like TCP over a LAN, but alas, I could not remember the full title of your tome (too many big words, you know), so I did not speak up.

Now, at this point I guess I should respond, point by point to your message. But frankly, who cares? I don't understand half the arguments you make, due to your use of excessively obscure english. I guess I can not just see your "larger picture". This isn't a personal attack, but I hope you take it personally(!) and clean-up your writing style so more of us can understand what you are trying to say.

With regards to who's atomized who: I have received (privately) one message congratulating me on stomping you with my last message. Although this is not conclusive proof, it is good for my ego.

Your turn (and please try to avoid using words which aren't found in my Webster's abridged dictionary, thanks).

/mtr

From: PADLIPSKY@A.ISI.EDU (Michael Padlipsky)
Newsgroups: comp.protocols.tcp-ip
Subject: Re: Does TCP/IP "comform" to ISO/OSI?

Marshall--

Gee, since you put it that way I guess I shouldn't even "invoke cloture"; how about I'll just call a halt?

'bye, map

P.S. If enough onlookers send private msgs requesting it I'll type in what my wrap-up msg would have been and send it back to them, also in private.

What do you want on your gravestone? Here are some possible ideas:

From: sreeni@hpisoa2.HP.COM (K. Sreenivasan)
Newsgroups: comp.unix.wizards
Subject: UNIX Remembered

It is known that Max Planck (the great physicist) chose to display the value of Planck constant on his grave stone. Max Born followed the same example and chose the commutative operator (for which he was awarded the Nobel Prize) for his grave stone.

Folklore has it that the defendants in the IBM JCL paternity suit are toying with the idea of using "**" and "&&" or "//" on grave stones.

Unix gurus are too young to think about grave stones but I am sure "\", "++". "?". "~", "^" are all spoken for. I have it on good authority that MBA's have usurped "$".

And now, back to aus.flame again. Tomasso is one of the few erudite flamers on the net:

From: osborn@nswitgould.OZ (Tom Osborn)
Newsgroups: aus.flame
Subject: Heimie von Stuch

During the early 1950's, the Dr Heimie von Stuch from Westfallen made a number of startling predictions which has recently been verified. Heimie was far in advance of his times. Indeed, on other occasions, he anticipated the Presidency of Ronald Reagan, the ascendency of IBM, ABBA, and, the rise and fall of digital watches. However, Heimie's greatest prediction, which has just been witnessed, concerned the sociological factors which influence and come to dominate a system of public or semi-

public communication which contains the three properties of "Lauczismut", absence of responsibility and high information volume without volume control.

["Lauczismut" is an ancient Czech term which has no exact translation into any other language. Its essential meaning is the kind of _power_ one has when one is placed within a hierarchy, where those above push incomprehensibly (even violently) hard for performance _and_ peace, whilst those below complain naively and incessantly that conditions have to be improved, and 'someone has to pay for our suffering'. The power consists of applying the push from above on those below and that from below on those above, without appearing to get involved. Delusional behaviour typically results after 6-18 months, in a belief in indispensibility and wisdom. Typically too, the individual can be observed to be surrounded by chaos from the rest of the hierarchy who are all kept very much in the dark - who all believe that things just can't be done for them, for reasons unspecified.]

Heimie's Predictions:

(1) Communications between Lauczismutic individuals from different organisations commences at a high level of abuse, with minimal information content.

[Here the term communication refers to globally witnessed two-way interchanges, not merely information posting and query answering.]

This has been confirmed on ACSnet in the persons of: Purdue, Maltby, Horsfall, Boyd, melbCAE (inclusive), Gibson, Callum, Wesley and Jeremy. (Sorry if anyone was left out, chappies).

(2) Lauczismutic opportunity sees the individuals spending more and more of their time engaged in the essentially useless communication. The establishment of a loose pecking order and in_groups commences, based on a working definition of 'eloquence' which emerges from vague loyalties to the intrinsic ethoses in each individual's heirarchy.

['Eloquence' styles become trademarks of individuals with little else to distract them - sure, they may work hard, but it doesn't distract them from developing eloquence - they use it back _within_ their organisation/hierarchy.]

This too has been confirmed on the ACS/etc network. Witness the three-way partition into 'industry', 'unis' and 'others'. The VMS vs UNIX* hype-up, the 'machine loyalty' fracas, and interstate, inter-transport-modal, and even inter-prejudice abusive interchanges.

(3) An organised meeting of an in_group of the individuals established the principle of 'in_group loyalty'. The communication system then degenerates further into intergroup rivalry.

[Like styles attract like]

For example, note the different attitudes to Horsfall since AUUGM. Also, the UNIX/VMS skirmish is getting more incisive, the drongoes

at melbCAE are copping heaps, Queensland and Vic are lampooned in generality, and the bosses are none the wiser of what's going on.

(4) One of the groups - the most established - will push, half-mokingly at first, for a formalisation of participation in communication.

Yes, this has hit. Next we see it formalised in the interests of control, of not wasting bandwidth/time/degrading_the_network.

BALLS!

Heimie was no mere critic or analyst of frightful disasters, he also advocated remedial measures. Academics would usually advocate peer review. In this case, it is not warranted, as Lauczismutic individuals are involved. No hint of responsibility is implied - in fact, their loyalty is to passing responsibility on. The primary remedy is to associate the less informative/persuasive postings with a cost to the poster, not the recipients. Secondly, the emergent eloquence _itself_ should be brought under question. If a piece of writing is not persuasive, then it should receive a 'fail' and be resubmitted. Finally, in_groups should be undermined at any cost, for example, by pointing out that the disclaimer at the end of a message does not imply that the boss is not interested.

Tomasso.

Refs

Von Stuch H., 1952, 'The rise and fall of power in open broadcast systems', Ann. Dr. Jn. Cyb., Vol 18, pp43-82.

Von Stuch H., 1953, 'Lauczismutic individuals and groups in public', J Acad. Psy. Czech., Vol 4, pp 2304-2319. (Translations may be obtained from Unisys Aust, on presentation of a very large check or Paul Keatings home phone no.)

Eco U. and Skinner B. F., 1985, 'Heimie von Stuch revisited: the demise of computer network news in the hands of system administrators and other abnormal types', Doklady nr Kbl (in Russian), 32, pp19-80.

Tanake U., Miwagana O. and Smythe-Johannesson D. V., 'Towards an eloquence of the Lauczismuds - they can't all be sacked, can they?', TR number 12, ICOT (to appear), also available from IBM Australia under the title: [sic] systems adminsitrators at UNIX* sights'. (Call and ask for Finny).

Disclaimer: This posting is news to them too.

We're still in aus.flame, and here are some beautiful graphic images:

From: smac@munmurra.mu.oz (Stuart McCormack)
Newsgroups: aus.flame
Subject: Re: Road Hogs

in article <360@chiton.cad.jmrc.eecs.unsw.oz>, skea@cad.jmrc.eecs.unsw.oz (Alan Skea
>
> In article <688@munmurra.mu.oz> sl@munmurra.mu.oz (Steven Lynch) writes:
>> [... lots of perceptive, intelligent comments ...]
>
> I agree whole-heartedly. In particular the point about reducing everybody
> to the lowest common denominator gives me the sh*ts.

Sometimes it's a case of reducing cars to the lowest common denominator.
How fast can a sh*t-heap get around this corner at night in the wet? 35K?
OK. Those yellow signs are precautionary.

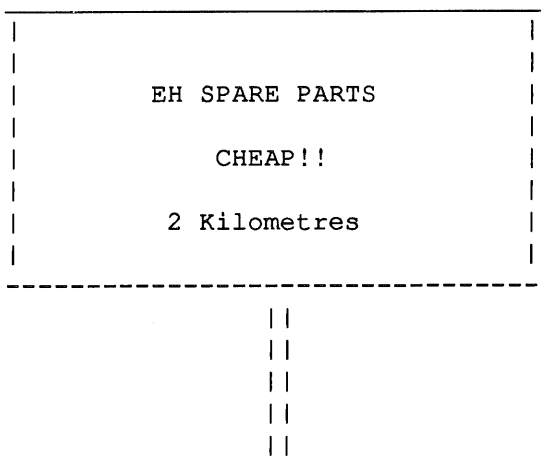
We could get rid of the lowest common denominator by making signs like this..

```
+-------------------------------------------------+
|                                                 |
|                   WARNING                       |
|                 CORNER AHEAD                    |
|                                                 |
|  Current Model European Sports Cars     95kph   |
|  Current Model Japanese Sports Cars     85kph   |
|  Current Model Australian Sports Cars   60kph   |
|  Current Model American Sports Cars     25kph   |
|  15" ( WankMaster ) Tyres               12kph   |
|  Volvos, make 3 point turn and reverse          |
|          around corner                          |
|  Lambourginis - Drop Dead You Capitalist Pig    |
|  Current Sedan/Saloon/Station Wagon     45kph   |
|  For every year on road please subtract  1kph   |
|  Beat-up EH sh*theap with four different        |
|       tyres ( All bald )               160kph   |
|                                                 |
+-------------------------------------------------+
```

But then of course you'd need...

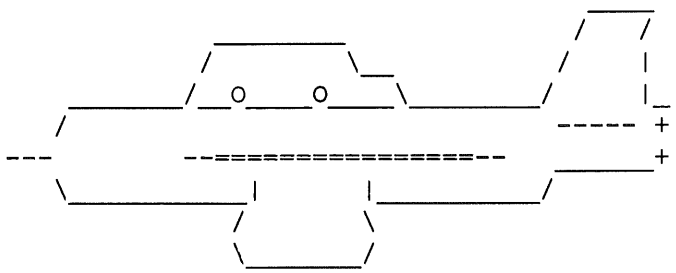
```
+---------------------------------+
|                                 |
|            WARNING              |
|                                 |
|          SIGN AHEAD             |
|                                 |
|       All But EH Drivers        |
|       Please Slow Down To       |
|                                 |
|             10kph               |
|                                 |
+---------------------------------+
                 ||
                 ||
                 ||
                 ||
```

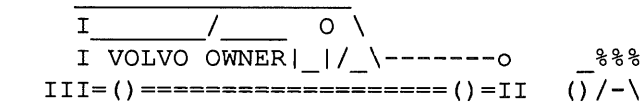
And before that....



For some time, a war has been raging in aus.flame between bicyclists and motorists - especially Volvo drivers. We'll leave it to John Mackin to make the final word:

From: john@basser.oz (John Mackin)
Newsgroups: aus.flame
Subject: Volvo meets something that doesn't _need_ turbocharging





''Strike, six three charlie has the Volvo visual, commencing strafing run.''

''Roger six three charlie, on completion climb as required to release bombs on a return from the southwest, avoiding the cyclist.''

This one from rec.humor.funny (long may Brad moderate it) is worth printing:

From: mosurm@mntgfx.UUCP (Mosur Mohan)
Newsgroups: rec.humor.funny
Subject: Tired and complaining

The following is an actual letter of complaint which I grabbed off the net many years ago (when it used to be called net.jokes, if you can remember that long ago!) Unfortunately, I don't have the original source anymore. Note the date sent and the prices quoted.
****

Atlanta, Georgia
September 13, 1970

Director
Billing Department
Shell Oil Company
P.O. Box XXXX
Tulsa, Oklahoma 74102

Dear Sir:

I have been a regular customer of the Shell Oil Company for several years now, and spend approximately $40.00 per month on Shell products. Until recently, I have been completely satisfied with the quality of Shell products and with the service of Shell employees.

Included in my most recent statement from your department was a bill for $12.00 for a tire which I purchased at the Lowell I. Reels Shell station in McAdenville, N.C. I stopped at this station for gasoline and to have a timing malfunction corrected. The gasoline cost $5.15; eight new plugs cost $9.36; labor on the points $2.50. All well and good.

Earlier in the day I had a flat tire, which the attendant at the Lowell I. Reels station informed me that he was unable to fix. He suggested that I purchase a tire from him in order that I have a spare for the remainder of my journey to Atlanta. I told him that I preferred to buy tires from home station in Atlanta, but he continued to stress the risk of driving without a spare. My reluctance to trade with an unknown dealer, even a Shell dealer, did not discourage him and finally, as I was leaving, he said that out of concern for my safety (my spare was not new) and because I had made a substantial expenditure at his station, he would make me a special deal. He produced a tire ("Hits a good one. Still has the tits on it. See them tits. Hits a twenty dollar tar.") which I purchased for twelve dollars and which he installed on the front left side for sixty-five cents. Fifty miles further down the highway, I had a blowout.

Not a puncture which brought a slow, flapping flat, nor a polite ladyfinger firecracker rubberburpple rupture (pop); but a howitzer blowout, which reared the the hood of my car up into my face, a blowout, sir, which tore a flap of rubber from this "tire" large enough to make soles for both sandals of a medium sized hippie. In a twinkling, then, I was driving down Interstate 85 at sixty miles per hour on three tires and one rim with rubber clinging to it in desparate shreds and patches, an instrument with a bent, revolving, steel-then-rubber-then-steel rim, whose sound can be approximated by the simultaneous placing of a handful of gravel and a young duck into

a Waring Blender.

The word "careen" does no justice whatever to the movement that the car then performed. According to the highway patrolman's report, the driver in the adjoining lane, the left hand-- who, incidentally, was attempting to pass me at the time-- ejaculated adrenalin all over the ceiling of his car. My own passengers were fused into a featureless quiver in the key of "G" in the back seat of my car. The rim was bent; the tits were gone; and you can f--k yourself with a cream cheese dildo if you entertain for one moment the delusion that I intend to pay the twelve dollars.

Sincerely yours,
/s/ T.B.T.

And now for a parody of the gross misinformation in the popular press:

From: daveb@gonzo.UUCP (Dave Brower)
Newsgroups: news.sysadmin,comp.protocols.tcp-ip
Subject: SF columnist on the Worm, suggests new terminology.

Jon Carrol's Wednesday column in the SF Chronicle is an interesting example of the level of "popular journalism". It gyrates wildly between keen insight and gross misinterpretation, arriving at something near a balanced perspective.

It does suggests a metaphor for these events, in the spirit of the tcp-ip swamp: The worm "frogged" the Internet. We could refer to painfully visible spelunking as "frogging".

-dB

"Random Thoughts About the Worm"

As faithful followers of the media and the hypermedia are already aware, a computer "worm" infected numerous extremely large computer systems around the country last week. The worm's name was ":sed
it did eat up an enormous amount of column inches and TV air time.

In conversations with people who know more about computers and worms that I do (approximately half the known world), the following facts and/or plausible opinions have emerged.

1. The difference between a worm and a virus is similar to the difference between a common cold and brain cancer. A worm does not eat up or change existing cells; it just fills up empty information cavities with disgusting gunk.

2. Interestingly, there is no direct cybernetic evidence that Robert Morse Jr. is the worm-master. Computer technology being what it is, the "telltale files" could have been planted by anyone, including Barry Manilow. I make no accusations here; I merely note the possibility.

3. What allowed the worm to enter the systsems was a programming structure called a "trapdoor," which is a device built into a computer system that allows the very smartest people to move more quickly through the system than the ordinary user. Such holes exists in virtually every system, including the ones that allow you to withdraw $100 on Saturday and the ones that launch large pizzas at Leningrad.

4. WHATEVER ITS CAUSES, the Worm Event was essentially a benign occurance. Given the sophistication of the worm program, it is easily possible that the worm master could have introduced a curious anomaly into, say FedWire, which is responsible for transferring $500 billion (think: half-a-trill) around the planet each and every day.

5. It is an interesting question, not yet decided, whether members of the affected networks (like the University of California) could sue UNIX (from whom they bought the program with the trapdoor that let the worm begin to burrow) for lost revenues. Although there is no case law on the matter, there is also no reason why not.

6. It is necessary to make a distinction beween "hackers" (who are simply people who understand computers very well and are unwilling to accept authoritarian definitions of what they should do with their knowledge) and "crackers," who invade extant computer system with malign intent. It seems probable that the worm-master was a hacker. To blame him (or, possibly her) for the trapdoor is like blaming Cassandra for the fall of Troy.

7. THE MOST IMPORTANT thing to understand is that computer programming is an extremeley intense art form. It is also a scientific discipline, but its addictive fascination lies in its creative component. Creating a truly innovative program is (for the artist) exactly like painting the roof of the Sistine Chapel.

Therefore, the hole that the worm went through may bee seen (in aesthetic terms) as a big blank piece between the figure of Adam and the figure of God on said ceiling. The hacker/artist saw the hole and said, "I'll bet the original artist meant for the two hands to be touching there. How stupid of him to have forgotten to put that in."

So the angry artist, upset at the unlovely disunity, decides to draw a big frog between God and Adam. "This," he says to himself, "will probably call attention to the problem."

- Jon Carrol

Back to aus.flame again - this time, it's the infamous spelling wars:

From: johnd@physiol.su.oz (John Dodson)
Newsgroups: aus.flame
Subject: Know What I Hate.

I HATE poeple who tell me...

"i before e except after c"

like wot it should'a done so this is just for them...

****

I am in _obeisance_ to the editors of the Oxford English Dictionary who may have exceptions to _their_ rule _albeit_ one that is almost _atheist_ .

I know that I am raving but would like to make _eirenicon_ with them even allowing for _their_ _absenteeism_ , _their_ _weird_ rules only _weigh_ on my mind and remind me of _apartheid_ so much that I need _caffeine_ , _codeine_ and _protein_ in order to maintain my _weight_ .

Being a simple man of moderate _height_ I occasionally go _beige_ at the _sleight_ of hand and _singeing_ comments of some _foreign_ _sovereign_ whose _counterfeit_ _reign_ sends _heinous_ chemicals pulsing through my _veins_ .

Even my _neighbour_ , who rides a _sleigh_ , is a tremendous _cueist_ and sometimes writes in _cuneiform_ , when _seeing_ such things dons a _veil_ and _deigns_ to _forfeit_ his _leisure_ .

In fact I get a _kaleidoscope_ of _seismic_ _seizures_ with a _surfeit_ of _poltergeist_ which causes me to _deify_ a _reindeer_ .

Even my _heifer_ _feigned_ and _feinted_ till the _eight_ pieces of _freight_ broke his _reins_ and fell into the _weir_ , the sound he made was a _neigh_ untill he _seized_ upon some _theism_ and _heighed_ his _heir_ to keep a good _rein_ upon himself.

Herein lies a story which I don't understand _either_ !

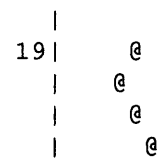
But _sobeit_ ...

johnd@physiol.su.oz

The content of aus.religion varies from time to time. Here, Chris Stuart has actually taken the trouble to do a little traffic analysis, and makes some interesting conclusions:

From: cjs@bruce.oz (Chris Stuart)
Newsgroups: aus.religion
Subject: Re: A plea to keep it short.

```
  |                      @ Just for fun. (I won't do it again.)
  |                      @
  |                     @  The graph shows the number of postings to
18|                  @      aus.religion over time. For each posting
  |              @          still on line at this site, there is a bell
  |         @               curve with standard deviation of 12 hours
```



and height of 2 character places, centred at the time (GMT) of the Date: in the header. These curves are superimposed to give the graph. Plots are at 6 hour intervals, and the numbers give new dates at midnight.

As is accepted practice with statistics, these parameters are carefully chosen to support a conclusion already reached by other means. To wit:

aus.religion is quite an active group. It was revitalized by discussions on Buddhism and Death (dating from the peak at the top). Some Christian topics are reappearing as the group recovers from a slump at around the 26th of November.

But why the slump? Lo! there was a complaint from our own well beloved Rev. Dr. Phil about long postings at Nov 25 7am GMT. IMMEDIATELY after this the graph plunges to its low point! As further evidence of the high regard for Phil in this newsdroop, there have been a spate of apologies from some who have begun to post again with more than trivial content.

Ironically, the original complaint was that the group was dying. It will die if people get too sensitive to post what they feel. The recovery 48 hours later should not be regarded with overmuch complacency, as it goes against my conclusions :-) uhh, I mean because a review of recent content shows a low signal to noise ratio.

It is especially ironic that the previously flourishing discussions on Buddhism and Death seem to have suffered the most, as we turn again to Christian topics. Don't get me wrong. I *like* the Christian topics; Drew's latest is excellent. But do also keep up the variety by posting articles on *your* areas of interest. Long ones, if you have a lot to say. With lots of carefully chosen quotes from other articles, if you want to help us follow a discussion. And short ones, sharp and to the point, if that is appropriate.

More, more, I'm still not satisfied...

Chris Stuart (who is actually rather a fan of the Rev Dr)

Still with me? Good. Here is some food for thought, for you poor benighted souls who are faced with the problem of time estimates.

From: john@basser.oz (John Mackin)
Newsgroups: comp.software-eng
Subject: Re: ethics

Well, back in the Bad Old Days when I used to work in the commercial world (as a programmer), I would very commonly get asked that dreaded question, 'How long will it take?' I knew there was no way I could answer that question with any accuracy, so I developed a method that kept me happy.

For those of you faced with this problem I offer the method. I call it 'Estimating by Threes'.

If you think the task will take under ten minutes, tell them three days.

If you think the task will take more than ten minutes but less than an hour, tell them three weeks.

If you think the task will take more than an hour but less than a day, tell them three months.

Otherwise, tell them three years.

You'll be amazed at how often you manage to predict the exact time it will in fact take.

From: whbr@cgchd6.uucp (Hellmuth Broda)
Newsgroups: comp.software-eng
Subject: Re: ethics

In article <941@vsi.COM> friedl@vsi.COM (Stephen J. Friedl) writes:
|In article <1616@basser.oz>, john@basser.oz (John Mackin) writes:
|< For those of you faced with this problem I offer the method.
|< I call it 'Estimating by Threes'.
|< [...]
|
|Alternate mechanism:
|
|* take your best guess as to how long the project will take,
| double it, then go up to the next unit of time :-).
|
| Steve

>From /usr/lib/cookies:

        Westheimer's Rule
                To estimate the time to do a task: estimate
                the time it should take, multiply by 2, and
                change the unit of measure to the next
                highest unit. Thus we allocate 2 days for a
                one hour task.

This wording sounds so scientific that you can put it into your design documentation as a motto on page 3 ;-}

And now, since this is the festive season, we close with a posting from none other than Mr S. Claus himself:

From: sclaus@north.pole.COM (Santa Claus)
Newsgroups: aus.flame
Subject: HO HO HO!!!

Ho ho ho, my little flamers! Which of you has been naughty, and which of you has been nice? Santa has been watching you know!! And for the nice one I have LOTS OF PRESENTS!!

As for the rest of you... well I have a forgiving nature, and so I'd like to wish you all a Very Very Merry Christmas and a Happy New Year. May you all try harder to be nice to one another in 1989.

And just a special message to those who have been spreading the rumour that I don't exist: F*CK OFF AND DIE, SCUMBAGS!

Ho Ho Ho,
Santa

That's it for now - see you in the next issue. Will one of _your_ postings be featured?

Dave Horsfall (VK2KFU), Alcatel-STC Australia, dave@stcns3.stc.oz
dave%stcns3.stc.oz.AU@uunet.UU.NET, ...munnari!stcns3.stc.oz.AU!dave
PCs haven't changed computing history - merely repeated it

UNIX In A Nutshell

O'Reilly & Associates, Inc.

A Book Review by David Purdue

Has this ever happened to you?

You are editing a file, say a C program, and you want to pipe the file through a command, say *cb*. But you have forgotten the flag on *cb* that limits line lengths. So you use *vi*'s shell escape and type, "*man cb*". But *man* takes about 5 minutes to present you with the info, because it has to reformat the page. Finally the manual entry for *cb* appears on your screen, but you have forgotten why you wanted to see it.

Maybe I just have a poor memory, but circumstances like the above happen to me all the time. And as much as I applaud the designers of UNIX† for having online manuals, doing "*man lp*" takes too long and presents too much information when all you really want to know is whether the "number of copies" flag is "-n" or "-#".

Which is why I am finding the new book on my shelf, *UNIX In A Nutshell*, so useful. I say "on my shelf," but the truth is that this book spends more time on my desk.

This book contains everything that you occasionally need to know about the UNIX user interface, and it is arranged in such a way that you can find what you need quickly. It is certainly not a book from which to learn about UNIX use, but as an *aide memoire* it is excellent.

Each chapter of the book covers a different aspect of the UNIX user interface. The chapters are:

1. *UNIX commands*
   - which lists all the UNIX commands by name, shows their syntax and options, and briefly explains their function.
2. *Shell Syntax*
   - which describes command syntax, special files and variables, and built in commands for the Bourne Shell and the C Shell. This chapter includes a comparison of those two shells.
3. *Pattern Matching*
   - a quick reference on how to build regular expressions.
4. *Editor Command Summary*
   - short descriptions of the commands for *vi*, *ex*, *sed*, and *awk*.
5. *Nroff and Troff*
   - describes *nroff/troff* requests, escape sequences, registers, and special characters.
6. *Macro Packages*
   - describes the *mm*, *ms*, and *me* macro packages for *nroff/troff*, in much the same manner as chapter 1 describes commands.
7. *Preprocessors*
   - *tbl*, *eqn*, and *pic* preprocessors for *nroff/troff*.
8. *Program Debugging*
   - a quick guide to using the *adb* and *sdb* debuggers.
9. *SCCS and Make*
   - a summary of how to use these two utilities.
10. *Error Messages*
    - a list of commonly encountered error messages, with explanation of what UNIX is actually complaining about, and telling which command generates each message.

† UNIX is a trademark of Bell Laboratories.

All the information in *UNIX In A Nutshell* is well presented. The pages are well arranged and laid out, so you can find what you are looking for quickly, and the text is concise yet clear and well written.

The only reservation I have about this book is that I would have liked to have seen more in it. For example, chapter 6 could have included the *man* macros. Also, I would have liked to have seen some help for programmers, such as a chapter on system calls and subroutines with the same brief descriptions; although such a chapter may well be considered beyond the scope of this book, and perhaps a seperate "programmer's handbook" would be in order. This volume is definitely a UNIX user's book.

One other worry about this book is that with the current UNIX developments the book may quickly become out of date. It must be noted, though, that O'Reilly seem committed to continuing work on this handbook and updating it to reflect the current UNIX user interface.

Apart from those concerns, the book fulfills its aim as the definitive UNIX quick reference. I strongly recommend *UNIX In A Nutshell*. It is one that I will keep near my terminal, and I feel that it is one of those reference books that helps to increse my productivity.

Nutshell Handbook Offer

Introduction

The "Nutshell Handbook" special offer for AUUG members is going ahead. This article contains a price list and order form; a follow up article will have descriptions of the books available (just in case you missed this before).

I stress again, this offer is for AUUG members only. If you wish to join AUUG, send a message (including your paper mail address) to Tim Roper <timr@labtam.oz>.

I will be placing an order for the books as soon as I receive enough orders. AUUG does not require payment immediately, but will require it before books will be delivered.

Ordering Information

Please complete the attached order form and post it to:

AUUG Nutshell Handbook Offer
Attention: David Purdue
c/o Labtam Limited
43 Malcolm Road
BRAESIDE, VIC, 3195

Payment need not accompany the order form, but books will not be sent until payment has been received (I will post an announcement when the books arrive in Australia). Purchase orders will only be accepted from Institutional members. Others please send cheques payable to AUUG Inc. If your oraganisation is paying, please organise your cheque now.

The prices appear on the order form and include postage and handling charges.

The order form must be signed by a member of AUUG. In the case of an Institutional Member it should either be signed by the Administrative Contact (the person who signed the current membership form) or stamped and signed by a representative of the institution.

Orders will be accepted from non-members only if they are accompanied by a completed membership application form and payment for membership.

Nutshell Handbook Order Form

Contact Details:

Name:

Phone:

Fax:

Net Address:

Postal Address:

Shipping Address (where the books will go):

Books Required:

Title	Copies	Price/copy	Total
Learning The UNIX Operating System		$ 15	$
Learning The Vi Editor		$ 19	$
Termcap And Terminfo		$ 24	$
Programming With Curses		$ 15	$
Managing UUCP and Usenet		$ 24	$
Using UUCP and Usenet		$ 21	$
Managing Projects With Make		$ 15	$
DOS Meets UNIX		$ 19	$
UNIX In A Nutshell (System V edition)		$ 24	$
UNIX In A Nutshell (BSD Edition)		$ 24	$
X Programming Manuals (2 vol set)		$ 89	$
X Window System User's Guide		$ 33	$
Hypercard UNIX In A Nutshell		$ 43	$
MKS Toolkit		$ 124	$
Checking C Programs With lint		$ 18	$
Understanding And Using COFF		$ 24	$
Grand Total			$

Membership Details:

Name of Member:

Category of Membership: Ordinary/Student/Institutional/Hon Life

Signature:

Name (please print):

Nutshell Handbook Summary

This article describes the Nutshell Handbooks being offered for sale by the AUUG. This information is taken mainly from the 'Nutshell News', a press release from the publishers.

UNIX is a trademark of AT&T Bell Laboratories.

Learning The UNIX Operating System

75 pages; AUUG price: $15

For those who are new to UNIX, this will teach just what you need to know to get started and no more. A better introduction than a 600 page book that contains many details irrelevant to the beginner. Topics include logging in and logging out, managing files and directories, redirecting i/o, and customising your account.

Learning The vi Editor

131 pages; AUUG price: $19

vi is the most commonly used text editor on UNIX systems, and is available on other systems aswell. This book teaches how to use vi by starting with basic concepts so that you can begin editing quickly, and then extending your skills so that you can use vi more powerfully.

Termcap And Terminfo

170 pages; AUUG price: $24

The termcap and terminfo databases are UNIX's solution to the dificulty of supporting a wide range of terminals without writing special drivers for each one. Unfortunately, like so many other essential UNIX features, they are poorly documented. This book tries to rectify that situation. Contents include:

- Terminal independence: the need for termcap and terminfo
- Reading termcap and terminfo entries
- Capability syntax
- How users should initialise the terminal environment
- Writing termcap and terminfo entries
- Converting between termcap and terminfo

Programming With curses

71 pages; AUUG price: $15

This book does not tell you what to yell at your terminal when your program won't compile. It does tell you how to use the curses library from C programs to provide a full screen user interface.

Managing UUCP and Usenet

242 pages; AUUG price: $24

This book is meant for system administrators who want to install and manage UUCP and Usenet software. It covers Honey-DanBer UUCP as well as standard Version 2 UUCP, with special notes on Xenix, SunOS and BSD4.3. This book was selected by Usenix's UUNET Communication Services for distribution to new customers.

Using UUCP And Usenet

185 pages; AUUG price: $21

A complete user's guide to UUCP and Usenet, including how to read and post news, send mail to other sites and how to execute commands on remote sytems and log on to remote systems via UUCP.

Manging Projects With make

77 pages; AUUG price: $15

Described as "the clearest book on make ever written" (and from my experience with Nutshell Books, I find that easy to believe). Topics include:

- Writing a simple makefile
- Shell variables
- Internal macros
- Suffix rules
- Special description file targets
- Maintaining libraries
- Invoking make recursively

DOS Meets UNIX

134 pages; AUUG price: $19

There are many applications that run under MS-DOS. UNIX lookalikes (XENIX, Microport, etc.) help you make better use of your PC. This book tells you how to get the best of both worlds. It describes the problems and the solutions available for integrating DOS and UNIX, including a coverage of products like PC-Interface, PC-NFS, Merge/386 and VP/ix that are used to run DOS applications under UNIX.

UNIX In A Nutshell

for System V: 289 pages; AUUG price: $24

for Berkeley: 306 pages; AUUG price: $24

The ultimate UNIX quick reference guides, these contain everything you need to know quickly. The "DEC Professional" (September 1987) states, "I highly recommend the 'UNIX In A Nutshell' handbooks as desktop references. [They] are complete and consise; in fact they pack more information into fewer pages than I've ever seen... These books are truly a bargain." See my review in the December '88 AUUGN. Note that there are different editions for System V and for BSD, so be careful that you order the right one.

X Programming Manuals (2 volume set)

vol1 611 pages, vol2 700 pages; AUUG price: $89

An awful lot of information about X. Volume 1 is a guide to programming with the X library, and takes the broad conceptual view. Volume 2 is a reference manual and gives a detailed description of each of the Xlib functions.

X Window System User's Guide

270 pages; AUUG price: $33

Describes window system concepts and provides a tutorial on the most common client applications. Later chapters describe how to customise the X environment.

Hypercard UNIX In A Nutshell

disk; AUUG price: $43

As for "UNIX In A Nutshell" above, but in the form of a Hypercard stack.

MKS Toolkit

disk; AUUG price: $124

Provides a set of UNIX utilities for DOS systems, including sed, awk, and ksh as a COMMAND.COM replacement.

NEW TITLES

All I know about the following books are their titles and prices:

"Checking C Programs With lint" - AUUG price: $18
"Understanding And Using COFF" - AUUG price: $24

David Purdue
AUUG Nutshell Handbook Coordinator

The Inauugral Software Distribution

Timothy Roper

Secretary, AUUG Inc.

Introduction

In response to the interest expressed in obtaining a timely copy of the source code for Release 3 of the X Window System, **AUUG** has arranged a software distribution that includes X11R3 as well as many other things often requested. This software distribution is being provided by **AUUG** as a free service to its members. If you are not a member this is an opportune time to join.

For more information about Release 3 of X, *USENET* recipients should see

Message-ID: <8810271924.AA01104@EXPO.LCS.MIT.EDU>

in Newsgroup **comp.windows.x**. Please note that this offer is for source code only and does not include any hardcopy, such as the book that you receive when ordering X11R3 from MIT.

What Media is Available

The software is available on the following media.

A. *tar* format on 2400 foot half inch tape written at 1600 BPI

B. *tar* format on 300XL/P (or 600A) quarter inch tape cartridge written in QIC-24 format.

C. *tar* format on 600A quarter inch tape cartridge written in QIC-24 format.

D. *tar* format on MegaTape MT-500 cartridge

Note that 6250 BPI half inch tape copies are not currently available but may be later.

What Software is Available

There are various sets of software available listed below. Each tape will be written with exactly one set. The letters A, B, C or D indicate the media types defined above and the numbers 1 to 11 indicate the contents of the set. There is some overlap between the contents of these sets. You should choose the sets you want according to your media preference and disk space. For example if you just want all the X11R3 stuff you could order A1 and A2 (severely compressed on 2400 foot half inch tapes), C9 and C10 (some files individually compressed on 600A quarter inch tapes), or B6, B7 and B8 (some files individually compressed on 300XL/P quarter inch tapes).

SETs A1 and B1
: X11R3 core distribution and user-contributed software for X: files *tar*ed together then compressed then *split*; includes source code of *(un)compress*.

SETs A2 and B2
: R3 specific user-contributed software for X; archive of **comp.sources.x**; a **gnu** distribution.

SETs A3 and B3
: TeX sources; archives of **comp.sources.misc** and **net.sources**; DARPA RFCs 606–1074.

SETs A4 and B4
: archives of **comp.sources.unix** and **comp.sources.games**

SETs A5 and B5
: a collection of software packages including pcip, ka9q, isode-4, kermit, NCSA Telnet, uupc, citi-macip, jetroff, elm, ietf and some kerberos documentation.

SETs A6 and B6
: X11R3 core distribution; R3 specific user-contributed software for X; archive of **comp.sources.x**: some files individually compressed; includes source code of *(un)compress*.

SETs A7 and B7
: part 1 of 2 of user-contributed software for X

SETs A8 and B8
: part 2 of 2 of user-contributed software for X

SET C9
: X11R3 core distribution; R3 specific user-contributed software for X; archive of **comp.sources.x**; part 1 of 2 of user-contributed software for X: some files individually compressed; includes source code of *(un)compress*.

SET C10
: part 2 of 2 of user-contributed software for X, ie. the companion of SET C9: some files individually compressed.

SET D11
: Contents of A1 through A5, C9 and C10 as seven separate files.

Ordering Details

Please use the attached order form. For current members of AUUG no payment is required which should allow most people to submit orders immediately. This is because you supply the media and the means for it to be returned to you. Note that your tapes will be returned in the packing in which they are received so please bear this in mind when packing and addressing. Just how long it takes to write and return your tapes depends on demand. Orders received between 19th December, 1988 and 6th January, 1989 may suffer longer delays. If you want them back in 1988 please send your order immediately.

Orders from non-members will be accepted only if they are accompanied by a completed membership application form and payment.

To order you must supply:

- a completed order form (see below).
  This must be signed by a member of AUUG. In the case of an Institutional member it should either be signed by the Administrative Contact (the person who signed the current membership form) or stamped and signed by a representative of the institution.
- blank media as per your order form each bearing an adhesive label on which you have written your name, network address and the desired contents of that tape (A1-A8, B1-B8, C9-C10, D11).
- return postage and a self-addressed adhesive mailing label *or* a suitable self-addressed pre-paid courier bag
- in the case of non-members of AUUG, a completed membership application form and payment

The above should be sent to:

AUUG Software Distribution
Attention: Timothy Roper
c/o Labtam Limited
43 Malcolm Road
Braeside VIC
Australia 3195

Membership Applications

Membership information and application forms may be found in a recent copy of the association's newsletter **AUUGN** or obtained by contacting **timr@labtam.oz** or telephoning (03) 5871444. Payment (cheque, money order or credit card) must be enclosed with the application: purchase orders will only be accepted for Institutional memberships.

Inauugral Software Distribution

ORDER FORM

Contact Details:

Name:

Phone:

Fax:

Net Address:

Postal Address:

Shipping Address (as on your enclosed adhesive mailing label or courier bag):

Software Required:

COPIES SET (A1-A8,B1-B8,C9-C10,D11)

Membership Details:

Name of Member:

Category of Membership: Ordinary/Student/Institutional/Hon Life

Signature:

Name (please print):

X Window System Version 11 Release 3 Annoncement

From: jim@EXPO.LCS.MIT.EDU (Jim Fulton)
Newsgroups: comp.windows.x
Subject: Announcing Release 3 of the X Window System
Message-ID: <8810271924.AA01104@EXPO.LCS.MIT.EDU>
Date: 27 Oct 88 19:24:34 GMT

The X Consortium and the MIT Laboratory for Computer Science are proud to announce the third release of the X Window System, Version 11. Highlights of this version include:

- o many bugs have been fixed
- o backing store and save-unders in sample servers
- o professionally designed fonts donated by Adobe/Digital and Bitstream
- o long font names, font name aliasing, font name wildcarding
- o client and server support (monochrome only) for MacII under A/UX
- o Xlib support for Cray supercomputers under UNICOS
- o server support for Parallax Graphics video controllers
- o correct arc code
- o version of the X Toolkit Intrinsics adopted by the X Consortium
- o improved documentation
- o new Display Manager for running X automatically
- o new utilities and demos
- o updated versions of Andrew, Xr11, and InterViews
- o lots of new user-contributed software, including:
  - awm, twm, and rtl window managers
  - fonts from X10, Berkeley Mac Users Group, and INFO-MAC
  - HP and Sony widget sets
  - lots of random utilities
  - previewers for DVI, TROFF, and PIC
  - mazewar, qix, xmille, and xtrek

The sample server, libraries, and applications are not in the public domain, but are freely available. No license is required and there are no royalties; vendors are actively encouraged to base products upon this software.

This release is available from the MIT Software Center, the DARPA Internet sites listed below, the UUNET Project, and several consulting/mail-order firms. It is organized into three pieces: the core software, user-contributed toolkits, and the rest of the user-contributed software. Sites that that have access to the Internet will be able to retreive all three pieces themselves from any of the following machines outside of normal business hours (9am-6pm at that machine) using anonymous ftp:

USA Location	Hostname	Internet Address	anonymous ftp directory
West	gatekeeper.dec.com	128.45.9.52	pub/X.V11R3/
Midwest	mordred.cs.purdue.edu	192.5.48.2	pub/X11/Release3/
East	uunet.uu.net	192.12.141.129	X/X.V11R3/
*	expo.lcs.mit.edu	18.30.0.212	pub/R3/

(*) in a galaxy far, far away.
(**) If mordred.cs.purdue.edu doesn't respond on 192.5.48.2, try 128.10.2.2.

The directories listed above contain a README (which you should read first), a PostScript version of the release notes, and three directories containing split, compressed tar files.

A set of three 1600bpi tapes in UNIX tar format plus printed versions of the major manuals and a copy of the new Gettys, Newman, and Scheifler book "X Window System: C Library and Protocol Reference" are available from the MIT Software Center for the following rates (prices include shipping):

	manuals, book	tapes, manuals, book
North America	$125	$400
Everywhere Else	$175	$500

To order, please send a letter and a check payable to MIT in US currency for the appropriate amount to:

MIT Software Center
Technology Licensing Office
room E32-300
77 Massachusetts Avenue
Cambridge, MA 02139

For ordering information, call the "X Ordering Hotline" at +1 (617) 258-8330 after 31 October 1988 or the Software Center at +1 (617) 253-6966.

Bob Scheifler, Jim Fulton, Keith Packard, Donna Converse, Michelle Leger
X Consortium
MIT Laboratory for Computer Science

Ralph Swick
MIT Project Athena

;login:

The USENIX Association Newsletter

Volume 13, Number 6 **November/December 1988**

CONTENTS

The closing date for submissions for the next issue of *;login:* is January 3, 1989

USENIX THE PROFESSIONAL AND TECHNICAL UNIX® ASSOCIATION

USENIX Winter Conference

Town & Country Hotel
San Diego, CA

January 30 - February 3, 1989

Tutorials

MONDAY, JANUARY 30

Introduction to 4.3BSD Internals

Thomas W. Doeppner, Jr., Brown University

You must be licensed for 4.2/3BSD source code, or licensed for either 32/V, System III or System V source code from AT&T in order to attend this tutorial.

Introduction to UNIX System V Internals

Steve Buroff, AT&T, & Curt Schimmeh, Key Computer Labs

An Introduction to C++

Robert Murray, AT&T Bell Laboratories

4BSD TCP/IP Performance Improvements*

Van Jacobson, Lawrence Berkeley Laboratory, & Mike Karels, University of California, Berkeley

UNIX System V Streams Module and Driver Design

Hari Pulijal & Stephen Rago, AT&T Bell Laboratories

Introduction to Programming the X Window System,† Version 11

Oliver Jones, Apollo Computer, Inc.

UNIX System V Remote File Sharing (RFS)

Michael Padovano & Michael Scheer, AT&T Bell Laboratories

The Andrew Toolkit (ATK): An Introduction

Andrew J. Palay & Nathaniel S. Borenstein, Carnegie-Mellon University

Open Systems Interconnection (OSI)*

Colin I'Anson, Hewlett-Packard

Managing a Network of NFS Systems

Ed Gould, mt Xinu

Security Issues in a Distributed UNIX Environment: The Kerberos Approach*

Dan Geer, Jennifer Steiner & Jon Rochlis, MIT

Using PostScript‡ as "Yet Another" UNIX Tool*

Dick Dunn, Interactive Systems Corp.

* This is the first time the USENIX Association has offered this tutorial.

† The X window System is a trademark of MIT.

‡ PostScript is a trademark of Adobe Systems, Inc.

TUESDAY, JANUARY 31

Beyond 4.3BSD: Advanced Kernel Topics

Mike Karels & Marshall Kirk McKusick, University of California, Berkeley

Advanced UNIX System V Internals – A Code Walk Through

Steve Buroff, AT&T, & Curt Schimmel, Key Computer Labs

You must be licensed for UNIX System V.3.1 source code in order to attend this tutorial.

Language Construction Tools on the UNIX System

Steve Johnson, Ardent Computer Corp.

UNIX 4.X BSD Systems Administration

Rob Kolstad, Prisma Inc., & Evi Nemeth, University of Colorado

XToolkit Intrinsics

Chuck Price, Digital Equipment Corp.

Open Network Computing (ONC) and NFS#

Sally Ahnger, John Corbin & Christopher Silveri, Sun Microsystems, Inc.

UNIX Device Driver Design (4.2/4.3BSD)

Daniel Klein, Software Engineering Institute

You must be licensed for 4.2/3BSD source code, or licensed for either 32/V, System III or System V source code from AT&T in order to attend this tutorial.

MACH

Avadis Tevanian, Jr., Next, Inc.

Network Computing System and Architecture: Overview and Tutorial in Writing Distributed Applications*

Nathaniel Mishkin & Paul J. Leach, Apollo Computer, Inc.

Network Extensible Window System (NeWS)

Owen M. Densmore & David A. LaVallee, Sun Microsystems, Inc.

Object Oriented Design on UNIX: The Eiffel Approach*

Bertrand Meyer, Interactive Software Engineering, Inc.

ONC and NFS are trademarks of Sun Microsystems, Inc.

* This is the first time the USENIX Association has offered this tutorial.

Conference Program

WEDNESDAY, FEBRUARY 1

Introduction and Keynote Chair: Peter Salus 9:00-10:30

Keynote:
William T. O'Shea, Vice President – Product Development, AT&T

Break 10:30-11:00

RPC and Distributed Systems Chair: Lori Grob 11:00-12:30

Experiences with PARPC
Bruce Martin, Hewlett-Packard
Charles Bergan & Walter Burkhard, University of California at San Diego
Jehan-Francois Paris, University of Houston

Dynamically synchronized locking – a lightweight locking protocol for resource locking in a stateless environment
Peter Gloor & Rudolf Marty, University of Zurich

Adding UNIX Hosts to Distributed Operating System
Geoffrey Lee, Lawrence Livermore National Laboratory

Lunch 12:30- 2:00

File Systems Chair: Melinda Shore 2:00- 3:30

The Automounter
Brent Callaghan & Tom Lyon, Sun Microsystems, Inc.

Improving the Performance and Correctness of an NFS Server
Chet Juszczak, Digital Equipment Corp.

RASH: A Rapid Access Storage Hierarchy
Robert Henderson, NASA Ames Research Center

Break 3:30- 4:00

Operating Systems I Chair: Avadis Tevanian 4:00- 5:30

Multiprocessor Aspects of the DG/UX 4.00 Kernel
Michael Kelley, Data General Corp.

MOSIX: An Integrated UNIX for Multiprocessor Workstation
Amnon Barak & Richard Wheeler, Hebrew University of Jerusalem

Sema: a Lint-like tool for Analyzing Semaphore Usage in a Multithreaded UNIX Kernel
Joe Korty, MODCOMP

THURSDAY, FEBRUARY 2

Window Systems Chair: Jeff Schwabb 9:00-10:30

Visualizing X11 Clients
David Lemke & David Rosenthal, Sun Microsystems, Inc.

PEX: A 3D Extension to X Windows
Spencer Thomas & Martin Friedmann, University of Michigan

XVT: Virtual Toolkit for Portability Between Window Systems
Mark Rochkind, Advanced Programming Institute

Break 10:30-11:00

Special Interest I Chair: Andrew Hume 11:00-12:30

Viral Attacks on UNIX System Security
Tom Duff, AT&T Bell Laboratories

A Faster *fsck* for BSD UNIX
Eric Bina & Perry Emrath, University of Illinois at Urbana-Champaign

Lessons of the New Oxford English Dictionary Project
Tim Bray, University of Waterloo

Lunch 12:30- 2:00

Internetworking Chair: Tom Narten 2:00- 3:30

Implementation of Dial-up IP for UNIX Systems
Leo Lanzillo & Craig Partridge, BBN Systems and Technologies Corp.

A UNIX Implementaion of the Simple Network Management Protocol
Wengyik Yeongw, Marty Schoffstall & Mark Fedor, NYSERNet, Inc.

Limiting Factors in the Performance of Jacobson TCP Algorithms
Allison Mankin & Kevin Thompson, MITRE Corp.

Break 3:30- 4:00

Objects & Memory Chair: Jim Lipkis 4:00- 5:30

The Shared Memory Server
Alessandro Forin, Joseph Barrera & Richard Sanzi, Carnegie-Mellon University

Minimalist Physical Memory Control in UNIX
Mark Holderbaugh & Scott Preece, Gould, Inc.

Software Configuration Management with an Object-Oriented Database
Eric Black, Atherton Technology

Supporting Objects in a Conventional Operating System
Prasun Dewan, Purdue University

Work in Progress Chair: Keith Bostic 4:00- 5:30

Ten minute presentations of current work

FRIDAY, FEBRUARY 3

Special Interest II Chair: Donn Seeley 9:00- 10:30

A Comparison of Compiler Utilization of Instruction Set Architectures
Daniel Klein, Software Engineering Institute

Discuss: An Electronic Conferencing System for a Distributed Computing Environment
Ken Raeburn, Jon Rochlis & Stan Zanarotti, Massachusetts Institute of Technology
William Sommerfeld, Apollo Computer, Inc.

A Partial Tour Through the UNIX Shell
Geoff Collyer, University of Toronto

Break 10:30-11:00

Operating Systems II Chair: John Kepecs 11:00-12:30

Job and Process Recovery in a UNIX-based Operating System
Brent Kingsbury & John Kline, Cray Research, Inc.

Session Management in System V Release 4
Tim Williams, AT&T Bell Laboratories

The Modix Kernel
Greg Snider & Jim Hayes, Hewlett-Packard

Lunch 12:30- 2:00

Processes Chair: Jim McGinness 2:00- 3:30

Evolving the UNIX System Interface to Support Multithreaded Programs
Paul McJones & Garret Swart, Digital Equipment Corp.

Variable Weight Processes With Flexible Shared Resources
Ziya Aral, Ilya Gertner & Greg Schaffer, Encore Computer Group

System V/MLS Labeling and Mandatory Policy Alternatives
Charles Flink & John Weiss, AT&T Bell Laboratories

Break 3:30- 4:00

Security Chair: Rick Lindsley 4:00- 5:30

Secure Multi-level Windowing in a B1 Certifiable Secure UNIX
Barbara Smith-Thomas, AT&T Bell Laboratories

Secure Window Systems
Mark Carson, Wen-Der Jiang, Jeremy Liang, Gary Luckenbaugh, & Debra Yakov, IBM Corp.

A Trusted Network Architecture for AIX Systems
Chii-Ren Tsai, Virgil Gligor, Wilhelm Burger, Mark Carson, Pau-Chen Cheng, Janet Cugini, Matthew Hecht, Shau-Ping Lo, Sohail Malik, & N. Vasudevan, IBM Corp.

Call for Papers

Workshop on Software Management

New Orleans Hilton and Towers
New Orleans, LA

April 3-4, 1989

David Tilbrook and Barry Shein will be chairing a workshop in New Orleans, LA on Monday and Tuesday, April 3-4, 1989. The workshop will concern the management and processing of source; the discipline of managing, maintaining, and distributing software. The ultimate objective of software management is unremarkable and painless installation of software and its subsequent upgrades at a remote site. The objective of the workshop is to present, discuss, and increase awareness of the issues involved with software management, in order to improve and facilitate the distribution and sharing of source throughout the UNIX community.

Release engineering
Configuration management
Installation tools and techniques
Construction tools and techniques
Source code control systems
Testing

The workshop will include full length papers, short presentations, and a panel discussion on tools (e.g., is PCTE a good or viable idea?).

Among the speakers already scheduled are: Vic Stenning (keynote), Steve Bourne, Andrew Hume, Kirk McKusick, and Evi Nemeth.

Abstracts of 350-700 words in PostScript or *troff* format should be submitted to *software@usenix*, by **January 25th, 1989**. Full papers will be required by **March 2nd, 1989**. Authors will be notified by **Febuary 6th, 1989** or at the San Diego Conference.

Call for Papers

Workshop on UNIX Transaction Processing

Pittsburgh Hilton Hotel
Pittsburgh, PA

May 1-2, 1989

It is expected that the UNIX System will play an increasingly important role in hosting production transaction processing systems. This first transaction processing workshop will explore existing technology applicable to UNIX-based transaction processing, and hopefully generate technical discussion on future requirements. The intent is to have short papers and presentations which include (but are not limited to) the following topics:

Transaction Integrity
Two-Phase Commit
Distributed Transactions
Client-Server Transaction models
Transaction queing and scheduling
Data Entry Systems
Transaction Benchmarking
Transaction system performance modelling
Operating System Support for Transaction systems

The workshop will focus on short papers and presentations. Please send electronically or on paper a one to two page single-spaced summary describing your paper or presentation to Doug Kevorkian by **February 1, 1989**. All submissions will be acknowledged, and authors will be notified of acceptance by March 15, 1989.

For further details about the workshop, contact the program chair:

Doug Kevorkian
AT&T Bell Laboratories
Room 5-340
190 River Road
Summit, New Jersey 07901

(201) 522-5086 (voice)
(201) 522-6621 (FAX)
attunix!dek

Call for Papers
Summer 1989 USENIX Conference

Baltimore, Maryland
June 12-16, 1989

Papers in all areas of UNIX-related research and development are solicited for formal review for the technical program of the 1989 Summer USENIX Conference. Accepted papers will be presented during the three days of technical sessions at the conference and published in the conference proceedings. The technical program is considered the leading forum for the presentation of new developments in work related to or based on the UNIX operating system.

Appropriate topics for technical presentations include, but are not limited to:

Performance:
- Kernel enhancements
- Compute and file servers
- Scaling issues resulting from more MIPS

File systems: CD-ROM, WORM, network, archival

Networks: WAN, LAN, UUCP, OSI, distributed services

User interfaces

High reliability/availability, fault tolerance

Heterogeneous environments: mainframes DOS/UNIX migration

Media: graphics, video, audio, art, education

System/network administration and security

Trends:
- Lightweight processes
- Neural networks
- Object-oriented extensions

All submissions should describe new and interesting work. Like recent USENIX conferences, the Baltimore conference is requiring the submission of full papers rather than extended abstracts. The review and production cycle will not allow time for rewrite and re-review. (Time is, however, scheduled for authors of accepted papers to perform minor revisions.) Acceptance or rejection of a paper will be based solely on the work as submitted.

To be considered for the conference, a paper should include an abstract of 100 to 300 words, a discussion of how the reported results relate to other work, illustrative figures, and citations to relevant literature. The paper should present sufficient detail of the work plus appropriate background or references to enable the reviewers to perform a fair comparison with other work submitted for the conference. Full papers should be 8-12 single spaced typeset pages. All final papers must be submitted in a format suitable for camera-ready copy. For authors that do not have access to a suitable output device, facilities will be provided.

An abstract should be submitted as soon as possible. Full details and requirements will be supplied to prospective authors. Copies of the full manuscript should be submitted by ordinary and electronic mail to the Program Chair. Electronic submissions are recommended; *troff -ms* if possible.

Four copies and one electronic copy of each submitted paper should be received by **February 8, 1989**. Papers not received by this date will not be considered. Papers which clearly do not meet USENIX's standards for applicability, originality, completeness, or page length may be rejected without review. Acceptance notification will be made by March 13, 1989, and final camera-ready papers will be due by **April 7, 1989**.

Neil Groundwater
Baltimore USENIX Technical Program
Sun Microsystems, Inc.
8219 Leesburg Pike #700
Vienna, Virginia 22180
(703) 883-1221

Abstracts, submissions, and questions:

usenet: {ucbvax,decvax,decwrl,seismo}!sun!balt-usenix

internet: balt-usenix@sun.com

Book Reviews

Programming in ANSI C

by Stephen Kochan
(Prentice Hall, Inc.)

Reviewed by Ed Gronke

Sun Microsystems, Inc.

In the preface to this book, the author states that the reason behind this book is that he wishes to teach ANSI C without discussing features in the "older" C. Though I might dispute the utility of this, I think that he has achieved that goal.

I should make a note here that the aim of this book is for the novice programmer learning C as a first or second computer language with little programming experience. I tried to review the book from the standpoint of just such a programmer.

The organization of the book is well thought out and all of the standard topics (variable declaration and use, expressions, loop construction, conditional expressions and their use in conditional execution) are well covered. There are many program examples and clear and concise explanations.

One chapter of particular note is the chapter on pointers. The author works into the concept of pointers naturally from the previous chapter on character strings and mixes text with simple figures to get the message across.

Also of note is the general discussion of programming style throughout the book. At strategic points, the various possibilities are discussed and recommendations made. For example, the placement of braces (forever a religious war) is discussed at the first program example. Also, within the discussion of macro expansion, the placement of parentheses in the definition of a macro define with arguments is well presented.

Another section of the book which is essential for novice programmers is an appendix entitled **Common Programming Mistakes**. It covers a number of common programming mistakes in C, such as the unwanted semicolon, the uninitialized pointer, and the & (address of) operator in scanf calls, to name a few.

This is not to say that I was totally pleased with the book either. Though I could applaud the aims and the wishes of the author, I do disagree with the fact that the only place that the current status of ANSI C (i.e., in draft form) is mentioned is in the preface. It gives the false impression to the reader that the dialect of C described is well established and readily available.

Another complaint that I have with the book is its length. Although I have heard many complaints from people about the terseness of the other standard text for learning the language, I find that this book goes to the other extreme. Though I found some of the chapters concise and terse, others, such as the chapter on conditional expressions and executions, are overly wordy.

Also, the overall description of declaration of variables, with respect to type and scope, was disjointed. The scope of variables was discussed in the chapter dealing with functions while the various types were discussed in a separate chapter. I feel that there should have been one place in the book where, as a reference, the reader could find out the complete description on how to declare variables.

This leads to the general lack of a good summary in the book. Only the first appendix makes an attempt at this, though it has the same problem with length as some of the rest of the book.

A list of all the program examples would have also been helpful, so that someone attempting to browse through the book by trying the examples could do it easily. (Many people learn by doing, not by reading.)

Finally, I found the index of the book totally inadequate. Especially in a book of this length, I would expect to be able to use the index to browse through specific topics and find related topics. Though I found no glaring

absences from the entries, I found a number of missing references within the entries and few cross-references.

But, barring these problems, I felt that the book was a reasonable treatment of the material for novice programmers, assuming they have access to an ANSI C compiler.

Operating Systems: Communicating with and Controlling the Computer

by Laurie S. Keller
(London: Prentice Hall Ltd.) 370 pp.

Reviewed by Marc D. Donner

IBM T. J. Watson Research Center
donner@IBM.com

The concept behind this book has some things to recommend it, but the execution is so inadequate as to render the result completely useless. The book seems to be intended to be used to educate technically naive people about operating systems, with a heavy emphasis on examples drawn from currently significant commercial systems and light emphasis on concepts. The coverage is so spotty and so thin, however, that nothing of note is really learnable from the book.

The book starts off with a thumbnail sketch of computers and operating systems, focusing primarily on management of cpu time and of memory and on multiprocessors. It isn't clear why these three topics were picked or why only a total of six pages in two different chapters were devoted to questions of nonvolatile storage, file systems, and naming.

The bulk of the book is a series of case studies, in which seven different operating systems are examined in faint detail and contrasted with one another. The seven systems selected are: CP/M, MS-DOS, the p-System, UNIX, TPF/II, VM/SP, and MVS. The treatments of MS-DOS, UNIX, and VM/SP, with all of which I have substantial experience, are superficial and, worse yet, wrong-headed. My experience with the other four is sufficiently limited to make it difficult for me to evaluate those case studies.

For a specific example, the chapter on UNIX goes into considerable detail about the hierarchical name space of the UNIX file system, but mentions only in passing the UNIX file model and doesn't consider any of the implications of the file model. As a consequence the entire treatment is unfocused and ad hoc.

The figures in the book generally contain more flash than actual content, reinforcing the inference that the intended audience is managers and marketeers who need to be fluent in the buzzwords and don't really need any real grasp of the technical content. The quality of the writing is abysmal. The abuse of *which* is inexcusable, even in a book produced in the UK. The author seems to have a strong affinity for having successive verbs alternate between present and past tenses, something that produces affects ranging from discomfort to confusion in the reader.

This book may be of some value to some community of readers, but to the technically astute USENIX community it is completely worthless.

Long-Term Calendar of UNIX Events†

1988 Dec 5-7	Sun User Group	Fontainebleau Hilton, Miami Beach, FL
1988 Dec 7-8	UNIX Fair '88	JUS; Tokyo, Japan
1988 Dec 12-16	ANSI X3J11	Seattle, WA
1988 Dec 13-15	System V.4 Software Dev.	AT&T and Sun; Washington, DC
1989 Jan 9-13	IEEE 1003	Embassy Suites, Ft. Lauderdale, FL
1989 Jan 17	Terminal Int. Ext. and Net. Serv.	NIST; MD
1989 Jan 30-Feb 3	USENIX	Town and Country, San Diego, CA
1989 Feb	UNIX in Government	Ottawa, Ont.
1989 Feb 28-Mar 3	UNIX Convention	AFUU; Paris, France
1989 Feb 28-Mar 3	UniForum	Moscone Center, San Francisco, CA
1989 Apr 3-4	*Software Management Workshop	New Orleans Hilton, New Orleans, LA
1989 Apr 3-7	EUUG	Palais des Congres, Brussels, Belgium
1989 Apr 10-11	ANSI X3J11	Phoenix, AZ
1989 Apr 24-28	IEEE 1003	Minneapolis-St. Paul, MN
1989 May 1-2	*Transaction Processing Workshop	Pittsburgh Hilton, Pittsburgh, PA
1989 May 8-12	DECUS	Atlanta, GA
1989 May 14-16	AMIX	Israel
1989 May 16	POSIX Application Workshop	NIST; MD
1989 May	UNIX 8x/etc	/usr/group/cdn; Toronto, Ont.
1989 Jun	NZSUGI	New Zealand
1989 Jun 12-16	USENIX	Hyatt Regency, Baltimore, MD
1989 Jul	JUS 13	Toyko, Japan
1989 Jul 10-14	IEEE 1003	San Francisco, CA
1989 Sep 18-22	EUUG	Vienna, Austria
1989 Oct 5-6	*Distributed Systems Workshop	Marriott Marina, Ft. Lauderdale, FL
1989 Oct 16-20	IEEE 1003	Brussels (or Amsterdam)?
1989 Nov 1-3	UNIX Expo	New York, NY
1989 Nov 6-10	DECUS	Anaheim, CA
1989 Nov/Dec	*Graphics Workshop V	?
1989 Nov	JUS 14	Osaka or Kobe, Japan
1989 Dec	JUS UNIX Fair	Toyko, Japan
1990 Jan 22-26	USENIX	Omni Shoreham, Washington, DC
1990 Jan 23-26	UniForum	Washington Hilton, Washington, DC
1990 Jan 29	IEEE 1003	New Orleans, LA
1990 Feb	UNIX in Government	Ottawa, Ont.
1990 Apr	IEEE 1003	Montreal, Que.
1990 Apr 23-27	EUUG	Munich, Germany (tentative)
1990 May 7-11	DECUS	New Orleans, LA
1990 May	UNIX 8x/etc	/usr/group/cdn; Toronto, Ont.
1990 Jun 11-15	USENIX	Marriott Hotel, Anaheim, CA
1990 Autumn	EUUG	south of France
1991 Jan 21-25	USENIX	Grand Kempinski, Dallas, TX
1991 Jan 22-25	UniForum	Infomart, Dallas, TX
1991 Jun 10-14	USENIX	Opryland, Nashville, TN

† Partly plagiarized from John S. Quarterman by PHS.

* USENIX Workshops

Future Events

USENIX 1989 Winter Technical Conference
San Diego, Jan. 30-Feb. 3, 1989
See page 3.

Workshop on Software Management
New Orleans, Apr. 3-4, 1989
See page 8.

EUUG Spring Conference
Brussels, Apr. 3-7, 1989
See page 9.

Workshop on UNIX Transaction Processing, Pittsburgh, May 1-2, 1989
See page 10.

USENIX 1989 Summer Conference and Exhibition, Baltimore, Jun. 12-16, 1989
See page 11.

Distributed Processing Workshop
Fort Lauderdale, Oct., 5-6, 1989

Graphics Workshop V, Nov. or Dec. 1989

Long-term USENIX & EUUG Schedule

Sep 18-22 '89	Vienna, Austria
Jan 22-26 '90	Omni Shoreham, Washington, DC
Apr 23-27 '90	Munich, W. Germany
Jun 11-15 '90	Marriott Hotel, Anaheim
Jan 21-25 '91	Grand Kempinski, Dallas
Jun 10-14 '91	Opryland, Nashville
Jan 20-24 '92	Hilton Square, San Francisco
Jun 8-12 '92	Marriott, San Antonio

UNIX Calendar

John Quarterman, of TIC, and Alain Williams, of EUUG, are planning to compile a calendar of world-wide UNIX events. If you have an event you wish to publicize on this calendar, contact John Quarterman at *jsq@longway.tic.com* or at *uunet!longway!jsq*.

2.10BSD Software Release

The "Second Berkeley Software Distribution" (2.10BSD), produced by the Computer Systems Research Group (CSRG) of the University of California, Berkeley, is being distributed by the USENIX Association. It is available to all V7, System III, System V, and 2.9BSD licensees for a price of $200. The release consists of two 2400 foot, 1600 BPI tapes (approximately 80Mb) and approximately 100 pages of documentation. If you require 800 BPI tapes, please contact USENIX for more information.

Sites wishing to run 2.10BSD should be aware that the networking is only lightly tested, and that certain hardware has yet to be ported. Contact Keith Bostic at the address below for current information as to the status of the networking. As of August 6, 1987, the complete 4.3BSD networking is in place and running, albeit with minor problems. The holdup is that only the Interlan Ethernet driver has been ported, as well as some major space constraints. Note, if we decide to go to a supervisor space networking, 2.10 networking will only run on:

11/44/53/70/73/83/84
11/45/50/55 with 18 bit addressing

If you have questions about the distribution of the release, please contact USENIX at:

2.10BSD
USENIX Association
PO Box 2299
Berkeley, CA 94710

+1 415 528-8649
{uunet,ucbvax}!usenix!office

If you have technical questions about the release, please contact Keith Bostic at:

{ucbvax,seismo}!keith
keith@okeeffe.berkeley.edu

+1 415 642-4948

Keith Bostic
Casey Leedom

NOTE: There are a few copies of 2.9BSD available. If you do not have split I&D and want to run UNIX on your PDP-11/x, write the USENIX office.

– PHS

Publications Available

The following publications are available from the Association Office. Prices and overseas postage charges are per copy. California residents please add applicable sales tax. Payment **must** be enclosed with the order and **must** be in US dollars payable on a US bank.

The EUUG Newsletter, which is published four times a year, is available for $4 per copy or $16 for a full-year subscription.

We hope to have EUUG tapes and conference proceedings available shortly.

Conference and Workshop Proceedings

Meeting	Location	Date	Price	Overseas Air
Large Installation Systems Admin. Workshop	Monterey	Nov. '88	$ 8	$ 7
C++ Conference	Denver	Oct. '88	20	20
UNIX and Supercomputers Workshop	Pittsburgh	Sep. '88	20	15
UNIX Security Workshop	Portland	Aug. '88	5	7
USENIX Conference	San Francisco	Jun. '88	20	20
C++ Workshop	Santa Fe	Nov. '87	20	20
Graphics Workshop IV	Cambridge	Oct. '87	10	15
USENIX Conference	Washington DC	Jan. '87	10	20
Graphics Workshop III	Monterey	Dec. '86	10	15

EUUG Proceedings for Spring 1988 (London) and Fall 1988 (Portugal) are available in limited numbers to North American customers at $40 per copy.

C++ Tape

The first 1988 USENIX software tape contains C++ software. It requires no AT&T nor UC license. It has been sent to all current Institutional and Supporting members of the Association.

Individual members of USENIX who wish to obtain a copy of the tape may request it from the Association Office, which will then send the requestor a "Tape Release Form." The form plus a check for $125 should be returned to the office, whereupon the tape will be sent out, postpaid in the US. Foreign individuals will be billed for the additional (air) postage/shipping.

The tape, in *tar* format at 1600 BPI, contains:

GNU C++ Version 1.21.0 (Michael Tiemann)
OOPS (Keith E. Gorlen)
Storage management class and String class... (Peter A. Buhr)
InterViews 2.3 (Mark A. Linton)
C++ Subroutines for string manipulation (Arthur Zemon)

The tape is not available to non-members of the USENIX Association.

NOTICE: Some copies of the C++ tape are faulty. If you cannot read your tape, return it to the USENIX Office.

4.3BSD Manuals

The USENIX Association now offers all members of the Association the opportunity to purchase 4.3BSD manuals.[†]

The 4.3BSD manual sets are significantly different from the 4.2BSD editions. Changes include many additional documents, better quality of reproductions, as well as a new and extensive index. All manuals are printed in a photo-reduced 6″×9″ format with individually colored and labeled plastic "GBC" bindings. All documents and manual pages have been freshly typeset and all manuals have "bleed tabs" and page headers and numbers to aid in the location of individual documents and manual sections.

A new Master Index has been created. It contains cross-references to all documents and manual pages contained within the other six volumes. The index was prepared with the aid of an "intelligent" automated indexing program from Thinking Machines Corp. along with considerable human intervention from Mark Seiden. Key words, phrases and concepts are referenced by abbreviated document name and page number.

While two of the manual sets contain three separate volumes, you may only order complete sets.

The costs shown below do not include applicable taxes or handling and shipping from the publisher in New Jersey, which will depend on the quantity ordered and the distance shipped. Those charges will be billed by the publisher (Howard Press).

Manuals are available now. To order, return a completed "4.3BSD Manual Reproduction Authorization and Order Form" to the USENIX office along with a check or purchase order for the cost of the manuals. You **must** be a USENIX Association member. Checks and purchase orders should be made out to "Howard Press." The manuals will be shipped to you directly by the publisher.

Manual	Cost*
User's Manual Set (3 volumes) User's Reference Manual User's Supplementary Documents Master Index	$25.00/set
Programmer's Manual Set (3 volumes) Programmer's Reference Manual Programmer's Supplementary Documents, Volume 1 Programmer's Supplementary Documents, Volume 2	$25.00/set
System Manager's Manual (1 volume)	$10.00

* Not including postage and handling or applicable taxes.

4.2BSD Manuals are No Longer Available

[†] Tom Ferrin of the University of California at San Francisco, a former member of the Board of Directors of the USENIX Association, has overseen the production of the 4.2 and 4.3BSD manuals.

4.3BSD Manual Reproduction Authorization and Order Form

This page may be duplicated for use as an order form

USENIX Member No.: _____

Purchase Order No.: _____

Date: _____

As a USENIX Association Member in good standing, and pursuant to the copyright notice as found on the rear of the cover page of the UNIX®/32V Programmer's Manual stating that

> "Holders of a UNIX®/32V software license are permitted to copy this document, or any portion of it, as necessary for licensed use of the software, provided this copyright notice and statement of permission are included,"

I hereby appoint the USENIX Association as my agent, to act on my behalf to duplicate and provide me with such copies of the Berkeley 4.3BSD Manuals as I may request.

Signed: _____

Institution (if Institutional Member): _____

Ship to:

Name: _____

_____

_____

_____

_____

Phone: _____

Billing address, if different:

Name: _____

_____

_____

_____

_____

Phone: _____

The prices below **do not** include shipping and handling charges or state or local taxes. All payments must be in US dollars drawn on a US bank.

4.3BSD User's Manual Set (3 vols.)	_____	at $25.00 each	= $_____
4.3BSD Programmer's Manual Set (3 vols.)	_____	at $25.00 each	= $_____
4.3BSD System Manager's Manual (1 vol.)	_____	at $10.00 each	= $_____
Total	_____		$_____

[ ] Purchase order enclosed; invoice required.
(Purchase orders **must** be enclosed with this order form.)

[ ] Check enclosed for the manuals: $_____
(Howard Press will send an invoice for the shipping and handling charges and applicable taxes.)

Make your check or purchase order out to "Howard Press" and mail it with this order form to:

Howard Press
c/o USENIX Association
P.O. Box 2299
Berkeley, CA 94710

Local User Groups

The USENIX Association will support local user groups by doing an initial mailing to assist the formation of a new group and publishing information on local groups in *;login:*. At least one member of the group must be a current member of the Association.

CA - Fresno: the Central California UNIX Users Group consists of a *uucp*-based electronic mailing list to which members may post questions or information. For connection information:

Educational and governmental institutions:

Brent Auernheimer (209) 294-4373
brent@CSUFresno.edu or csufres!brent

Commercial institutions or individuals:

Gordon Crumal (209) 875-8755
csufres!gordon (209) 298-8393

CA - Los Angeles: the Los Angeles UNIX Group meets on the 3rd Thursday of each month in Redondo Beach.

Drew Bullard (213) 535-1980
ucbvax!trwrb!bullard

Marc Ries (213) 535-1980
{decvax,sdcrdcf}!trwrb!ries

CO - Boulder: the Front Range UNIX Users Group meets monthly at different sites.

Steve Gaede (303) 938-2985
NBI, Inc.
P.O. Box 9001
Boulder, CO 80301

{boulder,hao}!nbires!gaede

FL - Coral Springs:

S. Shaw McQuinn (305) 344-8686
8557 W. Sample Road
Coral Springs, FL 33065

FL - Melbourne: the Space Coast UNIX Users Group meets at 8pm on the 3rd Wednesday of each month at the Florida Institute of Technology.

Bill Davis (407) 242-4449
bill@ccd.harris.com

FL - Orlando: the Central Florida UNIX Users Group meets the 3rd Thursday of each month.

Mike Geldner (305) 862-0949
codas!sunfla!mike

Ben Goldfarb (305) 275-2790
goldfarb@hcx9.ucf.edu

Mikel Manitius (305) 869-2462
{codas,attmail}!mikel

FL - Tampa Bay: the Tampa UNIX Users Group meets the 1st Thursday of each month, alternately in Largo and Tampa.

Scott Stone (813) 974-3307
uflorida!usfvax2!stone, stone@usf.edu

Bill Hargen (813) 530-8655
{codas,usfvax2}!pdn!hargen

George W. Leach (813) 530-2376
uunet!pdn!reggie

GA - Atlanta: meets on the 1st Monday of each month in White Hall, Emory University.

Atlanta UNIX Users Group
P.O. Box 12241
Atlanta, GA 30355-2241

Marc Merlin (404) 442-4772
Mark Landry (404) 365-8108

MI - Detroit/Ann Arbor: meets the 2nd Thursday of each month in Ann Arbor.

William Bulley (313) 995-6211
web@applga.uucp

Rich McGill (313) 971-5950
rich@oxtrap.uucp

Steve Simmons (313) 426-8981
scs@lokkur.uucp

MI - Detroit/Ann Arbor: dinner meetings the 1st Wednesday of each month.

Linda Mason (313) 855-4220
michigan!/usr/group
P.O. Box 189602
Farmington Hills, MI 48018-9602

MN - Minnetonka: meets the 1st Wednesday of each month.

UNIX Users of Minnesota
545 Ashland Avenue #3
St. Paul, MN 55102

Scott Anderson (612) 688-0089
scott@questar.mn.org

MO - St. Louis:

St. Louis UNIX Users Group
Plus Five Computer Services
765 Westwood, 10A
Clayton, MO 63105

Eric Kiebler (314) 725-9492
plus5!sluug

NE - Omaha: meets on the 2nd Thursday of each month.

/usr/group nebraska
P.O. Box 44112
Omaha, NE 68144

Kent Landfield (402) 291-8300
kent@ugn.uucp

New England - Northern: meets monthly at different sites.

Emily Bryant (603) 646-2999
Kiewit Computation Center
Dartmouth College
Hanover, NH 03755

David Marston (603) 883-3556
Daniel Webster College
University Drive
Nashua, NH 03063

decvax!dartvax!nneuug-contact

NJ - Princeton: the Princeton UNIX Users Group meets monthly.

Pat Parseghian (609) 452-6261
Dept. of Computer Science
Princeton University
Princeton, NJ 08544

pep@Princeton.EDU

NY - New York City:

Unigroup of New York
G.P.O. Box 1931
New York, NY 10116

Ed Taylor (212) 513-7777
{attunix,philabs}!pencom!taylor

New Zealand:

New Zealand UNIX Systems User Group
P.O. Box 13056
University of Waikato
Hamilton, New Zealand

OK - Tulsa:

Pete Rourke
$USR
7340 East 25th Place
Tulsa, OK 74129

PA - Philadelphia: the UNIX SIG of the Philadelphia Area Computer Society (PACS) meets the morning of the 3rd Saturday of each month at the Holroyd Science Building, LaSalle University.

G. Baun, UNIX SIG
c/o PACS
Box 312
La Salle University
Philadelphia, PA 19141

rutgers!{bpa,cbmvax}!
temvax!pacsbb!{gbaun,whutchi}

TX - Dallas/Fort Worth:

Dallas/Fort Worth UNIX Users Group
Seny Systems, Inc.
5327 N. Central, #320
Dallas, TX 75205

Jim Hummel (214) 522-2324

TX - San Antonio: the San Antonio UNIX Users (SATUU) meets the 3rd Thursday of each month.

Jeff Mason (512) 494-9336
Hewlett Packard
14100 San Pedro
San Antonio, TX 78232

gatech!petro!hpsatb!jeff

WA - Seattle: meets monthly.

Bill Campbell (206) 232-4164
Seattle UNIX Group Membership Information
6641 East Mercer Way
Mercer Island, WA 98040

uw-beaver!tikal!camco!bill

Washington, D.C.: meets the 1st Tuesday of each month.

Washington Area UNIX Users Group
2070 Chain Bridge Road, Suite 333
Vienna, VA 22180

Samuel Samalin (703) 448-1908

EUROPEAN UNIX® SYSTEMS USER GROUP NEWSLETTER

Volume 8, Number 2
Summer 1988

The JUNET Environment

Jun Murai
jun@utokyo-relay.csnet

University of Tokyo
2-11-16, Yayoi, Bunkyo
Tokyo, 113 JAPAN

JUNET has been developed in order to provide a testing environment for studies of computer networking and distributed processing by connecting a large number of computers and by providing actual services for users. The environment provided by the network represents the special requirements and problems of Japanese UNIX environment in general. Throughout the development stages of the JUNET environment, mechanisms to manage resource naming, Japanese character handling, and fast dial-up link using IP protocol have been developed for the network.

In this paper[1], the current status of JUNET focusing on the special environments and technologies to provide them are introduced.

Introduction

JUNET[1] has been developed as the first attempt to establish an electronic communication environment in Japanese research and development communities. Text message exchanges such as electronic mails and electronic news have been provided as well as other advanced network services. Several international links to world academic networks have also been established. Since this is the only working environment for experimental studies on computer networking in Japan, various research topics including physical communication technologies, network protocols, network interconnections and distributed processing environments are in progress.

The purpose in the very first stage of the network, thus, was to provide an actual network services to researchers and put Japanese communities into worldwide academic networks. And then, we started to work to solve problems existing on the network such as Japanese character handling, name handling for distributed resources, and communication technologies.

JUNET started its operation in October 1984 connecting local area networks in major Japanese universities in Tokyo area[2] and it provides users with the means of the worldwide communications via various international links[3].

The domain addressing over UUCPNET[4] was introduced on May 1985 with a system to generate the address conversion sendmail[5] rules. High speed dial-up modes have been studied in order to increase the transmission rate of the communication. To achieve this purpose, UUCP enhancement with kernel driver development were done, and the efficiency using a dial-up line increased up to more than 13Kbps. This encouraged us to migrate to TCP/IP protocol suite[6][7] even on dial-up lines as well as leased lines. As the result of the development, the dial-up IP link is providing as high as 8Kbps in end-to-end transmission. With general IP transmission over high-speed leased lines and over X.25 public packet switching network, this variety of links introduced us with immediate needs of advanced routing mechanisms which is one of our major field of studies at this point.

As for the internetworking between JUNET and other academic networks is concerned, two gateways are operating to exchanging electronic mails and news. One is Kokusai Denshin Denwa Co. Ltd. (KDD), an international telephone and telegraph company which serves a gateway between UUCPNET/USENET and JUNET, and University of Tokyo is providing a gateway function between CSNET[8] and JUNET which is the major path for most of the other academic networks in the world.

1. This paper was delivered at the EUUG Spring conference but was not submitted in time to be printed in the proceedings.

Generally, there are strong demands for Japanese character handling on computing environment and thus support of Japanese characters by means of computer communications are one of the primary characteristics of JUNET. In order to provide the functions, a JUNET standard *Kanji* code was chosen and conversions between the network standard *Kanji* code and operating system *Kanji* codes are provided by the network application available for JUNET. The general computing environment for Japanese character handling were established as well in order to cooperate with the network environment. The statistics introduced in this paper show drastic influences of Japanese character handling in computer network environment in Japan.

Profile of JUNET

JUNET currently connects more than 1300 hosts in 130 organisations. The geographic areas covered by the network expand from Hokkaido, the northern island, to Kyushu, the southern island, however, concentrations in Tokyo and Osaka areas are obvious as shown in Figure 1. Most of the links have been dial-up lines using 1200 bps or 2400bps modems, although special mechanisms have developed for the UNIX operating system and its communication software UUCP in order to use high speed dial-up modems with 9600bps or higher transmission rate. These mechanisms are also effective for TCP/IP protocols over dial-up telephone lines.

Organisations connecting to the network are Universities as listed in Table 1, as well as research laboratories of computer software/hardware companies, and research laboratories of telephone companies whose domain names are listed in Table 2. All the functions to operate the network have been administrated by administrators at each of the institutes in totally volunteer basis. Table 1: Second level domains for universities in JUNET (Apr. 1988)

Size of JUNET

The size of JUNET can be examined by various statistics. Among them, the constant growth in the number of organisations on the network is remarkable as shown in Figure 2.

One or two new organisations are being connected to JUNET every week on average.

News articles are posted to the network constantly in the `fj` news groups which are currently distributed only within Japan[2] The number of articles posted has been growing as in Figure 3, and 1807 articles are posted in October 1987, as the latest example. Note that only about 15 percent of the articles are written in English alphabet including articles in Romaji, an

Table 1: Second Level Domains for Universities in JUNET (Apr 1988)

aoyama	Aoyama Gakuin Univ.
chuo-u	Chuo Univ.
fit	Fukuoka Inst. of Tech.
fukuoka-u	Fukuoka Univ.
gunma-u	Gunma Univ.
hokudai	Hokkaido Univ.
kansai-u	Kansai Univ.
keio	Keio Univ.
kit	Kyoto Inst. of Tech.
kobe-u	Kobe Univ.
konan-u	Konan Univ.
kyoto-su	Kyoto Sangyou Univ.
kyoto-u	
kyushu-u	Kyushu Univ.
kyutech	Kyushu Inst. of Tech.
nagano	Nagano Univ.
nagoya-u	Nagoya Univ.
oita-u	Oita Univ.
osaka-u	Osaka Univ.
osakac	Osaka Elec. & Comm. Univ.
seikei	Seikei Univ.
sheart	Univ. of the Sacred Heart
shinshu-u	Shinshu Univ.
shizujoka	Shizukoka Univ.
sophia	Sophia Univ.
titech	Tokyo Insti. of Tech.
tohoku	Tohoku Univ.
tohoku-u	Tohoku Univ. (Computer Center)
tokuyama	Tokuyama National Technical College
toyo	Toyo Univ.
toyota-ti	Toyota Technological Inst.
tsuda	Tsuda College
tsukuba	Tsukuba Univ.
tuat	Tokyo Univ. of Agriculture & Tech.
tut	Toyohashi Univ. of Tech.
u-tokyo	Univ. of Tokyo
uec	Univ. of Electro-Comm.
ulis	Univ. of Library & Info. Sci.
waseda	Waseda Univ.
yamagata-u	Yamagata Univ
yamanashi	Yamanashi Univ.

2. Part of the `fj` news groups are distributed outside the country as requested.

Table 2: Second level domains other than universities in JUNET		
adin	asahi	ascii
asp	asr	astd
astec	atr	att-j
cac	canon	canopus
casio	cec	citoh
crl	csk	dcl
dec-j	decjrd	denken
dit	dnp	edr
etl	firmware	foretune
fujitsu	fujixerox	gctech
hitachi	hst	ibmtrl
icm	icot	ipa
jip	jsd	jus
jusoft	k3	kaba
kajima	kcs	kddlabs
kiic	kk	kubota
kyocera	m-giken	m-tsrd
matsubo	meiosk	melco
minpaku	mita	mri
msr	nacsis	ncc
ndg	nec	nig
nts	ntt	oki
omron	pentel	recruit
ricoh	riken	roland
rtri	sanyo	seclab
sharp	sigma	sony
sonytek	soum	sra
sumikin	sun-j	toshiba
tytlabs	uclosk	unisys
yamaha	yhp	ysc

alphabetical representation of Japanese language, however, even among these articles most of them are re-posted messages from various mailing-lists. The major JUNET sites handle about 22 M bytes of articles in a month; 4 M bytes of `fj` news groups and 18 M bytes of USENET news groups.

One of the systems in the JUNET backbone handles about 200 M bytes of information in a month, and 85 percent of them are for the network news and 15 percent are for electronic mails. Since the network news are compressed to half their size before the actual transmission, about 370 M bytes of text information is handled in one of the most busy systems in JUNET.

The international information exchanges are served by two gateways of JUNET; one is in University of Tokyo and named `ccut.cc.u-tokyo.junet`. Another is `kddlab.kddlabs.junet` of KDD laboratories. The `ccut.cc.u-tokyo.junet` is on CSNET as `utokyo-relay` and is operated as the JUNET-CSNET gateway, on the other hand, the `kddlab.kddlabs.junet` is widely known as one of the backbone sites in the UUCPNET and is operated as the JUNET-UUCPNET/USENET gateway.

The amount of international messages can be estimated by the total traffic at these two gateways. The example traffic of June 1987 is shown in Table 3.

Table 3: International Message Traffics		
gateway	mail	news
`kddlabs.junet`	10 MB	18MB
`u-tokyo.junet`	13MB	0
total	23MB	18MB

In summary, JUNET currently has approximately 41 M bytes of international traffic in a month, and it has been increasing about 3 M bytes per month in the last six months.

Figure 1: JUNET Geographical Map

JUNET Domain Addressing

In the hierarchy of JUNET domain structure, a domain called 'junet' is the top domain, although we are now preparing to employ ISO's country code (ISO 3116)[9] for Japan 'jp' as the top domain name[10]. The second level domains are called sub-domains, and each of them represents a name of an institute or an organisation. Lower level domains than the sub-domains may be determined at each of the sub-domains. In any cases, the lowest level domains are the names of hosts. The names of sub-domains usually are names well known to the

society, but such names sometimes differ in intra/inter-national environment.

Therefore, one or more names can be registered as synonyms for a sub-domain name to help users to address with general knowledge on the name of organisations. A name of a resource, thus, is defined in one of the domains.

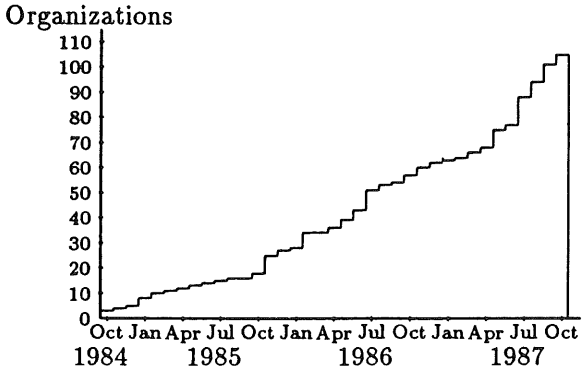


Figure 2: Number of domains

There is one of the distributed name server in each of the domains which handles definitions and deletions of names using a database dedicated to that domain. A name server of a domain thus has a database to define names of lower level domains adjacent to the domain, or names of resources, such as names of mailboxes. The information held by each of the distributed name server is used in retrieving information of resource names and in accessing resources. By this concept, a resource can be defined in a logical domain; a mailbox can be defined even in the top domain, `'junet'`. This provides a name space which is well-matched to the naming concept of the real world yet providing consistency and efficiency of operations.

Design of JUNET message delivery system

There exists a name server for each domain where a distributed resource name can be defined. There is at least one name server in every host; a name server for a domain representing that host. Other than that one, name servers which represent domains located along a path, from the root to this system in the domain tree, can exist in this system. Thus, name servers for the logical domains are managed by entities which are executed in a distributed manner.

A message delivery system using the above concept is implemented using sendmail[3] whose rules are generated by a rule generating system which plays a role of name servers of the JUNET naming concept.

Since the production rules of the sendmail system are different from site to site, the rules have to be generated at each site. To keep the consistency in the rules over JUNET sites, a configuration system to generate the necessary rules was designed and implemented. The configuration system to construct a name server receives information about domain names and about connections for communication among the name server. Then it generates sendmail rule as its output.

The domain database contains relations between the physical connections to the neighbour sites and the domain names which should be solved at that system. Other than the sendmail, the *rmail* command which receives messages through UUCP links was modified to handle JUNET addresses efficiently.

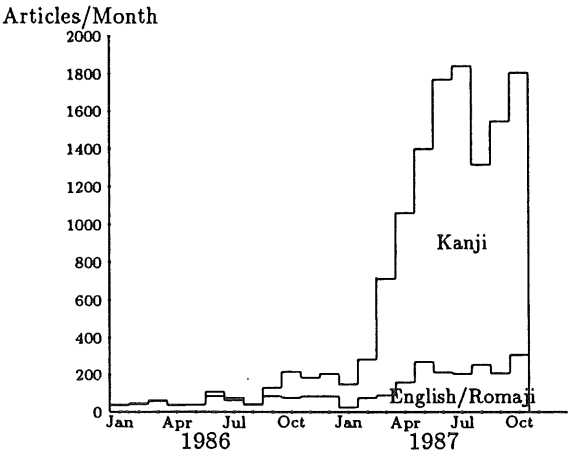


Figure 3: Number of Articles in `fj` newsgroups

International Information Exchange

As described in this paper, the number of messages exchanged in JUNET has increased rapidly, and the number of messages exchanged internationally via the two gateways has also increased as well. Users of other networks, however, sometimes complain about the insufficient information on world networks generated from JUNET. The primary reason for the complaints is obviously caused by the preference of Japanese characters with JUNET users.

The development of general purpose software to handle Japanese characters in non-special hardware environment as well as development of the hand-made *Kanji* fonts encourages us to distribute JUNET *Kanji* messages to other countries. The experimental delivery of the domestic news groups abroad was started to some universities in 1986.

On the other hand, submission of news articles from the USENET environment to JUNET can be achieved

by adding a news group called `fj.misc` to the news group list of an article. This is efficient because all of the JUNET sites are subscribing `fj` news groups while some sites are not subscribing USENET news groups. The news article posted this way is handled as a JUNET news article inside Japan.

It is known that the addresses whose top domains are 'junet' are properly handled at both `relay.cs.net` and `uunet.uu.net`. Therefore,

- *user%domain*`.junet@relay.cs.net`
- *user%domain*`.junet@uunet.uu.net`

are the most popular style of addresses from the ARPA Internet name space.

However, both international links are restricted links; mail to/from a JUNET user who does not register the addresses to the gateway cannot be served. This is due to the cost of the international message exchanges. Furthermore, the mails to/from the non-university users in JUNET cannot be passed through the link between `ccut.cc.u-tokyo.junet` and `relay.cs.net`. Therefore, you should check that your friends in JUNET are registered in the gateways before sending them mail. Any questions should be mailed to the JUNET administrators: `junet-admin@junet`.

Domestic Language Support

One of the primary philosophies of JUNET developments is the pursuit of the better computer communications. For that purpose, the native language supports on the network environment has been chosen as one of the goals. The statistics in Figure 3 show that most of the news articles are written in Japanese characters, or in *Kanji* codes. It is observed that a large amount of electronic mail is also exchanged using *Kanji* codes. Therefore, one of the remarkable characteristics of JUNET among academic networks in the world is *Kanji* message handling for text message exchanges. In order to discuss this topic, some basic concepts about Japanese character handling have to be discussed.

Kanji code

The history of *Kanji* code handling in computers is rather short, and there is still some confusion about the *Kanji* code itself. That is to say, several different '*Kanji* code standards' are actually used in the computer world. However, all the 'standards' refer to a single standard defined by JIS (Japanese Industrial Standard) X0208 as their way of defining a set of *Kanji* characters. JIS is always referred to because it can be introduced from any codes defined by International Standard Organisation using ISO 2022 extension guidelines. It defines as *Kanji* code by two 7-bit bytes without using the most significant bits (MSBs). Several different computer *Kanji* codes exist because the switching method defined by ISO 2022 is not practical for random access to text messages.

The JIS X0208 defines not only *Kanji* characters, but also English alphabets, digits, two types of 50 *Kana* characters (phonetic representation of the Japanese language), and special characters. Among them, *Kanji* characters defined by JIS X0208 are divided into two separate groups; one is called level one and another is called level two. The level one includes about 3500 *Kanji* characters and this set provides a sufficient number of *Kanji* characters for most ordinary texts such as technical writings. Although more complicated *Kanji* characters are needed for advanced text applications such text that includes names of people and places, or for literary text. For the purposes such as these, JIS defines a level two *Kanji* character set which includes another 3400 characters.

In total, about 7000 *Kanji* characters are necessary for providing Japanese character capability on a computer. Obviously, this needs more than one byte to represent it, and representing a single character with two bytes is a standard concept. The important issue here is that we still need to use ASCII codes in computers as well as *Kanji* codes. This means we have to establish a way to handle a mixture of ASCII codes and *Kanji* codes. In order to solve this problem, there has to be a way to distinguish ASCII code sequences from *Kanji* code sequences.

There are three major methods which can be used to distinguish *Kanji* sequences from ASCII sequences representating English alphabets and special characters:

1. JIS X0208 codes with JIS X0202 (ISO2022)
2. Extended UNIX Code (EUC)
3. Microsoft *Kanji* Code (Shift-JIS)

Method 1 works by a surrounding a *Kanji* sequence by a designating escape sequence (`ESC-$-@` or `ESC-$-B`) and a sequence of English alphabetics be a designating escape sequence (`ESC-(-J` or `ESC-(-B`).

2. was originally defined by AT&T UNIX Pacific to provide an internationalised version of UNIX

operating system[11] and is becoming to be a standard way to representing *Kanji* codes in the UNIX operating system in Japan. In the EUC, the MSB's of both bytes in a single *Kanji* character are set to 1 whereas the MSB is cleared in an ASCII character..

3. was originally defined for CP/M on personal computers by a Japanese subsidiary of Microsoft Corporation. This is now a *de-facto* standard for personal computers and is also supported in some of the Japanised UNIX environments. In the Microsoft *Kanji* codes, *Kanji* characters are mapped by some function so that first byte is in ranges of `0x81` — `0x9f` and `0xe0` — `0xff`.

In order to cooperate with the ISO standard method of handling character codes, and to utilise existing software which sometimes strips the MSBs off, we decided to use JIS X0208 two 7-bit codes surrounded with escape sequences as the network standard. This decision requires conversion functions from JIS X0208 to local operating system *Kanji* codes, namely EUC/JAPAN and Microsoft *Kanji* code.

Kanji code handling

In order to design network applications using *Kanji* codes, the following discussion has to be made regarding the average environments of existing JUNET systems.

1. There are several kinds of character codes including above codes which are actually used in operating systems as internal codes to represent Japanese characters including *Kanji* characters. Among them, JIS X0208 is not practical for internal code because of the complicated operations of switching modes when seeking a character in a byte sequence; random access to a sequence of byte is impossible. Thus, two 8-bit codes without escape sequence is preferable to two 7-bit codes with escape sequences as internal code for an operating system.
2. Since we have decided to us JIS X0208 as the JUNET network standard *Kanji* code, we have to provide conversion mechanisms from JIS X0208 to internal code of operating systems such as EUC and to Microsoft *Kanji* code.

JUNET approaches

By assuming the above issues, we have developed the following environment for JUNET:

Kanji code As we have described before, we are using JIS X0208 with ISO 2022 / JIS X0202 extension guide lines as a network standard *Kanji* code. Escape sequence introducing JIS X 0208 can either be `ESC-$-@` or `ESC-$-B`. These are two definitions of two minor versions of JIS X0208 and both are legal in a sense of standard definition. Escape sequence introducing ASCII code is `ESC-(-B` and introducing ROMAN character code of JIS is `ESC-(-J`. Only a few characters are different between the ASCII and ROMAN character sets: \ and ~ in ASCII are replaced by ¥ and ‾ in ROMAN respectively. We therefore treat both of them equally. Note that the default code set for JUNET text message is ASCII code; there is no introducing escape sequence necessary if a text starts with a ASCII code.

Control characters According to the ISO 2022 / JIS X0202, any characters appear in both *Kanji* sequence and ASCII sequence, however, deep backtracking maybe necessary to determine the character mode when a file is accessed randomly. So we decided not to allow control characters appear within *Kanji* code sequence; they can only appear in ASCII code sequence. This rule contributes to make software which handles text messages to be simple and transparent in terms of internationalising because a function to find a control character can transparently be defined.

Single byte Kana code There are another code set for representing Japanese character in a single byte; JIS X0201. In this code, only *Kana* characters are represented. This code set used to be used in mainframes because Japanising of software was easy. Since we now have *Kanji* code set anyway and all the *Kana* representations are included ijn JIS X 0208 using 2 bytes, we eliminate usage of JIS X 0201 to avoid the complexity caused from handling of three different code sets at a time.

Network Software In order to provide an environment for Japanese text message capabilities in JUNET, we have modified the following software to adopt the above strategies:

- Bnews[12] was modified to pass the escape sequences which used to be stripped off in the original version. In the standard implementation, the articles spooled are still represented in network code and the conversion functions to major internal *Kanji* codes are added. Other network news

interfaces such as rn and vn have been modified, too.

- MH[13] has been added the code conversion facility between the network standard code and the local code.
- GNUemacs and MicroEmacs were modified to handle *Kanji* codes so that we can edit Japanese mail and news articles.
- X Window System was also modified to represent *Kanji* characters on X.V10R3 and X.V10R4. Unfortunately, there were several different approaches to the modifications and offered us no interoperability. In order to establish an interoperability for X.V11. a group of researchers is organised with a mailing list in JUNET[14] for the purpose, and the recent discussions on the list have been very active. The actual work on X.V11 has been completed and is now in the distribution tape of X Window System as a contributed software.

  Level one *Kanji* fonts have been *hand-made* and posted to JUNET so that a user can read and write *Kanji* characters without special *Kanji* terminals.

An example to show the example JUNET environment of Japanese character handling is shown in Figure 4.

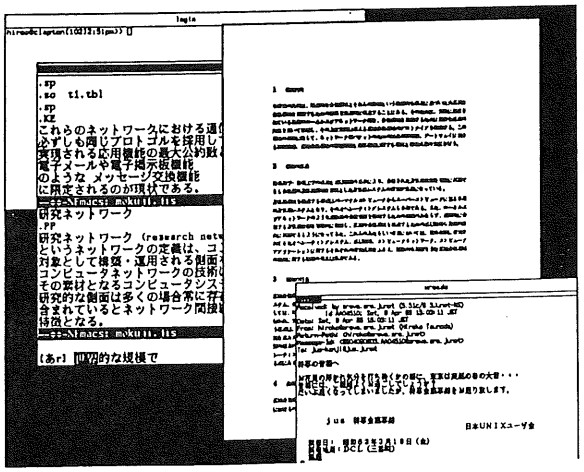


Figure 4: Example of JUNET Kanji Environment

Dial-up Links

Fast UUCP Links

The Primary goal of the development of JUNET as the first step was to construct a network providing functions for text message exchange over research communities in Japan. Since JUNET is a volunteer project to provide basis of researches on network communication, UUCP protocol over dial-up links was our choice to start the network because of the popular, easy to set-up and inexpensive features of UUCP technology. The dial-up modems used have been upgraded from V.21 300bps to V.22bis 2400bps and we have almost succeeded in eliminating the V.21 and V.22 1200bps modems.

However, UUCP over 2400bps communication links is not practical as the size of the network has grown and traffic has increased. One of the problems is its slow speed, and it can be reduced by using the high speed modems such as Telebit's Trailblazer or Microcom's AX/9624c. In order to make use of those high speed modems, the flow control mechanism between the modems and the hosts is required and f-protocol UUCP is well known for Xon/Xoff flow control which is one of the popular flow control mechanisms. It is good for X.25 links but is not efficient for the high speed modems because f-protocol UUCP does 7-bit encoding. Instead of f-protocol UUCP, j-protocol UUCP which assumes 8-bit transparent link with Xon/Xoff flow control has been developed. To reduce the overhead in the tty device driver, a new line discipline called `UTTYDISC` which handles only flow control has also been developed. With the combination of j-protocol UUCP and `UTTYDISC`, we can get more than 13Kbps of transmission rate using TrailBlazers or AX/9624c's. The advantage of new UUCP is not only its high performance but also reduction of the tty port occupation.

Dial-up Links — Motivation

The UUCP link is intended for file transfer in batch mode and the file systems at the destination machines are liable to overflow. This problem is serious; when a file system overflows, many new articles and even mails are lost. Therefore, reliable datalink protocols which replace UUCP efficiently are preferable.

As JUNET is a volunteer based network and as we do not have enough funds to get dedicated lines between organisations, we have to make use of the dial-up lines at least for several months. X.25 packet switching network is one of the candidates. We can pass the IP datagrams over X.25 links as defined in [15], however, it is not practical for the small organisations and for the newly coming organisations because of its cost. Another candidate is to pass the IP datagrams over the dial-up links.

Providing this facility, we can make the network more reliable without any additional hardware. We have developed a serial KIP module for dial-up links, called DL, derived from Rick Adams' SLIP.

Dial-up IP Link — Structure

IP connections over a dial-up link have been usually considered for personal computers at home[16], however, they are still effective for easy construction of an internetworking among universities, especially on the achieved performance with the high-speed modems. The system for dial-up IP link thus was designed and developed on Sun OS version 3.2 which is compatible with 4.2BSD UNIX.

The system consists of three modules. The *dldriver* resides in the kernel and it receives and sends IP datagrams with the IP module which also exists in the kernel space. It has communication entities as a 'special file' in the UNIX file space called `/dev/dlx`.

The major portion of the system, *dldaemon*, is a permanent user process which reads information from the *dldriver* and controls it through `/dev/dlz`. The *dldaemon* is invoked at a boot time and resides on the system permanently. As soon as the process is initiated, it registers a network interface (*if*) called `dlz` and sets up the routing information in the kernel space so that the IP module can pass IP fragments with addresses routed toward the link to the *dldriver*.

The *dldriver* includes a tty line discipline called `DLIPDISC`, which does character mapping to escape `DC1/DC3` and other control characters. Another module called *dlattach*, which is a user command, is invoked at login time to pass the information about the serial port to the *dldaemon*. Figure 5 illustrates the structure of the system.

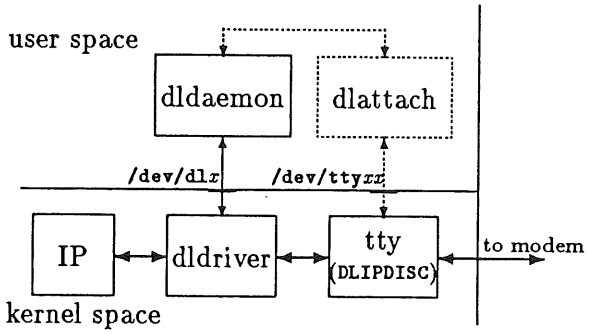


Figure 5: Structure of the DL

A connection established with the `DL` is a point-to-point connection between a master system which initiates the connection and a slave system. The following example of sequence shows an overview of the procedure and the functions of the system:

1. When an `IP` fragment is selected to the `dlz` interface by the routing function in the `IP` module, the *dldriver* passes the fragment to the attached tty line if the link has already been established. Otherwise, the *dldriver* requests the *dldaemon* to initiate dialing to the remote site.
2. The master *dldaemon* logs-in to the slave site and invokes the *dlattach* command.
3. The *dlattach* provides the *dldaemon* residing on the slave site with the information about the serial port and about the caller's name, as shown in the dotted line of Figure 5.
4. Then, each of the *dldaemons* at the both systems issues a system call to switch a line discipline to `DLIPDISC`. Finally, they complete establishment of a dial-up link ready for IP communication between the systems.
5. A connection is terminated by the *dldaemon* when it detects that 60 seconds has passed without any IP fragment transmissions.

The measured performance of the system with two TrailBlazers is approximately 7.5Kbps and two AX/9624c is approximately 8Kbps for end-to-end data transmission by ftp. Furthermore, we can obtain the performance of more than 10Kbps with an UDP-based file transfer protocol whose window size is large enough to cover the delay originated from the internal protocol of the modems.

Discussions

The DL have the following characteristics compared to other media:

1. Link is usually dead and activated by requests.
2. Dialing takes 20 — 60 seconds. The first datagram takes this long delay before reaching the destination.
3. Link speed is not so high. The propagation delay is relatively large because of the modems' internal protocol.
4. Link cost depends on the connection time rather than the amount of the traffic.

DL as a replacement of UUCP

The primary usage of DL is a replacement of UUCP. The messages are transferred in batched manner, however, each IP fragment is carried over the DL as in the dedicated lines. More precisely, *uucp* and *uux* replacements receive the file transfer requests and put the data once onto the spool directory. When the connection is successfully established, the processes which talk to the remote daemon with SMTP or NNTP are invoked. We have modified version of *nntpd* and a client process named *sendnntp*, and we can prevent the file system from overflowing.

The timing when the dial operation is initiated may be determined by an evaluation function depending on the amount of the data spooled for a particular destination, the elapsed time since last connection, and so on. The *dldaemon* executes such a function periodically, and determines whether it should initiate dialing.

Advanced usage of DL

For the DL as a replacement of UUCP, the existence of the link do not have be to inform to other sites. But the processes on the host other than the gateways want to send/receive the IP datagrams while the link is established. For this requirement, the gateways should propagate the routing information when the DL is established or closed.

Unlike the permanent network links, the DL is usually closed and initiated by the arrival of the IP datagram to the remote network. This means that the DL should be handled as 'active' in the routing information even the actual connection is not established. A process on a site other than gateway can issue a TCP connection request according to 'active' state in the routing information, however, it may time out because the dial operation may take 20-60 seconds. In such a case, the DL connections is established, while the TCP has been timed out and no actual data transmission is carried out.

We use the high speed modems for the DL to obtain good performance, however, those modems have relatively large delay time and we cannot expect the maximum performance with the ordinal TCP window size. Using a file transfer protocol with large window size over UDP, the performance of the DL exceeds 10Kbps, whereas 8Kbps with ftp.

In order to solve these problems and achieve the best use of the DL, a new routing procedure and congestion control technique suitable for the DL is now developing. The routing information should contain whether the path to a destination includes the DL or not. The time out mechanism and the window size determination mechanism in the TCP module are affected to such a information in the routing information. Apart from the efficiency problems, the researches on the security issue should also be worked to use the DL practically.

Conclusion

Development of JUNET started in October 1984. Since then various researches on computer networking and distributed environment have been actively done as the rapid growth in the size of the network. Among them, name management functions to construct a hierarchical domain name space, Japanese character handling, and communication technologies using the high-speed modems have been focused as primary research concerns.

The actual work for the addressing and routing of JUNET text messages is achieved by a name server concept and its implementation. This system receives control messages to specify the information about logical domain name, connections and methods to deliver messages, and generates a sendmail rule set. This software provides an environment where the logical naming definitions and physical routing issues are clearly separated so that reliability, efficiency, extensibility and flexibility of communication in the network are simultaneously achieved.

Internationalisation of computer software is one of the most important issues in JUNET and its communication software. The clear separation of network *Kanji* code and internal operating system code employed in JUNET software provides a transparent environment on Japanese character handling in computer networks. Availability of Japenese messages obviously encourages JUNET users very much to exchange messages over the network.

A new UUCP protocol and tty driver enhancement are developed for the requirements of higher transmission rate over the dial-up lines. As the result, more than 13Kbps UUCP transmission rate is achieved. This encouraged us to migrate onto TCP/IP suite using dial-up lines as well as using leased lines. Performance of the implementation of the dial-up IP link is about 8Kbps which is practical enough to construct a distributed environment over a widely interconnected network environment.

Progresses of JUNET technologies discussed in this paper lead us to many topics of future studies such

as:

- Enhancement of name servers which handles general distributed resources.
- Establishment of gateway technologies such as optimal routing strategies based upon constructions of IP-based network using the dial-up IP link and IP over leased lines.
- Supports of multi-media message exchanges enhances the environment with multi-language supports currently achieved.

The technologies developed in JUNET can generally be used to construct a network interconnection with inexpensive cost.

Acknowledgements

The authors would like to thank Youichi Shinoda, Keisuke Tanaka and Hiroshi Tachibana for their efforts in developing software. The dial-up IP link was achieved as a result of discussions with members of the JUNET project, especially with Susumu Sano.

References

[1] Jun Murai and Akira Kato. Researches in Network Development of JUNET. In *Proceedings of SIGCOMM '87 Workshop*, ACM, 1987.

[2] J. Murai and T. Asami. A network for research and development communications in Japan — JUNET —. In *Proceedings of First Pacific Computer Communications Symposium*, 1985.

[3] J.S. Quaterman and J.C. Hoskins. Notable Computer Networks. *CACM*, 29(10), October 1986.

[4] D.A. Nowitz and M.E. Lesk. *A Dial-Up Network of UNIX Systems*. Technical Report, Bell Telephone Laboratories, August 1987.

[5] Elic Allman. Sendmail — An Interconnecting Mail Rerouter, Version 4.2. In *UNIX Programmer's Manual, 4.2 Berkeley Software Distribution*, Univ. of California, Berkeley, 1983.

[6] J. Postel (ed.). Internet Protocol — DARPA Internet Program Protocol Specification. RFC 791. 1980.

[7] J. Postel (ed.). Transmission Control Protocol — DARPA cInternet Program Protocol Specification. RFC 793. 1981.

[8] D. Comer. The computer science research network CSNET: A history and status report. *CACM*, 26(10), October 1983.

[9] ISO. Codes for the Representation of Names of Countries. ISO 3116. 1981.

[10] J. Postel. Domain requirements. RFC 920. 1984.

[11] Hiromichi Kogure and Richard McGowan. A UNIX System V STREAMS TTY Implementation for Multiple Language Processing. In *USENIX Summer Conference Processings*, USENIX, 1987.

[12] S.L. Emerson. USENET: A bulletin board for UNIX users. *BYTE*, September 1983.

[13] R.S. Gaines, S. Borden and N.Z. Shapiro. *The MH Message Handling System: User's Manual*. Rand Corporation, 1979.

[14] Hiroshi Tachibana. PD *Kanji* font (tools and ascii fonts). Network news posted to `fj.sources` as `<1676@rika.cs.titech.JUNET>`, July 1987.

[15] J.T. Korb. A Standard for the transmission of IP Datagrams Over Public Data Networks. RFC 877. 1983.

[16] D.J. Farber, G.S. Delp and T.M. Conte. A Thinwire Protocol for connecting personal computers to the INETERNET. RFC 914. 1984.

Cake: a Fifth Generation Version of make

Zoltan Somogyi
UUCP: {uunet,mcvax,ukc}!munnari.oz!zs
zs@mulga.oz.au, zs@mulga.uucp
ARPA: zs%munnari.oz@uunet.uu.net
CSNET: zs%munnari.oz@australia

Department of Computer Science
University of Melbourne
Parkville, 3052 Victoria, Australia

Zoltan is a grad stude hacker droid who haunts the night corridors of the University of Melbourne. He is interested in Life, the Universe, and Everything. His main research interest is in parallel logic programming, but he sometimes cobbles together useful software in his spare time.

Zoltan lives in Australia.

Abstract

Make is a standard UNIX utility for maintaining computer programs. Cake is a rewrite of make from the ground up. The main difference is one of attitude: cake is considerably more general and flexible, and can be extended and customised to a much greater extent. It is applicable to a wide range of domains, not just program development.

Introduction

The UNIX utility make (Feldman, '79) was written to automate the compilation and recompilation of C programs. People have found make so successful in this domain that they do not wish to be without its services even when they are working in other domains. Since make was not designed with these domains in mind (some of which, e.g., VLSI design, did not even exist when make was written), this causes problems and complaints. Nevertheless, implied in these complaints is an enormous compliment to the designers of make; one does not hear many grumbles about programs with only a few users.

The version of make described in (Feldman, '79) is the standard utility. AT&T modified it in several respects for distribution with System V under the name augmented make (AT&T, '84). We know of two complete rewrites: enhanced make (Hirgelt, '83) and fourth generation make (Fowler, '85). All these versions remain oriented towards program maintenance[1].

Here at Melbourne we wanted something we could use for text processing. We had access only to standard make and spent a lot of time wrestling with makefiles that kept on getting bigger and bigger. For a while we thought about modifying the make source, but then decided to write something completely new. The basic problem was the

1. Since this paper was written, two other rewrites have come along: mk (Hume, '87) and nmake.

inflexibility of make's search algorithm, and this algorithm is too embedded in the make source to be changed easily.

The name cake is an historical accident. Cake follows two other programs whose names were also puns on make. One was bake, a variant of make with built-in rules for VLSI designs instead of C programs (Gedye, '84). The other was David Morley's shell script fake. Written at a time when disc space on our machine was extremely scarce, and full file systems frequently caused write failures, it copied the contents of a directory to /tmp and invoked make there.

The structure of the paper is as follows. Section 2 shows how cake solves the main problems with make, while section 3 describes the most important new features of cake. The topics of section 4 are portability and efficiency. The paper assumes that you have some knowledge of make.

The problems with make

ake has three principal problems. These are:

1. It supports only suffix-based rules.
2. Its search algorithm is not flexible enough.
3. It has no provisions for the sharing of new make rules.

These problems are built deep into make. To solve them we had to start again from scratch. We had to abandon backward compatibility because the make syntax is not rich enough to represent the complex relationships among the components of large systems. Nevertheless, the cake user interface is deliberately based on make's; this helps users to transfer their skills from make to cake. The *functionalities* of the two systems are sufficiently different that the risk of confusion is minimal[2].

Probably the biggest single difference between make and cake lies in their general attitudes. Make is focused on one domain: the maintenance of compiled programs. It has a lot of code specific to this domain (especially the later versions). And it crams all its functionality into some tight syntax that treats all sorts of special things (e.g., .SUFFIXES) as if they were files.

Cake, on the other hand, uses different syntax for different things, and keeps the number of its mechanisms to the minimum consistent with generality and flexibility. This attitude throws a lot of the functionality of make over the fence into the provinces of other programs. For example, where make has its own macro processor, cake uses the C preprocessor; and where make has special code to handle archives, cake has a general mechanism that *just happens* to be able to do substantially the same job.

Only suffix-based rules

All entries in a makefile have the same syntax. They do not, however, have the same semantics. The main division is between entries which describe simple dependencies (how to make file a from file b), and those which describe rules (how to make files with suffix .x from files with suffix .y)[3]. Make distinguishes the two cases by treating as a rule any dependency whose target is a concatenation of two suffixes.

For this scheme to work, make must assume three things. The first is that all interesting files have suffixes; the second is that suffixes always begin with a period; the third is that prefixes are not important. All three assumptions are violated in fairly common situations. Standard make cannot express the relationship between file and file.c (executable and source) because of assumption 1, between file and file,v (working file and RCS file) because of assumption 2, and between file.o and ../src/file.c (object and source) because of assumption 3. Enhanced make and fourth generation make have special forms for some of these cases, but these cannot be considered solutions because special forms will always lag behind demand for them (they are embedded in the make source, and are therefore harder to change than even the built-in rules).

Cake's solution is to do away with make-style rules altogether and instead to allow ordinary dependencies to function as rules by permitting them to contain variables. For example, a possible rule for compiling C programs is

2. This problem, called cognitive dissonance, is discussed in Weinberg's delightful book (Weinberg, '71).

3. For the moment we ignore entries whose targets are special entities like .IGNORE, .PRECIOUS, etc.

```
%.o:    %.c
        cc -c %.c
```

where the % is the variable symbol. This rule is actually a *template* for an infinite number of dependencies, each of which is obtained by consistently substituting a string for the variable %.

The way this works is as follows. First, as cake seeks to update a file, it matches the name of that file against all the targets in the description file. This matching process gives values to the variables in the target. These values are then substituted in the rest of the rule[4]. (The matching operation is a form of *unification*, the process at the heart of logic programming; this is the reason for the *fifth generation* bit in the title.)

Cake actually supports 11 variables: % and %0 to %9. A majority of rules in practice have only one variable (canonically called %), and most of the other rules have two (canonically called %1 and %2). These variables are local to their rules. Named variables are therefore not needed, though it would be easy to modify the cake source to allow them.

Example

If cake wanted to update prog.o, it would match prog.o against %.o, substitute prog for % throughout the entry, and then proceed as if the cakefile contained the entry

```
prog.o: prog.c
        cc -c prog.c
```

This arrangement has a number of advantages. One can write

```
%.o:    RCS/%.c,v
        co -u %.c
        cc -c %.c
```

without worrying about the fact that one of the files in the rule was in a different directory and that its suffix started with a nonstandard character. Another advantage is that rules are not restricted to having one source and one target file. This is useful in VLSI, where one frequently needs rules like

```
%.out:  %.in %.circuit
   simulator %.circuit < %.in > %.out
```

and it can also be useful to describe the full consequences of running yacc

```
%.c %.h:  %.y
          yacc -d %.y
          mv y.tab.c %.c
          mv y.tab.h %.h
```

Inflexible search algorithm

In trying to write a makefile for a domain other than program development, the biggest problem one faces is usually make's search algorithm. The basis of this algorithm is a special list of suffixes. When looking for ways to update a target file.x, make searches along this list from left to right. It uses the first suffix .y for which it has a rule .y.x and for which file.y exists.

The problem with this algorithm manifests itself when a problem divides naturally into a number of stages. Suppose that you have two rules .c.b and .b.a, that file.c exists and you want to issue the command make| file.a. Make will tell you that it doesn't know how to make file.a. The problem is that for the suffix .b make has a rule but no file, while for .c it has a file but no rule. Make needs a *transitive rule* .c.a to go direct from file.c to file.a.

The number of transitive rules increases as the square of the number of processing stages. It therefore becomes significant for program development only when one adds processing stages on either side of compilers. Under UNIX, these stages are typically the link editor ld and program generators like yacc and lex. Half of standard make's built-in rules are transitive ones, there to take care of these three programs. Even so, the builtin rules do not form a closure: some rare combinations of suffixes are missing (e.g., there is no rule for going from yacc source to assembler).

For builtin rules a slop factor of two may be acceptable. For rules supplied by the user it is not. A general-purpose makefile for text processing under UNIX needs at least six processing stages to handle nroff/troff and their preprocessors lbl, bib, pic, tbl, and eqn, to mention only the ones in common use at Melbourne University.

Cake's solution is simple: if file1 can be made from file2 but file2 does not exist, cake will try to *create* file2. Perhaps file2 can be made from file3, which can be made from file4, and

4. After this the rule should have no unexpanded variables in it. If it does, cake reports an error, as it has no way of finding out what the values of those variables should be.

so on, until we come to a file which does exist. Cake will give up only when there is *absolutely no way* for it to generate a feasible update path.

Both the standard and later versions of make consider missing files to be out of date. So if file1 depends on file2 which depends on file3, and file2 is missing, then make will remake first file2 and then file1, even if file1 is more recent than file3.

When using yacc, we frequently remove generated sources to prevent duplicate matches when we run egrep ... *.[chyl]. If cake adopted make's approach to missing files, it would do a lot of unnecessary work, running yacc and cc to generate the same parser object again and again[5].

Cake solves this problem by associating dates even with missing files. The *theoretical update time* of an existing file is its modify time 7 given by stat(2)); the theoretical update time of a missing file is the theoretical update time of its youngest ancestor. Suppose the yacc source parser.y is older than the parser object parser.o, and parser.c is missing. Cake will figure that if it recreated parser.c it would get a parser.c which *theoretically* was last modified at the same time as parser.y was, and since parser.o is younger than parser.y, theoretically it is younger than parser.c as well, and therefore up-to-date.

No provisions for sharing rules

Imagine that you have just written a program that would normally be invoked from a make rule, such as a compiler for a new language. You want to make both the program and the make rule widely available. With standard make, you have two choices. You can hand out copies of the rules and get users to include it in their individual makefiles; or you can modify the make source, specifically, the file containing the built-in rules. The first way is error-prone and quite inconvenient (all those rules cluttering up your makefile when you should never need to even look at them). The second way can be impractical; in the development stage because the rules can change frequently and after that because you want to distribute your program to sites that may lack the make source.

5. In this case make is rescued from this unnecessary work by its built-in transitive rules, but as shown above this should not be considered a *general* solution.

And of course two such modifications may conflict with one another.

Logically, your rules belong in a place that is less permanent than the make source but not as transitory as individual makefiles. A library file is such a place. The obvious way to access the contents of library files is with #include, so cake filters every cakefile through the C preprocessor.

Cake relies on this mechanism to the extent of not having *any* built-in rules at all. The standard cake rules live in files in a library directory (usually /usr/lib/cake). Each of these files contains rules about one tool or group of tools. Most user cakefiles #define some macros and then include some of these files. Given that the source for program prog is distributed among prog.c, aux1.c, aux2.c, and parser.y, all of which depend on def.h, the following would be a suitable cakefile:

```
#define MAIN    prog
#define FILES   prog aux1 aux2 parser
#define HDR     def

#include        <Yacc>
#include        <C>
#include        <Main>
```

The standard cakefiles Yacc and C, as might be expected, contain rules that invoke yacc and cc respectively. They also provide some definitions for the standard cakefile Main. This file contains rules about programs in general, and is adaptable to all compiled languages (e.g., it can handle NU-Prolog programs). One entry in Main links the object files together, another prints out all the sources, a third creates a tags file if the language has a command equivalent to ctags, and so on.

Make needs a specialised macro processor; without one it cannot substitute the proper filenames in rule bodies. Fourth generation make has not solved this problem but it still wants the extra functionality of the C preprocessor, so it grinds its makefiles through both macro processors! Cake solves the problem in another way, and can thus rely on the C preprocessor exclusively.

Standard make's macro facilities are quite rudimentary, as admitted by (Feldman, '79). Unfortunately, the C preprocessor is not without flaws either. The most annoying is that the bodies of macro definitions may begin with blanks, and will if the body is separated from the macro name and any

parameters by more than one blank (whether space or tab). Cake is distributed with a fix to this problem in the form of a one-line change to the preprocessor source, but this change probably will not work on all versions of UNIX and definitely will not work for binary-only sites.

The new features of cake

The above solutions to make's problems are useful, but they do not by themselves enable cake to handle new domains. For this cake employs two important new mechanisms: dynamic dependencies and conditional rules.

Dynamic dependencies

In some situations it is not convenient to list in advance the names of the files a target depends on. For example, an object file depends not only on the corresponding source file but also on the header files referenced in the source.

Standard make requires all these dependencies to be declared explicitly in the makefile. Since there can be rather a lot of these, most people either declare that all objects depend on all headers, which is wasteful, or declare a subset of the true dependencies, which is error-prone. A third alternative is to use a program (probably an awk script) to derive the dependencies and edit them into the makefile. (Walden, '84) describes one program that does both these things; there are others. These systems are usually called makedepend or some variation of this name.

The problems with this approach are that it is easy to alter the automatically-derived dependencies by mistake, and that if a new header dependency is added the programmer must remember to run makedepend again. The C preprocessor solves the first problem; the second, however, is the more important one. Its solution must involve scanning though the source file, checking if the programmer omitted to declare a header dependency. So why not use this scan to *find* the header dependencies in the first place?

Cake attacks this point directly by allowing parts of rules to be specified at run-time. A command enclosed in double square brackets[6] may appear in a rule anywhere a filename or a list of filenames may appear. For the example of the C header files, the rule would be

```
%.o:    %.c [[ccincl %.c]]
        cc -c %.c
```

signifying that x.o depends on the files whose names are listed in the output of the command ccincl x.c [7], as well as on x.c. The matching process would convert this rule to

```
x.o:    x.c [[ccincl x.c]]
        cc -c x.c
```

which in turn would be *command expanded* to

```
x.o:    x.c hdr.h
        cc -c x.c
```

if hdr.h were the only header included in x.c.

Command patterns provide replacements for fourth generation make's directory searches and special macros. [[find\ <dirs>\ -name\ <filename>\ -print]] does as good a job as the special-purpose make code in looking up source files scattered among a number of directories. [[basename\ <filename>\ <suffix>]] can do an even better job: make cannot extract the base from the name of an RCS file.

A number of tools intended to be used in just such contexts are distributed together with cake. Ccincl is one. Sub is another: its purpose is to perform substitutions. Its arguments are two patterns and some strings: it matches each string against the first pattern, giving values to its variables; then it applies those values to the second pattern and prints out the result of this substitution. For example, in the example of section 2.3 the cakefile main would invoke the command [[sub\ X\ X.o\ FILES]][8], the value of FILES being prog aux1 aux2 parser, to find that the object files it must link together to create the executable prog are

6. Single square brackets (like most special characters) are meaningful to csh: they denote character classes. However, we are not aware of any legitimate contexts where two square brackets *must* appear together. The order of members in such classes is irrelevant, so if a bracket must be a member of such a class it can be positioned away from the offending boundary (unless the class is a singleton, in which case there is no need for the class in the first place).

7. Ccincl prints out the names of the files that are #included in the file named by its argument. Since ccincl does not evaluate any of the C preprocessor's control lines, it may report a superset of the files actually included.

```
prog.o aux1.o aux2.o parser.o.
```

Cake allows commands to be nested inside one another. For example, the command `[[sub\ X.h\ X\ [[ccincl\ file.c]]]]` would strip the suffix `.h` from the names of the header files included in `file.c` [9].

Conditional rules

Sometimes it is natural to say that `file1` depends on `file2` *if* some condition holds. None of the make variants provide for this, but it was not too hard to incorporate conditional rules into cake.

A cake entry may have a condition associated with it. This condition, which is introduced by the reserved word `if`, is a boolean expression built up with the operators `and`, `or` and `not` from primitive conditions.

The most important primitive is a command enclosed in double curly braces. Whenever cake considers applying this rule, it will execute this command after matching, substitution and command expansion. The condition will return true if the command's exit status is zero. This runs counter to the intuition of C programmers, but it conforms to the UNIX convention of commands returning zero status when no abnormal conditions arise. For example, `{{grep\ xyzzy\ file}}` returns zero (i.e., true) if xyzzy occurs in `file` and nonzero (false) otherwise.

Conceptually, this one primitive is all one needs. However, it has considerable overhead, so cake includes other primitives to handle some special cases. These test whether a filename occurs in a list of filenames, whether a pattern matches another, and whether a file with a given name exists. Three others forms test the internal cake status of targets. This status is `ok` if the file was up-to-date when cake was invoked, `cando` if it wasn't but cake knows how to update it, and `noway` if cake does not know how to update it.

As an example, consider the rule for RCS.

```
%:      RCS/%,v     if exist RCS/%,v
        co -u %
```

Without the condition the rule would apply to all files, even ones which were not controlled by RCS, and even the RCS files themselves: there would be no way to stop the infinite recursion (`%` depends on `RCS/%,v` which depends on `RCS/RCS/%,v,v`...).

Note that conditions are command expanded just like other parts of entries, so it is possible to write

```
%:  archive   if % in [[ar t archive]]
    ar x archive %
```

The implementation

Portability

Cake was developed on a Pyramid 90x under 4.2bsd. At Melbourne University it now runs on a VAX under 4.3bsd, various Sun-3's under SunOS 3.4, an Encore Multimax under Umax 4.2, a Perkin-Elmer 3240 and an ELXSI 6400 under 4.2bsd, and on the same ELXSI under System V. It has not been tested on either System III or version 7.

Cake is written in standard C, with (hopefully) all machine dependencies isolated in the makefile and a header file. In a number of places it uses `#ifdef` to choose between pieces of code appropriate to the AT&T and Berkeley variants of UNIX (e.g., to choose between `time()` and `gettimeofday()`). In fact, the biggest hassle we have encountered in porting cake was caused by the standard header files. Some files had different locations on different machines (`/usr/include` vs. `/usr/include/sys`), and the some versions included other header files (typically `types.h`) while others did not.

As distributed cake is set up to work with `csh`, but it is a simple matter to specify another shell at installation time. (In any case, users may substitute their preferred shell by specifying a few options.) Some of the auxiliary commands are implemented as `csh` scripts, but these are small and it should be trivial to convert them to another shell if necessary.

Efficiency

Fourth generation make has a very effective optimisation system. First, it forks and execs only once. It creates one shell, and thereafter, it pipes commands to be executed to this shell and gets back status information via another pipe. Second, it compiles its `makefiles` into internal

8. Sub uses X as the character denoting variables. It cannot use %, as all %'s in the command will have been substituted for by cake by the time sub is invoked.

9. As the outputs of commands are substituted for the commands themselves, cake takes care not to scan the new text, lest it find new double square brackets and go into an infinite loop.

form, avoiding parsing except when the compiled version is out of date with respect to the master.

The first of these optimisations is an absolute winner. `Cake` does not have it for the simple reason that it requires a shell which can transmit status information back to its parent process, and we don't have access to one (this feature is provided by neither of the standard shells, `sh` and `csh`).

`Cake` could possibly make use of the second optimisation. It would involve keeping track of the files the C preprocessor includes, so that the `makefile` can be recompiled if one of them changes; this must be done by fourth generation make as well though (Fowler, '85) does not mention it. However, the idea is not as big a win for `cake` as it is for `make`. The reason is as follows.

The basic motivations for using `cake` rather than `make` is that it allows one to express more complex dependencies. This implies a bigger system, with more and slower commands than the ones `make` usually deals with. The times taken by `cake` and the preprocessor are insignificant when compared to the time taken by the programs it most often invokes at Melbourne. These programs, `ditroff` and `nc` (the NU-Prolog compiler that is itself written in NU-Prolog), are notorious CPU hogs.

Here are some statistics to back up this argument. The *overhead ratio* is given by the formula

$$\frac{\textit{cake process system time} + \textit{children user time} + \textit{children system time}}{\textit{cake process user time}}$$

This is justifiable given that the `cake` implementor has direct control only over the denominator; the kernel and the user's commands impose a lower limit on the numerator.

We have collected statistics on every `cake` run on two machines at Melbourne, mulga and munmurra[10]. These statistics show that the overhead ration on mulga is 11 while on munmurra it is 86. This suggests that the best way to lower total CPU time is not to tune `cake` itself but to reduce the number of child processes. To this end, `cake` caches the status returned by all condition commands `{{command}}` and the output of all command patterns `[[command]]`. The first cache has hit ratios of 42 and 54 percent on munmurra and mulga respectively, corresponding roughly to the typical practice in which a condition and its negation select one out of a pair of rules. The second cache has a hit ratio of about 80 percent on both machines; these hits are usually the second and later occurrences of macros whose values contain commands.

`Cake` also uses a second optimisation. This one is borrowed from standard `make`: when an action contains no constructs requiring a shell, `cake` itself will parse the action and invoke it through exec. We have no statistics to show what percentage of actions benefit from this, but a quick examination of the standard `cakefiles` leads us to believe that it is over 50 percent.

Overall, `cake` can do a lot more than `make`, but on things which *can* be handled by `make`, `cake` is slightly slower than standard `make` and a lot slower than fourth generation make. Since the main goal of `cake` is generality, not efficiency, this is understandable. If efficiency is important, `make` or one of its other successors is always available as a fallback.

Availability

The `cake` distribution contains the `cake` source, some auxiliary programs and shell scripts (many useful in their own right), diffs for the `lex` driver and the C preprocessor, library cakefiles, manual entries, and an earlier version of this paper (Somogyi, '87). It was posted to the Usenet newsgroup comp.sources.unix in October of 1987.

Acknowledgements

John Shepherd, Paul Maisano, David Morley and Jeff Schultz helped me to locate bugs by being brave enough to use early versions of `cake`. I would like to thank John for his comments on drafts of this paper.

This research was supported by a Commonwealth Postgraduate Research Award, the Australian Computer Research Board, and Pyramid Australia.

References

(AT&T, '84) Augmented version of make, in: *UNIX System V - release 2.0 support tools guide*, AT&T, April 1984.

(Feldman, '79) Stuart I. Feldman, Make - a program for maintaining computer programs, *Software - Practice and Experience*, 9:4 (April 1979), pp. 255-265.

10. On mulga (a Perkin-Elmer 3240), the main applications are text processing and the maintenance of a big bibliography (over 58000 references). On munmurra (an EXLSI 6400), the main application is NU-Prolog compilation.

(Fowler, '85) Glenn S. Fowler, A fourth generation make, *Proceedings of the USENIX 1985 Summer Conference*, Portland, Oregon, June 1985, pp. 159-174.

Gedye, '84 David Gedye, Cooking with CAD at UNSW, Joint Microelectronics Research Center, University of New South Wales, Sydney, Australia, 1984.

(Hirgelt, '83) Edward Hirgelt, Enhancing make or re-inventing a rounder wheel, *Proceedings of the USENIX 1983 Summer Conference*, Toronto, Ontario, Canada, July 1983, pp. 45-58.

(Hume, '87) Andrew Hume, Mk: a successor to make, *Proceedings of the USENIX 1987 Summer Conference*, Phoenix, Arizona, June 1987, pp. 445-457.

(Somogyi, '87) Zoltan Somogyi, Cake: a fifth generation version of make, *Australian UNIX system User Group Newsletter*, 7:6 (April 1987), pp. 22-31.

(Walden, '84) Kim Walden, Automatic generation of make dependencies *Software - Practice and Experience*, 14:6 (June 1984), pp. 575-585.

(Weinberg, '71) Gerald M. Weinberg, The psychology of computer programming, Van Nostrand Reinhold, New York, 1971.

Thank you, Zoltan

Zoltan has agreed that the latest version of 'cake' can be put on the next EUUG conference tape. Thank you, Zoltan.

Competitions at the London Conference

As is traditional at EUUG conferences a competition was held. As is not traditional there were two competitions.

The Signal Competition

The other competition was inspired by the excellent replies generated by the *errno* competition a couple of years ago. This time the task was to invent new signals and their meanings.

Signal	Explanation
SIGHTSEEING	Delegate lost
SIGTUBE	Bus error
SIGINGINTHERAIN	Pipe overflow
SIGFERRY	Data packet has crossed communication channel
SIGTUNNEL	Reserved for future use — will replace SIGFERRY
SIGTUBE	Attempt to pack ≥ bits into a byte
SIGLT	Double sigbus
SIGPEDESTRIAN	Slow data packet is overwritten by expedited data
SIGSIGSIGSIG	Excessive recursion depth
SIGBEKO	Segregation violation
SIGNORINA	36-24-36
SIGNAB	Tax evasion violation
SIGQUIT	You chief programmer just joined another company
SIGSUN	Processor superseded
SIGNAL	Not another language !
SIGJEDI	Use of "The Force" required to continue process
SIGSHUTTLE	Ring failure
SIGNEPHEW	A process, not a child of yours, has died
SIGMUND	Child process too close to motherboard
SIGPROC	Your paper missed the proceedings deadline
SIGNIFICANT	Too many digits
SIGILL	Too much curry
SIGCIA	Are we being bugged
SIGLIONS	I don't understand this
SIGBLAH	Unexpected file recovery
SIGMI5	Filename classified
SIG11	Last order (UK only)
SIGMAINSSPIKE	Incoming signal on power supply
SIGBSD	Your program is using too little memory
SIGCISC	Instruction too complex
SIGPTO	Page fault signal
SIGBLAH	Comment (not) found
SIGGWR	Train arrived on time
SIGMODEM	Unex ect d oss of car ier
SIGWESTEND	American tourist looking for Harrods
SIGLHR	Excessive aircraft noise

The winning entry was:

SIGTITANIC Floating point exception

Submitted by Martyn Tovey from BRS Europe.

There were many other entries of high standard. Here is a selection of the best:

SIGNET	Distributed system caught
SIGNUTCRACKER	Broken kernel
SIGAMNESIA	No more memory
SIGSTAB	Et tu IBM
SIGHUME	Large noise source detected
SIGPANIC	Tilbrook has logged in
SIGANSI	Language too large
SIGIBM	Corporation too large
SIGAPPLE	Program is too small
ISBGTYE	Byte swap error
SIGISO	Has yet to be defined
SIGHIC	Program confused due to excess alcohol
SIGHUP	Excess curry
SIG	Attempt to execute zero length program
SIGQEII	Security violation
SIGPLAN	Unknown programming language
SIGDOS	ENOTUNIX
CIGARETTE	ENOTOBACCO
SIGCIA	Classified information
SIGPUB	Sorry, we're closing in 10 minutes
SIGLUXO	Warning -lamp enabled
SIGDAS	You've been Suniled
SIGRISC	Unimplemented instruction
SIGFREID	Beware the Valkyries
SIGCAT	No mouse
SIGCHEESE	Lost contact with mouse
SIGHIC	Bottle empty
SIGMARTINI	Received any time, any place, any where
SIGAT&T	Failed to acknowledge trademark
SIGNORTH	Security label violation
SIGYAS	Yet another signal /* Dummy signal */
SIGEEC	System deadlocked
SIGBLITZ	Program bombed out
SIG#@$:	I'm not going to tell you where the file is
SIGHOTELFULL	No more space in process table
SIGDAS	Name reference count overflow
SIGASAPARROT	Ill-eagle instruction
SIGBLUE	Unknown hardware
SIGNAL	LAN running backwards
SIGFINE	Speed trap — caught breaking 1 MIP
SIGHIC	Process beer overflow
SINGAL	Program made a spelling mistake
SIGELBOW	Wakeup (boring talk finished)
SIGCHANGLING	Child process has been replaced
SIGGURU	You are not expected to understand this
SIGBUS	Transport service failure — you have to walk home
SIGRA	Attempt to execute Pyramid code on a Sun
SIGLUXO	Your ball is flat
SIGξאψ	Message from foreign host

The Plate Competition

The second was a balloon competition. A plate had been presented as a prize by HCR, this was of the UNIX founding fathers: Dennis & Ken — but what were they saying to each other at the time that the plate was made?

The results were rather poor, however a picture of the plate and the winning caption from Bob Gray of Eucs can be found after the conference proceedings, at the back of this newsletter.

The other entries of note are:

- D'you things UNIX will fly ?
  - It will if we implement it on a frisbee

- Dennis, I've got this new operating system.
  I call it Unix.
  - Twenty years from now you'll regret this

- Dennis, do you think that we are famous ?
  - No Ken, we should have started in a garage

- What do you think they will eat today ?
  - Inodes I hope. They don't stick in beards

- You write the TTY driver.
  - No. You write it

The Tie Breaker

Because it can be very difficult to judge a competition with a high standard of entries (as is found at an EUUG conference ☺) it was decided to have a tiebreaker.

What happened was that the tie breaker ended up being judged as another competition. Yet another first for London: the first conference with three competitions.

Entrants were asked to complete, in no more than 20 words, the sentence:

I attend EUUG conferences because .....

Some of those replying assumed that the judges were weak, vain creatures who could be swayed by simple flattery; they claimed *because:*

The competition committee are such nice people
I think the competition judges are great
The competition judges are so intelligent
Sunil Das is my hero (crawl, crawl)
I think Sunil Das is really wonderful

I have spared the authors of the above by not printing their names, and would like to point out that Sunil was not one the judges anyway.

Various other assorted reasons given were:

- I'm a Pratt — *I'm not arguing* — ED
- I'm waiting for an engineer to arrive and have nothing better to do
- It's fun
- It has funny films
- I want to meet other hackers
- I need a good laugh
- It's the only way I've found to visit London
- My boss told be to do so
- I'll do anything to get out of the office
- Booze, views, news and reviews
- They're there

But the outstanding reason was given by Per Holck of Bull (dk):

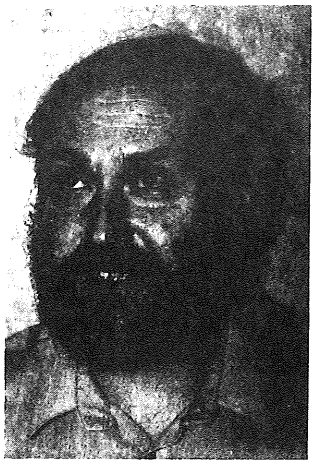
- I want to ask if anybody has received my mail

A Challenge

In the past EUUG competitions have been taken up and run by Usenix at their conferences. Opinions differ on the quality of replies obtained in the USA, they think that they are better. We disagree and assert that all the good ones are from visiting Europeans.

The EUUG hereby challenges Usenix to take up the signal competition and to try and better us at their summer conference in San Francisco.

AFUU Report — Convention UNIX '88



Philip Peake
philip@axis.uucp

Axis Digital
Boulogne
France

The most important event for the AFUU since the last article was without doubt the **Convention UNIX '88**. This was the first exhibition/conference organised directly by the AFUU, the previous exhibitions being organised by Network Events.

The preliminary information after the event gives the following statistics:

- Exhibition — 2100 square metres, with a hundred exhibitors.
- Visitors — 4700 people visited the conference, and 500 more the technical conference which was held in parallel.
- Conference and tutorials — obviously a success with that number of visitors. Since this was our first conference at a new conference centre, there were one or two organisational problems, but there will be resolved for next year.

For those interested, the proceedings of the conference are available from the AFUU.

Another important event is that the AFUU is moving. With 3 full time staff and two UNIX machines, the current office space at SUPELEC was becoming much too cramped. We are moving a little closer to Paris, in fact, just a few hundred metres from the Paris city boundary. The new address is:

AFUU
11 rue Carnot
94270 Le Kremlin-Bicetre
+33 1 46 70 95 90

This will be much easier for visitors, since it is about 50m from the nearest *metro*.

The United Kingdom UNIX Users' Group

Zdravko Podolski
Sunil K Das

sunil@nss.cs.ucl.ac.uk

Introduction

This memorandum is intended to describe the state and activities of the UKUUG. It is a modified document of that presented to the UKUUG Executive Committee, and the EUUG Governing Board and Executive Committee at the Governing Board Meeting held prior to the EUUG Spring 1988 Conference.

Membership

The membership consists of institutional, individual and honorary members. Our fees (ex VAT) are 105 pounds for the institutional membership, 50 pounds for the individual and nothing for the honorary.

The membership figure has now finally stabilised after the shocks of a couple of years ago. We would like to increase the membership and have held discussions with */usr/grp/UK* about merging which are covered later. One of the complications is that, although we recognise the need for a low cost individual membership, there needs to be control so as to avoid the tendency of everyone becoming an individual member. To encourage this, individual membership is required to be accompanied by a cheque drawn by a single person, not one drawn by a company. We look to the EUUG for help in checking at conferences etc that the person applying is a bona fide member of an institution that is an institutional member, or is an individual member in their own right. We are interested in and would support a differential charging structure, to reflect that the individual can draw much less on the services of the group than an institutional member.

The membership are very fee sensitive and we are keeping the fees at the same level as last year. We expect with a rising membership, fees can be held under reasonable control in 1989.

Meetings

There were two meetings during 1987, one at Newcastle in July and one at City University in December. The format of two half days has been successful, allowing people to travel to the meeting in the first morning, attend one session, have the evening to make and renew contacts and then have another session in the morning.

The Newcastle meeting was a straight forward technical event, with Michael Lesk as the keynote speaker, while the City event was a UKUUG workshop on computer networking in the UK attended by over 200 delegates. The workshop was closed to UKUUG members in recognition that only institutional UKUUG members are allowed to join UKNET.

The proceedings of the meetings have been published as is our normal practice, and a copy has been sent to the EUUG National Group contacts. We welcome at our events any other EUUG National Group member on the same terms as our own members. At the City meeting, for the first time and in recognition of the improving relations with */usr/grp/UK*, their members were invited to attend the meeting.

Last month, the UKUUG were hosts to the EUUG's Spring Conference held in London. We are collaborating with */usr/grp/UK* about the 'European Unix Users' Show' in London in July and holding a technical meeting in December in Cambridge. The EUUG's London meeting has consumed vast amounts of the Committee's and others' effort. The number of delegates, attendees and helpers reached a record for EUUG by touching the 600 mark.

Contacts with other groups

During the year we have kept in touch with the UK Sun Users' Group. This group is funded mostly by Sun, so it is naturally much influenced by Sun. The contacts were started with a view to some sort of

collaboration, especially concerning meetings, but in the event the most we could interest them in was to avoid clashing with each others' meeting dates. We shall try harder to encourage them to cooperate with us.

We have also held several meetings with */usr/grp/UK* with a view to merging the two groups. The situation in the UK is anomalous, being the only European country with two user groups. This is an historical accident. With UKUUG/EUUG originally being seen as mostly an academic user group, the vendors started one of their own. The situation is now less clear cut, with about half the UKUUG institutional members being commercial users or vendors, while */usr/grp/UK* has been increasing the ranks of users amongst their members. We all agree on the desirability of ending this confused state of affairs and presenting a single user group to the UK UNIX users.

The structure of */usr/grp/UK* is similar to that of the UKUUG, with individual and institutional members, and fees being 200 pounds and 50 pounds respectively. However, the vast majority of their members are individual. They run a major exhibition every year (contracted out to EMAP), publish a newsletter and have various other activities and relationships, most notably with */usr/grp* in the USA.

Possibly because of their greater advertising clout they have a larger membership than us. If a merger could be achieved, the combined membership would then be in the region of 650 members. The merger discussions have reached agreement on all points except one: the EUUG fees. For a member of the new combined group, the fees would be roughly the current */usr/grp/UK* fees (50 and 200 pounds respectively) plus the EUUG 40 pounds. This would virtually double the individual members' fees overnight. There is also a problem with doubling the current UKUUG institutional fees. While savings can undoubtedly be found because of the size of the combined group, these are unlikely to be large, as both the UKUUG and */usr/grp/UK* run on a non profit basis. Various possibilities exist for solving this problem. It may be possible to reduce drastically the individual membership fee, or unbundle the various EUUG services so that those who want them just pay extra for them, or have an 'associate member' who does not get a member's reduction for conferences. We would welcome advice from the Governing Board, the EUUG executive and our membership on these points.

Another difficulty we had during our attempts to sell EUUG affiliation to */usr/grp/UK* was the lack of EUUG publicity material. The visible activities of the EUUG are well known, the conferences, the Newsletter, EUnet. However there is a whole host of efforts that can and should be loudly advertised. Links with other groups outside Europe, input to standardisation bodies, European networking initiatives, etc, etc. Some obviously should be kept quiet about until fruition, but mostly publicity would be helpful. Otherwise the EUUG is seen as a body with an expensive Newsletter and an even more expensive bureaucracy, without tangible benefits. We tried hard to disabuse them about this, and made much headway, but the issue of cost still remains unresolved.

We now need help from the EUUG to progress this initiative, which would be of immense benefit to users everywhere. The EUUG would gain many new members at a stroke while at the same time cementing the unity between users all over Europe.

USENIX Association News for EUUG Members

Donnalyn Frey
donnalyn@uunet.uucp

Fairfax
Virginia, 22031
USA

Ms. Frey is the USENIX Association Press Liaison. She provides members of the press, USENIX Association members, and EUUG members with information on the activities of the USENIX Association. Ms. Frey has been a UNIX technical writer for five years. She now writes journal and newspaper articles on UNIX-related subjects, as well as working as the Association's press liaison.

The USENIX Association has grown significantly over the last few years. Part of this growth has been the expanding contact between the members of the USENIX Association and the EUUG. This column was created to help foster this contact. The column will provide EUUG members with information on current and upcoming activities of the USENIX Association. Addresses and telephone numbers for upcoming activities are included in the column to assist EUUG members in contacting USENIX Association representatives.

Summer 1988 Conference

Most EUUG members already know about the Summer 1988 USENIX Conference and Technical Exhibition to be held in San Francisco, California on June 20—24. This is expected to be the largest USENIX Association conference to date. Several new tutorials will be presented this summer. The technical sessions include papers on subjects such as window systems, file systems, security, programming languages, and networking. Approximately 100 exhibitors will be displaying their wares at the technical exhibition. A reception will be held at the Exploratorium, a hands-on science museum in San Francisco.

The next USENIX Association Conference will be held in San Diego, California in early 1989. For information on the technical conferences or any other USENIX Association conference, contact the USENIX Conference office at P.O. Box 385, Sunset Beach, CA 90742, USA. The telephone number is +1 213 592 1381. The email address is judy@usenix.uucp.

The Facesaver Project

The Facesaver Project will continue to record new faces and registration information at the San Francisco conference. The Facesaver combines photographs and registration information onto a sticky label that conference attendees can give to exhibitors and other attendees to help them remember each other. After the conference, the attendee list mailed to conference attendees will feature a postage stamp size portrait beside each name and address.

The Facesaver portraits are captured via a video camera using AT&T Targa M8 graphics boards installed in Bell Technologies PC AT clones running the SCO XENIX version of the UNIX operating system. Portraits are printed using a Postscript laser printer.

The Facesaver project is run by Lou Katz. It is sponsored by the Association to aid in improving attendee recognition at the conference. A sample Facesaver label is reproduced below.


FaceSaver 4/88

Donnalyn Frey
+1 703 764 9789
uunet!donnalyn
USENIX Press Liaison
P.O. Box 2051
Fairfax, VA 22031

C++ Conference

The USENIX Association is about to embark on a new series of conferences P the C++ Conferences. The Association has held successful C++ workshops in the past. However, the last workshop, in Santa Fe, New Mexico, was so well-attended that the Association decided to expand the format. The result is the first open C++ conference, scheduled for October 17—20, 1988 in Denver, Colorado. For information on technical submissions for the conference, contact Andrew Koenig at ark@europa.att.com. For registration information on this new conference, contact the USENIX Conference office at P.O. Box 385, Sunset Beach, CA, 90742, USA. The telephone number is +1 213 592 1381. The email address is judy@usenix.uucp.

EUUG members still trying to get copies of the Proceedings of the 1987 Santa Fe, New Mexico C++ Workshop will be pleased to know that the Proceedings have been reprinted and are again available for purchase. For information on the Proceedings, contact the USENIX Association office at P.O. Box 2299, Berkeley, CA 94710, USA. The telephone number is +1 514 528 8649. The email address is office@usenix.uucp.

Large Installation System Administration II Workshop

The second Large Installation System Administration Workshop will be held November 17 & 18, 1988 in Monterey, California. The call for papers will appear in the May-June and later issues of the Association's **;login:** newsletter. For information on submissions for the workshop, contact Alix Vasilatos of MIT's Project Athena at alix@athena.mit.edu. For registration information, contact the USENIX Conference office at P.O. Box 385, Sunset Beach, California 90742, USA, by telephone at +1 213 592 1381, or by email at judy@usenix.uucp.

UNIX Security Workshop

The UNIX Security Workshop will be held August 29 & 30, 1988 in Portland, Oregon. The workshop will bring together researchers in UNIX computer security and system administrators trying to use UNIX in environments where security is of the utmost importance.

Some topics to be included in the workshop include password security, network security, and file system security. For information on submissions for the workshop, contact Matt Bishop at bishop%bear.dartmouth.edu@relay.cs.net or at {ihnp4,decvax}!dartvax!bear!bishop. For registration information, contact the USENIX Conference office at P.O. Box 385, Sunset Beach, California 90742, USA, by telephone at +1 213 592 1381, or by email at judy@usenix.uucp.

Workshop on UNIX and Supercomputers

The Workshop on UNIX and Supercomputers will be held September 26 & 27, 1988 in Pittsburgh, Pennsylvania. The workshop will consider the general problems of running UNIX on supercomputers and will cover both practical and abstract topics. Areas of interest will include system administration, file systems, networking, monitoring performance, shared memory management, and very large files.

For information on technical submissions for the workshop, contact Lori Grob at grob@lori.ultra.nyu.edu or Melinda Shore at shore@reason.psc.edu. For registration information, contact the USENIX Conference office at P.O. Box 385, Sunset Beach, California 90742 USA, by telephone at +1 213 592 1381, or by email at {ucbvax,uunet}!usenix!judy.

1988—1989 USENIX Association Board of Directors

The USENIX Association recently held elections for the 1988—1990 Board of Directors. The new board is composed of:

Alan Nemeth	President
Deborah Scherrer	Vice-President
Rob Kolstad	Secretary

Steve Johnson	Treasurer
Kirk McKusick	Director
Michael O'Dell	Director
John Quarterman	Director
Sharon Murrel	Director.

Michael O'Dell and Sharon Murrel are new board members. The new Board of Directors will take office at the San Francisco USENIX Association Conference in June.

2.10BSD Operating System

The USENIX Association is continuing to distribute the 2.10BSD operating system. 2.10BSD is a UNIX operating system for Digital Equipment Corporation PDP-11 series computers. It was released in October 1987 by the Computer Systems Research Group (CSRG) of the University of California at Berkeley.

2.10BSD is a basic port of 4.3BSD functionality to a PDP-11. This release is most useful for sites that have 4.3BSD programs or machines and who would like consistency across the environment. It is also useful for sites that want a faster, cleaner version of 2.9BSD.

2.10BSD is faster than 2.9BSD and includes 22-bit Qbus support; 4.3BSD networking (TCP/IP, SLIP); 4.3BSD serial line drivers; 4.3BSD C library; most 4.3BSD applications programs; 4.3BSD system calls; RAM disk; inode, core and swap caching; and the conversion of the system to a 4.3BSD structure.

This release is being handled by the USENIX Association and is available to all UNIX V7, System III, System V, and 2.9BSD licensees. The Association will continue to maintain the non-profit price of $200 plus shipping for overseas sites, as was charged by CSRG. The release consists of two 2400 foot, 1600 bpi tapes (approximately 80 MB), bug fixes, and approximately 80 pages of documentation. If a site requires 800 bpi tapes, they should contact the Association for further information.

The tapes that the USENIX Association is distributing only support machines with split I/D and floating point hardware. At this time, the Interlan Ethernet, DEQNA, DEUNA,and 3COM drivers have been ported. 2.10BSD runs on the following machines: PDP-11 /24 /34 /44 /53 /60 /70 /83 /84 and PDP-11/23

For information on the distribution of the 2.10BSD release, contact the USENIX Association 2.10BSD, P.O. Box 2299, Berkeley, CA 94710, USA. The telephone number is +1 415 528 8649. The email address is office@usenix.uucp.

More to Come...

This column will again appear in the Autumn EUUG newsletter, providing EUUG members with more news of the USENIX Association. Any (favourable) comments on the column should be sent to me.

Computing for the 1990's

Malcolm Stayner
malcolm@nezsidc.uucp

ICL New Zealand Ltd
Wellington
New Zealand

The forthcoming NZUSUGI (New Zealand UNIX Systems User Group Inc.) Conference and Exhibition at the Plaza International Hotel, Wellington on 9—11th June promises to be the biggest and best yet. With the sale of several large UNIX systems marking the very definite arrival of UNIX as an operating system for commercial installations in New Zealand, interest from vendors has been very high. All the exhibition space was sold by early March and visitors to the exhibition can expect to see some exciting developments in the UNIX world. The deluxe exhibitors will be ICL, NCR, NEC and Olivetti who will, no doubt, be exhibiting their products to the utmost. Premium stand holders are Altos, Phoenix, Unisys and STC. Finally, also exhibiting will be Apple, Barson, CBA, Northrop, Rakon, Today and Xact. The exhibition will be open to the public on all three days.

'Computing for the 1990's' is the theme for the conference, chosen because UNIX is seen as providing the vehicle for departmental computing, a major growth area for the next decade. It is perhaps worth pointing out that it is not envisaged that departmental systems will oust PCs or mainframes but rather form a worthy complement to these well established areas of information processing. While the individual will still require computing power on his or her desk and corporations will still require centralised data storage and processing, there is a large middle ground where information has to be shared amongst a work group. It is here that UNIX-based systems are seen as providing a solution to a pressing business need, which is the requirement for two or more people with a common business function (e.g., scheduling sales calls or ordering supplies) to be able to communicate and operate effectively, thereby increasing their contribution to the organisation. As such, topics such as standards and connectivity are major issues and the conference will address these. This year the conference will specifically target government departments as well as our traditional audience, as NZUSUGI believes that there is much scope for UNIX-based systems amongst this community.

We are honoured to have Peter Dunne, Under Secretary for Trade and Industry address the conference on the opening day and formally open the exhibition.

Once again, there is an excellent line-up of international and local speakers. From overseas there is Stuart Hooper, Director of The Santa Cruz Operation, who will speak on the significance of the merging of UNIX and XENIX; Jim Bennett of Convergent Technology will speak on Windows under UNIX and Network File Sharing and James Clark of the Singapore UNIX Association will speak on the Universality of UNIX in the Asian culture.

Our cast of overseas speakers will be complemented by ten local speakers addressing various UNIX-related issues, including electronic mail, office automation and UNIX trends in government worldwide. Doug Marker of IBM will talk about IBM's UNIX strategy and the differences between AIX and OS/2. Keith hopper, a NZUSUGI stalwart, will talk on POSIX, Mark III and beyond.

We are delighted to have as chairpersons for our stream sessions this year Ian Howard and Roger Hicks, co-directors of Rakon Independent Software Division, and John Hine from the Department of Computer Science of Victoria University, Wellington.

Finally, the conference will be brought to a close by what promises to be a very lively debate among a panel of vendors and users. The topic for discussion will be 'Standards, are they effective?' This will be chaired by Gordon Hogg, former Chief Executive of Databank.

The annual NZUSUGI conference provides a valuable opportunity for NZUSUGI members and those involved with UNIX to mix and mingle in a convivial atmosphere. The conference dinner on the evening of 9th June will be one of the social highlights. Mike Dubrall, Director of the Independent Software Marketing Division for the Pacific Group of NCR, who has attended two past NZUSUGI conferences said in UNIX/World (Oct'87) that the NZUSUGI conference was "arguably one of the best UNIX gatherings in the world today". He went on to say, "all in all, the New Zealand UNIX Group is as vibrant as any in the world. UNIX aficionados would do themselves a favour by combining a vacation with the business of attending the next NZUSUGI conference in Wellington". Such an accolade from the United States is a fearsome one indeed, and one that this year's Conference Committee will be endeavouring to surpass.

Malcolm Stayner
1988 NZUSUGI Conference Chairman

For further details contact:

Malcolm Stayner
Technical Manager, Software Industry Development Centre
ICL New Zealand Ltd
P.O. Box 394
Wellington
New Zealand

Tel: +64 4 724-884
FAX: +64 4 726-737

For a registration form contact:

Jan Tonkin
CMS
P.O.Box 12-442
Auckland

Tel: +64 9 525 1240
FAX: +64 9 525 1243

European UNIX® systems User Group
Owles Hall, Buntingford, Herts. SG9 9PL, UK
Tel (+44) 763 73039

PRELIMINARY ANNOUNCEMENT

and

CALL FOR PAPERS

-=-=-=-=-=-

EUUG SPRING '89 CONFERENCE

Brussels, 10–14 April 1989

-=-=-=-=-=-

UNIX: EUROPEAN CHALLENGES

Preliminary Announcement

The BUUG will host the Spring '89 European UNIX systems User Group Technical Conference in Brussels, Belgium.

Technical tutorials on UNIX and closely related subjects will be held on Monday 10th and Tuesday 11th April, followed by the three day conference with commercial exhibition finishing on Friday 14th April.

A pre-conference registration pack containing detailed information will be issued in early December 1988.

Call for Papers

The EUUG invite abstracts from those wishing to present their work. All submitted papers will be refereed to be judged with respect to their quality, originality and relevance.

Suggested subject areas include:

- Real time
- Networking
- Security issues
- Graphics
- Internationalisation
- Distributed processing
- Fault tolerance
- New architectures
- Transaction processing
- Window systems and environments
- Supercomputing
- Standards and conformance tests

Submissions from students are particularly encouraged under the EUUG Student Encouragement Scheme, details of which are available from the EUUG Secretariat.

Important Dates

Abstract deadline	**November 30th, 1988**
Acceptance notification	**January 15th, 1989**
Final paper received	**February 1st, 1989**

Method of submission

Abstracts **must** be submitted by post to the EUUG Secretariat. All submissions will be acknowledged by return of post.

Papers may be submitted electronically to Prof Nyssen, but this is not the formal method of submission.

Tutorial Solicitation

Tutorials are an important part of the EUUG's biannual events providing detailed coverage of a number of topics. Past tutorials have been taught by leading experts. Those interested in offering a tutorial should contact the EUUG Tutorial Officer as soon as possible.

Additional Information

The Programme Chair, Prof Marc Nyssen, will be pleased to provide advice to potential speakers. Prof Nyssen may be contacted at the address below.

If you wish to receive a personal copy of any further information about this, and future EUUG events, please write, or send electronic mail, to the Secretariat.

Useful Addresses

Secretariat	**Tutorial Officer**	**Programme Chair**
EUUG Owles Hall Owles Lane BUNTINGFORD Herts SG9 9PL UK	Neil Todd IST 60 Albert Court Prince Consort Road LONDON SW7 2BH UK	Prof Marc Nyssen Medical Informaticas Dept Vrije Universiteit Brussel Laarbeeklaan 103 B-1090 JETTE B-1090 BELGIUM
Phone: (+44) 763 73039 Fax: (+44) 763 73255 Telex: Email: euug@inset.uucp	(+44) 1 581 8155 (+44) 1 581 5147 928476 ISTECH G neil@ist.co.uk	(+32) 2 477 44 24 (+32) 2 477 40 00 marc@minf.vub.uucp

Macro Expansion as Defined by the ANSI/ISO C Draft Standard

Cornelia Boldyreff
Cornelia.Boldyreff@brunel.ac.uk

PRACTITIONER PROJECT
Department of Computer Science
Brunel University

Cornelia Boldyreff is a member of the British Standards Institution technical committee on Application Systems, Environments and Programming Languages. She acts as Convenor and Chairman of the BSI C Language Panel; and is one of the UK Principal Experts on the ISO Working Group on C. She is also Convenor and Chairman of the BSI POSIX Panel; and is one of the UK Principal Experts on the ISO Working Group on POSIX.

Introduction

The ANSI/ISO C Draft Standard contains a definition of the C Preprocessing Directives in Section 3.8. As the standard is not a tutorial, it simply aims to provide a definitive guide for implementors and programmers. Occasionally examples are included in the text of the standard as illustrations; however, the non-tutorial character of the document precludes any detailed explanation of these examples. In this note, the examples given to illustrate aspects of macro expansion are discussed in more detail.

This note arose out of explanations prepared in response to comments submitted to ISO purporting to have found an inconsistency and an error in the preprocessing examples. Examples 1 and 1a: Illustrations of macro-replaced `#include` directives

Example 1 found in section 3.8.2 is unlikely to surprise current users of C; under the proposed ANSI/ISO C standard, the preprocessing tokens following the `#include` are subject to processing, that is any macro names found will be subject to replacement. The result must be a header name as defined in section 3.1.7; basically, a header name is a sequence of characters delimited by either angle brackets (<>) or double quotes with some restrictions on allowable characters.

In the example given, one of a series of macro definitions is selected on the basis of a nested set of conditional inclusions. This macro name following the `#include` is then replaced by the replacement list in the selected definition; and the result forms a valid header name as required.

```
#if VERSION == 1
   #define INCFILE "vers1.h"
#elif VERSION == 2
   #define INCFILE "vers2.h"
#else
   #define INCFILE "versN.h"
#endif
#include INCFILE
     /* Example 1 */
```

An example with a similar effect can be extracted from those given in the draft to illustrate rules for creating character string literals and concatenating tokens; it is given below.

```
#define str(s) # s
#define xstr(s) str(s)
#define INCFILE(n) vers ## n
#include xstr(INCFILE(2).h)
     /* Example 1a */
```

This example works because an argument may consist of any number of (but at least one) preprocessor tokens; and any argument not corresponding to a parameter that is an operand of either the # operator or the ## operator is subject to complete macro replacement before substitution. By invoking the macro `str` through the macro `xstr`, this example ensures that any macros in the argument of `str` have been completely replaced.

A similar effect may achieved by explicitly invoking a `paste` macro either directly or through another macro, e.g.

```
#define paste(a,b) a ## b
#define xpaste(a,b) paste(a,b)
#define str(s) # s
#define xstr(s) str(s)
#include xstr(xpaste(paste(vers, 2), .h))
```

This explicit form is more general as Example 1a only works where the macro invocation can be clearly distinguished in the argument. The following would not succeed:

```
#define str(s) # s
#define xstr(s) str(s)
#define INCFILE vers2
#include xstr(INCFILEheader)
```

Example 2: Illustrations of Redefinition and Re-examination

This example is best considered as several smaller examples. The directives given define a series of macros; these are listed below:

```
#define x 3
#define f(a) f(x * (a))
#undef x
#define x 2
#define g f
#define z z[0]
#define h g(~
#define m(a) a(w)
#define w 0,1
#define t(a) a
```

The scope of the initial definition of `x` [line 1] extends to the corresponding `#undef` directive [line 3]; as there are no relevant occurrences of `x` in this section of the file, these two lines may be ignored. The expression given

in the text:

```
f(y+1) + f(f(z)) % t(t(g)(0) + t)(1);
g(x+(3,4)-w) | h 5) & m
      (f)^m(m);
```

yields the result:

```
f(2 * (y+1)) + f(2 * (f(2 * (z[0])))) % f(2 * (0)) + t(1);
f(2 * (2+(3,4)-0,1)) | f(2 * (~ 5)) & f(2 * (0,1))^m(0,1);
```

To clarify how this result is obtained, consider the expansion of the following terms:

Number	Term	Result
1	`f(y+1)`	f(2 * (y+1))
2	`f(f(z))`	f(2 * (f(2 * (z[0]))))
3	`t(t(g)(0) + t)`	f(2 * (0)) + t
4	`g`	f(2 * (2+(3,4)-0,1))
5	`h`	f(2 * (~5))
6	`m(f)`	f(2 * (0,1))
7	`m(m)`	m(0,1)

The two key phrases to note are:

— Before substitution, each argument's preprocessing tokens are completely macro replaced as if they formed the rest of the source file; no other preprocessing tokens are available. [section 3.8.3.1]

and

— [Under 3.8.3.4 Rescanning and further replacement] If the name of the macro being replaced is found during this scan of the replacement list [i.e., the scan after substitution] (not including ...), it is not replaced. Further, if any nested replacements encounter the name of the macro being replaced, it is not replaced. These non-replaced macro name preprocessing tokens are no longer available for further replacement ... [This effectively rules out recursion. See note in Rationale.]

The processing of term 1 is simply an expansion of macro `f` with argument `y+1`. As the argument contains no macros, after substitution `f(y+1)` becomes: `f(x * (y+1))` and `x` is replaced by `2` on the rescan giving: `f(2 * (y+1))`.

The expansion of macro `f` in term 2 is as follows: Parameter a matches argument `f(z)`. Before substitution, `f(z)` is expanded with a matching `z`; and then `z` in turn is expanded to `z[0]` and this `z` is not subject to any further replacement. Thus after substitution, `f(z)` becomes: `f(x * (z[0]))` and `x` is replaced by `2` on the rescan giving: `f(2 * (z[0]))`. So the original expression `f(f(z))` becomes: `f(x * (f(2 * (z[0]))))` after substitution. Macro `x` is replaced by `2` on rescan giving the following as the final result: `f(2 * (f(2 * (z[0]))))`.

The expansion of term 3 is as follows: parameter `a` matches argument `t(g)(0) + t` which requires further expansion of `t(g)` before substitution. So `t(g)` is expanded with parameter `a` matching `g`; and `g` is further expanded to `f`; so on substitution, `t(g)` becomes `f`. On substitution in the original expression, the replacement list becomes `f(0) + t` which on rescan results in expansion of `f(0)` and no expansion of `t` as it is ruled out by section 3.8.3.4. The subexpression `f(0)` expands to `f(x * (0))` and with `x` being replaced by `2` becomes `f(2 * (0))`. Thus, the final result is `f(2 * (0)) + t`.

Term 4 is simply a replacement of the macro `g` by the replacement list `f`. This token is followed by a left parenthesis so the following tokens are interpreted as the argument to an invocation of the macro `f`. The result of invoking `f` with the argument `x+(3,4)-w` is given by replacing the `x` and `w` macros, and then substituting the result in `f`. The argument before substitution is `2+(3,4)-0,1` and after substitution and replacement of `x` with 2, the final result is: `f(2 * (2+(3,4)-0,1))`.

In term 5, `h` is replaced by `g(~`; on rescan `g` is replaced by `f` and on the next rescan `f` followed by a left parenthesis is recognised as an invocation of the macro `f` with argument `~5`. After substitution and replacement of `x` with, the result is: `f(2 * (~5))`.

The sixth term, `m(f)`, is an invocation of macro `m`; after substitution, the result is: `f(w)`. Note the intervening newline in the source is simply interpreted as white space which is lost during replacement. On the rescan, this recognised as an invocation of `f` with argument `w` which expands to `0,1` before substitution; and `f(x * (0,1))` after substitution. After replacement of `x` by `2`, the final result is: `f(2 * (0,1))`.

The last term is quite straight forward to explain; this is an invocation of macro `m` with argument `m` matching parameter `a`. After substitution, the result is: `m(w)`. On rescan, the `m` is not subject to further replacement by the rule given in 3.8.3.4; and the `w` is replaced by `0,1` giving the final result: `m(0,1)`.

Examples 3, 4 and 5: Illustration of rules for creating character string literals and concatenating tokens

The series of definitions listed here can be separated into four separate examples; one has already been discussed as 1a above. The relevant definition for example 3 are:

```
#define debug(s, t) printf("x" # s "= %d, x" # t "= %s", \
                           x ## s, x ## t)
```

and the term to be processed consists of: `debug(1, 2)`. After substitution, the result is:
`printf("x" "1" "=%d, x" "2" "=%s", x1, x2).`

Example 4 illustrates the special handling required for producing the spelling of string literals and character constants. The relevant definition is:

```
#define str(s) # s
```

and the terms to be expanded are:

```
str(strcmp("abc\0d", "abc", '\4') == 0)
str(: @\n)
```

Only in the first case does the argument receive special handling to deal with string literals and character constants; in the latter case, there are none. Thus `str(:@\n)` simply expands to `": @\n"`. In the first case, a backslash is placed before each double quote and `\` character of a character constant or string literal before this argument is transformed into a single character string literal which replaces the list `# s`. Thus, the result after replacement is:

```
"strcmp(\"abc\\0d\", \"abc\", '\\4' == 0)"
```

The fifth example simply illustrates the difference between invoking the `paste` operation directly and indirectly. The relevant definitions are:

```
#define glue(a, b)  a ## b
#define xglue(a, b)  glue(a, b)
#define HIGHLOW  "hello"
#define LOW LOW  ", world"
```

The two terms to be expanded are: `glue(HIGH, LOW)` and `xglue(HIGH, LOW)`. The first simply results in `HIGHLOW` after substitution; and on the rescan, `HIGHLOW` is replaced by `"hello"`.

The second requires the expansion of the argument `LOW` to `LOW ", world"` before substitution. After substitution of `HIGH` for parameter `a` and `LOW ", world"` for parameter `b`, the result is: `HIGH ## LOW ", world"` and before the rescan, the `##` is deleted and tokens `HIGH` and `LOW` are concatenated giving: `HIGHLOW ", world"`. After rescan, this becomes: `"hello" ", world"`.

Summary

The preprocessing examples given in the draft standard illustrate some of the new facilities introduced by the draft. This note has given a fuller explanation of these examples for new readers of the draft.

Report on POSIX Meeting: 2-4 March 1988, London

Cornelia Boldyreff
Cornelia.Boldyreff@brunel.ac.uk

PRACTITIONER PROJECT
Department of Computer Science
Brunel University

Introduction

This was the inaugural meeting of WG15. It was attended by delegates from Canada, Denmark, the UK and the USA; as well as several observers and representatives from SC18/WG9 (User Interfaces) and SC21/WG5 (OSCRL). Jim Isaak, the IEEE P1003 Chairman and convenor of WG15, chaired the meeting; and Roger Martin of the NBS acted as secretary to the meeting. The meeting was held over three days.

Liaison Activities and Status Reports

The first day consisted of establishment of liaison activities with related standards work such as OSCRL, C language, Ada, and liaison activities with related organisations such as NBS, X/OPEN and */usr/group*. Brief reports were made of the status of related standards and the progress of associated IEEE activities. The table below summarises these:

Standard	Status
1003.1 - POSIX	Balloting
1003.2 - Shell and Utilities	1Q88
X3J11 'C'	2Q88
1003.3 - Test Methods	3Q88
1003.4 - Real Time	1Q90
1003.5 - Ada	2Q89
1003.6 - Security	1Q90
1003.0 - Guide to POSIX Based Systems	1Q89

With respect to the work of 1003.3, Roger Martin of the National Bureau of Standards spoke on their aim of defining what must be tested, i.e., test assertions and not how testing must be done. He mentioned the recent NBS POSIX FIPS indicating that this was an interim document, and that NBS intended to move to a FIPS based on the full use version of the POSIX standard when it was available. In conjunction with the FIPS, the NBS has developed the PCTS — POSIX Conformance Test Suite. The PCTS is based on the SVVS Subset from ATT. It will be available in source code form in the public domain; it will be distributed by the National Technical Information Service. NBS will maintain the test suite as the standard evolves. They aim to encourage use of the test suite by third party testing services.

The remainder of the first day was taken up with an introduction to work on User Interfaces by SC18/WG9 and SC21/WG5 OSCRL. Although the former work is not directly relevant to the current POSIX work; it was acknowledged that some elements of the OSCRL work placed constraints (e.g., security) on POSIX applications. The OSCRL work presents the user with a different system view and as such is complementary to POSIX. It was agreed that the WG would express a willingness to accept the OSCRL work item; this would allow the current OSCRL documents to be accepted as technical reports for consideration in the future work of WG15.

Main Items of Business on Days 2 and 3

The most pressing business of the meeting was to review the results of the SC22 Ballot and coordinate a response. The Ballot results consisted of only two votes of disapproval from Canada and Japan. The issues raised by Canada were resolved at the meeting; unfortunately, the Japanese comments were not available during the meeting. All other comments accompanying votes of approval from Denmark, France, Germany and the UK were processed. A Disposition of Comments document was drafted.

The WG discussed future work including a division of the work item (see Resolution 9 below); and the following dates of future biannual meetings were agreed:

20-21 Oct 88 -Tokyo
Apr 89 - Ottowa, Canada
Oct 89 - Brussels
Apr 90 - Paris
Oct 90 - USA

It was agreed that subject to satisfactory resolution of the Japanese comments, the WG should put forward DP9945 for registration as a DIS. It is anticipated that if DP9945 is registered as a DIS in June or July this year, the six-month ISO Ballot will have closed in good time before the WG meeting planned in the spring of 1989.

Resolutions of WG15

The text below is taken from my own notes; and may appear in a revised form in the official minutes.

Resolution 1

JTC1/SC22/WG15 requests that the US member body incorporate the recommendations made in and resolve the outstanding issues in JTC1/SC22/WG15/N___, "Disposition of Comments on DP9945" into the revised POSIX document.

Resolution 2

JTC1/SC22/WG15 requests that the US member body take into consideration recommendations made in JTC1/SC22/WG15/N___, "Disposition of Comments on DP9945", including the UK and French member bodies comments on language independent specifications when developing future versions.

Resolution 3

JTC1/SC22/WG15 recommends that the revised document adopted by the IEEE which incorporates responses to SC22 comments be registered as a DIS.

Resolution 4

JTC1/SC22/WG15 requests that the SC22 Secretariat coordinate determination of the proper document format and forward that information to the POSIX document editor as soon as possible.

Resolution 5

The following comments shall be submitted to the SC22 Secretariat regarding recommendations made in TC97/N1935:

1. The work item scope has been modified as proposed. JTC1/SC33/WG15 is committed to the development of a language independent specification of POSIX.
2. JTC1/SC22/WG15 met representations of the OSCRL committee and issues resolutions pertaining thereto.
3. The POSIX trademark has been dropped by IEEE.

Resolution 6

JTC1/SC22/WG15 requests that the US member body develop a single interchange format that resolves the concerns raised regarding CPIO and TAR, and provide such format for inclusion in a future version of the POSIX document. The existing definitions of both CPIO and TAR formats in POSIX should be retained until a new, single interchange format definition is available.

Resolution 7

JTC1/SC22/WG15 wishes to express its thanks to the SWG on POSIX, SSI and Related Issues. Following the recommendations of the SWG (TC97 N1935, Att. D.), the definition of the scope of work was clarified, and the POSIX work item was accepted and assigned to SC22 for development.

The WG also wishes to thank SC22 for its prompt action on the POSIX NWI in Sept 1987. By unanimously voting to accept registration of DP9945 allowing WG15 to proceed immediately to a DP Ballot, work on this standard has been greatly expedited.

Resolution 8 — Regarding OSCRL

JTC1/SC22/WG15 expresses to the the SC22 Secretariat its willingness to accept the OSCRL work item. If OSCRL is assigned to WG15, the current OSCRL documents would be accepted as technical reports for consideration in the future work of WG15.

Resolution 9

JTC1/SC22/WG15 instructs the WG15 Convenor to initiate a division of the work item, and submit this to the SC22 Secretariat for action. Said division should reflect the following WG15 activities:

1. Continued work beyond 9945 to address:
   - — Language Independent Bindings
   - — Ada and C Bindings
   - — Additional Service Definitions for Terminal Control, Data Interchange, etc
2. A separate standard document pertaining to the Operating System Environment based on the IEEE 1003.2 work shall be initiated.

3. Formation of a Rapatour Group on Security to do preliminary investigations of the scope and requirements for work in this area, and to coordinate with the IEEE 1003.6 effort.

WG15 will also need complementary work to be done in the areas of — testing, windows, application/human interfaces, systems administration, distributed systems, and network interfaces.

Resolution 10

JTC1/SC22/WG15 instructs the WG15 convenor to respond to the Japanese Comments on DP9945 in consultation with WG15 experts and prepare an addendum to the Disposition of Comments document.

More Obfuscated C Code

Mick Farmer
mick@cs.bbk.ac.uk

Here's a program we wrote recently. It originated in a discussion I had with Dave Tilbrook and my seeing the winners of the 'Obfuscated C Code Contest' for 1986.

```
**********************************************************************
#define 1000 char
#define 1001 mess
#define 1010 [
#define 1011 13
#define 1100 ]
#define 1101 =
#define 1110 {
#define 1111 'H'
#define 1001 ,
#define 1010 'e'
#define 1011 'l'
#define 1100 'o'
#define 1101 ' '
#define 1110 'w'
#define 1111 'r'
#define 1011 'd'
#define 1011 '0
#define 1101 ' '
#define 1101 }
#define 1110 ;
#define 1110 main
#define 1111 (
#define 1111 )
#define 1111 printf
1000 1001 1010 1011 1100 1101 1110 1111 1001 1010 1001
1011 1001 1011 1001 1100 1001 1101 1001 1110 1001 1100
1001 1111 1001 1011 1001 1011 1001 1011 1001 1101 1101
1110 1110 1111 1111 1110 1111 1111 1001 1111 1110 1101
```

UNIX Clinic

Colston Sanger
olibc1!colston@olgb1.oliv.co.uk

Olivetti International Education Centre

Colston Sanger is a lecturer at the Olivetti International Education Centre, Haslemere, UK and a visiting lecturer in the Faculty of Engineering, Science and Mathematics at Middlesex Polytechnic, London.
He is *not* the Musical Director of the Pangbourne & District Silver Band...

About this column

At the recent EUUG conference in London Alain Williams asked if I would like take over this column. I said "OK", so — here goes.

About this column: I see it as being primarily for the post–1984, UNIX System V generation. That's to say, I see it as being for users who want or need to use the UNIX System to accomplish practical, real–world tasks. Most of you (or us — because I'm also one of the post–1984 generation) are working in the commercial rather than academic world, and have UNIX binary as opposed to source licences. In short, then, it's for beginners rather than people who use `cat` to write device–drivers.

As to the flavour of UNIX, the emphasis is likely to be on System V: not only because that's "The Right Choice" (as the advertisement says), but also because that's the kit available to me. Where I work — at the Olivetti International Education Centre — we have a Starlan network of Olivetti–AT&T 3B2's running UNIX System V Release 3.1 and RFS, together with various Olivetti PC's running MS–DOS and the AT&T DOS Server package, but that's all. Sorry, but I just don't have a VAX.

That said — the introductory stuff, I mean — let's get into it.

Screen management in shell

There's been a lot of talk lately about user interfaces: News, X, Andrew, Open Look and so on. I thought it might be interesting to look at what can be done using plain old `/bin/sh`.

The basics — `echo` and `read`

First a quick review of the basics: `echo` and `read`. You know what `echo` does, but the System V shell built–in `echo` also takes a set of special escape characters:

`\b`	backspace
`\c`	do not append a newline
`\f`	formfeed
`\r`	carriage return
`\t`	tab
`\v`	vertical tab
`\\`	backslash
`\0n`	where `n` is the eight–bit character whose ASCII code is the one–, two– or three–digit octal number representing that character.

For example:

```
$ echo "\n\tPlease enter name: \c"

        Please enter name: _
```

(The underscore here is meant to be the cursor on your terminal screen.) Berkeley people will have to use

```
$ echo -n "\n\tPlease enter name: "
```

instead of the quoted `\c`.

`read` is pretty simple too. Within a shell script, just:

```
echo "\n\tPlease enter name: \c"
read NAME
```

where NAME is a shell variable.

Using screen attributes — `tput`

Suppose you want to use screen attributes such as reverse video, blink, underlining and so on? This is where it gets interesting — and also where the System V `tput` command comes in.

Try:

```
$ tput blink
$ echo Hello world
```

OK, if you can't stand it any longer, type:

```
$ tput sgr0
```

There's a complete list of terminal attributes (and potential arguments to `tput` — depending on the capabilities of your terminal) under *terminfo*(4) in the UNIX System V Programmer's Reference Manual. For now, some useful attributes are:

`clear`	clear the screen and leave the cursor at the top–left (home) position
`bold`	turn on bold (extra–bright) mode
`smso`	begin standout (reverse–video) mode
`rmso`	end standout mode
`smul`	begin underscore mode
`rmul`	end underscore mode
`blink`	turn on blinking
`dim`	turn on dim (half–bright) mode
`bel`	ring the terminal bell
`sgr0`	turn off all attributes.

Note for Berkeley people: I don't think you have an equivalent to the System V `tput` command, although there are public–domain versions of `tput` around that use the `termcap` database.

A note of style

It's good style to assign screen attributes to shell variables (by command substitution) at the beginning of your shell script as in this example — a companion for the classic `phone`, everybody's first shell script:

```
# phone.add - enter phone nos in $HOME/lib/phone.nos

CLEAR=`/usr/bin/tput clear`
STANDOUT_ON=`/usr/bin/tput smso`
STANDOUT_OFF=`/usr/bin/tput rmso`

YN=YES
while [ "${YN}" = "YES" ]
do
        echo "${CLEAR}Phone.add v1.0\n"
        NAME=
        echo "\t${STANDOUT_ON} Name: ${STANDOUT_OFF} \c"
        read NAME
        echo "\n\t${STANDOUT_ON} Address: ${STANDOUT_OFF} \c"
        ADDRESS=
        while read A
        do
                echo "\t\t     \c"
                case ${A} in
                        '')     # blank line ends address
                                break ;;
                        *)      ADDRESS="${ADDRESS} ${A}" ;;
                esac
        done
        echo "\n\t${STANDOUT_ON} Phone: ${STANDOUT_OFF} \c"
        read PHONE
        echo "${NAME}\t${ADDRESS}\t${PHONE}" >> ${HOME}/lib/phone.nos
        echo "\n\tAdd another name (Y/N) ? \c"
        read YN
        case ${YN} in
                [Yy]*)  CONTINUE=YES ;;
                *)      CONTINUE=NO ;;
        esac
done

sort -u ${HOME}/lib/phone.nos -o ${HOME}/lib/phone.nos
```

Why is it good style? Well, first, it parameterises things: if you want to use underline instead of reverse-video for highlighting you only need to change two lines at the top. Second, it's more efficient than

```
        echo "\t`tput smso` Name: `tput rmso`\c"
```

because you're not spawning an extra process for every `tput` command: just echoing a shell variable.

Making menus

Given `echo`, `read` and `tput`, it's easy to make menus — though it has to be said that genuinely 'user-friendly' menus are quite hard to do. Here is the outline of a fairly traditional menu:

```
# monitor - main menu
# trap interrupts
trap ' ' 1 2 3

# set standard variables
MACHINE=`/bin/uname`    # machine name
BIN=${HOME}/monitor/bin   # dir where progs are
```

```
export MACHINE BIN

# set screen attributes
CLEAR=`/usr/bin/tput clear`
STANDOUT_ON=`/usr/bin/tput smso`
STANDOUT_OFF=`/usr/bin/tput rmso`
NORMAL=`/usr/bin/tput sgr0`
BELL=`/usr/bin/tput bel`
export CLEAR STANDOUT_ON STANDOUT_OFF NORMAL BELL

HEADER="${CLEAR}${STANDOUT_ON} ${MACHINE} monitor
                `/bin/date` ${STANDOUT_OFF}"
export HEADER

while :
do  echo "${HEADER}\n\n\n"
    echo "\t\t\t1 - Usage monitoring"
    echo "\t\t\t    (whodo, who -u, ps -ef.)\n"
    echo "\t\t\t2 - Performance monitoring"
    echo "\t\t\t    (Run sar, print or "
    echo "\t\t\t    display results.)0
    echo "\t\t\t3 - System accounting"
    echo "\t\t\t    (Print daily/monthly"
    echo "\t\t\t    reports.)\n"
    echo "\t\t\tq - Quit.\n\n"
    echo "\tPlease type the number of your selection"
    echo "\tand press the RETURN key  > \c"
    read SELECTION
    case ${SELECTION} in
        1)  ${BIN}/monitor.1 ;;
        2)  ${BIN}/monitor.2 ;;
        3)  ${BIN}/monitor.3 ;;
        [Qq]*)  exit ;;
        !*) COMMAND='echo ${SELECTION} | cut -c2-'
            eval ${COMMAND}
        echo "\n\tPress return to continue > \c"
        read RESPONSE ;;
        *) echo "\n\t${BELL}Unknown option - please try again."
           echo "\n\tPress return to continue > \c"
           read RESPONSE ;;
    esac
done
exit 0
```

Form–filling

Another style of interface that is sometimes called for is form–filling. Here, for example, is an outline for a form–filling front–end to `pr`, the draft formatter:

```
# fpr - form-filling front-end to pr

CLEAR=`/usr/bin/tput clear`
REVERSE=`/usr/bin/tput smso`
UL=`/usr/bin/tput smul`
NORMAL=`/usr/bin/tput sgr0`
```

```
LINE="--------------------------------------\
----------------------------------------"
COMMAND=/bin/pr

trap 'echo $CLEAR ; exit' 0 1 2 3

echo "${CLEAR}\n${REVERSE} pr - draft \
formatter ${NORMAL}\n\n"

echo "${LINE}"

while test -z "${FILE}"
do
    echo "\n\t${UL}Please enter \
filename${NORMAL}..... \c"
    read FILE
    case ${FILE} in
        '')     /usr/bin/tput bel
                /usr/bin/tput cuu1
                /usr/bin/tput cuu1 ;;
         *)     break ;;
    esac
done

echo "\n\t${UL}Title to use (RETURN for \
filename)${NORMAL}... \c"
read TITLE
case ${TITLE} in
    '')     ;; # default is filename+date+page-no
     *)     COMMAND="${COMMAND} -h ${TITLE}" ;;
esac

echo "\n\t${UL}Page length (66)${NORMAL}......\
..... \c"
read P_LNGTH
case ${P_LNGTH} in
    '')     ;; # default is 66 lines per page
     *)     COMMAND="${COMMAND} -l${P_LNGTH}" ;;
esac

echo "\n\t${UL}Line length (72)${NORMAL}......\
..... \c"
read L_LNGTH
case ${L_LNGTH} in
    '')     ;; # default is 72
     *)     COMMAND="${COMMAND} -w${L_LNGTH}" ;;
esac

echo "\n\t${UL}Single or double-spaced \
(S/D)${NORMAL}..... \c"
read SPACING
case ${SPACING} in
    [Dd]*)      COMMAND="${COMMAND} -d" ;;
        *)      ;;
```

```
esac

echo "\n\t${UL}Send output to \
(Terminal)${NORMAL}..... \c"
read PRINTER
case ${PRINTER} in
        # default is the terminal
    '') ;;
        # pipe the output to PRINTER
     *) PRINTER=" | lp -d${PRINTER}" ;;
esac

echo "\n${LINE}"

echo "\n${REVERSE} Command is:\
 ${NORMAL}${COMMAND} ${FILE} ${PRINTER}"
echo "\n\t${UL}Execute (Y/N)${NORMAL}? \c"
read X
case ${X} in
    [Yy]*)    eval "${COMMAND} ${FILE} ${PRINTER}" ;;
        *)    ;;
esac
exit 0
```

This is no more than an outline. Obviously, it would be helpful to know early on that the file to be formatted exists and is readable.

Cursor addressing

In UNIX System V Release 3 you can type:

```
$ tput cup 10 5
```

to move the cursor to row 10, column 5. In UNIX System V Release 2 it's not quite so easy.

There are two ways of doing cursor addressing in UNIX System V Release 2. The first way is fast, but not really to be recommended because you have to hardcode the terminfo `cup` (cursor addressing) capability within your shell script. The second way is to call a C program.

The first way (in System V Release 2)

Here is an example of the first way, the hard way:

```
# cursor_pos - cursor addressing the hard way

# parameterised cup capability
case $TERM in
        vt100)  begin=^[[; mid=';'; end=H ;;
        50)     begin=^[a; mid=R; end=C ;;
esac
tput clear
echo "\n\tPlease enter ROW: \c"
read ROW
echo "\tPlease enter COL: \c"
read COL
echo "${begin}${ROW}${mid}${COL}${end}Hi there! \c"
exit 0
```

What this means is that to move the cursor to a selected row and column on a vt100 (for instance), you have to send the terminal:

```
ESC [

followed by the row number
(in decimal)

followed by a ;

followed by the column number
(in decimal)

followed by an H
```

How do you know this? Because you have looked it up in the terminal manual, or because you have tried typing

```
$ tput -Tvt100 cup | cat -v
^[[%i%p1%d;%p2%dH
```

And to find out what *that* means, you'll have look up *terminfo*(4) in the UNIX System V Programmer's Reference Manual.

The second way (in System V Release 2)

The second way is to call a C program. For example:

```
# del_reg - delegate registration
#           cursor addressing example

PS1=
CURSOR=`/usr/bin/tput cup`
CLEAR=`/usr/bin/tput clear`
UL=`/usr/bin/tput smul`
REVERSE=`/usr/bin/tput smso`
NORMAL=`/usr/bin/tput sgr0`
BEL=`/usr/bin/tput bel`
export CURSOR

# display screen form
echo "${CLEAR}"

cursor 1 25 ; echo "${UL} Delegate Registration\
 ${NORMAL}"
cursor 1 65 ; echo "`/bin/date '+%d %m %y'`"
cursor 4 5 ; echo "First Name: ..............."
cursor 4 35 ; echo "Last Name: ........................."
cursor 6 10 ; echo "Home address: ......................\
............."
cursor 7 24 ; echo "....................................."
cursor 8 24 ; echo "....................................."
cursor 10 10 ; echo "Post code: ........"
cursor 10 46 ; echo "Tel. no: ............."
cursor 13 10 ; echo "Car reg. no: ........."
cursor 16 5 ; echo "Company: ........................\
..................."
cursor 19 5 ; echo "Course: ............................."
```

```
cursor 19 46 ; echo "Room no: ...."

# read input
```

`cursor` is the C program.

Speeding things up

If you try `del_reg` as it is, it will run quite slowly. However, one simple trick will make it much faster:

```
$ del_reg > dr2.scr_vt100
```

So now, here is version 2 of the *del_reg* script:

```
# del_reg2 - delegate registration (version 2)
#            cursor addressing example

PS1=
CURSOR=`/usr/bin/tput cup`
BELL=`/usr/bin/tput bel`
export CURSOR

LIB=${HOME}/lib

# trap interrupts
trap ' ' 1 2 3

# display screen form
if [ "${TERM}" = "vt100" ]
then
      cat ${LIB}/ dr2.scr_vt100
elif [ "${TERM}" = "50" ]
then
      cat ${LIB}/ dr2.scr_50
else
      echo "Sorry, this example only works with vt100"
      echo " or Wyse 50 terminals" 1>&2 ; exit 1
fi

# read input
cursor 4 17 ; echo "${BELL}\c" ; read F_NAME
```

Finally, here is `cursor.c`. I stole it, then modified it slightly from Rod Manis and Mark H.Meyer's *The UNIX Shell Programming Language*, Howard W.Sams, 1986 (ISBN: 0–672–22497–6), which I think is the best of the four books about the shell (`sh`, `csh` and `ksh`) published in the last eighteen months. The good news for Berkeley people is that it should also run on your system.

```
/*
* cursor.c - move the cursor to a specified
*            line and column on the screen.
*
* cursor is intended to be used within a
* shell script to provide cursor addressing.
* So it's very handy for constructing
* prototype data-entry screens.
*
* Normal usage would be (within a shell script):
*
```

```
*      PS1=
*      CURSOR=`tput cup`
*      ...
*      cursor 5 10 ; echo "Please enter your name: \c"
*
* Alternatively (from the command line):
*
*      cursor 5 10 `tput cup`
*
*
* Compile as: cc -O -s -o $HOME/bin/cursor cursor.c -lcurses
*
* or (for Berkeley systems)
*              cc -O -s -o $HOME/bin/cursor cursor.c -ltermlib
*/

#include <curses.h>
#include <term.h>

#define USAGE  "Usage: cursor line col0
#define EUSAGE 1
#define NOCURSOR "%s: no CURSOR argument or \
                          $CURSOR shell variable.0
#define ENOCURSOR 2
#define LINE 1
#define COL 2
#define CURSOR 3

char *tgoto();
char *getenv();
main(argc, argv)
int argc;
char *argv[];
{

    char *cup;

    setupterm(0,1,0);

    if (argc < 3) {
        fprintf(stderr, USAGE);
        exit(EUSAGE);
    }
    else if (argc == 3) {
         if (cup = getenv("CURSOR")) {
        /*
         * tgoto is an old function, included for
         * compatibility with termcap.
         */
         printf("%s", tgoto(cup, atoi(argv[COL]),
                                 atoi(argv[LINE])));
         } else {
            fprintf(stderr, NOCURSOR, argv[0]);
            exit(ENOCURSOR);
```

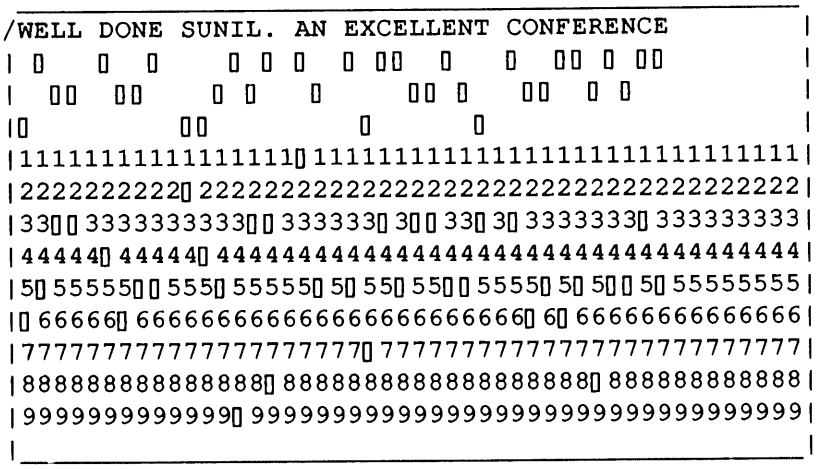
```
            }
        } else {
        printf("%s", tgoto(argv[CURSOR], atoi(argv[COL]),
                                        atoi(argv[LINE])));
    }

    reset_shell_mode();
    resetterm();
    exit(0);
}
```

In the next issue

That's it for this column. Let me know (by e–mail) whether you find this stuff interesting or useful. In the next issue, unless anyone suggests a better topic, I'm planning to write about efficiency considerations and shell functions in the System V shell.

Found on the Floor ...

Found by Mick Farmer on the floor of the QEII conference centre ...



C++ In Print

John Carolan
john@puschi.uucp

Glockenspiel Ltd.
Dublin

John Carolan is the current chairperson of the Irish UUG. He is also managing director of Glockenspiel Ltd. of Dublin. Glockenspiel has been using C++ since 1985, and John has presented several technical papers on C++. His present work includes the development of C++ class libraries common between OS/2 and X-Windows on UNIX.

In March '88, the only book you could buy on C++ was Bjarne Stroustrup's.

Come April, you're not exactly spoilt for choice, but clearly the deluge is beginning.

First of all, there's conference proceedings.

The Santa Fe conference on C++, I reviewed last issue. Also included in the conference proceedings are a few really good papers that the editor thought would interest people...

Experience in Using C++ for Software System Development

by Bill Hopkins

An excellent paper by a real end-user of C++. Bill's team at AT&T in Denver built a very large software product in C++ before anyone had heard of C++. The paper describes C++ concisely and enumerates the benefits that accrued from using it. We badly need more papers like this from practitioners of object-oriented programming.

Possible Directions for C++

by Bjarne Stroustrup

Catalogues the opportunities which the Perpetrator sees for improving the language. Top of the list are 'parameterised types' (which would give to C++ classes a functionality similar to generics in Ada) and 'exception handling' (which would improve the interaction of `setjmp/longjmp` constructs with scoped operations such as destructor invocation).

A Set of C++ Classes for Co-Routine Style Programming

by Stroustrup & Shopiro

Describes a working implementation from the Labs. Rumor: release 2.0 of AT&T's C++ will have a better 'tasks' library.

C++ vs. Lisp

by Howard Trickey

Compares by code-size, compile-time and various similar metrics the same slightly expert graphics software done in C++ and LISP. In my opinion, it is totally worthless because:

1. The LISP used was an obscure deviant from Common LISP;
2. The architecture of the program was preserved instead of being redesigned to use inheritance and polymorphism Despite the unreasonable basis for comparison, the paper comes out strongly in favor of C++.

Several other papers included are of the general form "Let's hack on the C++ language to do X". For the present, I believe the only guys who should

hack on the language are Bell Labs — otherwise we will end up with a disaster instead of a standard language.

Next, on the subject of conferences, the EUUG conference in London included many papers describing projects implemented in C++. The proceedings include three papers ostensibly about C++.

Yacc Meets C++

by Steve Johnson

This paper presents an extension to yacc to allow inheritance in `yacc` grammars.

Formatted I/O in C++

by Mark Rafter

This paper illustrates magnificently how you can add functionality to C++ by elegant extension of an existing class library — in this case 'streams'.

Software Re-Engineering using C++

by Anderson & Gossain

Compares the same program in C and C++. The approach here was to redesign a graphics presentation program in an object-oriented way and implement it in C++. The comparison consists of both qualitative issues such as re-usability and metrics such as a count of magic numbers present in the code.

Now, on to the first text-book on C++ since 1986.

An Introduction to Object-Oriented Programming and C++

by Wiener & Pinson
Addison-Wesley
ISBN 0-201-15413-7

The book presents a good introduction to the jargon of OOP, at least until it gets to page 8. On page 8 it presents the first of many serious mis-understandings. The book tries to describe the process of designing an inheritance hierarchy. It uses for an example an automobile with components such as engine and gear-box. It implies that engine and spark-plug should be derived from automobile, which misses the following point: Functional decomposition of a problem and class derivation are orthogonal design processes. Both processes must be pursued to achieve a good solution. The book also confuses bottom-up design with specialisation. Apart from the view of OO design being seriously damaged, the book does not consider that OOP should extend to human interface issues at all. Most of the worked examples come from older books on algorithmic programming. Hoare's QuickSort algorithm and much list-processing code occur in the book. Why I object to these examples is that they dissipate the reader's energy on following the algorithms. It must be possible to choose examples which illustrate the OO approach to design without so much implementation details. Examples which deal with human interface to some extent would be more interesting and could show how OOP assists consistency in the human interface.

As a textbook on C++, W&P has serious flaws. I counted over twenty errors. Here's an example from page 126:

> "The public qualifier in the derived class definition indicates that objects of the derived class can use all the methods of the parent class, unless these methods have been redefined in the derived class."

Stroustrup's book may be hard going, but at least it is accurate. Nowhere does W&P offer a Rosetta Stone for translating between C++ terminology and OOP terminology. It calls all data members of classes 'fields', for example. Now, in C and C++, only members with the form:

`< declarator : constant-expression >`

are called fields. It confuses objects with classes and declarations with definitions in several places. For example, on page 21:

> "The operators in the cin, cout and cerr classes can be overloaded within a programmer-defined class."

W&P is very misleading on the interaction between `typedef`, in built types and classes. The book omits several important features of C++. The most noticeable being programmed assignment, programmed conversion, coercions and pointers to member functions.

The last chapter contains three example programs. The first is a spelling checker. In it, there is a class 'word' which has a member function `'open_file()'`. Now, how can a word have a behaviour such as open file? In a book which attempts to instruct people on OOP, this is unacceptable. There should be a class such as WordFile on which Word is not dependent. The second example is a golden oldie from Simula which does an event-driven simulation of queues in a bank. The book follows the OOP approach nicely until it comes to the human interface. Then ZAP!, objects are nowhere to be seen. The output from the

program is very difficult to interpret simply because the OOP paradigm gets dropped. The third example is a simple function Interpreter. It parses a calculator language using an expression tree and it does syntax checking using a finite state machine. The syntax checker does not re-use any of the code in the parser. However, the program is riddled with friend declarations, which weaken the abstractions unnecessarily. The finite state machine declares each state as a separate class with one instance, which immediately rings warning bells. Well, the warning bells are correct. It is simpler to code this particular FSM in a table-driven form. The classes should be state transition tables, not states. If you are in any doubt about my criticism of this example, then repeat the following exercise: Add a unary minus operator to the Function Interpreter and count in how many places you have to change the code. Lastly, this example restricts the user interface just because W&P want to use operator overloading in a place where it is not appropriate. The user of the program types a formula where the arguments are mentioned explicitly. Each invocation of the formula asks the user to say how many arguments again. Not a good user interface.

In summary, the book covers neither nor C++ very well. It needs a lot more proof reading by practitioners of C++ before I would recommend it to someone learning C++.

Tech Tip

Here is a posting from Jonathan Shopiro which describes the usage of assignment and initialisation operators very neatly...

If a copy-initialiser is not supplied by the programmer, the compiler generates the default version, which looks like the following for a class with arbitrary base classes and members:

```
class D : B1, B2 ... {
        T1      m1;     // m1 is a data member, T1 is a type
        T2      m2;
        ...
};

D::D(const D& d) : B1(d.B1), B2(d.B2) ... m1(d.m1), m2(d.m2) ... {}
```

Each initialisation is the copy-initialisation for the appropriate type. They are executed left to right. Of course, such copy-initialisation is usually optimised to "blast the bits from the source to the target". If you leave the `const` specification out, you won't be able to copy a constant object. Following the same principles, the default assignment function is:

```
D&
D::operator=(const D& d)
{
        *(B1 *)this = (B1 &)d;
        *(B2 *)this = (B2 &)d;
        ...
        m1 = d.m1;
        m2 = d.m2;
        ...
        return *this;
}
```

It too is usually optimised to "blast the bits."

STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS ST OP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS

This year's C++ conference will be in Denver, Colorado October 17 to 20. If you would like to submit a paper then send a 2 to 4 page abstract to:

Andy Koenig
fax: +1 201 580 4127
uucp: ....!research!ark

before June 14th

STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS ST OP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS STOP PRESS

Price for proceedings

The Santa Fe conference proceedings are available from the EUUG secretariat (Owles Hall) at a price of £30 includings postage & taxes.

This is the best read around on C++ that you will find in Europe. There are only a limited number of copies — don't be disappointed.

EUnet in Finland

Peter Houlder
uknet@ukc.ac.uk

Computing Laboratory,
University of Kent

Introduction

This time we are grateful to Finland for this article. Juha Heinanen has provided a clear interesting account of the Finnish network. We now have had details on Finland, France, The Netherlands, Sweden, The United Kingdom and West Germany along with details of the overall network structure. Hopefully the other missing EUnet countries will soon fill in gaps, but other articles on:

Type of Mailers
Mail Interfaces
File Transfer
X400
News Software
Local and Wide Area Networks

and any other aspect of networking would be very welcome. Please send all contributions to the above (uknet@ukc).

Juha Heinanen
jh@tut.fi

Tampere University of Technology
Finland

Juha Heinanen is currently an associate professor in computer science at Tampere University of Technology in Finland. Networking has been his hobby since 1983 when we established the EUNET link from University of Tampere to Enea Data Ab in Sweden. Juha is also member of the steering committee of the Finnish University and Research Network Project (Funet).

EUnet in Finland

History

This all began in 1983 when a UUCP mail link was established from University of Tampere to Enea Data Ab in Sweden. The first UUCP machine was a PDP 11/34 and, of course, its address/disk space was too small to run the news software.

In 1984 Helsinki University of Technology (HUT) got a link to mcvax and the enea link was transferred from Tampere University to the neighbouring

Tampere University of Technology (TUT). The news was received by TUT from enea while HUT handled mail with mcvax. In 1985 FUUG (Finnish UNIX Users' Group) established Finland's first official EUnet backbone at Penetron Oy, which in 1986 also started to receive the news directly from mcvax. Mainly because of shortage of 'free' manpower at Penetron, the official backbone was moved in the beginning of 1987 to TUT.

TUT now handles international connections and serves the 'rural' parts of the country. The sites in the Helsinki area are served by HUT that has taken the responsibility of a second backbone.

Rate of Growth

During the five years of existence, EUnet in Finland (SFNET) now consists of 46 sites 11 of which receive the news. The main reason for the low number of news sites is the high cost, which currently is around $200/month.

The number of sites has grown steadily at the rate of some 10 sites/year. The new sites are almost exclusively commercial companies, typically software houses, since almost all of the universities are already subscribing to SFNET.

A new development at universities has been the installation of Internet domain naming in computer centre mainframes running VMS and VM operating systems. This has increased the international through traffic at TUT considerably, since earlier SFNET services were mainly used by computer science departments.

At the time of writing, 25 organisations have registered an Internet subdomain under .fi.

Backbone Personnel

At TUT, the mail and news service is now in practice run by Vesa Keinanen and I'm proud how fast he has become a real guru, mastering the mysteries of sendmail and other networking software. As a backup, we have Hannu-Matti Jarvinen who has several years' SFNET experience. At HUT, the work is done by Jukka Virtanen who runs a machine called santra. Jukka also maintains the EARN and CSNet gateways. Currently SFNET is totally uncommercial, i.e., we don't charge the sites for any of the salaries or other overhead. If such costs were added on top of already high news costs, the network would probably become too expensive for some of our current subscribers.

Links with Other Networks

SFNET has direct gateways to FUNET's (Finnish University and Research Network) TCP/IP and DECNet networks, EARN, CSNet, and the RARE experimental X.400 network. All other networks are reached via mcvax or enea.

Future Plans and Problems

All the universities in Finland will soon by linked by 64Kb leased lines managed by FUNET. For SFNET this means division of the sites into two quite different groups: the commercial companies running UUCP over X.25 or telephone lines, and the academia running LAN-protocols over the fast backbone.

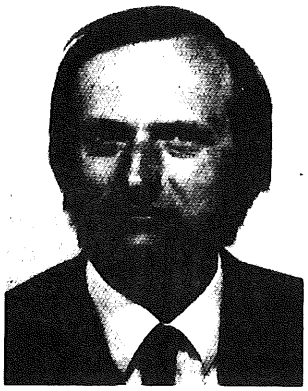
So far all SFNET traffic has been allowed free of charge on the FUNET backbone. In the future, some kind of formal agreement might be needed between SFNET and FUNET concerning the through traffic. The relations between SFNET and FUNET are very good and there has been no pressure from either side to take over other's operations or compete with them.

For the future benefit of SFNET, it might be desirable to find a wealthy commercial company that would take responsibility to develop further the non-academic part of SFNET. An ideal site would be one that has a large research department with a lot of contacts with both universities and software companies.

At the time of writing there are very interesting developments going on internationally, which hopefully will lead to a better and cheaper EUNET infrastructure within Scandinavian countries and also to Central Europe. The next report will tell how this all worked out.

X/Open Midterm Report

John Totman
Director of European Program



X/Open Company Ltd.
Lovelace Rd
Southern Industrial Estate
Bracknell
Berks RG12 4SN
UK
+44 344 424842 Extn 2748

John Totman is an electronics development engineer who has been involved in the engineering support, development and marketing of operating systems since the early 1970's.

He more recently strayed into UNIX when managing the development of commercial applications for departmental users and became a convert to the cause of standards and applications portability.

John is now a full time employee of the X/Open company and is responsible for European marketing activities.

Opening wide the X/Open Door

As we planned, X/Open has now widened its industry influence and involvement in two ways: by increased share holder memberships and by the establishment, in 1987, of our User and ISV Advisory Councils.

In terms of shareholders, NCR and Sun Microsystems joined in January 1988 bringing out total membership to 13 of the world's largest computer vendors. Our continuing growth in membership is acknowledgement of the unstoppable market trend towards vendor independent computing.

Our advisory councils, formed from major users such as General Motors, Eastman Kodak, British Airways, Daimler Benz, and software vendors such as Uniplex, Informix, Oracle, Quatratron, and Multihouse give direct advice to X/Open on the future development and expansion of the Common Applications Environment. Both these Councils meet regularly with X/Open and have already provided valuable input to our future plans and strategies.

In addition to the ISV Council, which can only be a small group to be fast moving and responsive, we have also established a programme of wider software vendor participations in X/Open by means of our Software Partnership scheme. This scheme is designed to give direct support to software vendors though out Europe and the USA who wish to produce and market software to the X/Open standard.

Software vendors wishing to take part in the scheme should contact me at the above address.

Technical "Hot Spots"

On the technical front, POSIX has been a major area of activity for the X/Open Technical Committees. By the time that this is published, we expect POSIX (i.e., P1003.1) to be formally endorsed as the official full use IEEE standard. On this basis X/Open will be incorporating POSIX functionality into the 1988 edition of the *Portability Guide*.

In addition to POSIX convergence X/Open has also developed a standard transport interface definition for networking (XTI) which provides a protocol independent system program transport interface; a user interface (a windowing standard based on X-

Window); the COBOL definition has been updated from a COBOL 74 to a COBOL 85 base; and the the GSA standard ADA has been adopted. All these will be published in the 1988 edition of the Portability Guide.

1988 will also be the year that verified X/Open conforming systems will be available in the market. Although compliant systems from X/Open members have been available since early last year it is only now that formally verified and branded systems will be available. This has been made possible by the formulation of conformance guidelines, establishment of public verification centres, and, later in the year, licencing of the X/Open system supplier to test for compliance to the X/Open standards and to brand compliant products.

1988 Outlook

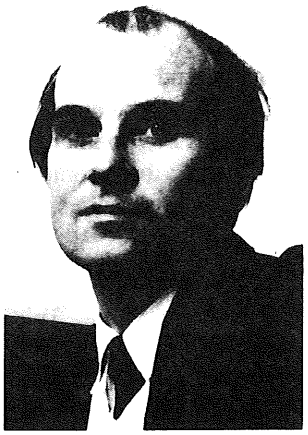
An aggressive marketing activity in the USA, coupled with the establishment of the X/Open Software Partners programme, is now attracting a wide base of industry support for X/Open standards. It is out expectations that by the end of 1988 major users, both in the US and Europe, will be specifying the X/Open Common Applications Environment as a procurement standard, confident that compliant products will be available from many suppliers.

Open Comments

X/Open now publishes a quarterly magazine. This is called *Open Comments* and is for those who wish to know more about developments within X/Open and its standards. If you would like to receive a copy please contact me.

Receiving News at a Small Commercial Site: Is It Worth It?

Dominic Dunlop
domo@sphinx.co.uk



Sphinx Ltd.
Maidenhead
United Kingdom

Dominic Dunlop is the Research and Development director of Sphinx Ltd, a UK software distribution and services company he co-founded in 1983, after experience in supporting Zilog's range of super-micro computers. Sphinx centers its operations around non-proprietary operating environments, selling in a variety of third-party and self-written software products across hardware from name different vendors.

Dominic's current rôle is that of bringing complex new products into Sphinx' offering by first understanding the technical and marketing issues involved, then working to address them in the context of the company's current capabilities and activities.

My company, Sphinx Ltd., subscribes to Usenet, the UNIX world's distributed bulletin board system[1]. In the past, the net's user base has been dominated mainly by academic institutions, and by large, technically-oriented commercial companies. Today, however, more and more small companies are subscribing, even though the subscription involves significant costs, both in hard cash and in staff commitment.

Why is this? Trying to answer this question — in response to a query which had appeared on the net — I came up with the following summary of the benefits, the costs — and the traumas — of bringing news into a young, marketing-based company.

Justification

1. Access to the net keeps technical people happy. Considered this way, it's as much a perk as the coffee machine (and about as expensive to run — see below).
2. The net keeps us well informed of what's happening in the UNIX world, and, to a slightly lesser extent, in the industry at large. As a consultant, I find this knowledge increases my worth and the usefulness of my services to others, both inside and outside the company. And the non-computer-related postings broaden my mind!
3. As my company sells software products, it is very useful for us to learn of the good and bad experiences of existing users of software we already carry, which we may carry in the future, which is competitive to products we carry, or to which we may have to interface. Receiving this information from US sites, which tend to get product ahead of Europe, is an added bonus. Similar arguments undoubtedly hold for suppliers of computers and peripheral hardware.
4. My company runs what has come to be called a heterogeneous installation: we have many types of computer system from many different suppliers. Administration is a constant problem, as no standard tools yet exist for this task in such an environment. Sources and hints from the net have helped us on a number of occasions, the most recent being when the

1. Strictly speaking, we subscribe to **uknet**, the Britsh part of **eunet**, which, in turn, links to the US **Usenet**.

public domain *tar* was distributed late last year. We now use this program, which has compiled without problem for us in several environments, for cross-machine back-up as a matter of course. On its own, this program would justify many months of network charges.

5. The net is an educational tool: it can teach readers much about many aspects of programming.

Problems

1. You can spend a lot of time reading news — time when you ought, perhaps, to be doing other things. An intelligent browser, such as *rn*, is *de rigeur*. Even then, users have to spend the time to learn how it can best help them to cut down the volume of the news they see to a manageable level.

2. You can also spend a lot of time replying to news[2] (although surveys indicate that the vast majority of news readers are passive, and don't post followups).

3. Getting the news service running can be like searching for the light switch in a strange and totally dark room, involving a lot of groping around, and falling over things you didn't know were in your way. The situation getting better, however — see below.

4. News needs maintenance. I'd estimate that it costs us between two and four man-hours per week.

5. Good references and 'how-to' information are hard to find. Of course, user guides are posted to the net, but, firstly, you can't see them until the news service is running — the classic bootstrap problem; and secondly, they don't address the issues of administration and management at all. I can recommend two Nutshell handbooks published by O'Reilly and Associates: *Managing uucp and usenet* (ISBN 0-937175-09-9), and *Using uucp and usenet* (ISBN 0-937175-10-2). They are obtainable from specialist UNIX book stores[3].

6. News uses a lot of disk space, and quite a lot of CPU cycles and memory. The latter can be noticeable if several users are reading news at busy times of day. (The CPU costs of receiving news generally occur at night, and so don't cause problems.)

7. News is distributed on a 'best efforts' basis: failures at points throughout the network are commonplace, and can result in degradation or total loss of service for indefinite periods. To put it another way, the net is not a reliable medium. The level of service delivered would not be acceptable were it provided on a commercial basis. For a non-profit co-operative venture, it's what one might expect. The USA, until recently badly overloaded and disorganised, is getting a lot better as result of the commissioning of the central site, *uunet*. Europe, on the other hand, was running very smoothly until recently, but is beginning to creak a bit. (Plans are afoot to address these problems on short order.)

Practicalities

1. My company acts as a leaf node: we get our feed by polling Reading university (*reading*), which is a local telephone call away, at 1200 baud, and feed no other sites. The feed is compressed, giving a 'worthwhile' saving in transmission times, and, consequently, line charges.

2. We run news on an Olivetti/AT&T 3B2/400, and, following disk overflow problems on shared file systems, have dedicated a 17 MB file system to */usr/spool/news*. This is marginal: we keep the volume of news held on-line in check by not having a number of groups delivered to us, and by expiring a number of groups after one week rather than two. I'd estimate that 30 MB would be needed to have enough headroom to meet all eventualities. This is a lot of space on a comparatively small shared system. In addition, at least 2 MB of spool space are needed under */usr/spool/uucp* as temporary storage for incoming news. (For non-leaf sites, the situation is worse: they also need *uucp* spool space for outgoing files, which can quickly mount up if leaf sites fail to poll or respond to calls from their feed for any reason.)

2. For example, I spent a morning writing this article in response to a news posting...

3. And also (quick commercial) from Sphinx Ltd.

3. Apart from machine costs, which we do not budget, news costs us about $300 per quarter, split evenly between phone charges and our share of uknet's costs. These cash costs include some modem usage other than that required solely for news — in particular, UNIX mail, and cu and *kermit* dial-out connections to other sites. Uknet would charge us less if we provided primary news feeds to other sites. However, this would require us to dedicate to news more machine resources and more administration time than we do at present.

4. Getting news running was a real game, not least because it was all done in people's spare time, and not project-managed in any way. We obtained the source code from EUUG's 1986 public-domain software distribution tape (fine if you have a nine-track tape drive...), and compiled for our target by trial and error. (By the way, compiling *rn* was no problem at all: the program has one of the best installation scripts that I've ever seen.) Initial set-up was also a problem: the Nutshell books can help a lot here. Users of SCO XENIX and some other variants of UNIX may be able to get unsupported netnews binaries through the organisation which provides their operating system support. This should help quite a bit, but still leaves a fair amount of work to be done. All in all, I'd estimate the cost of getting our site reliably connected to others over dial-up links (including the reception of news) at approaching a man-month — decidedly non-trivial. This time could have been reduced had we not been so ambitious (for example, we spent a fair amount of time creating a script to drive a multi-standard smart modem bidirectionally).

5. Getting a news feed was quite easy, because I knew who to ask. Officially, you go to the central site in your country (note the bootstrap problem again), and they refer you to likely local feed sites, leaving you to take it from there. We know *uucp* well, and it took us about a week to bed down our connection to Reading. Again, the Nutshell books can take some of the pain out of this process.

6. The mailers delivered with systems are either dumb, or have a dumb default configuration. We've had to install a public-domain mailer (*smail*) on our 3B2 so as to handle domain addressing, and to make a reasonable attempt at handling the weird return addresses and paths that pop out of news postings. Even now, we're relying on *reading* and *ukc* to do most of the hard routing work. I anticipate any site that is serious about the news service and Usenet will find that the configuration job involves in similar work. (Happily, more and more systems are being delivered with *sendmail* these days, so you merely have to edit the configuration file.) (A touch of satire there...)

So, then, have the benefits of the news service outweighed the effort and expense of getting it running at our site?

I'd say that they have.

If you don't yet subscribe to news but want to, I hope this article has either given you some positive arguments to take to your management — or has spurred you towards presenting a *fait accompli* after you've got news up and running!

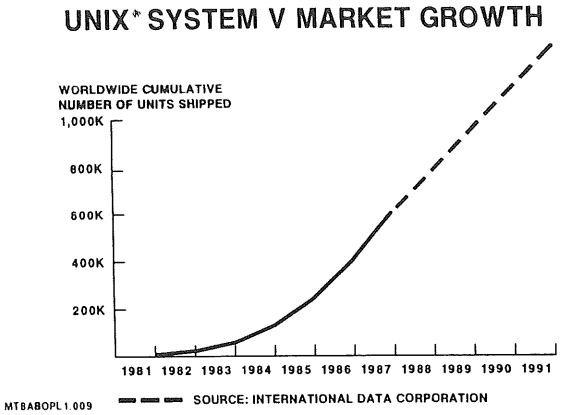
Unification and Openness

Janet Davis
janet@uel.uucp

AT&T UNIX Europe

AT&T announced plans for the next major release of UNIX System V and reaffirmed the openness of the operating system at UniForum in Dallas, Texas, in February of this year.

As statistics show more computers users are moving to UNIX Systems, In the last year the number of shipments of computers based on the UNIX Operating System increased by 60% and the International Data Corporation estimate that this growth will continue at an annual rate of 30% over the next three years.



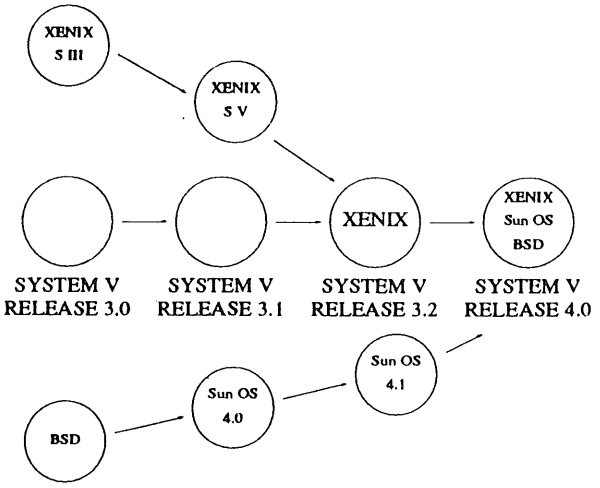
AT&T's plans for the next release of UNIX System V will support the growth of the UNIX Systems-based market. Two critical factors behind the success of UNIX Systems have been the move towards standardisation and the openness of UNIX System V.

AT&T's continuing commitment to both these areas will be backed up by a number of activities. AT&T will continue to provide both customers and the industry with information on the direction of UNIX System V. Seminars, such as the series planned for this summer covering UNIX System V Release 4.0, will be held on both technical and business issues. AT&T will also establish a UNIX System V OEM Licensee Group, open to all licensed commercial vendors. This group will meet regularly to share up-to-date information on UNIX System V developments and ideas for promoting the use of open systems. The aim behind all of these activities is to support the introduction of UNIX System V Release 4.0, and underscores AT&T's commitment to open systems.

Unification

Another major factor in the growth of the UNIX Systems market over the last year has been the move towards the unification of the major UNIX System derivatives, a move that will be of increasing importance over the next few years as the industry looks to a consolidation of the UNIX Systems market.

In 1987 the move towards unification occurred on two fronts. One being an alliance with Microsoft whereby XENIX System V compatibility would be consolidated into UNIX System V, the other being an alliance with Sun Microsystems whereby features of the Berkeley 4.2 and 4.3 systems (BSD) and Sun OS would be consolidated into UNIX System V.



a move that will be of increasing importance over the next few years as the industry looks to a consolidation of the UNIX Systems market.

In 1987 the move towards unification occurred on two fronts. One being an alliance with Microsoft whereby XENIX System V compatibility would be consolidated into UNIX System V, the other being an alliance with Sun Microsystems whereby features of the Berkeley 4.2 and 4.3 systems (BSD) and Sun OS would be consolidated into UNIX System V.

Application Binary Interface

As part of the strategy of unification AT&T have established a standard Application Binary Interface (ABI) for both the Intel 80386 and SPARC chip. Similar work is being carried out elsewhere for the Motorola 68000 chips. The ABI defines the binary interface just as the System V Interface Definition (SVID) defines the source code interface.

By providing application binary portability within an architecture, the ABI offers the type of portability normally associated only with PC applications which in turn will enlarge the market for software applications and preserve customers investments. Associated with the ABIs will be a Trademark Licensing Program allowing licensees to use the trademark UNIX to identify their sublicensed products.

Release 3.2

The first aspects of this unification work were seen at UniForum this year when AT&T demonstrated UNIX System V/386 Release 3.2. This is the first version of UNIX System V to incorporate XENIX System V compatibility and is expected to become the preferred multi-user, multi-tasking operating system for 80386-based computers. Running a wide selection of software, including software packages written for UNIX System V and XENIX System V for the Intel 80286 and 80386 chips, UNIX System V/386 Release 3.2 will also work with AT&T's Simul-Task software to allow users to run multiple MS DOS applications simultaneously. Licensing of UNIX System V, Release 3.2 will begin in mid-1988 with Release 3.2 for the 3B2 followed by the 80386 version. Sub-Licensees of the binary UNIX System V/386 Release 3.2 will have an option to license and use the trademark UNIX. Where the trademark is used the product must conform to AT&T's product and trademark usage specifications. For end-users this will mean that when they purchase an 80386-based system running UNIX System V they will be assured of the binary portability of applications.

Release 4.0

Release 4.0, the next major release of UNIX System V, will broaden the market for UNIX System software by further unifying the major derivatives of the UNIX Operating System. As well as incorporating features of XENIX and BSD and Sun OS, new features such as real-time capabilities, improvements to systems operations, administration and management, enhanced networking and features designed for international markets will be included.

The merging of BSD and Sun OS features into UNIX System V will take place in three phases. Phase 1 will see Sun OS 4.1 developed to comply with the SVID. In Phase 2 AT&T will merge Berkeley features into UNIX System V, Release 4.0 and add new AT&T features. This version will comply with the IEEE POSIX standard and will be consistent with the operating system specifications of X/Open. The third and final phase will see the restructuring of the kernel.

BSD and Sun OS features widely used in the industry will be incorporated into Release 4.0 so that users familiar with these features will find it easy to migrate to a single common UNIX System V base. Along with RFS, NFS will provide distributed access and operations on UNIX System file system objects. RFS and NFS will each be supported as an independent package with an administrative interface that will give an administrator a single, consistent view of distributed file access. Remote Procedure Call (RPC), a tool for developing distributed processing applications for a variety of manufacturers' hardware will also be incorporated. RPC lets a process executing on one machine issue

service requests to a process on another machine.

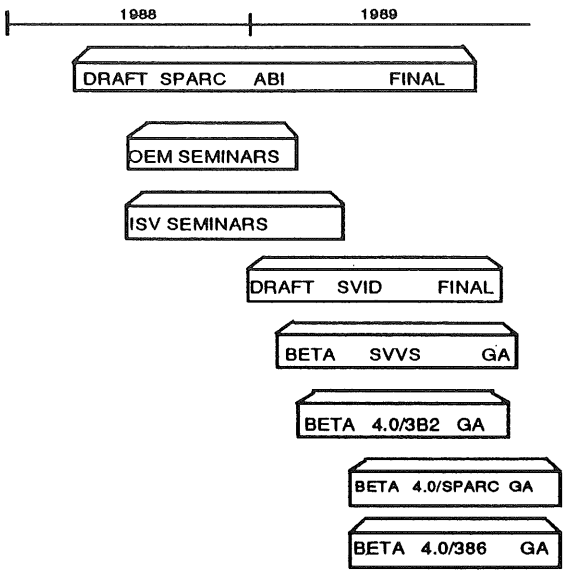
Significant enhancements are also planned in the areas of system operations, administration and maintenance will include backup and restore, configuration management software installation and distribution, message handling facility and remote operation. Real time enhancements include a scheduler, general events mechanism and asynchronous I/O while internationalisation features such as a C compiler that conforms to the ANSI X3J11 international standards and a message handling facility will be included.

Conformance to Industry Standards

With Release 4.0 AT&T reaffirms its commitment to supporting industry standards. Release 4.0 will conform to the forthcoming IEEE POSIX standard, which is due to be published and adopted in 1988. Release 4.0 will also be consistent with the operating system specifications of X/Open in line with AT&T's commitment of providing the industry with products that are compatible with the X/Open Common Application Environment.

When Will All This Happen?

In preparation for the introduction of Release 4.0, AT&T will be holding briefing seminars on Release 4.0 for computer manufacturers and software developers during 1988.



This year will also see the OEM Licensee Group established to set up a channel of communication and input on the development and direction of UNIX System V. As well as preparing much of the groundwork for Release 4.0, AT&T will bring UNIX System V Release 3.2 to the market in 1988. This release will first be delivered on the 3B2 and then on the 80386 chip. 1988 will also see work commence on a draft update of the ABI for the SPARC chip, work which will be completed some time in 1989. During 1989 AT&T will update the SVID and prepare a version of the System V Verification Suite for beta release to customers wanting to test their systems conformance to the SVID. Finally the move towards the unification of UNIX Systems will culminate in 1989 when beta versions of Release 4.0 will be made available first for the 3B2 followed by the 80386 and SPARC chips.

UNIX System V has been successful in the marketplace because of its consistency, openness, widespread availability and powerful capabilities. The next two years will see AT&T expand its open systems policy and continue to develop and enhance UNIX System V.

OPEN LOOK

AT&T have announced the OPEN LOOK user interface, it employs commonsense graphic symbols instead of written commands to help users work more efficiently with their UNIX System V-based computers.

"OPEN LOOK will change the way the industry thinks of the UNIX system," said Vittorio Cassoni, president of AT&T's Data Systems Group. "This interface brings the benefits of the UNIX system to a whole new group of users who otherwise might never have taken advantage of the power of a UNIX system-based computer."

The OPEN LOOK technology was designed for AT&T by Sun Microsystems Inc. of Mountain View, Calif. Sun's design is based on original work, contributions from AT&T, and on technology licensed from Xerox Corporation, which originated many of the concepts present in today's computer interfaces.

The OPEN LOOK interface's graphic symbols include push pins to 'pin' important menus to the screen for further reference and an elevator to move up or down in the text. To print or store files, users move a hand-held mouse to push labeled buttons designed to look like those on a household appliance.

As the name implies the OPEN LOOK user interface supports AT&T's commitment to open systems and the need for a standard user interface. Scott McNealy, president of Sun Microsystems, says that

it represents the next critical step in truly expanding the UNIX marketplace. Applications developed with the OPEN LOOK interface can vie for a larger market because the interface is standard.

The interface has already generated endorsements from key computer system suppliers, PC and workstation software suppliers and system suppliers.

OPEN LOOK combined with ABI, will provide a common application platform which has, until now, only been found on MS-DOS machines. Because of this OPEN LOOK has been endorsed by some of the big names in the PC world, these include: Lotus, Ashton-Tate, and Xerox.

In addition to being easy for them to learn, the OPEN LOOK interface will make users more productive because it allows them to create multiple 'windows' on their computer screens, each of which can perform a different task simultaneously.

Programmers will find that the various Application Programmer Interface (API) Toolkits AT&T plans to release will give them a set of tools — or pre-programmed components — to make it more efficient to write new applications by reducing the amount of code that needs to be written per function. As OPEN LOOK will have a defined program interface, software portability will be increased.

AT&T will circulate OPEN LOOK specifications for comment this summer and will make them available in the third quarter of this year. These will include a specification of the common style for applications — The Applications Style Guide — as well as descriptions of the programming interface for OPEN LOOK under two toolkits, both of which AT&T will support via a single graphics system platform. They are the XT toolkit based on the X Windows and the NDE toolkit based on NeWS.

The first availability of OPEN LOOK features in an AT&T product will be this summer in a window manager for the 6386 workstation, followed by an XT toolkit in the first quarter 1989.

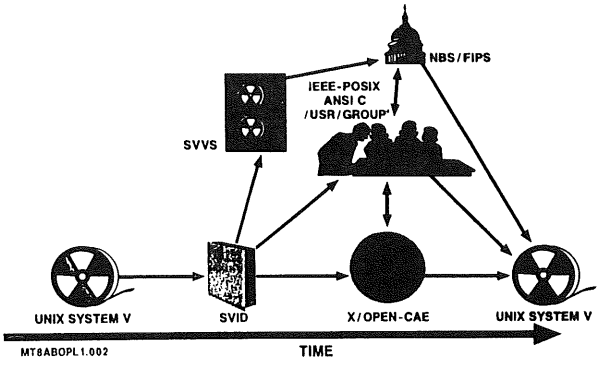
In keeping with its commitment to support standards, AT&T said that as they become accepted, the company would support API's for emerging standard interfaces. AT&T also will license source code for the various toolkits supporting the OPEN LOOK user interface.

The OPEN LOOK user interface toolkits are scheduled to be available in source form in early 1989.

The OPEN LOOK user interface is designed to be useful into the 1990's. For instance, unlike some graphical interfaces, the OPEN LOOK interface is designed for a wide range of applications from simple document processing to much more sophisticated computer-aided engineering (CAE). In addition, the graphics perform well whether they appear on a PC or a high resolution engineering workstation. Also, the interface will support a variety of terminals accessing different applications.

AT&T today also announced it will co-sponsor with Sun Microsystem a series of eight, three-day conferences around the world beginning in September to give independent software vendors, value-added resellers and large corporate users a preview of the key technical features of UNIX System V Release 4.0, including the newly announced OPEN LOOK interface.

UNIX® SYSTEM STANDARDIZATION



Book Review: 2 Books on ANSI C (Draft)

Reviewed by
Andrew Macpherson
andrew@stl.stc.co.uk

STC Technology Ltd.

The ANSI Draft standard has been out for public comment for almost a year, and we are now seeing the books which will guide us through the exact formal definitions to a working appreciation of where the language has gone.

There is an attitude which says that once a language is set in a standard it becomes useless. C shows little signs of this failing, but those who subscribe to this viewpoint can take heart: in both cases C++ was used to check the code!

The C Book

Featuring the draft ANSI C Standard.
Mike Banahan, The Instruction Set Series — Addison Wesley, 1988, ISBN 0–201–17370–0. (U.K.) price 15.95, Soft Back, 268 pp, Size 23.5 cm x 15.5 cm.

To many the acronym X3J11 is just so much gibberish. To the C programming community however, the J11 committee of ANSI's X3 secretariat is a medium term source of fairly fundamental change to the language. Mike Banahan's early involvement in that committee, and his well-known skills as a communicator, make him an ideal candidate to present the changes and the rationale for some of the apparent inconsistencies introduced or addressed by the Draft Standard.

Not content with this, *The C Book* is more than just a catalogue of changes and how they will affect the practice of programming and porting C. Its scope also encompasses a balanced tutorial on C for practising programmers who wish to become familiar with the language.

The format is very much an informal tutorial style — the difficult concepts are reserved for the later stages, and each new concept has an illustrative fragment of code to demonstrate its use. In fact the code fragments are usually complete working programs, all of which have been tested from the text (with one exception, noted in the text on page 207). Significantly, Mike recognises that some of the working of C is best reserved until one has experience of the language, and recommends that portions of the book be skipped, or at least skimmed, on first reading, and only returned to when the reader has six months familiarity with the language. Each chapter has, at the end, a review exercise to test the reader's understanding and full answers are given at the end of the book.

It works well. The first six chapters deal with the language *per se*, with only a few difficult points reserved for chapter 8. The style is easy to follow, and the occasional wry wit helps make points while improving the readability. Chapter 7, the pre-processor, becomes clear on a second reading, but I would advise skipping past the informal discussion of tokens at the bottom of page 158 as it tends to cloud the issue, only coming back to it if you have to. The abuse of the second language — the pre-processor — is well warned against:

"The urge to maim the author of a piece of code becomes very strong when you suddenly come across

```
#else
        }
#endif
```

with no `#if` or whatever immediately visible above."

Also don't be confused by the missing `__DATE__`, on page 166, it is explained on the opposite page.

Chapter 8 is a discussion of many things which are not, in general, needed to make use of the language, in particular the new and unfamiliar storage qualifiers `const` and `volatile`, along with a discussion of linkage, sequence points (where side-effects are resolved) and `typedefs`. This last is put in the realm of system header files, rather than a useful tool to the application programmer's hand, and may reflect, in part, the same historical

perspective that led to the explanation that `enums` did not exist when Mike learned the language.

The discussion of the standard libraries is divided by function (or header file), and quickly establishes that the UNIX interface has not been taken up *in toto*. Goodbye `creat`, goodbye `unlink`, hello `delete`. I expect this to be the most useful section of the book for me in the long-term.

This is not a book for the computer novice. It does not pretend to be. It is certainly suitable as a course-book both for home-study and formal instruction. I would rate it as essential reading for someone contemplating their first C language project, who needs to know what C will and will not do for them, the limits of the automatic error detection at compile time, and what is left to the programmer.

The C Programming Language

Second Edition, Based on Draft-Proposed ANSI C. Brian W Kernighan and Dennis M Ritchie, Prentice-Hall Software Series, 1988, ISBN 0–13–110362–8. (U.K.) Price 24.95, Soft Back, 272 pp, Size 23.5 cm x 17.7 cm.
Available — Real Soon Now

The White Book, K&R, the C programmer's bible — a book so deeply engrained into one's working environment for the past ten years that it is nigh on impossible to look objectively at it, and here it is in a revised second edition with the emphasis shifted to what may be.

This is not a new book, but it will be a best-seller in the UNIX community. It is a rewrite of the 1978 C book, with new examples, a C–declaration translator `dcl`, a new chapter on the ANSI standard libraries, and the benefit of improved diagrams (`pic`).

Saying it like that gives the wrong impression, however. The first edition of this book has for so long been the standard for the language, a sufficiently usable standard that compilers could be written and tested against the text, that every working C programmer must have had a copy. This second edition is a major revision of text, order, examples; it gives credibility to the ANSI standard.

How then does the new book differ from the old? Other than the new typesetting, and the unacknowledged font used in chapter and section headings, the text has been thoroughly revised. No more is it a tutorial and reference manual for C, rather a tutorial and reference manual for ANSI Standard C. All the new features that have been added to the language are dealt with, and considerable emphasis is placed on the standard libraries.

For those who want a quick introduction to the new language, Appendix C is a summary of the changes introduced by the standard, while Appendix A is the language reference manual. The reference manual contains the bones of a `yacc` parser for the new grammar, but otherwise follows the format of the first edition. Appendix B deals with the standard library, it is equivalent to chapter 9 of *The C Book*, with slightly less explanatory text, but is a more useful quick reference.

While usable as a study document, this, like the previous edition, is a working programmers book. Anyone seriously using ANSI C, particularly on UNIX, should have a copy.

Book Review: UNIX Products for the Office

Reviewed by
Tony Bamford
afb@phcomp.co.uk

Parliament Hill Computers Ltd.

UNIX Products for the Office

The National Centre for Information Technology, Published by The National Computing Centre Ltd., 1988, ISBN 0-85012-701-7. Price (UK) 45, Soft back, 266 pp.

Its all very well knowing where your next GNU EMACS update is coming from but what do you do when you want a prospect tracking system to run on your UNIX box? Never fear, NCC have the answer.

UNIX Products for the Office is a catalogue of UNIX products available within the UK that are designed and intended specifically for the office environment.

It is divided into 4 main parts, an introduction, a detailed description of products, an index by description and an index by supplier.

The introduction deals with what UNIX is, why UNIX is wonderful for office applications and how to choose a UNIX system if you don't have one already. The product description forms the major part of the handbook and consists of about 200 pages, each product getting half an A4 page. The half page of information gives a few lines describing what the product does, a list of applications for which the product is suitable, a list of users the product addresses and some details of the type of hardware needed to run it. Then come financial matters, how much it costs to buy or, where applicable, rent and how much annual maintenance costs. Finally a UK supplier of the product is named.

This book is a great idea, a guide to UNIX products in the marketplace is long overdue.

However, there are some problems with the catalogue, a fair proportion of the entries don't mention prices. Also, only one supplier is listed per product, this is to be expected for software written by small software houses, but one UK supplier for INFORMIX? Other problems appear when the catalogue is used and suppliers say "sorry, we don't deal with that product any more".

The key to the problem is outlined in the introduction to the catalogue, which states:

> Any volume of this type is absolutely dependent on the good will of the product suppliers in completing the questionnaires ...

This implies that many suppliers returned half-completed questionnaires. Thus, we can only hope that the book becomes popular and suppliers are encouraged to provide the NCC with complete information.

So my conclusion is good as far as it goes, but the second edition should be more useful.

EUUG Conference Proceedings

Here are the abstracts of the papers delivered at the EUUG 10th Anniversary Conference (Part II) in London this April. Please contact the authors if you would like a copy of a paper.

Thanks are due to Stuart Mc Robert (sm@doc.ic.ac.uk) who also typeset the proceedings.

OFS — an Optical View of a UNIX File System

Paulo Amaral

GIPSI-SM90[1]
c/o INRIA
BP105
78153 LE CHESNAY CEDEX
FRANCE
mcvax!inria!gipsi!paulo

Abstract

The design and implementation of the Optical File System (OFS) is described. It was conceived to run under UNIX and to deal with Optical Disks. We explain our view on how to develop a file system, at UNIX user level, with a WORM (Write Once—Read Many) device. OFS manipulates multiple file versions automatically. It also works upon an implementation of atomic transactions: fault tolerance implications are studied. Finally, we describe our experience using the OFS by means of a backup utility, that has been used by our software research team since October 1987.

Software Re-engineering using C++

Bruce Anderson
Sanjiv Gossain

Electronic Systems Engineering
University of Essex
bruce@ese.essex.ac.uk,
goss@ese.essex.ac.uk

The plan for our experiment was to take a piece of software and rewrite it in C++. We wanted the program in question to be locally-generated, written in C, widely used and to be a generic program, one which was typical of a class of programs that were either actually written or likely to be needed. Our idea was to proceed in small steps and to reflect on each step.

Measuring File System Activity in the UNIX System

Maurice J. Bach[2]
Ron Gomes

AT&T Information Systems
190 River Road, Summit, NJ 07901

Abstract

We describe an analysis of system call activity (particularly file system activity) made on several UNIX systems in a software development lab. The measurements were motivated by design work in support of distributed file systems; the intent was to characterise such things as system call frequencies, file system access patterns, and caching behaviour, and to identify performance bottlenecks which might have been missed by existing measurement tools. Among the more important results of the study:

- Most system call activity is file system activity.
- Most *reads* and *writes* are not matched to the file system block size.
- Most terminal I/O is done a single character at a time
- Operations on directories dominate buffer cache activity even though only a small part of the cache contains directory data.
- Most buffer cache hits result from repeated access by a single process.

A UNIX Environment for the GOTHIC Kernel

Pascale Le Certen,
Béeatrice Michel,
Bull Recherche.

Gilles Muller,
IRISA-INRIA
Campus de Beaulieu, 35042 Rennes-CEDEX

Creating an operating system on an open machine, implies the use of development methods. This report describes the way chosen to implement the kernel of the GOTHIC distributed system and a well suited environment. Our goal is to design a development system which can run on the successive versions of the kernel. The major advantage of that method is to intensively test the kernel for programming and design errors.

UNIX Around the World

Sunil K Das

City University London,
Computer Science Department
London EC1V 0HB, UK
sunil@cs.city.ac.uk

Abstract

In the Preface to the Eighth and Ninth Editions of the Programmer's Manual for the UNIX Time-Sharing System, Doug McIlroy says that the volumes describe the lineal descent of the original operating system pioneered by Ken Thompson and Dennis Ritchie. Distributed computing proved to be the distinctive theme of the landmark Eighth Edition: Dennis Ritchie's coroutine-based stream IO system, and the Datakit virtual circuit switch realisation by Lee McMahon and Bill Marshall, provided the basis for networking, Peter Weinberger's remote file systems made it painless, and Rob Pike's software for the Teletype 5620 moved system action

1. GIPSI-SM90 is sponsored by the French Ministry of Research and Technology under the contracts 83-B1032 84-E0651 85-B0524

2. Author's current address: IBM Israel Scientific Center, Technion City, Haifa, Israel.

right out to the terminal.

Users distributed around the world is the theme of the Spring 1988 EUUG Conference. The Conference Chairman discusses here why users around the world have demanded to use UNIX, why UNIX has proved successful around the world, and the future of the UNIX system in the world marketplace.

The paper finishes with a citation of the original and innovatory contributions made by many of the speakers who travelled from all over the world to be at the the EUUG's Conference held at the Queen Elizabeth II Conference Centre, London in April 1988.

UNO: USENET News on Optical Disk

A. Garibbo,
L. Regoli,
G. Succi

University of Genoa
Italy

Introduction

The size of a WORM optical disk is greater than a Gbyte and it is likely to grow fast within few years; moreover storing and retrieving USENET news is becoming tedious and difficult: at present time a user has easy access to news if he knows exactly which ones he wants to consult; besides reading daily news takes little time using standard read-news tools.

Troubles arise when one wants to find some news only knowing few features because the help he has is merely a hierarchical organisation of the news supplied by the USENET system: actually, such a tree-shaped framework seems to be quite unsuitable as long as:

i. the structure is not strongly enforced

ii. quite different leaves lie in the same directory

Owing to the high rate of news traffic, lots of space is needed, and usually each local network connected with USENET either devotes too much space to archiving or it needs frequent backup on tape.

UNO – USENET News on Optical disk – attempts to solve this kind of problems, since more than four years of full news, at the present rate, can be archived on a WORM disk.

All facilities provided by standard readnews tools are enclosed in UNO; moreover it supports an incremental knowledge driven search, which allows interactive data retrieving without either knowing exactly the wanted news or having to deal with all the news of a USENET directory.

UNO provides easy interaction through a smart query language, remote query and intelligent programmable selection of relevant news.

UNO was developed an a workstation named Arianna, based on a National 32032 processor, which runs UNIX System V.3. A WORM disk is fully integrated in the global file system; UNO is designed in C++ according to object oriented programming and software engineering criteria.

Evolution of the SunOS Programming Environment

Robert A. Gingell

Sun Microsystems, Inc.
2550 Garcia Ave.
Mountain View, CA 94043 USA

Abstract

Recent changes to Sun's implementation of the UNIX operating system (SunOS) have provided new functionality, primarily file mapping and shared libraries. These capabilities, and the mechanisms used to build them, have made significant changes to the programming environment the system offers. Assimilating these new facilities presents many opportunities and challenges to the application programmer, and these are explored in this paper.

The new mechanisms also provide the application programmer with a flexibility comparable to that previously reserved for the operating system developer. Much of this flexibility is based on mechanisms for dynamic linking that support interposition. The future developments and ramifications of these mechanisms, as well as other areas for similar system refinements, are also explored.

Multiprocessor UNIX: Separate Processing of I/O

A J. van de Goor,
Delft University of Technology,
Department of Electrical Engineering,
Mekelweg 4,
P.O. Box 5031,
2600 GA Delft,
The Netherlands.
vdgoor@dutesta.UUCP

A. Moolenaar,
Oce Nederland B.V.,
St. Urbanusweg 126,
P.O. Box 101,
5900 MA Venlo,
The Netherlands.
mool@oce.nl.UUCP

J.M. Mulder,
Delft University of Technology,
hansm@dutesta.UUCP

Abstract

Making UNIX suitable for a multiprocessor system is a logical step because of the wide acceptance of UNIX and the decreasing cost of hardware. The multiprocessor adaptation, however, is not trivial because of some of the assumptions the UNIX kernel is based on. This paper illustrates, on a high level, the performance considerations which guided the design of a UNIX multiprocessor, and it describes specifically the modifications required to implement the I/O kernel layers on dedicated I/O processors. This implementation was based on the concepts of horizontal and vertical data sharing.

System V Release 3, Diskless Workstations and NFS

Robert Cranmer-Gordon,
Bill Fraser-Campbell,
Mike Kelly,
Peter Tyrrell

The Instruction Set
rob@inset.co.uk,
bill@inset.co.uk,

wot@inset.co.uk,
petet@inset.co.uk

Abstract
Diskless UNIX workstations are becoming a fashionable way of providing users with high levels of facilities and performance at low cost. To date, most implementations of UNIX for diskless computers have been based on 4.2BSD. This paper describes some major modifications made to Motorola System V/68 to produce a version of System V Release 3 capable of supporting diskless machines.

To make diskless operation possible using Sun's Network File System (NFS over Ethernet, the File System Switch feature of System V Release 3 has been replaced with the Sun Virtual File System (VFS) switching arrangement. However, the Release 3 STREAMS architecture has been retained as a framework for Internet protocol software to permit a (comparatively) easy switch to OSI protocols in the future. The BOOTP (RFC 951) and TFTP protocols are used for bootstrapping diskless machines.

The paper presents details of the method used to marry NFS and STREAMS, performance enhancements to the Sun distributed record locking and experiences with BOOTP. It also lists those awkward places where the requirements of NFS and the System V Interface Definition (SVID) conflict.

Implementation of X.25 PLP in ISO 8802 LAN Environments

S.A. Hussain,
J. Ølnes,
T. Grimstad

Norsk Regnesentral,
Blindern
0314 Oslo 3
anwar%vax.nr.uninett@tor.nta.no

Abstract
The X.25 Packet Layer (ISO 8208) and Class II of LLC (ISO 8802/2) are both implemented in the kernel of Berkeley UNIX 4.2BSD on a VAX 11/750 as a new communication domain (AF_XLAN). It is accessible using the IPC primitives provided by 4.2BSD. X.25 PLP's stream services are accessible via stream sockets. Class II of the LLC's datagram services are accessible via raw datagram sockets and stream services via raw stream sockets.

General Purpose Transaction Support Features for the UNIX Operating System

S. G. Marcie
R. L. Holt

NCR Corporation
E&M Columbia
W. Columbia, South Carolina 29169

Abstract
This paper describes the features of NCR's General Purpose Transaction Facility (GPTF), an extension to NCR's implement-ation of UNIX System V for the TOWER supermicrocomputer. Timer signals with millisecond resolution are presented. Performance of process synchronisation and interprocess communication is improved via a set of semaphore primitives which executes in the user program environment and operates on structures which exist in standard UNIX System V shared memory. A scheduler is described which reduces process switching latency and provides process scheduling among both realtime and timesharing priority classes.

Additionally, a mechanism is provided to lock a process in memory so that it is immune to paging. Scheduling latency is reduced through voluntary preemption within the kernel. A novel disk I/O scheduler provides the ability to schedule disk requests according to process priority, seek distance, or some configurable combination of both parameters.

User access to the transaction processing facilities is provided via a set of system calls and shell commands. A user friendly interface is provided to allow a superuser to control such access.

Grep Wars

Andrew Hume

AT&T Bell Laboratories
Murray Hill, New Jersey 07974
research!andrew

Abstract
Subsequent to the Sixth Edition of the UNIX system there have been different versions of the searching tool *grep* using different algorithms tuned for different types of search patterns. Friendly competition between the tools has yielded a succession of performance enhancements.

We describe the latest round of improvements, based on the *fio* fast I/O library and incorporating the Boyer-Moore algorithm. Although *grep* is now 3–4 times faster than it was, *egrep* is now typically 8–10 (for some common patterns 30–40) times faster than the new *grep*.

Yacc Meets C++

Stephen C. Johnson

Ardent Computer Corp.[3]
880 W. Maude Ave.
Sunnyvale, CA, USA, 94086

Abstract
The fundamental notion of attribute grammars is that values are associated with the components of a grammar rule; these values may be computed by *synthesising* the values of the left component from those of the right components, or *inheriting* the values of the right components from those of the left component.

The `yacc` parser generator, in use for over 15 years, allows attributes to be synthesised; in fact, arbitrary segments of code can be executed as parsing takes place. For the last decade, `yacc` has supported arbitrary data types as synthesised values and performed type checking on these synthesised values. It is natural to think of this synthesis as associating a value of a particular type to a grammar symbol when a grammar rule deriving that symbol is recognised.

Languages such as C++ support abstract data types that permit functions as well as values to be associated with objects of a given type. In this framework, it appears natural to extend the

3. Much of this work was done when the author was employed by AT&T Information Systems

idea of computing a value at a grammar rule to that of defining a function at a rule. The definition of the function for a given object of a given type depends on the rule used to construct that object.

In fact, this notion can be used to generalise both inherited and synthesised attributes, unifying them and allowing even more expressive power.

This paper explores these notions, and shows how this rule-based definition of functions allows for easier definitions and much more flexibility in some cases. Several examples are given that are hard to express using traditional techniques, but are naturally expressed using this framework.

Software Tools for Music
- or -
Communications Standard Works!

David Keeffe

Siemens Ltd., Systems Development Group,
Woodley, Reading, UK
ukclsiesoft!dk

Introduction

Described here is the evolution of a small suite of programs for the composition and performance of music. They started life as a personal interest, inspired in part by Peter Langston's work [Langston 86]. As they developed, however, a use for the programs was seen as an unusual and illustrative aid for exhibiting computing equipment.

At the Systems Development Group, we are involved variously in developing systems which address the problems of UNIX and DOS communication, in developing software for a graphics workstation, and generally in improving the flexibility and usability of Siemens range of UNIX and DOS machines. Would it not then be a good idea if an 'earcatching' package could be built which combined all this?

As such a system is to receive close scrutiny, the musical ideas must have a reasonable foundation: the computer should not be seen as simply a glorified tape machine.

The aim of this paper, then, is to present several facets of the music system: of course, the musical ideas are central, but there are also other lessons to be learnt. There are two central foundations of the design of the system: the first is the Musical Instrument Digital Interface, or MIDI: there will not be much said about it, as there isn't much to say — the standard itself is only about one-third the length of this paper! What should become clear is not only the way MIDI allows the system to function but also how such a useful standard can make writing other music programs so much easier and more effective. The second foundation is the legacy of the UNIX operating system: it is that legacy which makes the whole thing fit together.

Why music? The composition of music is generally thought of as one of the most abstract of human activities. While not ever hoping to replace the human musician, the computer can be used to experiment with music, as well as providing a base for a diverting exercise in analysis and programming. Also, computer music is strangely attractive, like high-quality intelligent speech systems.

An Overview of the Gothix Distributed System

Alain Kermarrec

IRISA – Campus de Beaulieu –
35042 RENNES-CEDEX - FRANCE –
kermarre@irisa.irisa.fr

Introduction

Currently under development at the IRISA/INRIA, GOTHIC is intended to be an integrated distributed system implemented on a network of multi-processor machine BULL SPS7. Since the development of the GOTHIC kernel is assumed to take a rather long time, it was decided to build on UNIX machines (a Network of SUN running under UNIX 4.2BSD) a system which provides the same interface as GOTHIC in order to start the development of applications. The first release of this system called GOTHIX is currently under test. This paper first describes the concepts developed in both systems and then discusses some implementation details of GOTHIX.

A Tool-based 3-D Modelling and Animation Workstation

Samuel J. Leffler
Eben F. Ostby
William T. Reeves

Animation Research and Development Group
Pixar
3240 Kerner Blvd.
San Rafael, CA. 94901

Abstract

A tool-based system for 3-D modelling and animation is presented. Each *tool* is a separate program that operates as an independent UNIX process. Tools utilise a window-oriented display package, an event-based input system, and a large graphics database that resides in shared memory in providing interactive and non-interactive functions. The system described here is being developed for use in the production of 3-D animated sequences and as a testbed for research in 3-D modelling and animation. The architecture of the system and the motivation behind the tool-based approach is described.

Word Manipulation in Online Catalog Searching: Using the UNIX System for Library Experiments

Michael Lesk

Department of Computer Science
University College London
Gower St
London WC1E 6BT

Bellcore
435 South St
Morristown, NJ 07960

Abstract

Online public access catalogs are often plagued with very short queries and very short document descriptions. As a result performance may be poor and the users are dissatisfied. To improve recall, in particular to deal with query terms not found in the collection, a machine-readable dictionary can be used to identify related terms by overlap of defining words. To improve precision, phrases can be retrieved and the user asked to pick the appropriate ones. A demonstration system is running on 72,000 records from the British Library Eighteenth Century Short Title Catalog.

A UNIX system is a good way to implement this software, because of its advantages of easy programming, availability on small machines, and advanced data base routines.

Help! I'm Losing My Files!

John Lions

University of New South Wales
Kensington 2033
Australia

Abstract

Managing large collections of miscellaneous files can present a problem for individual users of a UNIX system. Keeping track of files that are still wanted and useful, finding and eliminating files that are no longer needed, and reorganising the file hierarchy from time to time may not be trivial if the set of files is large. Outlines are drawn for a partial solution involving index files, embedded keyword lists, a procedure for revising file pathnames and the implementation of a daemon secretary to keep everything tidy.

A Toolkit for Software Configuration Management

Axel Mahler,
Andreas Lampen

Technische Universität Berlin

Abstract

For almost ten years, `make` has been a most important tool for development and maintenance of software systems. Its general usefulness and the simple formalism of the `makefile` made `make` one of the most popular UNIX tools. However, with the increased upcoming of software production environments, there is a growing awareness for the matter of *software configuration management* which unveiled a number of shortcomings of `make`. Particularly the lack of support for version control and project organisation imposed a hard limit on the suitability of `make` for more complex development and maintenance applications.

Recently, several programs have been developed to tackle some of the problems not sufficiently solved by `make`. `Shape`, the system described in this paper, integrates a sophisticated version control system with a significantly improved `make` functionality, while retaining full upward compatibility with `makefiles`. `Shape`'s procedure of identifying appropriate component versions that together form a meaningful system configuration, may be completely controlled by user-supplied *configuration selection rules*. Selection rules are placed in the `shapefile`, `shape`'s counterpart to the `makefile`.

The `shape` system consists of commands for version control and the `shape` program itself. It is implemented on top of the *Attribute File System* (AFS) interface. The AFS is an abstraction from an underlying data storage facility, such as the UNIX filesystem. The AFS allows to attach any number of attributes to document instances (e.g., one particular version) and to retrieve them by specifying a set of desired attributes rather than giving just a (path–) name. This approach gives an application transparent access to all instances of a document without the need to know anything about their representation. So, it is also possible to employ different data storage facilities, as for instance dedicated software engineering databases.

The project organisation scheme of `shape` provides support for small (one man), medium, and large projects (multiple programmers/workstation network).

Design of and Experience with a Software Documentation Tool

José A. Mañas
Tomás de Miguel

Dept. Ingeniería Telemática
E.T.S.I. Telecomunicación
Ciudad Universitaria
E-28040 MADRID
SPAIN
jmanas@goya.uucp
tmiguel@goya.uucp

Abstract

A UNIX tool is presented that permits to write documented code in a text oriented fashion, looking for humans that have to read, understand, and maintain it, rather than thinking for language processors that have to compile it. The tool permits handling any text processing system, as well as any target language. Several files may be documented and maintained as a single unit, thus helping in keeping them coherent. The tool may easily and productively interact with standard UNIX tools. The design criteria, basic features, and some real examples of utilisation are presented.

UNIX Past, Present, and Future: Changing Roles, Changing Technologies

John R. Mashey

MIPS Computer Systems
Sunnyvale, CA 94086

Introduction

The UNIX operating system seems to defy the laws of physics by remaining in perpetual motion. This paper takes a brief look at where it's been, where it is, and where it might be going. In particular, UNIX stands as a major beneficiary of the the current developments in RISC microprocessors.

Multilevel Security with Fewer Fetters

M. D. McIlroy
J. A. Reeds

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

Abstract

We have built an experimental UNIX system that provides security labels (document classifications), where the security labels are calculated dynamically at the granularity of kernel activity, namely, at each data transfer between files and processes. Labels follow data through the system and maintain the lowest possible classification level consistent with the requirement that the labels of outputs dominate the labels of inputs from which they were computed. More rigid control is exerted over the labels of data passing out of reach of the system to and from tapes, communication lines, terminals, and the like. Necessary exceptions to the security rules (as for system administration, user authentication, or document declassification) are handled by a simple, but general, privilege mechanism that can restrict the exceptions to trusted programs run by 'licensed' users. Privileges are subdivided; there is no omnipotent superuser. Carefully arranged data structures and checking algorithms accomplish this fine-grained security

control at a cost of only a few percent in running time.

Dynamic labels should help mitigate the suffocating tendencies of multilevel security. At the same time dynamic labels admit covert channels by which dishonest, but authorised, users can leak data to unauthorised places at modest rates. The system is still highly resistant to other kinds of threat: intrusion, corruption of data by unauthorised users, Trojan horses, administrative mistakes, and joyriding superusers. In most real settings, we believe, worries about potential leaks will be far outweighed by these latter concerns and by the overriding consideration of utility.

Directly Mapped Files

Andreas Meyer

Stollmann GmbH,
Max Brauer Allee 81
D-2000 Hamburg 50
West Germany

Abstract

Directly Mapped Files is a file access method implemented under UNIX System V Release 3.

The entire file appears to the user process like a large byte array in the virtual address space and may be accessed without any read, write or seek operations. Thus, many programs, especially those working on data bases, intermediate files, or complex data structures, may be written much more easily and run faster. The paper describes the implementation in the UNIX kernel and its direct relationship to the demand paging algorithms. An example (*ar*) demonstrates the issues for user programs.

SunOS Virtual Memory Implementation

Joseph P. Moran

Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043 USA

Abstract

The implementation of a new virtual memory (VM) system for Sun's implementation of the UNIX operating system (SunOS[4]) is described. The new VM system was designed for extensibility and portability using an object-oriented design carefully constructed to not compromise efficiency. The basic implementation abstractions of the new VM system and how they are managed are described. Some of the more interesting problems encountered with a system based on mapped objects and the resolution taken to these problems are described.

Adventures in UNIX Arithmetic

Robert Morris

National Computer Security Center
Fort George Meade
Maryland 20755
USA
RMorris@dockmáster.arpa

4. SunOS is a trademark of Sun Microsystems.

The problems of writing mathematical software under UNIX are described. This comes about as the accuracy of standard mathematical functions (e.g., *sin*, or *log* is something rarely considered important by the supplier of a computer system, the *guru* that does the port will probably comment that 'floating point is for users' and forget this important issue.

This paper is a plea to manufacturers to provide something that works to the accuracy that the hardware can support.

The JUNET Environment

Jun Murai

Computer Centre
University of Tokyo
Japan
jun@u-tokyo.junet

Abstract

The JUNET environment consists of various Kanjified public domain utilities and some original tools providing Kanji capabilities. The design and implementation of this environment will be described, as well as the current status of JUNET itself.

POSIX — A Standard Interface

Jim R Oldroyd

The Instruction Set
jr@inset.co.uk

Abstract

There is a bewilderingly large number of UNIX systems in existence today. Most are derived from one of two main 'flavours' of the operating system — System V and 4.*BSD.

However, these derivatives vary considerably in a variety of ways, both expected and unexpected. Differences exist in the behaviour of functions, their types, the type and number of arguments, location and contents of header files, etc; also the commands and utilities may differ or take different options, etc.

These differences provide headaches to authors of *portable* applications. Although it is possible to write software that will compile without modification and run correctly on a large number of existing systems, considerable expertise and knowledge of the different systems is required to do this. Acquisition of this expertise can be a time-consuming and costly overhead.

Developing software which is portable across different UNIX operating systems suffers from a major problem: the software is still likely to need modification when another new implementation of the system appears.

The POSIX System has been developed to ease this problem. The interfaces (system calls, libraries and commands) described in POSIX have evolved from those on existing UNIX systems and, where things differ on existing systems, the POSIX interfaces represent a compromise or an improvement.

The POSIX interface can be implemented on all existing UNIX systems (and, in fact on non-UNIX systems too). Porting applications to new systems will be considerably simplified, if both source and target systems are POSIX compatible.

This paper presents a technical overview of POSIX and looks at areas which differ from existing systems. The paper takes the view of an Applications Writer, but in so doing, highlights

areas which will be of interest to those responsible for making a system *POSIX-compatible*.

An overview of the position of POSIX and impact in the market place is also given.

The Andrew Toolkit — an Overview

Andrew Palay
et al.

Carnegie Mellon University

Abstract

The Andrew Toolkit is an object-orientated system designed to provide a foundation on which a large number of diverse user-interface applications can be developed. With the Toolkit, the programmer can piece together components such as text, buttons,and scroll bars to form more complex components. It also allows for the embedding of components inside other components, such as a table inside of text or a drawing inside of a table, Some of the components included in the Toolkit are multi-font text, tables, spreadsheets, drawings, equation, rasters, and simple animations. Using these components we have built a multi-media editor, a mail-system, and a help system. The Toolkit is written in C, using a simple preprocessor to provide an object-oriented environment. That environment also provides for the dynamic loading and linking of code. The dynamic facility provides a powerful extension mechanism and allows the set of components used by an application to be virtually unlimited. The Andrew Toolkit has been designed to be window-system independent. It currently runs on two window systems, including X.11, and can be ported easily to others.

Plan 9 from Bell Labs – The Network

David Leo Presotto

AT&T Bell Laboratories
Murray Hill, New jersey 07974
research!presotto
presotto@att.arpa

Abstract

This paper describes a new computing environment and the networking that underlies it. We expect the environment to accommodate either a small group or a large organisation. Although our initial implementation is targeted at 100 researchers, our goal is a system that can encompass all of AT&T's research and development.

Our design runs counter to the popular trend in computing environments, workstations connected by local area networks. We have found this solution to be both expensive and awkward. This is especially apparent in large organisations. Instead, we propose a system based on clusters of file servers and execute servers connected by high speed networks. User interfaces, similar to workstations, access the servers via lower speed distribution networks. Among other things, this simplifies administration and allows the home and work computing environment to be the same.

Formatted I/O in C++

Mark Rafter

Computer Science Department
Warwick University
Coventry
England
../mcvax!warwick!rafter

Abstract

The *fmtio* library extends C++ stream I/O to include formatted I/O in the style of stdio. This extension is layered on top of stream I/O, and only requires minor changes to `<stream.h>`. The key traits of the original stream I/O system, namely extensibility and type-security, are retained. An example of its use is:

```
cout[ "log of %d is:%9f\n" ] << 5 << log(5);
```

which prints

```
log of 5 is: 1.609438
```

The *fmtio* library is presented as a suitable framework in which to conduct further experiments with formatted I/O systems. The methods used in the library are sketched, and its overall structure outlined. An example is given of how to equip a datatype with formatted I/O by interfacing it to the *fmtio* library.

A Protocol for the Communication between Objects

R. Schragl
UNA EDV-Beratung GmbH, München
D. Lauber
Siemens AG, München

Abstract

In object-oriented systems objects communicate with each other via messages. An object activates processing by sending a message to another object and waiting for its termination. Most of the existing implementations (e.g., SMALLTALK 80) have chosen this procedure. Normally, they are available as stand-alone systems, so that no specific protocols are required. When offering an object-oriented user interface, integrated in a conventional command-oriented system, and with tools running in a local or distributed environment, application protocols are required. This contribution defines a protocol with a service, comparable to the session-layer of the ISO reference model, suitable for this application. The characteristics of the protocol are described, and an implementation is shown within a UNIX system using the programming language C. The concepts are validated in a distributed software development environment, where system software for mainframes is developed using connected workstations based on UNIX.

UNIX V.3 and Beyond

Ian Stewartson

Data Logic Limited
System Software Development Group

Abstract

The object of this paper is to provide an overview of the current state of the UNIX Operating System environment with specific reference to the latest release (V.3) from AT&T. As UNIX has been selected as the basis for a portable operating system by a number of standards bodies, the work being done by these groups is also reviewed. Finally, the paper highlights possible and likely future developments of UNIX that are designed to improve its commercial viability.

An Overview of Miranda

David Turner

Computing Laboratory
University of Kent
Canterbury CT2 7NF
ENGLAND

Abstract

Miranda[5] is an advanced functional programming system which runs under the UNIX operating system. The aim of the Miranda system is to provide a modern functional programming language, embedded in an 'industrial quality' programming environment. It is now being used at a growing number of sites for teaching functional programming and as a vehicle for the rapid prototyping of software.

5. Miranda is a trademark of Research Software Ltd.

EUROPEAN UNIX® SYSTEMS USER GROUP NEWSLETTER

Volume 8, Number 3
Autumn 1988

ACCES

Jean-Louis Faraut
inria!pastpe!cerisi!faraut

Ingenieur de recherche
Université de Nice

Abstract

ACCES - A new tool to allow management of access rights to commands on multi-user UNIX systems.
ACCES - Un nouvel outil pour faciliter la gestion des droits d'acces aux commandes d' Unix multi-utilisateurs.

Introduction

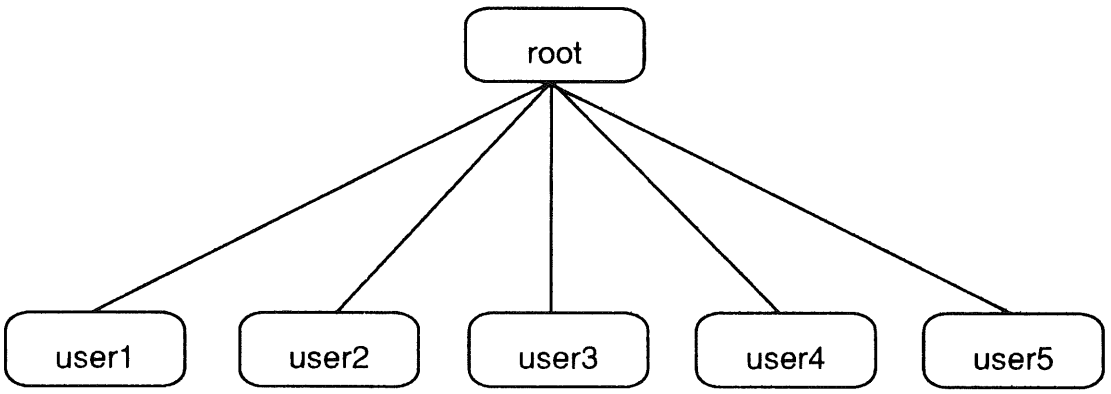
All UNIX systems have a ''super-user'' (normally called ''root''), which can bypass all protections, and modify protections so as to limit the permissions of each user.

In the case of a large machine, with many users this organisation is usually very simple, one super-user, and lots of other users, all at the same level, with basically the same rights. The diagrammatic representation of this schema resembles a ''rake'':

Introduction

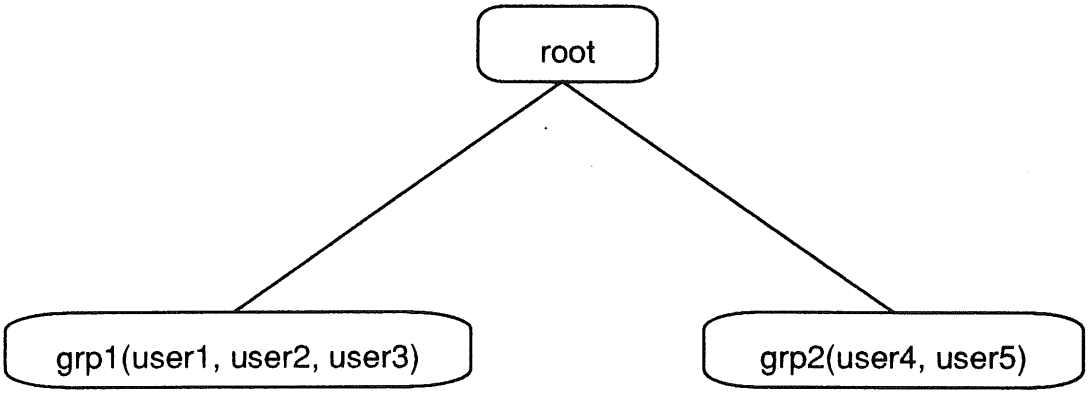
Tout systeme Unix suppose l'existence d'un utilisateur nomme ''root'' qui peut franchir toutes les protections (fichiers, processus, memoire) et les modifier afin de delimiter les droits de chacun.

Dans le cas d'environnement lourd (machine puissante avec utilisateurs nombreux), cette organisation qui est essentiellement du type ''rateau'', c.a.d. avec une tete et tout le monde sur le meme niveau peut ne plus convenir.



The group permissions can be used to further sub-divide this schema giving more selective access permissions:

On peut alors avoir recours a des repartitions par groupe d'utilisateurs a qui l'on accordera selectivement les droits d'acces:



The user passes from one group to the other using either the *newgrp* command (AT&T systems), or by being a member of several groups simultaneously on a BSD-based system.

This organisation, although efficient, presents several inconveniences.

Firstly, there is a confusion about access rights; the same attribute (group membership) controls two resources of a different natures, file access and command access.

Secondly, this system is quite limited in its possibilities; suppose that some group has access to an external network, a second group has access to a laser printer, and that a third has accesses to some other system resource, etc.

The limits of the system are rapidly reached, in that the management these groups becomes a nightmare, and the ease of use for the user non-existent.

Lastly, there is no hierarchical scheme for groups of users. This means that one cannot give a sub-group of a group any extra privileges without giving them to the whole group, or without upsetting the whole organisation by re-defining the whole permissions structure. There is also always a strong possibility that any such re-organisation will introduce errors as the system becomes more complex.

Also, how can one manage privileges that need to be given on a temporary basis?

et on passera d'un groupe a l'autre par la commande *newgrp* (chez ATT) ou en mettant les gens dans plusieurs groupes simultanement (chez Berkeley).

Cette organisation bien que plus efficace presente cependant plusieurs inconvenients.

En premier lieu, elle induit une confusion dans les droits d'acces; en effet, un meme attribut, l'appartenance a un groupe, conditionne l'acces a deux ressources de natures differentes: les fichiers et les commandes.

En second lieu, elle est limitee; supposons que l'on veuille accorder a tel groupe l'acces au reseau exterieur, a tel autre l'acces a l'imprimante laser, a un troisieme encore l'acces aux sources du systeme, etc.

On atteind ainsi rapidement les limites du syteme et la gestion devient tellement lourde que le confort des utilisateurs et leur securite risquent de s'en ressentir.

Enfin, il n'y a pas de hierarchie de groupes d'utilisateurs. Ceci implique qu' on ne peut pas accorder a un sous-ensemble d'un groupe (un "sous-groupe") des privileges particuliers sans les accorder a tout le groupe ou sans bouleverser toute l'organisation au risque de commettre des erreurs et d'introduire des failles pas toujours faciles a deceler dans les protections.

Et comment gerer les privileges que l'on ne veut accorder que temporairement?

Proposed improvement

The proposed solution consists of giving temporary rights (root permissions) to one or more users under strictly controlled conditions.

Note that this possibility already exists in the form of the SUID bit:

chmod u+s UNIX_file

But this only partially solves the problem, since the problem of sub-groups, and temporary privileges still remains. Also, the indiscriminate use of the SUID bit is not particularly compatible with a secure system.

We can thus imagine a single command with the SUID bit in place, which works with a data-base, modifyable only by ''root''.

This data-base contains a list of all the limited-rights commands, together with a list of users authorised to use the command with root permissions.

A limit-date could also be included to automatically revoke permissions after that date.

Implementation details

The implementation is based upon an Access Control List (ACL), with a defined syntax.

The program is in two modules, 'access.lex' and 'access.yacc', respectively lexical and syntactic analysers for the ACL file. The main program is written in C.

Syntax of the ACL file (in yacc syntax):

```
acl    : block
       | block acl
       ;
```

⇒ the file is a sequence of blocks

```
block : command_line    user_block
      ;
```

⇒ each block is a command and a user block

Amelioration proposee

La solution proposee consiste a donner temporairement les droits de root a un ou plusieurs utilisateurs dans des conditions bien determinees.

Notons que cela est deja possible en positionnant le "USER-bit" d'une commande appartenant a root avec la commande:

chmod u+s fichier_Unix

Mais cela ne repond que partiellement a la question, car le pb des sous-groupes ou des privileges temporaires demeure. D'autre part la multiplication des modes "s" n'est jamais tellement souhaitable pour la securite.

On peut alors imaginer une commande en mode "s" (et une seule) travaillant a partir d'une base de donnees (un simple fichier modifiable seulement par ''root'').

Ce fichier comprendrait autant d'enregistrements que de commandes Unix concernees; dans chacun d'eux l'on trouverait la liste des utilisateurs autorises

Une date limite pourrait etre affectee a chaque utilisateur, l'empechant d'utiliser cette commande au dela.

Details de l'implementation

La realisation est articulee autour d'une liste de controle d'acces (fichier "ACL") dont la syntaxe est definie une fois pour toutes.

Le programme comporte 2 modules acces.lex et acces.yacc constituant respectivement un analyseur lexical et un analyseur syntaxique pour le fichier ACL. Le programme principal se trouve dans l'analyseur syntaxique et est ecrit en C.

Syntaxe du fichier ACL (en langage de description ''yacc'')

```
acl    : bloc
       | bloc acl
       ;
```

⇒ le fichier acl est un ensemble de blocs

```
bloc : ligne_de_commande    bloc_utilisateur
     ;
```

⇒ chaque bloc comporte une ligne de commande et un bloc utilisateur

```
user_block : command_line
           | command_line  user_block
           ;
```

⇒ each user block is a collection of user lines

```
command_line : NAME   PATH
             | NAME   PATH1   PATH2
             ;
```

⇒ a command line consists of a command name, spaces, a pathname used to access the command, and, optionaly, a pathname for a log file to log each use of the command.

```
user_line : USER STAR
          | USER DAY SLASH MONTH
            SLASH YEAR
          ;
```

⇒ a user line consists of a login name, space, and a limit date which is the expiry date for the permissions, be this dd/mm/yyyy or an asterisk indicating that there is no time limit.

Refusal

Access is refused with one of the following error messages:

```
"Usage: acces command [args]"
                          : bad command syntax
"Unknown user"
                          : unknown user
"Who are you?"
                          : unknown user
"Can't open ACL file"
                          : ACL file open failed
"Command not in ACL file"
                          : command not in ACL file
"Permission denied"
                          : non-listed user
"Expiry date exceeded"
                          : expiry date passed
"Exec failure"
                          : exec failed
"Syntax error in ACL file"
                          : syntax error in ACL file
```

```
bloc_utilisateur : ligne_utilisateur
                 |ligne_utilisateur bloc_utilisateur
                 ;
```

⇒ chaque bloc utilisateur est un ensemble de lignes utilisateur

```
ligne_de_commande : NOM   PATH
                  | NOM   PATH1   PATH2
                  ;
```

⇒ une ligne de commande comprend un nom de commande, des espaces, un chemin pour acceder a la commande et eventuellement un chemin pour acceder a un fichier ou l'on enregistre tous les acces

```
ligne_utilisateur : USER   ETOILE
                  | USER   JOUR   SLASH
                    MOIS   SLASH   ANNEE
                  ;
```

⇒ une ligne utilisateur comprend un nom d'utilisateur, des espaces et une date limite qui est soit jj/mm/aaaa soit une etoile indiquant qu'il n'y a pas de limite d'acces.

Cas de refus

L'acces est refuse avec un message d'erreur dans les cas suivants :

```
"Usage: acces commande [args]"
                          : mauvaise syntaxe de la commande
"Utilisateur inconnu"
                          : getlogin n'a rien trouve
"Qui etes-vous ?"
                          : getpwuid n'a rien trouve (?)
"impossible d'ouvrir le fichier ACL"
                          : fichier ACL non lisible ou non accessible
"Commande ACL non trouvee"
                          : commande inexistante dans le fichier
"Permission refusee"
                          : utilisateur non liste pour la commande donnee
"Date limite depassee"
                          : fin de droits
"erreur d'exec"
                          : pb d'exec de la commande
"erreur de syntaxe dans fichier ACL"
                          : erreur de syntaxe ACL
```

Second chance

In certain cases (ACL file not found, permission refused, date limit exceeded) a second chance is given, by asking for the root password:

Password:

In the case of the password being correct, the command is exece'd in the traditional manner (using the PATH of the user), and executed as SUID root.

Conclusion

The tool described here does not replace the existing system administration tools. Nor does it disturb existing system organisation, it is simply an improvement. An improvement made without - at least we hope! - introducing an Achilles heel into the system.

This solution solves the majority of problems of access rights to UNIX commands, without disturbing the current groupings of users.

Actually, the tool applies not only to UNIX commands, independently of file access rights. Putting a user into a group, not only gives that user access to commands accessible by that group, but also to all files accessible by them - this is not always acceptable.

Thus, the system administrator has a more supple means at his disposal in the organisation of access rights. Problems of temporary access rights which have been solved before ''by hand'' or by *cron* can now be solved much more easily.

Deuxieme chance

Dans certains cas (commande ACL non trouvee, permission refusee, date limite depassee), on donne une deuxieme chance a l'utilisateur en lui proposant de taper le mot de passe root en reponse a la question :

Password :

Si le mot de passe tape est conforme a celui qui se trouve dans /etc/passwd, alors on cherche la commande dans l'environnement de l'utilisateur (variable PATH) et une fois trouvee, on essaye de l'executer avec les droits de root.

Conclusion

L'outil propose ici ne se substitue pas aux outils d'administration existants. Il ne remet pas non plus en cause l'organisation du systeme, mais vient plutot completer en l'ameliorant, et sans intoduire de talon d'Achille - du moins nous l'esperons - dans la gestion du systeme Unix.

La solution retenue devrait permettre de resoudre la plupart des problemes d'acces aux commandes Unix "sensibles" sans remettre en question la repartition des utilisateurs en groupes de travail.

En effet, l'outil propose ne s'applique qu'aux commandes Unix, independamment des fichiers; a l'oppose de mettre l'utilisateur dans un groupe c'est lui donner acces aux commandes accessibles par le groupe mais aussi aux fichiers accessibles par ce meme groupe ce qui n'est peut-etre pas toujours souhaitable.

Une plus grande souplesse dans la gestion et la repartition des droits est ainsi apportee a l'administrateur du systeme. De plus une reponse est apportee aux problemes des droits temporaires que l'on gerait habituellement a la main ou par *cron*.

Home-Directory Mail System

Dr. Andrew J Findlay
Andrew.Findlay@brunel.ac.uk

Manufacturing and Engineering Systems
Brunel University
Uxbridge
UB8 3PH

Andrew Findlay is a lecturer on the Special Engineering Programme at Brunel University, teaching computing and electronics. He is system manager for some of the University's networked computers, and is responsible for electronic mail systems across the whole campus.

Abstract

To make the mail system behave in a more obvious way on networks of workstations, mailboxes are moved into users' home directories. User Agent programs are modified, and a new mail notification system is introduced.

Background

Brunel University has a large network of Unix-based computers. There are about fifty machines covering seven departmental subdomains. There are four different processor types. All machines use Sun's NFS, and all users have a single home directory that is exported to all machines where they might log in. Most workstations are in communal areas so a user is likely to log in on a different machine each day.

The mail system is based on the *UK sendmail 1.4a* package [1], and users create *.forward* files to specify which machine their mail should be delivered to. This approach worked well when there were only a few machines, but has become increasingly awkward as the network has grown. To avoid locked-up machines, most user filesystems are soft-mounted. This means that there can be times when a machine is running but a group of user directories are not present. If mail arrives at such a time, the *.forward* file is not effective. A user's mail could be spread around the whole network, with one or two messages on each machine.

With increasing use being made of the mail system, it was felt that some improvements were needed.

Options

- *Use a big alias file*
  This would avoid the 'invisible forward file' problem, but it still requires each user to nominate *one* machine where they will read mail.
- *Mount a single /usr/spool/mail directory for the whole Campus*
  This is probably the simplest solution, but there would be problems with mail delivery

agents that run as root. For security reasons, NFS maps *root* into *nobody* when accessing remote filesystems. Mailers that expect to use lockfiles or *flock(2)* would have problems too.

- *Mount one /usr/spool/mail directory for each subdomain*
  It would be possible to arrange that only the (NFS) server machine ever delivered mail. This would remove the root-mapping problem, but would not solve the lockfile problem. Many users are registered in more than one subdomain, so this solution would not help them.
- *Create a Campus-wide 'mail server' as a TCP service*
  This is possibly the best solution, as it would be able to serve non-NFS and even non-Unix machines. The work involved is considerable, as all the User Agent (UA) programs currently in use expect mailboxes to be *files* in v7 mbox format.
- *Move the mailbox out of /usr/spool/mail into the user's home directory*
  Although the UAs still need modifications, it was felt that this would be the best compromise between effort expended and results obtained.

Mailbox in Home Directory

The main benefits of this approach are: The system behaves in an 'obvious' way.
If the user's home directory is available, then so is their mailbox. Mailboxes become subject to the same quota limits as other files. The existing mailbox format can be retained.

There are some problems remaining: Mail User Agent programs need modifying to reflect the new mailbox position. Some UAs react badly to the disappearance of a soft-mounted filesystem.

Files and Lockfiles

The v7/BSD convention places all mailboxes in /usr/spool/mail and names them with the username of their owner. Lockfiles are named by appending '.lock' to the mailbox name.

The home-directory mail system names all mailboxes '.mailbox' and all lockfiles '.mailbox.lock'. Mailboxes are identified by the directory they are placed in. The /usr/spool/mail area is not used at all.

Mailbox locking is the least satisfactory part of the home-directory mail system. Sun NFS is stateless: this means that the conventional 'exclusive create' mechanism cannot be trusted. None of the traditional atomic file operations can be used, as NFS does not guarantee atomicity. The 'correct' way to lock files is to use *flock*, which makes use of the lock daemon to handle local and remote files in the same manner. Unfortunately, not all implementations of NFS support the lock daemon. As a compromise, it was decided that home-directory mail would use lockfiles *and flock*. This would give complete safety between machines supporting *flock* and reasonable safety on all others.

The mail delivery agent may have to create a new mailbox to deliver into. In this case, the lockfile must be created before the mailbox. The *flock*-style lock is therefore always applied to the lockfile.

Delivery

A new program called *hdmail* was written to handle mail delivery. It is simplified by not doubling as a User Agent. *hdmail* is aware of the peculiarities of soft-mounted filesystems: if a directory is missing or a file suddenly becomes inaccessible, the value EX_TEMPFAIL is returned to *sendmail*. Mail for users whose home directories are not available will be held in the mail queue, and delivery will be attempted again at each queue run. This has the useful side-effect that *sendmail* will look for the *.forward* file again at each queue run, making the forwarding mechanism much more reliable. If the necessary directory does not appear within a reasonable time, *sendmail* will return the message to the sender.

Hdmail runs as the recipient user, not as root. This gets round the root-mapping problem on remote filesystems, and also improves security. When secure NFS [2] is introduced, it may be necessary to have mail delivered only by the machine that supplies the user's home directory.

After the mail has been delivered, *hdmail* arranges to notify the user of its arrival.

Notification

The home-directory mail system allows a user to read and send mail when logged in on any of the fifty-plus Unix machines on Campus. When new mail arrives, the user expects to be told about it.

Under the BSD version of the mail system, this function is handled by the *biff* daemon. *Biff* is a UDP (datagram) service; it is sent a packet containing a username and a mailbox-offset. If the named user is logged on, the mailbox is inspected from the given offset to construct a suitable message for display on their terminal.

Biff provides a good service on individual computers. It is not so useful on a large network of workstations, as a user may not be logged in on the machine that delivers the mail.

Four approaches to the notification problem were considered: Modify *biff* to read the new mailboxes, and invoke it on all known machines. Use a background 'watcher' process, such as *newmail* from the *elm* suite. Use the shell's MAIL variable. Create a new network service.

With the number of workstations and the volume of mail both increasing, it was felt that (1) would soon impose an excessive load on the system.

Solution (2) was ruled out as it would almost double the number of active processes on undergraduate teaching machines.

Solution (3) would be a backward step, as users would only find out about new mail when they returned to the shell.

It was decided to create a new network service, similar to *biff*, but with less work done by the daemon. The new service, *tell*, is a UDP service that receives a packet containing a username and a message. If the named user is logged in and has set the '*biff*' bit on their tty, the message is written to that terminal. For security reasons, only a limited set of control characters is allowed in the message. Any 'illegal' characters are converted to spaces by the daemon. The *tell* daemon provides a more general service than *biff*. It is not limited to mail notification, though that is its only function at the moment.

To avoid having to invoke *tell* on every machine on the network, another file is maintained in the user's home directory. The '.whereami' file records the name of each machine where the user has logged in, together with the time of login. When *hdmail* is ready to notify the user that new mail has arrived, it scans '.whereami' and sends a *tell* packet to every machine where the user has logged in within the last 48 hours. If the file does not exist or if no recent logins are recorded, a *tell* packet is sent to 'localhost'.

The '.whereami' file is maintained by a program called *mailcheck* that is invoked at login time from '.login' or '.profile'. *mailcheck* also prints a message if mail is waiting to be read, thus avoiding the need to modify the shells and the login program.

User Agents

The major UAs in use at Brunel are *elm*, *Mail* (/usr/ucb/mail), and *mailtool* (under SunWindows only). Since *mailtool* is a driver for *Mail*, it was only necessary to modify two UAs to satisfy almost all the users.

Elm was tackled first, taking version 1.5b as the starting point. References to mailboxes are scattered in the source code, which made the conversion rather tricky. The utility programs had to be modified individually. *elm* still uses lockfiles only; it is hoped that version 1.7 or 2.0 will have a cleaner interface to the mailbox, allowing the full locking protocol to be used.

Mail was modified using source code from the 4.3BSD distribution. The changes required were confined to two source files. Unfortunately, the 4.3 version of Mail is not compatible with *mailtool* so this work had to be repeated when Sun source code was obtained.

Other UA programs have been converted, including *prmail*, but not */bin/mail* (which is superseded by */usr/ucb/mail*). The interface between mail and *emacs* has not yet been converted. Other minor links to the mail system (such as the News '(Mail)' prompt) can safely be ignored.

Mail Transport Agents

Brunel uses *sendmail* for transport and routing of mail. The only modifications needed were in the part of the configuration file describing the 'local' mailer. *hdmail* replaces */bin/mail* for local mail delivery. The only major difference is that *hdmail* can only deliver to one user per invocation.

Operating Experience

Nobody noticed....

Apart from one incident when an automatic updating system installed the modified UAs on a machine that was not due for conversion, the transition has been smooth. Problems were encountered when re-compiling *Mail* on certain machine types: in one case caused by an optimiser bug, and in the other by bad memory-

allocation code.

The home-directory mail system is now operating in three of Brunel's six Unix-based subdomains, on two different CPU types. It is to be installed in the remaining subdomains shortly, which will include two more CPU types.

References

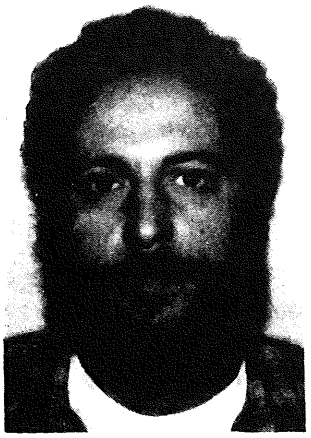
Crammond, J., "Domain Mailers: Sendmail", Workshop on Networking, UKUUG, December 1987

Taylor, B., "A framework for Network Security", Sun Technology, Spring 1988.

i2u Annual Convention

Joy Marino
joy@ugdist.uucp

i2u Annual Convention
June 6th-8th, World Trade Center, Milano-Assago



Joy Marino is associate professor of Computer Science at DIST (Dipartimento di Informatica, Sistemistica e Telematica), University of Genoa. He started using UNIX in 1980, when he was interested in ADA: now he is still using (and teaching) UNIX, C and C++, but no more ADA.

He has been involved with *i2u* since 1984, first as editor of the *i2u* newsletter (**UNI***forum*) then (since 1986) as member of the Board. He has also been appointed as *i2u* representative in the EUUG governing board (probably because other members *think* that he is good at English).

Introduction

I cannot claim to be objective in this technical report on the Annual Convention of UNIX in Italy: of all the parties represented here there is one that I would like to give a special mention to, since it is one I am involved with. I am of course referring to the *users*, better: the Italian UNIX systems Users Group, which this Convention has definitely recognised as an essential reference point for the Italian IT industry. If I allow myself to be a little sentimental, as someone who has always believed in UNIX, from the time when they could say "it's crazy to have the same operating system for microcomputers and mainframes" without people laughing, as someone who remembers when *i2u* brought together 60 people for a first Round Table on UNIX (1984), as someone like me, the high point - the feeling of finally having ''made it'' - was when, on the morning of the first day of the Conference, the partitions were opened up to transform a 400 seater hall (by that time with standing room only) into a 700 seater.

That's enough sentimentalism for now - let's get down to brass tacks.

Programme

Monday 6th June: Tutorials

9:00-12:30 An introduction to X Windows System.

9:00-12:30 Evolution of the C programming language.

14:00-18:00 An overview of UNIX standards: from BSD and SVID to POSIX.

14:00-18:00 UNIX tools for Office Automation.

Tuesday 7th June

8:30 Registration.

9:30 Opening session (A. Camici, Chairman of *i2u*).

9:45 UNIX: the state of the art (J. Marino, *i2u*).

10:15 UNIX in the U.S.A. (P. Gray, /usr/group).

10:45 Coffee break.

11:30 UNIX: International market survey and trends (R. Masiero, IDC).

12:00 The UNIX market in Italy (D. Gerundino, Mate).

12:30 Lunch.

14:30 Open Session on "The future of UNIX systems".

- X/Open Directions (J. Totman, X/Open).
- Convergence of new AT&T Releases (J. Knowles, AT&T).
- New developments at SUN (B. Joy, SUN Microsystems).

16:30 Coffee break.

17:15 Open Session on "The future of UNIX systems".

Apollo - Digital Equipment - Hewlett Packard - Honeywell Bull - IBM.

Wednesday 8th June

9:00 Open Session on "The future of UNIX systems".
Olivetti - Philips - Siemens - UNISYS.

10:45 Coffee break.

11:30 Integration of UNIX and PC world (H. Chuck, Microport).

12:00 UNIX for department systems: a case study (M. Crescioli, Unirel).

12:30 Lunch.

15:00 Session on user experiences and applications.

16:30 Coffee break.

17:00 UNIX in Europe: the EUUG (K. Biel-Nielsen, EUUG).

17:40 EUnet and the UNIX network in Italy: strategies and perspectives (*i2u*).

18:45 End of the conference.

Introduction

The following notes are my personal thoughts about the conference. Usually *i2u* meetings are devoted to industry people, and are not technically oriented as EUUG's or other national groups' are. Furthermore, this time the Conference was scheduled to be held 15 days after the first public announcement of OSF, and we have had to rearrange the progamme in "real time" in order to find place for all voices and points of view.

Never the less, we could lever on the interest that OSF has built up about UNIX to a wide audience; *i2u* Convention has been one of the first opportunities in Europe for confronting the different points of view about the evolution of UNIX in the light of this announcement.

As a whole the Convention has been quite successful: 560 attendees at the Conference, over 350 for tutorials, a full floor of Centro Congressi - Milano Fiori filled up with exhibition booths for three days.

The mysterious object

My own statistics (obviously a rough one) of the most frequently used words came up with: 1) "OSF", 2) "POSIX", 3) "ABI", 4) "X/Open", and then all the rest. It is clear that everyone's attention is focused on the Open Software Foundation.

I would like to give special mention to Dave Nelson, appearing before the convention with two hats: as the Vice-President of Apollo, and as the official speaker for the much discussed Foundation. He gave his talk mainly with his second hat, but could not totally avoid his role as spokesman for Apollo.

His vision of the IT world outlined in his address, a very complex world with many different operating systems, countless communication networks and systems, very astutely set the scene for going on to describe the advantages of a new world based on UNIX (and OSF), but how could one fail to notice that many of the complications of the proprietary world are actually due to the partners who are now seated around the same table in the Foundation?

So what is the OSF? It is an autonomous organisation, organised as a non-profit making foundation, to which the founders have given either grants of several million dollars (90) - sufficient to produce the "critical mass" - and a fraction (large or small - this remains to be seen) of their wealth of know-how on open systems. OSF will not produce new standards, it will write lines of code; and in this world, where standardisation committees abound and where the real programmers stay well clear, one can only commend their initiative.

Two of the better proposals of OSF cannot be disputed: to have a process for actively encouraging new proposals and new technologies,

to promote technological progress through universities and research centres. In a word: to recreate the "virtuous circle" that made such an important contribution to UNIX at the time of Bell Labs and the first releases of the University of Berkeley. I'm not too convinced about the third proposal: "to have a product range based on the *relevant* industrial standards". Who decides what is "relevant"?

Quoting his words, the aims that OSF sets out to achieve are therefore:

— a decision-making process not modelled by any particular manufacturer (as is happening with AT&T);

— equal right of access to the specifications and to the initial development phases (and not 6 months delay in getting the sources of the next release of System V from AT&T);

— reasonable and consistent conditions for licencing agreements (and not the mess that AT&T put together for the licences of System V.3);

— implementations that really are hardware independent.

As you can see, a scenario where AT&T appears in quite a bad light, almost as if it were the "enemy" that wants to take over UNIX.

In practice things are not quite like this, not only does AT&T stand no chance of transforming UNIX into a proprietary product, but also - as Pamela Gray, Chairperson of /usr/group, said - it's organisational structure is so complex that it cannot be credited with a "perverse mentality" (even if it is capable of acting like a bull in a china shop).

Nelson answered a lot of questions, all aimed at clarifying the shape of this mysterious object. Here is a summary of them. Licence costs? - we will offer the best products at the lowest cost. When will OSF be available ? - we hope within 18 months, but this is not definite. Will there be validation procedures? - yes, the usual procedures, but these will not be strictly adhered to. Will proprietary expansions be possible? - of course, each producer will be able to expand as he thinks best.

The last two points give rise to certain doubts: what future is there for a product that from the start does not impose a guarantee of uniformity for the various implementations?

After Nelson spoke, the other partners in OSF[1] also outlined their views on the strategies for the UNIX environment, even if not all the addresses had a "strategic" line and at times provoked murmurs of disapproval and even a spontaneous applause for the Chairman of the session, Gregorio Lerma (Banco Popolare di Bergamo, but here acting in the capacity of Vice-President of *i2u*), who stood up to rebut some particularly heated comments.

The talk by Ron Kita, DEC, was not exactly riveting, but his allusion to possible future POSIX compatibility of VMS re-awakened the interest of some of the audience, who had attended the tutorial on "Standards in comparison" concluding with an open discussion on the role of POSIX. Here are a couple of questions which I would like to see answered at the next *i2u* Convention: "What place will non-UNIX operating systems have in POSIX?" or even "Will the POSIX compatibility of VMS have any relevance in the US Government tenders?"

AIX, IBM's version of UNIX, is to play a leading role in OSF. Many doubts have been voiced on the current and future AIX characteristics, doubts which Koch (who spoke for IBM) failed to remove completely in his talk. As far as certainties go, we can point to IBM's commitment to UNIX, with the announcements of AIX on the PS2 and on the 937X, but as far as uncertainties go, we can point to the AIX characteristics on which OSF is going to build its future, an AIX that is different from the current one, but derived from it - i.e. an operating system for which IBM still has to pay its licence fees to AT&T.

It would take too long to list all the speeches that made reference to OSF, especially since almost all of them broached the subject more or less directly. Chuck Hickey, of Microport, speaking about the integration between UNIX and the PC world, had to say his piece as well, and immediately pinpointed the central problem: OSF was born as a response to the notion of ABI - "Application

1. For those who did not know them, names were listed in one of the slides in his presentation: Apollo, Bull, DEC, HP, IBM. The rest - Siemens, Nixdorf, Philips - came later on, and apparently there was not enough time (?) to redesign the slide...

Binary Interface" - i.e. binary compatibility at the level of the individual applications. ABI was announced at the Uniforum in Dallas, initially as a proposal only for the SPARC processor by SUN (SPARC-ABI) but immediately afterwards for other major microprocessors as well, starting with the Intel 80386 (386- ABI). According to Hickey, this was not merely a response to a possible hegemonic attitude by the SUN/AT&T alliance, it was primarily worry about a newborn phenomenon that might develop in the same way as has happened in the PC world, where binary compatibility has made it impossible for any large corporation to control the market. The fact that the UNIX market, for the wide range of products from PC's to mainframe, is potentially far more extensive than the PC market, gives us grounds for excitement or anxiety (depending on our point of view...)

I have tried to act as an impartial observer at the Convention, and my introductory speech also tried to outline all the positive aspects of the two proposals and strategies, SPARC/ABI and OSF, which are at the centre of so much debate. But the Convention has left me somewhat perplexed: on paper OSF looks as good as it can be, but the various partners who belong to it are still too fragmented, the motives driving them to this step are too varied, just as their expectations seem to differ as well.

OSF can hope to really succeed only if it shows that it is truly authoritative, empowered with the type of authority that cannot be derived from proportionate sums of capital of the corporations that founded it (addition that does not have any significance anyway), but from the cultural prestige of an authentic "neutral" body. Perhaps Kim Biel-Nielsen, attending the Conference as the EUUG delegate, is right when he says that everyone ought to join OSF, especially all the user groups such as EUUG, to ensure that this initiative really does provide a means of progress for UNIX.

SUNny Bill

Bill Joy shone like the Californian sun - not just because he was dressed in a casual white and yellow outfit, nor because his company is called SUN, nor because he was the main point of reference that the AT&T speaker, Jim Knowles, had to turn to when replying to questions. Delivering an epochal style speech from on high, he steered well clear of the squabbles that other players in the field are embroiled in. Or rather, this is how things ought to have been, had it not been for a few punches below the belt which he managed to get in. Having sat next to Bill Joy on the podium and seen the simple scratch pad he used to base his presentation on, I can only express great admiration for his ability to construct a strategic presentation in "real time", but I can also understand why many corporations are not pleased with the explosive mix of AT&T capital and the SUN-strokes of genius. Bill Joy says that the future lies with standards, and quotes a long list ranging from ANSI C, to the floating point format of the IEEE standard, to the SCSI peripherals bus, putting in the middle of the list the "SPARC" (the RISC processor that SUN has publicised so brilliantly) as if this were a consolidated standard as well. At this point I would like to put forward two theories: either Bill Joy is "trying it on", or he really is a few steps ahead of everyone else, and has come to tell us what our immediate future looks like. I have no way of deciding which is correct.

This immediate future, which Bill Joy reckons is 1991, can also provide the recipe for making supercomputers: take 50 boards, each one with a 100 MIPS processor and about 100 MB memory, all of which is available from your High Street dealer, add magnetic or optical disks as necessary, and for a fraction of the cost of a CRAY you can build a 5000 MIPS machine, with 5000 MB main memory, and so on, not to mention reliability, with a potentially infinite MTBF. There was only one fleeting mention of the glue needed to tie together this array of processors: didn't someone perhaps say that UNIX is the glue of the IT industry?

I would have agreed with Catellan, from HP, who branded this "multiprocessing paradise" an example of superficiality, if it were not for one detail: exactly one year ago Bill Joy gave a talk at the EUUG convention in Helsinki, where he presented the same promises of increase in the computing power of the single processor (doubling every year from 1984 to 1992), but, on answering a specific question, excluded the possibility of development in the short term for multiprocessor architectures. We need to await the first results of the Menlo Park laboratory (where the real programmers of SUN and AT&T are working side by side) before we know whether the UNIX System V Release 5 will be a serious argument in support of these "visionary" talks.

Libert, Egalit, Fraternit

The *i2u* Convention was not just a conference and debate on strategic subjects, it also put on tutorials and product exhibitions.

The success of the tutorials is, I believe, confirmation that the Italians are much closer to the Japanese than to the Americans and are therefore "hungry for knowledge", and are interested in attending a UNIX convention to gain more insight into technical subjects, as well as hearing about AT&T and OSF. Incidentally, the tutorials on the first day were perhaps the only time when the organisation of the Convention proved a little shaky, especially since the "desire for knowledge" of people interested in UNIX was overestimated. This was a fault that is bound to be easily rectified next year.

The exhibition was rather drab, with stands all designed the same, the exhibitors names on impersonal and simply laid out cards. A whole set of "signals" that convey a *message* that even the most unaware visitor could not fail to miss: UNIX levels out everyone - large and small. Market opportunities can be full filled by offering better machines at more competitive prices: the starting point must be the same for everyone. Of the 31 exhibitors present, all of the main suppliers selling systems were there, and some of the software houses more active in the UNIX environment.

Conclusions

Anyone attending the *i2u* Convention to find out if it is worth investing in UNIX came away with definite answers: Masiero (IDC) was very clear on this point: "you can count on UNIX, now" is the moral of his talk, strengthened by dozen of bar-charts, pie-charts and trend analysis. "You can count on UNIX" has been also the six-column title of a report about the Convention that appeared afterward in the main Italian economical newspaper.

Anyone attending to see whether we will finally manage to combine all the versions into a single standard came away with provisional answers; the situation is certainly better than in the past, but we have still not reached a single UNIX version that is the same for everyone: anyone deluding themselves that the announcement of convergence between System V and SUN/OS (i.e. UNIX BSD) was the decisive step, would have been dismayed to hear the announcement of OSF. In my view the panorama that is unfolding is however acceptable; as we can see now, we have shifted from a past scenario with 3 or 4 versions (System V, Xenix, BSD, and X/Open), all quite different from each other, to a future scenario where we will have two versions at the most: System V.4 and derivatives on the one hand, and OSF on the other, with an extremely high degree of overlap between them (since both refer to the same consolidated standards: POSIX, X/Open). The world of UNIX has always been used to having different versions at the same time: continuing to follow the path already mapped out presents no big problem.

If I can quote Henry Spencer, one of the technical "gurus" in UNIX, who says: at a given point in time there is a set of functionalities that everyone treats as "standards", and another set seen as "expansions" on which there is no agreement as yet; at a later point in time the set that agreement has been reached on is enlarged, because consensus is reached, but at the same time the same set of expansions broadens, since other technological areas are incorporated or new applications are required.

1988 has been a significant time for both these sets.

The Trouble with PostScript and Device-independent Troff

Markku Sakkinen
markku@jytko.jyu.fi

Department of Computer Science, University of Jyväskylä
Seminaarinkatu 15, SF-40100 Jyväskylä, Finland

Graduated in mathematics in 1971 and got the Licentiate degree in 1972. However, had already lost his heart to computers and was not quite bright enough for a mathematical career. Worked at the university computing centre and also at the department of physics (putting up a laboratory computer system). Then did several years of Real Work in the Real World (sort of industrial automation mostly). Returned to his dear old university in 1985, partly in the hope to get a PhD before retirement age, but got to bear hardships, e.g. UNIX™.

Why the trouble to write this article?

I have done a lot of work, somewhat intermittently during a rather long time, trying to make the device-independent Troff formatter and a PostScript™ printer serve well the text-producing needs of our department. The documents range from one-page letters to entire books, very typical being journal and conference papers such as this one. The main languages are Finnish and English. There is not much mathematical notation on the average, because the principal area of research is information systems; the department belongs to the faculty of Social Sciences. The principal computers of the department's own are a DEC™ VAX™ (with 4.3BSD) and a newer Sun-3™.

Reflecting on the experience, I concluded that there are sufficiently important drawbacks in all software products concerned that they be brought to the attention of both those products' developers and the general public. This is all the more true since PostScript is now fashionable everywhere and the Roff family of formatters (Ditroff and the whole Documenter's Workbench™ especially) probably continue to be utilised by a significant part of UNIX installations for some time to come. Also, I have not encountered much critique on PostScript even in books like [Earn] and [Holz], at least not on those problems that I will stress in the sequel. This paper will not be only a list of complaints; I hope that it can help other people to install and exploit text formatting systems more smoothly, and give some useful or interesting insight into the principles of these tools.

PostScript

Basics

PostScript [Adob1, Adob2] is a page description language and a stack-oriented programming language at the same time. While a print job for a conventional printer is a sequence of printable characters interspersed with control characters and escape sequences where needed for special printer functions, a print job for a PostScript printer is a PostScript programme, in which the actual text to be printed is normally, but not necessarily, interspersed as literal strings. The print job can also output "anything" back to the client (host computer). The PostScript language is very interesting, flexible, and powerful in many respects: it contains e.g. a rich set of graphic operations. Nevertheless, there are surprising limitations in its primary function, namely the printing of characters.

The names by which fonts are known in PostScript are long and explanatory, like 'Times-Roman'™ or 'NewCenturySchlbk-BoldItalic'. The basic defining entity for the character set of any built-in PostScript font is a dictionary (associative array) called **CharStrings**. Its keys are the names of all characters in the font; for the upper and lower case letters of the English alphabet, the names are the letters themselves as one-character ASCII strings, for other characters, they are such rather self-explanatory words as 'three', 'comma', 'asterisk', 'parenleft', 'Sigma'. The value (object) associated with each key is the actual shape description that PostScript uses to draw the character.

According to the general principles of PostScript, it should be possible to get a shape description from **CharStrings** by using a character name as a key, convert it into executable and execute. In fact, this is how the **show** operator finally prints every single character, according to section 5.4 of [Adob2]. However, if one tries to do the same directly in a PostScript programme, nothing happens. My conclusion is that the true shape descriptions are completely protected even from execution (presumably for proprietary reasons) and that only the **show** operator has special access to them. Therefore, the only way to get characters printed is to use the **show** (or **kshow**) operator to strings of *encoded* characters; it is the encoding that proves to be a bottleneck in the business.

Character encoding

PostScript defines *strings* as special arrays with small nonnegative (0 to 255) integer elements. When a string is to be printed, the language itself does not define how these codes should be mapped to characters. In order to interpret the codes, each font description must contain an array of 256 character names called **Encoding** — user-defined fonts will not be accepted if they lack this array. String elements are simply used as indices into **Encoding**. There is an array in the system, **StandardEncoding**, which maps all printable ASCII codes and many codes between 161 and 255 to PostScript characters; it is used by most standard fonts as their **Encoding**.

In the normal case, a host computer communicates with a PostScript server (printer) by 8-bit or 7-bit transmission. The whole PostScript communication except for strings needs only 7-bit ASCII characters. When 7-bit transmission is employed, string characters with codes above 127 cannot be sent as such but as a four-character sequence: backslash followed by a three-digit octal number (digits coded in ASCII). This method, borrowed from the C language, can also be used to pass codes that would otherwise conflict with ASCII control characters.

The above encoding scheme is quite flexible. There are examples in the book [Adob1] on how to change the encoding to add Scandinavian characters and how to define an encoding to accept EBCDIC instead of ASCII characters in strings. Unfortunately, the scheme has drawbacks also. Because of possible conflicts with ASCII control characters, 67 codes should in principle not be used for printing characters, leaving 189 usable codes. This precaution can be necessary due rather to support software than to PostScript itself (cf. chapter 3), as the octal escapes are available. The standard encoding actually defines only 149 characters, and therefore 61 characters that exist in the normal fonts were left unencoded already in PostScript version 25.0. That is why one has to change the encoding first in order to get Scandinavian characters printed, as mentioned above. In PostScript version 47.0, other built-in standard text fonts have 228 and the Courier fonts 260 different characters; there is thus no way to have all of them encoded at the same time.

It is real lack of foresight that a page description language designed in the 1980's has been constrained to an 8-bit character code. After all, 8-bit EBCDIC code was introduced with the IBM® System/360 in the sixties, though most of the 256 possible characters were initially undefined there. With the backslash-octal number notation as it stands, one could express 512 different encoded characters. By devising a more efficient scheme, one could access even much larger internal character sets with "escape sequences" of just three (instead of four) ASCII characters. The most frequent characters would still be transmitted as themselves. Larger encoding vectors would demand more storage space, but they would not need to be excessively large. As regards the space taken by strings to be output, a PostScript server would in typical cases need to store only one word at a time in string form, in principle. The primitive memory management of current implementations (cf. section 1.4) evidently implies that in practice it can be necessary to store simultaneously all strings belonging to one print page.

As explained, a PostScript user is presently stuck with the 189-character limitation in the worst case, and cannot keep all characters that actually exist in a font mapped. In consequence, the need to temporarily access an unencoded character can arise. There would appear to be two ways to do it: either modify the encoding temporarily, or try to get the character printed directly from **CharStrings**, thus bypassing the encoding. Unfortunately, the second, more natural and straightforward alternative does not seem to work (cf. section 1.1), so we will look at the first one.

More about character sets

Modifying an encoding vector is not as simple as one might think, mainly because one must first copy a lot of structures, then do the actual modification, and lastly define the modified font as a new font to PostScript. (See the "cookbook" programme 18 in [Adob1].) The source code to accomplish this can be downloaded as a PostScript procedure once and invoked with appropriate parameters every time one or more new characters must be encoded. Even then, the extra work to be done by the PostScript engine will slow down printing. Moreover, there is a snag in the whole principle of re-encoding: Some (accented) characters are internally represented as composites of two or more simple characters of the same font. If such a composite character is encoded but one of its components is not, the character code will not cause anything to be printed. One must *guess* all such dependences, they are not clearly documented anywhere.

The composite character snag mentioned in the previous paragraph seems to imply that even the access of the **show** operator to the character descriptions is queerly bound to the encoding vector of each font. It is queer because surely the description of a composite character cannot refer to the components by code; otherwise the encoding of any component characters could not be changed without causing bizarre results. The PostScript language should be made more consistent and orthogonal in the handling of characters. Since an **execute-only** access attribute can be defined for any PostScript object, it would hardly constitute a risk for secrecy to let every character description be directly executed. This method to print seldom-needed characters would be much more efficient than the modification of encoding vectors.

Almost all *non-PostScript* printing devices and many display terminals of today recognise **national character sets** as a dimension orthogonal to **fonts**. At least as long as we have a lot of 7-bit-oriented hardware and software, this will remain an important distinction. An alternative to national character sets is one comprehensive international character set. PostScript does strive in this direction — the standard fonts of version 47.0 contain even the Thorn and Eth of Icelandic, so that virtually all languages with a Latin-based alphabet are catered for — but the encoding bottleneck makes it difficult to utilise the large character set efficiently, as explained earlier. Probably the most convenient way to introduce national character sets into PostScript would be to allow user modification of the **StandardEncoding** vector so that all fonts bound to it would automatically be affected. (If this article had been printed "at home" instead of being sent electronically as troff source text, you would see the Icelandic letters above; cf. section 2.3.)

There is in current PostScript one Symbol font, which comprises Greek letters, a host of mathematical and other special symbols not found in "standard text" fonts, and some punctuation characters common with other fonts. This approach is familiar from the various *roff programmes. It is probably a convention from the times when typesetter fonts had a very limited number of characters to regard this as a **font**. More naturally, the special symbols would be regarded just as an extension of the character set; they should be available in several different fonts, at least as Roman, italic, bold, and bold italic variants, why not in more than one typeface family as well.

A memory problem

One "programming-language" aspect of PostScript can have somewhat surprising effects on printer operations (cf. section 3.7 of [Adob2]). PostScript does not store composite objects (arrays, dictionaries, etc.) on the operand stack, only pointers (descriptors) to them; the actual objects are stored in heap (dynamic) memory, called *virtual memory* in PostScript parlance. But there is no explicit operator to deallocate a composite object that is no longer needed, nor is there automatic garbage collection. Therefore, memory gradually fills up during the execution of a PostScript programme. This is generally

harmless in ordinary printing jobs, because virtual memory is cleared between jobs (programmes). Also, there is an operator that restores the whole virtual memory to a state saved earlier; this is typically called between pages.

However, it is possible to put objects into virtual memory so that they will remain there from job to job, until the printer is physically switched off.. This facility is deemed so machine-dependent that it has been left out of [Adob2] (cf. section 1.6); it is (or should be) described in the specific manual of every PostScript printer. One would often like to download both re-encodings of built-in fonts and complete non-standard fonts just once and then be able to utilise them in all jobs. In a typical spooling environment, a user cannot always be sure whether such a downloading has already been done after the most recent powering up of the printer or not, and may thus send the file again. What happens is that a new set of objects is created but the old ones are not deleted. If this is repeated, virtual memory fills up and print jobs start to fail, first those which create the most complex pages. To prevent this problem, any downloading file (PostScript programme) should be designed to test first and exit if the downloading appears to have already been done.

Client-server communication

One drawback of PostScript in comparison to most other page description languages *is* mentioned fairly often, e.g. in [Earn]: because of its human-readable style, long keywords, etc., it requires much more characters to be transmitted from client to server for the same printed text. This can be an important factor when slow communication links must be used. However, in typical cases client and server are physically close and have a relatively fast connexion (maybe even a parallel interface or a LAN); the data transmission will hardly be a bottleneck in the printing process.

Perhaps a more serious problem in PostScript communication is the protocol. PostScript printers can provide versatile feedback to the client, which is marked progress from conventional printers. But the data stream from printer to host computer may in the worst case contain both output explicitly generated by the current PostScript programme (print job), spontaneous messages from the printer, and responses to the host's status requests, all arbitrarily interspersed. This makes life hard for the host interface software. For instance, the filters in the TranScript package (section 3.2) can get so confused in some problem situations that the super-user must do several clean-up tricks (besides powering the printer off and on) before they will transmit anything again.

How to get information

There apparently are at least two different versions of the reference manual [Adob2] around. One version contains the device-specific information for Apple™ LaserWriter, of which most is applicable to other PostScript printers as well, but its main text part corresponds to PostScript release 23.0 (even in a 7. printing from August 1987!). The other version is updated with the features of PostScript release 25.0, e.g. packed arrays and a couple of new standard characters, but it does not contain the device-specific appendix. I do not know about any documentation that would tell about what new things there might be in release 47.0. The manuals [Adob3] and [Qume] do not even mention the considerable number of new characters that have been added to the built-in fonts — I had to find out myself.

Adobe Systems have defined a standard format called AFM (short for Adobe Font Metrics, obviously) for files describing metric (and encoding) information of PostScript fonts. This information is essential to host computer software, of course, but it is also meant to be readable by people. It may not always be easy to get an AFM file for every font in one's printer anywhere: the manufacturer or representative of the machine does not know or care, and support software packages typically support only a few fonts. Fortunately, it is possible to dig out approximately the same information out of the printer itself with a clever PostScript programme, but that requires a lot of hard work. We will return to these questions in chapter 3.

Device-independent Troff

Introduction

The device-independent (or typesetter-independent) Troff [Kern1] is a member of the Roff family of batch text formatters. In several contexts, e.g. the AT&T documentation, it is called simply 'troff'; if it is needed to speak about the old troff also, the latter is called 'otroff'. To avoid such confusion, we will henceforth call the

new formatter 'Ditroff'. It is part of the UNIX Documenter's Workbench (DWB) — the whole DWB package includes several preprocessors, a new version of Nroff, and other stuff [AT&T1, AT&T2], but Ditroff is practically the only component that one can install (with some effort) even on a 4.2BSD or 4.3BSD system. Some BSD-based UNIX systems have enough System V compatibility features to install practically all of DWB, but on a "pure BSD" system you cannot even read in the distribution tape directly because for some reason it is in **cpio** format. This discussion is based on release 2 of DWB: at least rather recently the release 1 has still been distributed as a standard or optional part of some microcomputer UNIX systems.

There are three important facets in Ditroff: what its input is like, how it works, and what its output is like. The input is essentially the same as for Troff (the old typesetter-specific formatter), but with some very useful additions. The main difference in the working is that Ditroff needs a a separate device description file and respective font description files for each supported printer type. The output is totally different from Troff output: it is printable ASCII throughout, always needs a postprocessor for actual printing, and thus is normally directed into a file or pipe. We will discuss the output format first.

Output format

The general format of Ditroff output is clear and simple: it consists of commands, each having zero or more (but typically one) parameters after the command character. White space never harms between commands, and it is even required after some commands to clearly delimit their last parameter. There are commands for motions, character printing, drawing, size and font specification, and device control functions. Very important for many purposes is that word, line, and page boundaries are explicitly marked with special commands, even though they cause no action in the default case when the whole file is printed on the type of device it has originally been intended for. Certainly, many people besides myself have sometimes been driven crazy by the stupidity of Nroff, which cannot be even forced to insert honest form feeds into its output.

Every page in a Ditroff output document is self-contained in the sense that all context setting such as mounting fonts and defining point sizes is done at the beginning of the page. Therefore, if a postprocessor is asked to print only selected pages from a large document, it is sufficient to read the general header information at the beginning of the file and the required pages, which can be found easily. It is possible also to feed Ditroff output to the postprocessor of another device than it was originally created for — this exactly is device independence. Besides such things that the new target printer physically cannot accomplish (e.g. impossible font sizes), most likely to go wrong then are non-standard (perhaps even some "should be standard") special characters. — The only information about the original target printer in a Ditroff output file is its name and resolution; this scarcity can further reduce the possibility of optimal printing on another kind of device, no matter how clever the postprocessor.

There is one exception to the above in the Ditroff output format, and that is its most common command: '*xya*', where *x* and *y* are decimal digits. This specifies that the current position is to be moved *xy* units to the right and then the character *a* is to be printed. It is thus a shorthand for the combination of a movement and a printing command: '**h***xy***c***a*'. However, the shorthand cannot be used if the movement is greater than 99 units. Surprisingly, there is no notation for a string of output characters to be printed with their default spacings; this would often reduce the size of an output file by more than 50% (cf. section 3.1). At least, besides '**c**', there could be a command to print one character and then advance the current position by its nominal width.

A nice flexibility in Ditroff is that the set of special characters is not fixed within the programme itself but defined in each device description file. Special characters are denoted in Ditroff output by the command character '**C**' followed by the special character's name followed by white space. This syntax would make long names possible, but they are not allowed in the input. It would be very practical if one could use PostScript character names for unusual characters in Ditroff input, too. It is hard to invent unique and mnemonic two-character names for large character sets. Font names cannot be longer than two characters, either; as many present printers have dozens of fonts available, the same difficulties arise as with special character names. If one gets some support software or Ditroff input text from two different sources, they will almost certainly conflict in their use of both character and font names.

There are no commands in Ditroff output (nor input, of course) to control text and paper orientation. It would be useful, though. Even many non-PostScript printers today allow both horizontal and vertical printing, at least. PostScript allows any oblique direction of text lines as well. The simple case in which a whole document should be oriented horizontally can be handled outside Ditroff, in a postprocessor or printing filter.

Fonts and character sets

Ditroff requires that there be, in a certain public directory, a respective subdirectory for each different output device, containing a device description file and font description files. It is this subdirectory that the device selection option '-T*name*' of Ditroff actually specifies. For every known font, the name of the font description file is the font's own name (at most 2 characters) suffixed by '.out'. Different PostScript printers may very well be described to Ditroff as the same device type.

The same non-existence of national character sets that was mentioned in PostScript holds *a fortiori* for Ditroff. All characters outside the U.S. ASCII set must be defined as special characters in the device and font description files used by Ditroff. Fortunately, in the input text one can avoid writing the special characters tediously everywhere: the **.tr** (translate) directive can be used to substitute special characters for ordinary ASCII characters. Thus, if one's terminal has the same national character set as the intended printing device, one can type even the non-ASCII characters as themselves when preparing the input text. An exception occurs because the escape character of Ditroff is the backslash, whose code is one of those officially reserved by ISO for national variants. For instance, in Finland and Sweden '\' is replaced by 'Ö'. Ditroff itself does allow the escape character to be redefined, but all standard macro packages count on its being the standard backslash.

The set of standard special characters defined in [AT&T2] is much too limited. For instance, it does not include any non-English letters except the Greek alphabet. This omission has the effect that Ditroff source text that contains such special characters is not portable between installations. Because the text of this paper was sent by electronic mail to the editor as Ditroff source text, I had to use one of the predefined strings of the **mm** macro package to obtain the letter 'Ö' in the previous paragraph — with rather inferior quality. In my own installation, I could have used the appropriate special character to print the PostScript 'Ö' with typographic quality. The same goes for the 'ä' letters in the heading of this article. In section 1.3, I had to omit the letters Thorn and Eth, so those readers who were unfamiliar with them before will still not know what they look like.

The alternative way for an installation to offer a national character set is to duplicate fonts — this seems to be a common solution, though not entirely attractive. For instance, many current PostScript printers have 33 different "standard text" fonts (and the number will increase with each PostScript release and printer generation). Even if just one national variant of each is needed in addition to ASCII, installations have trouble managing all of them and allocating Ditroff names to them. If one can remain in the same national character set in the whole of a document, this solution is adequate for the end user. If there is need e.g. to print ASCII braces and brackets, they can be defined as special characters in the national character set. However, if it is often necessary to change between national character sets, writing the input text becomes difficult. One difficulty is that always when changing temporarily to bold or italic style or another typeface, one must take care to specify the right variant of the required temporary font. Considering macro packages and trap handling, this can become complicated indeed. We will return to this question in the next chapter.

A user may run into a couple of font handling problems with the current release of Ditroff. It cannot interpret correctly font change escape sequences of the type '\f*N*' where *N* is a number, unless some font has been *explicitly* mounted on position *N* (the **me** macro package uses such sequences extensively). One may also get the error message 'Font *XY* too big for position *N*' when trying to mount a font; the cause can sometimes be that **biggestfont** (cf. below) has not been correctly defined when the device and font descriptions have been created.

Defining fonts

When one has to create or modify device or font descriptions, it becomes most apparent that Ditroff is not a perfectly mature and robust software product yet, although the original version

was taken into internal use at Bell Laboratories already in 1979, according to [Kern1]. There are severe limitations, not even hinted at in the manual [AT&T2], to what one can actually define. Most of these limitations are only detected when, after a seemingly successful generation of the description files, Ditroff simply crashes with a 'Segmentation fault' (or something equally informative) when trying to run with the new device. Customers with a source distribution can glean some of the limitations from various source files.

One limitation that is mentioned in the sources (and can even be changed by editing the appropriate header file) is that no more that 10 initially loaded fonts must be defined. Another is that the total number of distinct characters (that includes both ASCII and special characters) allowed in all fonts together is quite restricted. At least in the VAX source distribution, it appears to be 512. In contrast, the built-in fonts of PostScript version 47.0 already comprise over 600 different characters (counting the Zapf Dingbats™ font of fancy figures). Customers with a source distribution can extend this limit, too. But there can be at most 224 characters in a *single* font; one must wait for the next release of DWB to get this restriction (hopefully) lifted.

When installing font descriptions, one must take care that no font contains more characters (alias names are not counted) than the number **biggestfont** defined in the device description; otherwise the above-mentioned run-time error will appear. I have not tried out what happens if one tries to run Ditroff with a font exceeding the 224-character limit: whether one gets wrong characters, an error message, or a core dump.

The handling of special fonts is too primitive for some purposes. The definition classifies a font as 'special' or 'normal'. Whenever Ditroff finds that a character to be printed is missing from the current font, it searches for it in the mounted special fonts in the order of their mount positions. An explicit change to a special font is seemingly accepted by Ditroff but actually causes no effect! (There are even more subtleties with special fonts.) One should be able to specify a sequence of alternate fonts for each font separately. If this were possible, the font size limitation would not be serious.

The set of properties that can be described in the *device* description is not very complete or well thought out. For instance, it includes the list of possible point sizes, but it is not possible to tell whether the printer can perform height and slant modifications. Neither can the available drawing functions be specified. The width and length of paper, on the other hand, *are* in the device description. It is questionable if they belong there, since most printers can handle different paper sizes; at least there should be overriding options in the Ditroff command.

Input language

Everybody who has processed text with any *roff knows how difficult and awkward its commands and escape sequences can be. On the other hand, they are quite powerful. Various macro packages try to enable writing the input in terms of higher-level commands and in more structured form. Unfortunately, the effects of a macro package can interfere with those things that can only be done on the *roff level. This pertains especially to novel features of Ditroff, because many macro packages such as the Berkeley **me** are older and cannot take them into account.

Ditroff allows the slant of characters to be modified, as well as the height to be changed without affecting the width. Such operations are naturally not possible on all output devices, but on PostScript printers they are easy. For some reason, these operations are not available as ordinary commands, only as escape sequences, which can sometimes be impractical. Also, there are no predefined number registers in which Ditroff would store the current slant and height values, as it does to other similar parameters. Slant and height definitions seem to remain in force even across font and size changes.

Another novelty of Ditroff that cannot be implemented on all printers but can very easily be translated into PostScript are the drawing functions. There are escape sequences for drawing line segments, circles and circular arcs, ellipses, and B-splines. Of course, these cause very little work within Ditroff except calculating the position after the figure and passing the information to the output file.

A traditional lack in *roff capabilities that still remains in Ditroff is that it is surprisingly hard to get a desired amount of white space at the top of a page, at least on the first page of a document. Another is that one cannot specify character spaces to be increased or decreased in a similar

manner as line spacing can be adjusted with the .vs command. Such a capability would sometimes be useful for standout text. More importantly, some PostScript printers have so tight spacing in several fonts that characters touch each other; many people would like to avoid this displeasing effect. The lack is somewhat surprising since even such simple devices as Diablo® daisy-wheel printers have long offered 'increase character spacing' as a hardware operation.

Preprocessors and macro packages

Documenter's Workbench includes versions of the traditional *roff preprocessors Tbl (for tables) and Eqn/Neqn (for mathematical text and equations). A new preprocessor that can be used in connection with Ditroff only is Pic [Kern2]. It has primitives for picture elements such as line segments, boxes, arcs, ellipses, and splines. Very obviously, Pic and the new geometric facilities of Ditroff have been designed together. In release 2 of DWB, there is further a preprocessor to Pic, called Grap; its purpose is to plot time series and x-y graphs in an easy and flexible way.

The Eqn preprocessor is, unfortunately, not as device-independent as Ditroff itself: it has a command option for specifying the intended output device but it cannot use any external tables — device-specific information on the two supported printers is programmed in. Fortunately, this does not seem to matter much. An auxiliary file, **eqnchar**, is normally used in connection with Eqn to define more symbols (e.g. logical and set-theoretical connectives). There are two versions of **eqnchar**, each designed to fit one of the DWB-supported printers. Most of the symbols become bad-looking or even unrecognizable on other (e.g. PostScript) printers.

The Pic preprocessor has hard-coded device dependencies like Eqn. When drawing vertical and horizontal lines, Pic tries to exploit appropriate special characters as much as possible, in preference to Ditroff's line-drawing primitives. Since Pic does not access the font tables of a non-supported printer, the sizes and alignments of those horizontal and vertical bars are not necessarily what it excepts; lines and boxes are thus not likely to come out well. The Pic version distributed with release 1 of DWB allowed the user to specify the resolution of a non-supported output device [AT&T3]. The Pic *manual* from release 1 also mentions that there is a '-D' option that instructs Pic to draw all lines by using graphic escape sequences. Fortunately, current Pic still seems to know this option, and so the just-mentioned line alignment problem can be avoided.

As was mentioned in the previous section, people normally do not write "raw" *roff input code, but use some standard or home-brewed macro package that offers a higher level of abstraction. "Standard" should really be placed into quotes because there are so many "standard" macro packages and rather widely differing versions of each. There is only one that is included in virtually every UNIX system: **man**, which is strictly specialised to formatting traditional UNIX-style manual pages. As for general-purpose macro packages, in BSD environments there are **ms** and **me**, whereas System V sports **mm**. (DWB further includes the macro packages **mv** for view graphs and slides, and **mptx** for formatting permuted indexes.) Consequently, I have always worked with the **me** macros, but when I got the EUUG article template and directions from the editor (for this very paper, which was then already written to the greater part), they suggested using **mm**. Well, I have converted my text, and could even test it by using the version of **mm** distributed with the DWB, but let us see if the macro package at The Instruction Set is compatible or not. If I had originally used really fancy features of **me**, the conversion to another set of macros would have been very tedious if not impossible.

Exploiting a "standard" macro package can thus impede the portability of *roff source text. One's own macros will not cause this drawback, as they can always be prepended to the main text. Another kind of difficulty arises when one needs to do some operations on the *roff level: it may interfere in nonobvious ways with the workings of the macros. One must be especially careful against conflicts in the names of number registers, strings, and macros — the packages have many internal macros in addition to those that can be called directly by the user.

Middlemen between Ditroff & PostScript

Principles

As mentioned in chapter 2.1, Ditroff always needs a postprocessor for the actual printing. To be exact, there is an option to the Ditroff command that produces "a printable ASCII approximation of the results", but it is interesting only for

preview purposes. Because the output language is simple, it is true to a large extent what [Kern1] states:

> "... it is straightforward to write a prototype driver for a particular typesetter, especially when one can steal an existing one as a model, although it may take some effort to make one of production quality."

Still, some qualification is necessary for this statement. Putting up the font description tables, which are normally used by both Ditroff and the postprocessor, can be a bigger task than writing the driver / postprocessor itself.

If the model of the printing process employed by a printing device is very different from that of Ditroff, it can be more difficult to write a good postprocessor, or perhaps even impossible to exploit all capabilities of the printer. For instance, this is true about the national character sets. I have written, for a non-PostScript printer, a rudimentary postprocessor which understands some special characters (with zero width) as instructions to change to different national character sets, but this is possible only because the fonts of the printer concerned are of fixed width (Ditroff does not know of the change when looking up the characters' widths). Another difficulty can arise from a too large character set in the printer. The same kind of trickery — using some special characters to switch from one part of the character set to another — could be imagined even here if there are enough pairs (triples, quadruples, ...) of characters of the same sizes so that they can be mapped on the same Ditroff characters by pairs (triples, quadruples, ...). It would certainly not be easy. If there are only more characters *in one font* than Ditroff can support, then one must define the printer font as two or more Ditroff fonts. This can make it very unconvenient to use the font in practice.

Writing a Ditroff-to-PostScript translator cannot be very easy for the reason that PostScript is a complex language and difficult to debug. Furthermore, the postprocessor alone is not enough: at least the character width tables of all fonts are absolutely necessary for Ditroff to work. Therefore, one is not very tempted to "do it oneself" but rather to buy a commercial postprocessor. Normally one will get some useful goodies on the bargain, in addition to the absolute necessities.

PostScript documents have not necessarily the same kind of page structure as Ditroff output: in the general case, they are programmes rather than "documents". Nevertheless, for the overwhelming majority of PostScript programmes that are straight documents, there is a page-structuring convention recommended in [Adob2]. Postprocessors usually follow that convention. It enables printing filters and utility programmes to select pages from a document, to print in reverse order if the sheets will stack in right order that way, to find out quickly all fonts required by a document, etc.

The necessity of explicit spacing information between characters in Ditroff output (section 2.2) looks somewhat silly especially when the output is converted to PostScript: within each word, all explicit spacing disappears again in normal cases. Thus, a source text word 'Computer' may appear in Ditroff output as 'cC72o50m89p57u61t39e50r', but in PostScript again as '(Computer)'.

TranScript

TranScript™ is a PostScript support software package for UNIX systems originated by Adobe Systems Inc., the developers of PostScript itself. The following discussion pertains to the version adopted for Sun workstations and supported by Sun Microsystems, release 2.0 [Sun1]. The package contains filters that convert plain text, Ditroff and old Troff output, Sun raster files, and several other device-specific output formats, as well as **plot** output, to PostScript. A great number of PostScript programmes comes with TranScript, including all "cookbook" examples of [Adob1]. However, we will concentrate on things that have to do with Ditroff.

Some of the positive sides of TranScript are the following: It has a good collection of print filters, which can also collect any output coming back from the PostScript printer to a log file. Even otherwise, the standard spooling system as such would not be able to handle PostScript communication intelligently. Most of the previously mentioned conversions can be effected simply by the appropriate **lpr** options, which is especially handy for remote jobs. The actual Ditroff postprocessor has options to orientate the output horizontally and to set the printer in manual feed mode for the duration of a job that needs special stationery. It is possible for a knowledgeable user to embed PostScript code within Ditroff input. The distribution includes

complete AFM files of all supported fonts.

Some of the negative sides of TranScript, according to personal experience, are as follows: At least here in Finland, the package was very expensive (even the one-CPU binary licence). It does not support the newer PostScript fonts, only the Times, Helvetica™, and Courier families and Symbol. Naturally, it does not know about the extended character set of the current PostScript versions. The truly *catastrophic defect*, however, is that there is no possibility to use Scandinavian letters from Ditroff (after several months' waiting, we got support from the Sun representative for printing *plain text* in the Finnish national character set). Another part of this "U.S. only" attitude is that it is not possible to specify paper sizes, the American parochial standard size is silently assumed. This can be fixed somehow for Ditroff by editing the appropriate PostScript prologue file (anything that looks like 11 or 8.5 inches there has a high probability to mean paper length or width, respectively). With the **enscript** programme that converts plain text to PostScript, the user can do nothing because the paper size is programmed in.

Devps

Devps™ [Pipe] is a PostScript support package developed by Pipeline Associates, Inc. Its European representative is Penetron Oy, who have added national character set support to their distribution. They have also added a programme to generate Ditroff device and font description files, because that is not included in the *binary* DWB distribution kit from AT&T. Our department has used Devps mainly to fill the "holes" in TranScript, but the reader should note that it lacks some sophisticated features that TranScript does have, in turn.

Some aspects in which Devps is better than TranScript follow: It is available at a reasonable price, even as multi-CPU source licence. It supports the Palatino™ font family and includes some downloadable fonts (outline and Hershey fonts); there is also advice on how to make desired downloaded fonts resident beyond jobs. It includes PostScript programmes that can e.g. extract font metric information from *the printer*, albeit slowly, either in AFM format or ready for Ditroff font description. The documentation is more complete. Most important to us are the national character sets; they are implemented as different fonts so that there is a U.S. Times-Roman and a Finnish Times-Roman, etc.

Some aspects in which Devps is clearly worse than TranScript are: There are conversion programmes for Ditroff output and plain text only, the latter being much simpler than the corresponding filter in TranScript. There is no print filter that takes care of any output back from the printer (some tricks are suggested in the manual for that purpose; their success may depend on the particular UNIX version). No AFM files are included in the distribution.

There are many other support software packages similar to TranScript and Devps on the market, e.g. the third-party software catalogue [Sun2] lists several of them. They are often bundled with DWB, which is practical for the customer. I have no experience on any others than the two packages just described.

Some own work

The combination of an input-logging print filter from TranScript and information-extracting PostScript programmes from Devps encouraged me to some investigations and further developments of my own. In the beginning, making even trivial-looking modifications to PostScript programmes tended to make them crash. PostScript is really difficult to use as a real programming language, mainly because one must take care of the stack being always in the right state for each operation. Nevertheless, I got listed the names of *all* characters (the unencoded ones included) in each built-in font from its **CharStrings** dictionary and saw that there were so many new characters unmentioned in any documentation (cf. section 1.2).

The next step was modifying the PostScript programme that constructed metric information, to handle more characters. Ditroff names had to be invented for the new characters. Also, the PostScript programme that changed the encoding for Scandinavian fonts (in the printer) was enhanced so that almost every code between 128 and 255 is now mapped to some PostScript character. I finally got rid from using alternate (U.S.A./Finland) fonts — the 7-bit codes now have their Finnish interpretation and ASCII characters such as '[' and '\' have special character names. There was quite a lot of work before all device and font descriptions for Ditroff were in order. Fortunately, we have a source distribution of DWB: Ditroff and its auxiliary **makedev** programme had to be rebuilt with several array sizes enlarged (cf. section 2.4).

These developments have been made around the postprocessor of Devps, for which there was already a good beginning from Penetron. One would have needed still much more work for the postprocessor of TranScript; I studied that possibility first and fairly extensively. A problem that has surfaced later is that some kinds of complicated input (e.g. as produced by **eqn**) can crash Devps on the Sun, but fortunately not on the VAX. The complete work flow in document formatting at our department is thus funny: Ditroff is normally run on the machine each particular user happens to have his/her session on, either Sun or VAX; its output must in some cases be sent from the Sun to the VAX for postprocessing by Devps (i.e. converting to PostScript), but in most cases Devps could be run on either computer; the PostScript document is always printed on the Sun (to which the printer is physically connected), using the filter from TranScript.

Conclusion

Both device-independent Troff and PostScript (printers) are useful tools, but presently their cooperation is not as smooth as could be hoped. You can expect to utilise them together without a lot of initial work and involvement on your side mainly if you need only English letters and a limited set of special characters in your texts. However, this depends on the Ditroff-PostScript support software. When selecting support software, check very carefully that it really has all the features your application needs — compare at least two or three packages. You cannot exploit many of the fancy capabilities of PostScript (that can sometimes be stressed in printer advertisements) in combination with Ditroff. And, unless your UNIX is System V or very well compatible with it, you may not be able to install Ditroff at all, not to speak about other components of the Documenter's Workbench. The portability of Ditroff source text is problematic.

It is really to be hoped that a new release of DWB would appear soon, be a production-quality tool, and remove at least the most irritating limitations of the current release. As for PostScript, some obvious improvements would be welcome in order to maintain its position as a leading page definition language. The updating of available PostScript documentation to describe the newest releases of the language would be really urgent, of course.

Acknowledgements

This work was supported by the Ministry of Education (doctoral programme in information technology) and the Academy of Finland.

References

[Adob1] Adobe Systems Incorporated, PostScript Language Tutorial and Cookbook, Addison-Wesley 1985.

[Adob2] Adobe Systems Incorporated, PostScript Language Reference Manual, Addison-Wesley 1985.

[Adob3] PostScript Language Supplement for the Qume ScripTEN (Version 47.0 Revision 1), Adobe Systems Incorporated 1987 *(Reorder no. 32147)*.

[AT&T1] UNIX System V Documenter's Workbench Software Release 2.0 User's Guide, AT&T 1986 *(310-004 Issue 1)*.

[AT&T2] UNIX System V Documenter's Workbench Software Release 2.0 Technical Discussion and Reference Manual, AT&T 1986 *(310-005 Issue 1)*.

[AT&T3] UNIX System V Documenter's Workbench Software 1.0 Preprocessor Reference, AT&T 1984 *(307-153 Issue 2)*.

[Earn] Rae A. Earnshaw (ed.), Workstations and Publication Systems, Springer-Verlag 1987.

[Holz] David A. Holzgang, Understanding PostScript Programming, SYBEX (San

Francisco) 1987.

[Kern1] Brian W. Kernighan, A Typesetter-independent TROFF, *Computing Science Technical Report No. 97*, Bell Laboratories 1982.

[Kern2] Brian W. Kernighan, PIC — A Graphics Language for Typesetting, Revised User Manual, *Computing Science Technical Report No. 116*, AT&T Bell Laboratories 1984.

[Pipe] Devps™ User's Guide *(and associated manual pages)*, Pipeline Associates, Inc.

[Qume] ScripTEN Guide To Operations : Printer Installation and Operation, Qume Corp. 1987 *(Reorder no. 32206)*.

[Sun1] Installing the Sun™ LaserWriter *(and associated manual pages)*, Sun Microsystems, Inc. 1986 *(Part No. 800-1249-04, Rev. A)*.

[Sun2] Catalyst, January 1988, Sun Microsystems, Inc. 1987.

Warning: The reports [Kern1] and [Kern2] can be hard to obtain and very expensive in proportion to their extents of 13 and 26 pages respectively.

Note: See chapter 1.6 on the different versions of [Adob2].

News from the Netherlands

Frances M.T. Brazier
frances@psy.vu.nl
vupsy!frances

Department of Cognitive Psychology
Vrije Universiteit
Amsterdam

Frances is the secretary of the board of the NLUUG, and is their representative in the EUUG's Governing Board.

Conferences

Spring Conference

Our last conference in May was on Integrated Project Support Environments - IPSEs. As our experience with poster sessions at the spring conference a year before had been quite positive, they were again held at this conference. In contrast to our autumn conference, the spring conference is not adorned with an exhibition.

The Board of the NLUUG experimented with a slightly different form of organisation for this conference. Instead of appointing one member of the Board to be responsible for the programme, a programme committee was formed in which three members of the Board and one member of the NLUUG (with more knowledge of the subject) participated. This formula worked to our satisfaction and will therefore be followed in the future.

The following presentations were given in May (if more detailed information is required addresses are available):

— Introduction to IPSEs
Phil Mair, The National Computing Center Ltd, Manchester

In the opening talk, an overview of the architecture and levels of integration of IPSES was provided. The basis for discussions on developments within this area was thus formed: the Alvey Committee's proposal was discussed at length, the two current standards initiatives PCTE and CAIS were placed in perspective, and current implementations of commercial-DP IPSES and real-time IPSEs were considered.

— Object oriented design with workstation-based tools and the importance of the EDIF standard
Jelle Visser, Pierre Schmit, Koning en Hartman Elektrotechniek, Delft, Cadre Technologies, Switzerland.

The possibilities available for the implementation of an ADA developmental environment in all phases of a project, were discussed. The first step taken in this direction is the design and implementation of a Buhr

diagram-editor. Further development of tools as such, however, requires specification of the interface. A uniform interface standard for the exchange of data between CASE, CAE and ATE environments is necessary. The EDIF standard is an attempt to bridge the gap.

— IPSEs: background information, categories and expectations
Frans Ververs, Technical University, Delft

The problems encountered in this field of research were introduced on the basis of the terminology employed. The possible and plausible combinations of the terms: integrated, development, software, programming, system, engineering and environment employed, do not necessarily provide the most efficient categorisation of the systems currently in development. The types of users for which such systems are designed is more likely to succeed on this count. The advantages of UNIX and the current state of art were discussed. Optimistic views of the situation in both the near (1995) and the far future (2nd generation IPSEs), were presented and discussed.

— The Network Software Environment - NSE
Dick Mansvelder, Sun Microsystems Nederland BV, Soest

NSE is an environment designed for the support of large software development projects for the management of parallel development in heterogeneous networks. The major concepts employed in this system and a few of the problems encountered were addressed. Naming and management facilities for all types of objects were mentioned, as were facilities for hierarchical distribution and configuration of objects during development.

— PIMS - Project Integrated Management System
Dick Biekart, BSO, Eindhoven

The first prototype of PIMS, a system that is being developed within the EEG ESPRIT programme, has been implemented in a SUN-3 environment in Common Lisp and Flavors. PIMS is an integrated set of tools designed to support the management of software projects. Tools for task definition, planning resource management, tracking, reporting and risk analysis have been implemented, all with the same interface and all within the same object oriented project model. Each tool presents a different view of the model, depending on the relevancy of the information required. The object oriented information manager is not only responsible for the the model itself but also for the knowledge base (with data on project management, and parts of the user interface. The tools communicate with these modules via the information manager. The information manager is furthermore responsible for communication between the tools themselves.

— Computer aided development and maintenance of complex systems using EPOS
Peter Göhner, Gesselchaft für Prozesrechnerprogrammierung mbH, Germany

The fundamentals of this engineering and project management oriented support system were discussed in detail. The seven interactive modules: Requirements, System Design, Project Specification, Analysis, Management, Documentation and Communication were described separately, but also in connection with the rest of the system. The advantages of such systems (consistent, complete and unambiguous system or software requirements documents, well-structured, well-documented software design and code with very low residual error levels, time effective project management, etc) were discussed on the basis of this quite extensive system.

— Project Support Environments, Integrated?
Dik Fikker, Fysisch en Elektronisch Lab/TNO, Den Haag

The FEL is involved in a number of international, (some military) initiatives concerning the development of PSEs. An overview of the initiatives, goals and purposes of the international projects was presented. The need for integration was again emphasised; the problems involved were discussed on the basis of a number of models (APSE, Special ER-model). The insight thus provided was strengthened by an explanation of PSE, the proposed model for the implementation of PTIs (interface sets). The success of integration was, however, as yet not achieved. This model was not sufficient for an IPSE: the syntax has been defined (in PTIs), but the semantics are still missing.

— UNIX and RTEE: the Real-Time Engineering Environment
M.J.A. Steltman, Westmount Technology, Delft

The title suggests a broader topic than was discussed. The RHTX debug environment, a component of the RTEE, was the topic actually addressed. RHTX is a host-target development system which enables programs to be developed and generated on a host-workstation and debugged and tested on a real-time target microprocessor system connected to the host. The architecture of the system was described in detail.

Autumn Conference

Pre-announcement

NLUUG Autumn Conference 1988
UNIX and Standardisation
November 10, 1988 at De Reehorst
Ede, The Netherlands

Unix and standardisation
With the increasing importance of UNIX for a large number of applications and for diverse areas of data processing, the need for the definition of formal standards has increased. Effort is currently being spent, by various organisations, to produce UNIX-related standards.

For instance:

Portable Operating System Interface
- POSIX (IEEE 1003)
System V Interface Definition
- SVID (AT&T)
Common Applications Environment
- X/Open
C Language
- ANSI X3J11
Systems Application Architecture
- (IBM)

In addition, standardisation activities in the areas of communication and window management, of great importance to UNIX users, should not go unmentioned, for example:

ISO OSI	TCP/IP
NeWs	X-Windows

Not all developments follow standard specifications: universities develop their own "standards".

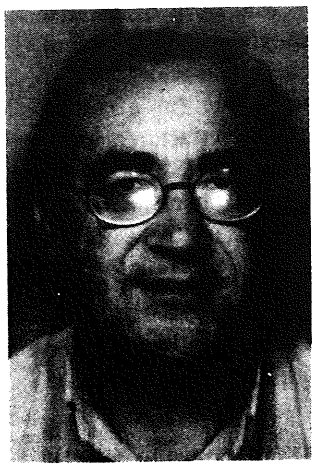
The Dutch Unix Users Group (NLUUG) is organising a one-day conference on November 10, 1988, on the above mentioned standardisation activities. The programme will include technical sessions, a number of tutorial sessions and an exhibition of UNIX hardware and software products.

All EUUG members are more than welcome.

AFUU News

Philippe Dax
dax@enst.enst.fr

Ecole Nationale Supérieur des Telecoms



Philippe Dax has been the editor of Tribunix (the French Unix Newsletter) since 1985. He discovered Unix V7 on an Onyx in 1981. He was part of the bunch that created the AFUU in 1982. Philippe also wrote a book on C in 1983.

Philippe is an engineer in the Computer Centre at ENST (Ecole Nationale Supérieur des Telecoms), where he struggles with "Unity" under VMS in the spirit of Unix philosophy, e-mail, and daily clears our spool directories. He is waiting for Godot (native Unix workstations)!

Chers adhérents,

Après l'annonce de la constitution de l'OSF, ``l'Open Software Foundation'', par IBM, DEC, HP, Apollo, Bull, Siemens et Nixdorf, de nombreux adhérents ont demandé á l'AFUU quelle était sa position. De plus une discussion s'est engagée sur le réseau Eunet pour susciter commentaires et réactions.

L'actualité exige de commenter ce mouvement. Certains ont parlé du partage de la Planète, de guerre ouverte entre deux géants, de ''Yalta de l'informatique'', de torpillage de la standardisation, bref, de quoi se régaler dans le choix des titres á sensation. D'autres préfèrent ne pas trop alimenter le débat, pensant que cela ferait une mauvaise publicité á UNIX.

A mon sens (je ne parle pas au nom de l'AFUU mais exprime ici mon avis personnel), toute cette histoire démontre qu'UNIX est devenu le véritable enjeu des années 90 jusqu'á l'horizon de l'an 2000 et probablement au-delá (sous une autre forme).

Dear members,

After the announcement of the creation of the Open Software Foundation (OSF), by IBM, DEC, HP, Apollo, Bull, Siemens and Nixdorf, many members have asked the AFUU what its position is on this development. There has also been a discussion on the network (EUnet), trying to find out what are people's attitudes.

Certainly, some comment is needed. Some have spoken of 'Partitioning the planet', of 'Open war between two giants', of 'The Yalta of the computing world', 'Torpedoing of the standardisation efforts'. Ohers prefer not to engage in open debate, believing that it can do nothing but damage the reputation of UNIX.

My personal opinion (ie, that of Philippe Dax, not the AFUU) is that this event simply shows that UNIX has really become the real 'game' in the computing world, the most significant OS for the '90's and into the 21st century.

Enfin!. les ''Grands'' ont choisi UNIX comme la solution la plus appropriée par rapport á la pression du monde des utilisateurs. Les utilisateurs, c'est eux, qui, á un certain degré, ont engendré inconsciemment cet état de fait. C'est eux, qui, au cours des années 80 ont manifesté leur ''ras le bol'' vis-á-vis des constructeurs qui se croyaient toujours en terrain conquis ou en situation de monopole. C'est eux, qui, confrontés á une informatique de plus en plus hétérogène, ont poussé á la standardisation. C'est eux qui ont découvert UNIX comme le seul système permettant la portabilité des applications sur tout un éventail quasi-mondial de machines. C'est eux qui dictent d'une certaine manière quelle direction doivent prendre les constructeurs. Ce sont, enfin, les constructeurs qui l'ont compris et tentent maintenant d'anticiper le mouvement des utilisateurs.

Si UNIX, comme cela a été annoncé, atteindra près de 25"%" du marché mondial vers les années 92-94, il est tout á fait normal que le gâteau soit convoité, tout comme celui du monde PC l'a été avec les compatibles asiatiques.

Ceux qui croyaient en UNIX sont probablement effrayés ou agassés par ce tapage, mais qu'ils se réjouissent car leur cause a été entendue. Ceux qui n'y croyaient pas, ou ne voulaient pas y croire, des noms!!!, sont maintenant contraints d'y croire.

Maintenant..., que sera l'UNIX des années 90? Sera-t-il OSF? ATT-SUN? Sera-t-il ouvert? Sera-t-il unifié? Lá est tout le débat, mais une chose est sûre, il sera POSIX. De toute manière, les analyses, les commentaires, les prises de position ne manqueront pas dans les mois qui viennent, et á ce propos, lors des ''Journées UNIX de Grenoble'' jumellées avec la ''Convention AFUU 88'' les 17 et 18 novembre prochains, vous en saurez plus après les exposés qui traiteront sur ce sujet (voir dans ce numéro ''Prochaines manifestations'').

At last!. The 'giants' have chosen UNIX as the most appropriate solution, following the pressure generated by users. 'THE USERS', are the people who have, only half conciously maybe, shown during the course of the 1980's that they have had enough of manufacturers wanting to do except conquor teritory, and hold it in a monopolistic fashion. It is they, who, confronted by a computing industry which becomes more and more hetrogenous, have pushed for standardisation. It is they who have discovered UNIX as the only system giving portability of their applications onto a world-wide set of machines. It is they who dictate (in a certain fashion) which direction the manufacturers must take. These (the members of OSF) are the manufacturers who have finaly understood, and are trying to anticipate the wishes of 'the users'.

If UNIX, as has been predicted, takes 25% of the world computing market in 1992-94, it is completly normal that the 'cake' will have to be shared, as was the case with the IBM PC, and the Asiatic compatibles.

Those who believe in UNIX are probably worried by this 'incursion', but they should rejoice because their cause has been enlarged. Those who don't believe this, or who didn't want to believe in this cause will now be forced to believe!

Now ... where will UNIX be in the '90s? Will it be OSF?, AT&T/SUN? Will it be 'open'? will it be unified? There is the debate, but one thing is certain, it will be POSIX. In any event, the analysis, the comments, the taking of sides will not be lacking in the comming months. Apropos, during the UNIX days, at Grenoble twined with the "Convention AFUU 88" on the 17th and 18th of November, you can learn more about this subject, as several exhibitors propose to examine this subject.

Rappelons ici, que le rôle de l'AFUU est de promouvoir UNIX sous toutes ses formes et ses tendances, ce qu'elle a déjá réalisée dans son passé á propos des UNIX : Version 7, System 3, Xenix, BSD 4.? et System V; des Stantards : /usr/group, SVID, X/Open, IEEE, POSIX. L'AFUU n'a jamais pris parti ni pour l'un ni pour l'autre, et entre la bataille OSF - ATT/SUN, son attitude sera d'informer les adhérents, de susciter des échanges d'opinions et d'organiser des débats. Ces colonnes vous sont ouvertes...

Remember, the role of the AFUU is to promote UNIX, in ALL its forms and tendancies. This it had done in the past with the different versions of UNIX : V7, S3, Xenix, BSD4.?, SV; and the standards: /usr/group, SVID, X/OPEN, IEEE/POSIX. The AFUU has never taken sides, and in the battle OSF - AT&T/SUN its attitude will be simply to inform all its members of what is happening, to solicit exchanges of opinion, to organise debates etc - it is YOU, the members who must take the final decisions. The Newsletter columns are always open to you...

POSIX Standardisation Developments

Cornelia Boldyreff
corn@cs.brunel.ac.uk

UK POSIX Panel Convenor
Department of Computer Science
Brunel University
Uxbridge UB8 3PH
England

Cornelia Boldyreff is a member of the British Standards Institution technical committee on Application Systems, Environments and Programming Languages. She acts as Convenor and Chairman of the BSI C Language Panel; and is one of the UK Principal Experts on the ISO Working Group on C. She is also Convenor and Chairman of the BSI POSIX Panel; and is one of the UK Principal Experts on the ISO Working Group on POSIX.

Recent Developments

The recent ballot on the IEEE POSIX 1003.1 which closed in July was successful with over 85% in favour. The plan is to bring out a final revision, draft 12.4, which will be published at the beginning of September; and seek IEEE Standards Board approval of this draft when the board next meets in October 1988. It was agreed at the March meeting of the ISO Working Group on POSIX that this revised document adopted by the IEEE would be registered as the Draft International Standard. So by the end of 1988, we can expect to see an approved IEEE Standard for POSIX P1003.1; and a registered DIS. Approval as an ANSI standard and a full ISO standard should follow in the coming year. The ISO DIS will be subject to six month ballot; at present, there are no outstanding unresolved issues likely to hold up progression to an ISO standard.

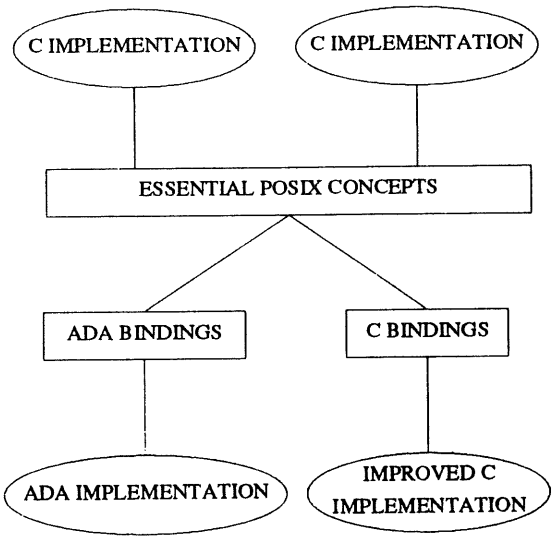
An issue which has been much discussed in the past few months has been the options allowed by the present draft. Early in the year, the US National Bureau of Standards brought out an interim FIPS which in effect removed options. This move and concern from the ISO community regards the undesirability of options has resulted in the elimination of most options from the standard; those remaining are as follows: the multiple groups option, the job control option and the administrative/security option which restricts *chown*. A more precise term in place of options to describe this aspect of the standard is alternatives; thus, for example, if the symbol, _POSIX_JOB_CONTROL is defined, it indicates that job control is supported and otherwise, that it is not.

Future Work

At the ISO WG15 meeting, it was agreed to initiate a division of the work item recognising that P1003.1 is simply the platform for a family of related standards. The next version of POSIX 1003.1 is committed to being a language independent definition of the interfaces currently defined in terms of the C language. Work within the IEEE developing an Ada binding for POSIX

has already begun to address this issue: and there is much interest within the ISO group in progressing this work.

The process of abstracting the essential POSIX concepts from their various implementations which underlies this work on language bindings is bound to benefit the users of the UNIX family of operating systems by resulting in a better conceptual understanding and subsequently better implementations.



Recent and Future Meetings:

27 Jun -1 Jul	P1003	Denver, CO
7 Sept	POSIX Panel	BSI, London
20-21 Oct	WG15	Tokyo
24-28 Oct	P1003	Hawaii
9 Nov	POSIX Panel	BSI, London
9-13 Jan 89	P1003	Fort Lauderdale, FL
Apr 89	WG15	Ottowa, Canada
17-21 Apr 89	P1003	St. Paul, MN
Jul 89	P1003	Monterey, CA
Oct 89	WG15	Brussels
Oct 89	P1003	Brussels
Jan 90	P1003	New Orleans, LA
Apr 90	WG15	Paris
Oct 90	WG15	USA

British Standards Institute - Ada Evaluation Reports

These are now becoming available. BSI tell the EUUGN that the reports point out the strengths and weaknesses of compilers enabling selection of the compiler best suited to a project, and allow more effective use of the chosen compiler.

You can either buy one copy at £250 (+ VAT), or by using their subscription service you can have all the reports produced in one year for £2000.

For more details please contact:

Terry Calvert
BSI Marketing
PO Box 375
Milton Keynes
UK MK14 6LL
+44 908 220908 Extn 2094

Draft Proposed ANSI/ISO C Standard: Developments

Cornelia Boldyreff
corn@cs.brunel.ac.uk

UK C Panel Convenor

Recent Meetings

The ISO Working Group Meeting

ISO Progress
The London meeting of WG14 held from the 13-14 June 1988 was attended by representatives from Denmark, the Netherlands, the UK and the USA. At this meeting, Bill Plauger, the ISO Convenor, tabled the X3J11 Response Document to comments from the second public review: this document included a large section devoted to the comments submitted by the ISO community.

Bill Plauger agreed to ensure ISO comments would be processed at the September meeting of X3J11. At present, the UK has no outstanding unresolved comments which would require other than editorial changes to the draft. However, WG14 was unable to resolve the Danish NO vote in the recent ISO DP ballot. Resolution of this requires a further substantive change to the ANSI draft and would require a further public review before ANSI approval could be sought. The crux of the Danish objection is that the trigraphs for { } and [] are unusable; and that an alternative is required.

It was agreed by WG14 not to submit any further draft for ISO ballots until a stable X3J11 document was available. If following the third review, a new draft is voted in by X3J11 at their September meeting and this draft is submitted to ANSI for approval, then WG14 agreed to either submit this draft to ISO for registration as a DIS if it is acceptable to Denmark and the UK, or to resubmit it for a further DP ballot.

X3J11 Progress

At their April meeting, X3J11 processed comments from the second public review; and made a number of substantive changes to the draft standard. The most controversial of these was dropping *noalias*. As a result, a third public review of three months is required. This review is currently in progress. If you wish to comment on the May 13, 1988 draft under review, you should obtain a copy via your ISO representative or from Global Engineering at the address below:

Global Engineering Documents, Inc
by calling +1 800 854 7179 or +1 714 540 9870.
Expected Single Copy Price US$65.00 (draft standard and rationale).

Global Engineering is located in Santa Ana, California, USA, which is in the Pacific Standard Time (PST) zone.

Future Meetings and Projected Targets

Any comments on the X3J11 May 13, 1988 draft will be processed at X3J11's September meeting. If following the third public review, no substantive changes have been made to the draft by X3J11 in September, it will go forward for administrative processing by ANSI; and emerge as an ANSI C Standard in the latter part of 1988 or earlier next year. If the ANSI standard is acceptable, WG14 will put it forward for registration as a DIS; otherwise there will be another ISO ballot on the draft as a DP. There is strong support amongst the ISO community for a common standard which does not deviate from the ANSI standard.

Future Meetings:

9th Aug 1988	BSI IST/5/14	London
26-30 Sept 1988	ANSI X3J11	Cupertino, CA
8th Nov 1988	BSI IST/5/14	London
12-16 Dec 1988	ANSI X3J11	Seattle, WA
10-12 Apr 1989	ANSI X3J11	Phoenix, AZ
Dec or Mar	WG15	with X3J11

OPEN LOOK Graphical User Interface

Janet Davis
janet@uel.co.uk

Market Communications
Unix Europe Limited (UEL)
International House
Ealing Broadway
London W5 5DB
+44 1 567 7711

Janet Davis is Market Communications Manager for AT&T Unix Europe.

She has been with AT&T since September 1987 and is responsible for the promotion of UNIX System V and related products and services throughout Europe.

Janet was previously marketing executive with Thorn EMI Computer Software.

The OPEN LOOK graphical user interface, announced in April 1988, employs common sense graphic symbols instead of written commands to help users work more efficiently with UNIX® System V based computers.

OPEN LOOK supports AT&T's commitment to open systems and the need for a standard user interface. It is the next step in expanding the UNIX Systems marketplace. By providing a standard interface across various hardware environments, applications developed with the OPEN LOOK interface will have access to a larger market. The interface has already generated endorsements from key industry players including Lotus, Ashton-Tate and Xerox.

The UNIX operating system is nearly 20 years old. Many parts of the system have evolved over the last two decades, but the user interface of the standard system has not. With the introduction of the OPEN LOOK User Interface, that situation is changing.

The UNIX system user interface evolved in the late 1960s and early 1970s with the emergence of time-sharing as the leading new computing style. Users were primarily scientists and engineers who used terminals to share minicomputers.

This interface was revolutionary in its time for its simplicity and power. A "shell", or command interpreter read command lines from the keyboard and executed a separate process for each command. The system also provided powerful programs to accommodate a variety of terminals for common editing operations so that applications could be made much simpler and still support a consistent user interface.

The Need for a New UNIX System User Interface

In the late 1980's, the UNIX system is being used by customers who have wide ranging needs.

® UNIX is a registered trademark of AT&T in the U.S.A. and other countries

Technology is changing. Even inexpensive personal computers now have displays capable of high quality graphic presentations and mouse pointing devices that increase usability. New computer input and output technologies such as voice recognition and on-screen video are beginning to appear in desktop computers.

Therefore it is time for a new user interface for the UNIX system; one that is accessible, consistent, and meets the needs of its new users. A new UNIX system interface should be easy to use and offer a comprehensive flexible way for programmers to take advantage of it. It must also preserve the investments of the past and prepare us for the future. The OPEN LOOK User Interface is such an interface.

Why call it The OPEN LOOK User interface?

A major trend in computer systems today is the trend to open systems. Open systems are those which are available from multiple vendors, so procurements can be competitive and so common platforms can be developed that preserve customer investments in applications software, documentation and training.

As part of its commitment to open systems AT&T is participating with the industry in defining standards for the UNIX system and its derivatives. The POSIX standard being developed in the United States, the X/Open standard being developed in Europe, and the efforts of the Sigma group in Japan aim to expand a common definition of a portable and open operating system. This common definition can then be used by any number of vendors to produce different implementations of the same operating system.

To support open systems, AT&T will make the OPEN LOOK User Interface available for use in all standard systems, including appropriate licences to all relevant copyrights, patents, and trademarks that pertain to the OPEN-LOOK User Interface and the standard Application Programmer's Interface toolkits.

How did AT&T decide on OPEN-LOOK User Interface?

AT&T, with industry partners such as Sun Microsystems investigated the requirements for the right UNIX system user interface with care, researching works inside and outside the UNIX system environment. This research covered interface designs from the 1970s through the 1980s, weighing the features, motivations, and resulting successes.

Today's most recent user interface developments were scrutinised. AT&T's research experience was combined with reviews from leading industry consultants. AT&T's work on graphical user interfaces dates back to 1965 and a project known as GRAPHIC I - a remote graphical display console. Subsequent work on GRAPHIC II resulted in a key graphics patent, as did work on a bit-mapped graphic workstation and the overlaying of graphic windows.

None of the system user interfaces in the marketplace today was judged suitable for the UNIX system of the 1990s. Each had its good and bad points, but none combined the necessary simplicity, power and openness. As a result, the OPEN LOOK technology was designed for AT&T by Sun Microsystems, Inc. of Mountain View, California. Sun's design is based on original work, contributions from AT&T, and on technology licenced from the Xerox Corporation, which originated many of the concepts present in today's computer interfaces.

What were the Design Goals?

Simplicity:

A user doing a new task on the computer, with no model from prior experience, needs the interface to be simple. This makes the interface easy to learn and to master.

Consistency:

A user doing a new task but having a model from prior experience, either on a computer or from real world experience wants the interface to be consistent. This allows the user to learn the new system very quickly.

Efficiency:

Once familiar with a task, a user will perform it with the least effort - the fewest keystrokes, the shortest movements of the cursor.

What is the environment for OPEN LOOK User Interface?

Because the UNIX system is highly flexible, it has always been used for a wide range of purposes. The designers of OPEN LOOK wanted it to be as adaptable as UNIX systems. They wanted it to work for:

- standard as well as large screens
- a variety of screen resolutions
- a true multitasking environment
- interface technology that is advancing rapidly

Each aspect of the OPEN LOOK User Interface design was examined to see how well it worked. Some common interface designs were abandoned as too unwieldy for one use or the other. New designs were invented to meet needs that are just emerging.

What are the Design Principles of OPEN LOOK User Interface?

Good designs are the result of just a few, carefully considered principles, and the consistency of the whole is more important than the brilliance of a single part. These are the design principles that guided the OPEN LOOK User Interface design:

Good Graphics and Visual Metaphors:

Careful graphics designs help make the interface intuitive, easy to learn, efficient to use. Visual metaphors were used to make operations clear. Ideas that could not be well expressed graphically were discarded.

Functional Balance:

The goals of simplicity, consistency and efficiency sometimes conflict. All parts of the design must balance the three and work for new users, for experienced users doing new tasks, and for experts.

Less is More:

One concept is better than two. One way to do an operation is better than two. There is less to learn, less to remember. These design principles lead to a more detailed list of design rules that are "laws of physics" of the OPEN LOOK User Interface system.

Select then Operate:

Operations follow the familiar noun-verb model: select the object, then pick the action.

Objects have Properties:

OPEN LOOK User Interface objects can have properties. These can be examined and set in a consistent way.

Help for Everything:

Pushing the Help key always gives a message.

Cut/Copy/Paste:

Cut, copy, paste are used to move information.

Stay Up is Universal:

A general mechanism is provided to allow any part of the user interface to be "pinned" to the desk for repeated use.

Visual Controls:

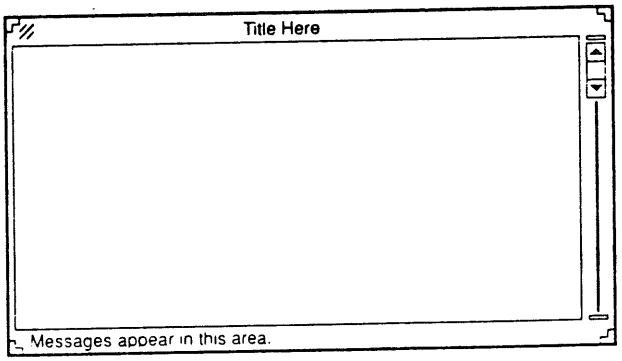
All controls are accessible on a visible control panel.

The OPEN LOOK System

The elements of the OPEN LOOK User Interface:

The Mouse

Basic movement of the cursor is controlled by a one, two or three button mouse. On a three-button mouse, the left button is used to select the object to be manipulated, the middle button to extend or adjust the selection (i.e. select more text) and the right button produces a menu - either for an object or a region of the screen. Selecting, adjusting and producing menus can be performed if the mouse has only one or two buttons.



OPEN LOOK Window with Scroll Bar

Desktop

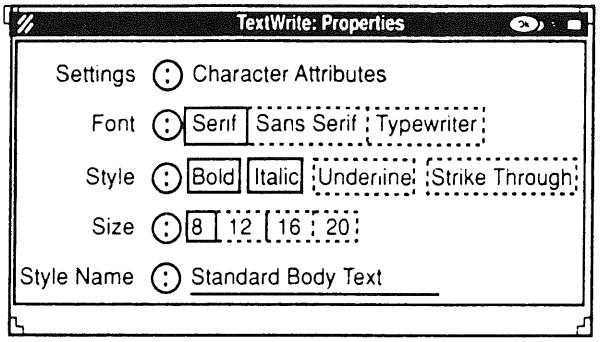
The desktop, or basic blank screen, uses windows that are capable of either overlapping or remaining separate. To move them out of the way, a window may be "closed" or reduced to a small icon or symbol in the corner of the desktop. Every object on the desktop is moved in the same fashion; using a mouse to move the cursor to the object to be selected then "dragging" the object across the screen by holding down the mouse button and moving the mouse. In the same way,

windows may be "stretched" by selecting and dragging the window frame corners using the mouse. These concepts are applied the same way in the OPEN LOOK user interface so the user only has to learn this basic concept once.

Windows, Command Boxes, Property Sheets

Windows, command boxes and property sheets are variations that allow the user to see and control information that pertains to the function being performed. All three are controlled in a consistent way. For example, the box in the upper left hand corner closes the object, whether it's a window, command box or property sheet.

- windows are the centre stage, where text is edited or the engineering design displayed.
- Command boxes pop up to allow the user to fill in more information after he or she has initiated a command. For instance, a user who's just saved a document will see a command box appear so the file name may be entered.
- Property sheets are used to set the properties of objects - such as the tab setting in a document or the font for a paragraph.



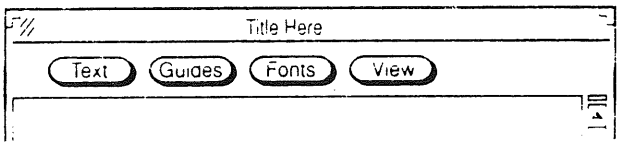
OPEN LOOK Property Sheet

Push Pin

The push pin is used to "pin" to the screen such things as property boxes that may be needed repeatedly and would otherwise disappear each time they were used. For instance, a complicated piece of text may require many variations in fonts - such as heavy use of italics. Making these property changes would be a long and cumbersome process if the property box disappeared after each change. However, pinning the property box to the screen allows the user to refer to it again and again thus saving keystrokes and time.

Control Panels

The controls for an application are on "control panels". These control panels are like the dashboard of a car. They contain the buttons to push to make things happen, like printing or editing a document. The buttons are "pushed" by moving the cursor to be button and clicking the select key on the mouse.

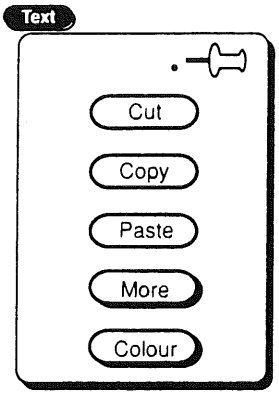


OPEN LOOK Control Panel at top of Window

An application can have more than one control panel. For example, when a spreadsheet and a graph are on the screen at the same time, it makes sense to put a control for the graph next to the graph and a separate control panel next to the spreadsheet.

Menus

The OPEN LOOK Interface allows users to access menus in a couple of ways. Some buttons on control panels are shaded so they appear to be the top of a stack of buttons. Manoeuvreing the mouse to these control panel buttons and clicking the mouse's menu button displays the stack as a menu.



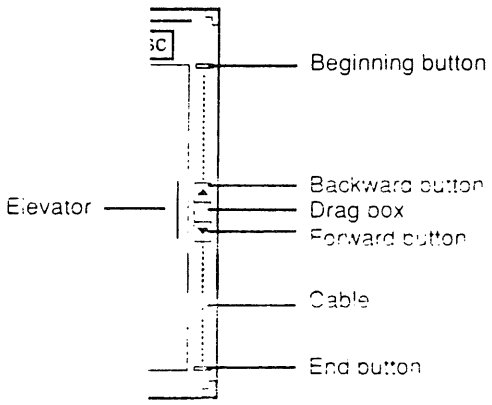
OPEN LOOK Menu

In addition, menus can also be made to pop up when they're needed. For instance, a user who wants to insert, delete or move text first selects the text to be edited, then clicks the mouse button to access a menu from which choices can be made to perform the editing function.

For the user's convenience, no matter where he or she is on the screen or what application is being performed, a menu can be made to pop up to give quick access to important controls. Unlike some user interfaces where a user must manipulate and search to find the appropriate control, the control comes to the cursor, rather than the cursor searching out the control.

Scroll Bar

Scrolling through text occurs through the use of an "elevator" symbol and gives the user the ability to move up and down in the text quickly and easily. The bold portion of the elevator cable provides a visual clue as to the size of the displayed portion in relation to the entire text.



OPEN LOOK Scroll Bar

Pushing the up arrow on top of the elevator car moves it up one line or down one line if the down arrow is pushed. The control buttons are placed next to the car so the cursor need not be walked to the top or bottom of the window to push the buttons. In addition, the cursor moves up or down with the elevator's control button so the user can click the cursor repeatedly without having to reposition the cursor with every click, thus saving keystrokes and movement.

Multitasking

Like the multitasking abilities of the UNIX system, windows on an OPEN LOOK interface can perform functions simultaneously. Users can start an operation in one window while working in another. The title bars of working windows turn grey so the user knows at a glance which windows are busy and which have completed their work. To avoid confusion, each window has its own control panel. Like any other object, windows - either individually or several at a time - can be moved, closed, enlarged or manipulated.

Toolkits for OPEN LOOK

Toolkits contain pre-programmed components or "tools" that applications programmers use to build a user interface for a particular application. For instance, the "scrolling" function is a tool. A programmer who wants to adapt this function to a particular application doesn't have to start from scratch to develop it because it's already part of the toolkit. The OPEN LOOK Toolkits will establish a consistent "Open Look" style among different applications. Providing the tools will encourage developers to support this "look" and they'll be less likely to want to recreate the interface for a new application.

Because the toolkit is "object-oriented" - which means it provides the objects such as scroll bars, menus and command boxes - it can easily adapt to new objects or applications. Its flexibility allows applications programmers to customise elements of another interface while remaining within the OPEN LOOK interface environment. This flexibility will allow the toolkits to conform to the varied requirements of UNIX system applications developers now and in the future.

Initially AT&T will be providing two toolkits:

— OPEN LOOK XT toolkit based on X-Windows systems

— OPEN LOOK NDE toolkit based on X11/NeWS

AT&T will circulate OPEN LOOK specifications for comment this summer and will make them available in the third quarter of this year.

Please contact the author if you would like a copy of this document.

These will include specification of the common style for applications, the Application Style Guide, as well as descriptions of the programming interface for OPEN LOOK under two toolkits, both of which AT&T will support via a single graphics platform.

The first availability of OPEN LOOK features in an AT&T product will be this summer in a window manager for the 6386 workstation, followed by an XT toolkit in the fourth quarter 1988 and an NDE toolkit in the first quarter 1989. Source code for the OPEN LOOK user interface toolkits are scheduled to be available in early 1989.

USENIX Association News for EUUG Members

Donnalyn Frey
donnalyn@uunet.UU.NET

Frey Communications

Ms. Frey is the USENIX Association Press Liaison. She provides members of the press, USENIX Association members, and EUUG members with information on the activities of the USENIX Association.

Summer 1988 Conference

The San Francisco Summer 1988 Technical Conference and Exhibition was informative and enjoyable. Over 3,200 people attended the conference, the largest in the Association's history. All of the tutorial sessions sold out, some well before the conference. The technical presentations presented new and interesting information and were well received by attendees. The BOF sessions were, as a whole, well attended. One session, the Open Software Foundation BOF, attracted a large crowd. The speakers at the BOF, technical managers at OSF, answered questions and spoke of their plans for the future.

The conference reception was held Thursday night at the Exploratorium, a hands-on science museum. Attendees dined among physics, chemistry, and biology experiments that invited participation.

The Proceedings of the San Francisco conference sold out on the last day of the conference. A second printing will make additional proceedings available soon. To request copies of the San Francisco proceedings, please send mail to {uunet,ucbvax}!usenix!office.

USENIX Signal CONTEST

The gauntlet was thrown and the challenge was accepted. The winners of the USENIX signal contest was not an European this year, but an American. The judging committee consisted of both Americans and Europeans to avoid any bias toward Americans. Thousands of entries were received and a truck was used to move all the entries to the hotel bar for the judging. As the evening wore on, and the judges had too much fun, the winners were culled from the losers.

The next day the conference attendees anxiously awaited the announcement of the contest winners. First prize was copy of "The C Programming Language", autographed by Dennis Ritchie and donated by Prentice-Hall. Second prize was an EUUG penknife, donated by the EUUG.

The presentation was made under an EUUG banner. The contest announcement and awards were presented by Peter Colinson, British, with the assistance of Jean Wood, American, and Andrew Hume, the Bell Labs Murray Hill resident Australian announcing the first prize winner.

An attempt by EUUG to commandeer the awards ceremony was put down, but the EUUG standard still flew valiantly over the smoldering hulk of the podium. After order was restored, it was reported

by the usually reliable suspects that Peter Colinson grudgingly acknowledged defeat by the brilliant American entries. When confronted with the allegation, Colinson refused to deny it.

The winning entry in the USENIX signal CONTEST was:

SIGSEUSS

I meant what I said
And I said what I meant
Your program is buggered
One hundred Percent

The FaceSaver Project

The FaceSaver Project was again successful at the San Francisco conference. The FaceSaver combines photographs and registration information onto sticky labels that conference attendees could give to exhibitors and other attendees. After the conference, the attendee list will feature a postage stamp size portrait beside each name and address.

For the San Francisco conference, attendees who previously had their faces saved found three pages of labels already in their registration packets. Both photographs and registration information could later be updated in the FaceSaver room.

The FaceSaver portraits are captured via a video camera using AT&T Targa M8 graphics boards installed in Bell Technologies PC AT clones running the SCO XENIX version of the UNIX operating system. Portraits are printed using a Postscript laser printer.

The FaceSaver project is run by Lou Katz. It is sponsored by the USENIX Association to aid in improving attendee recognition at the conference. A sample FaceSaver label is reproduced below.


FaceSaver 2/88

Donnalyn Frey
703-764-9789
uunet!donnalyn
Frey Communications
PO Box 2051
Fairfax, VA 22031


FaceSaver 2/88

Donnalyn Frey
703-764-9789
uunet!donnalyn
Frey Communications
PO Box 2051
Fairfax, VA 22031

Winter 1989 San Diego Conference

The next USENIX Association technical conference is scheduled for January 30 - February 3 in San Diego, California. Tutorials will be given on Monday and Tuesday, January 30 - 31. The technical sessions will be presented Wednesday, Thursday, and Friday, February 1 - 3. The call for papers for the San Diego conference was sent to USENIX Association members in July. The deadline for papers submitted for the conference is October 7, 1988. Papers are requested for formal review as candidates for inclusion in the conference. Papers that are accepted will be presented at the conference and published in the conference proceedings. Suggested topics for papers includes (but are not limited to): Performance Analysis and Tuning New User Interfaces and Applications System and Network Security Networking and Distributed Services RISC versus CISC in UNIX Software and System Management Tools Standards Graphics and Electronic Publishing Evolution of UNIX for the 1990s

All papers should describe new and interesting work. For full information on submitting a paper for the San Diego conference, contact the USENIX conference office at P.O. Box 385, Sunset Beach, CA 90742, USA and request a copy of the call for papers for the conference. Technical questions should be referred to Greg Hidley or Keith Muller at sd@usenix.ucsd.edu. The deadline for submissions is October 7, 1988.

C++ Mini-Conference

The USENIX Association's first C++ Mini-Conference will be held October 17 - 21, 1988 in Boulder, Colorado. The mini-conference was begun to accommodate all the people who wanted to attend the smaller 1987 C++ workshop. The conference includes two days of tutorials and three days of technical sessions. The conference is intended to provide information for both new and experienced C++ programmers. For information on the conference, contact the USENIX conference office at P.O. Box 385, Sunset Beach, CA 90742, USA.

Conferences and Workshops

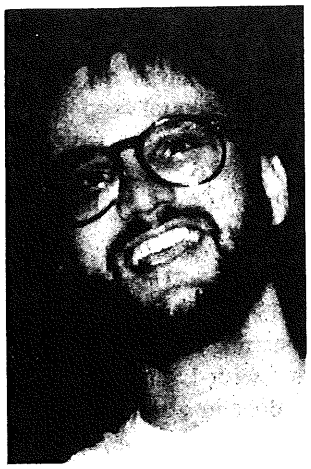
If you need further information on upcoming annual USENIX Association conferences or workshops, contact the USENIX conference office at P.O. Box 385, Sunset Beach, CA 90742

USA. The conference office can provide you with information on the annual Computer Graphics, Large Installation Systems Administration, UNIX Security, and UNIX and Supercomputers workshops. The office can also provide information on the C++ mini-conference and the semi-annual technical conferences.

EUnet News

Daniel Karrenberg
dfk@cwi.nl

Centrum voor Wiskunde en Informatica



Daniel received a graduate degree in Computer Science from the University of Dortmund, West Germany, in 1987. During his studies he was exposed to the Unix Operating System and worked as a systems manager & consultant. He is a co founder of the GUUG & was heavily involved in the starting stages of the German part of EUnet. He currently works at the CWI (Centrum voor Wiskunde en Informatica) in Amsterdam, the Netherlands, where his responsibilities include local & wide area networking as well as the operation of the EUnet international backbone node called "mcvax". He also is a member of the the EUUG Executive Committee with responsibility for EUnet. He enjoys being involved in both managerial & operational aspects of EUnet.

Abstract

Short news items from Europe's largest network for the research and development community.

This is intended for those interested in news about recent developments in EUnet. If you would like to know more, come to the next EUUG conference in Portugal where I will give a more detailed report (which might be printed in this forum if the editor bears with me).

News Flash

The load average of mcvax (a poor little vax 11/750) just reached 25.4 again. The average average :-) is about 18.

Infrastructure

Our international workhorse mcvax is really bogged down at the moment, but relief has arrived. The Dutch group (NLUUG) have received a HP 800 series machine as a kind donation from Hewlett Packard. This machine was christened hp4nl. It has been placed next to mcvax and the news feed for the Dutch sites is being transferred as I write this. hp4nl is equipped with the new "Trailblazer" modems which support uucp and achieve transfer speeds around 14000bit/s (no spelling error, I counted the 0s twice!).

This really speeds up local news feeds!

While hp4nl already provides some relief to mcvax, we are looking to replace mcvax as well. The EUUG has received several generous offers. Testing of some machines is currently under way and I hope to bring good news to Portugal.

The 9600bit/s link to the news and mail service of the Usenix Association uunet has been running smoothly and is carrying almost all of EUnet's mail to and from the United States. Introduction of the link helped stabilise costs for mail and news exchange with the US while the volume is increasing steadily. The line is still fully paid for by EUnet, but it is expected that Usenix will start to cover a share of the costs this year.

Due to the work of Simon Kenyon we have also received reliable signs from the U.S. Internet administration that EUnet will be granted full IP access to the Internet in the next few weeks. This will make it possible for backbones to deliver mail even more quickly and to provide such services as fetching information available only via "anonymous ftp". The details of this still have to

be worked out. These developments and the expected completion of a new trans-atlantic fibre optics cable have prompted us to order an upgrade of the U.S. link to 64kbit/s. This will be only marginally more expensive than the current 9600bit/s link. This is a sign that the European PTTs are becoming more competitive. We actually received a very glossy folder from the Dutch PTT offering their leased line services including bargain discount schemes. We were stunned!

Inside Europe the Eunet infrastructure continues to grow and as traffic grows it becomes more and more attractive to replace links over public X.25 networks with leased lines. A line between CWI and INRIA, the French backbone site has been in operation for some time running IP over X.25. A line to the University of Kent (England) is due to be delivered in August. A 64kbit/s circuit between the campus where CWI is housed and CERN in Geneva was ordered last week. It will be jointly financed and shared with the high energy physics community's HEPNET. This is the first example of such cooperation between two European networking organisations and hopefully it's the start of a trend. There are rumours from unido that a link between CWI and Germany is going to be ordered real soon now. Another candidate for a fast connection to EUnet is the Scandinavian community with its new powerful infrastructure called Nordunet.

While EUnet extends it's infrastructure other European networking organisations do likewise. Talks have been going on over the last year to explore the possibilities of sharing some of the infrastructures in order to reduce costs and enhance services. Piet Beertema and Bjorn Eriksen are participating in a joint working party with representatives from other major networks to create a shared pan-European X.25 infrastructure for research and development. Piet has written a report that shows that with current funding the bandwidth on the most crucial links can be increased by at least an order of magnitude, if – the infrastructure is shared by all participants. Some money has been promised by the European community and the government of Norway to start this activity off this year, but a lot of formal problems will have to be solved to create a stable shared infrastructure. EUnet will continue it's own plans while participating in the sharing exercise until the situation is more stable.

News Flash

The load average on mcvax has dropped below 8. Something must be wrong, I'll have to check!
– X.25 connection fixed again. I'll have to hurry now, deadline is tomorrow!

New EUnet Services

There have been many calls for news archiving services for groups such as comp.sources.* and that need is recognised by the backbones. Some of them already operate archive servers and some plan to do this in the near future. Once the new Dutch and European backbone machines are in operation we will start with some central support and coordination activities in this area. The same goes for general information servers holding documents, site information and the like. Suggestions from users are welcome.

A study has been completed on how EUnet could be migrated to the use of ISO protocols. The conclusion is that it's worth starting now, it can be done, that resources are needed and most importantly that it should be done in a user driven way rather than by coercion. I'll give a detailed account of this in Portugal.

A printed electronic mail directory is currently under development. It will hopefully contain information about as many European organisations connected to R&D networks as possible. The directory will be useful as an "off line" reference tool usable especially by people not yet on the net. Backbone sites often get requests for comprehensive information about reachable organisations from people who want to connect to EUnet. This directory is hoped to fill that gap. Of course care is being taken that it can't be used as a junk mail list. We still aim to have them available in time for Portugal.

Well, that's about it. The load on mcvax is back above 20, my pot of coffee is empty. See you in Portugal.

UNIX Clinic

Colston Sanger
olibc1!colston@olgb1.oliv.co.uk

Olivetti International Education Centre

Colston Sanger is a lecturer at the Olivetti International Education Centre, Haslemere, UK and a visiting lecturer in the Faculty of Engineering, Science and Mathematics at Middlesex Polytechnic, London. As soon as he finishes this article he is going off to get married...

Efficiency considerations and shell functions in the SVR2 shell

This month's topic is efficiency considerations in the UNIX SVR2 shell. At the end, I'll also talk about shell functions: how you can use them to add new commands to your login shell, and also as subroutines within shell-scripts.

Does it need to be more efficient?

Way back before the beginning of time (1 January 1970 in the UNIX world) when computers were big and slow and expensive, programmers put a lot of effort into making their programs more 'efficient'. Today, computers are cheap and programmers are expensive. Efficiency, while still a consideration, now primarily means saving programmer, not machine, time.

The first question, then, is: does it it need to be more efficient? There is little point in spending weeks making a script more efficient if it is only going to be run once a year.

The second question is: have you timed it with `time` or `timex`? Either will give you the elapsed, user and system times taken in the execution of a command, but `time` reports in tenths of a second whereas the newer `timex` reports in hundredths (and also has several useful options). As an example, here is `timex` run on a shell script `xwho` (extended `who`):

```
$ timex xwho
sue       tty62      Jul 16 08:49
yvonne    tty34      Jul 16 08:56
tony      tty14      Jul 16 09:00
peter     tty72      Jul 16 13:27
sandy     tty61      Jul 16 13:27
colston   tty11      Jul 16 13:44

real         1.22
user         0.34
sys          0.50
```

'Real' is the elapsed wall-clock time; 'user' is time spent actually executing the user program; and 'sys' is time spent inside the kernel executing system calls on behalf of the user program.

PATH organisation

Given that your script really does need to be more efficient, what can you do? First of all, you could look at `PATH` organisation. The default value of `PATH`, the environment variable that defines the search-path for commands, is:

```
:/bin:/usr/bin
```

meaning the current directory (the null string before the first colon), /bin and /usr/bin, in that order. For most users this setting is inefficient. Most of the time they are executing programs in /bin or /usr/bin. It makes sense, therefore, to change the default value of PATH (the best place is in /etc/profile) to:

```
/bin:/usr/bin:.
```

so that the current directory is searched last. Note that current directory is also defined here explicitly as . 'dot'.

Another practice that may result in minimal searching is to reset PATH within your shell script. You may also want to use the absolute or relative pathnames of commands. (If a command name contains a /, then the search-path defined by PATH is not used.) Here is a shell script, paths, illustrating some of these techniques. If you are a system manager, you may want to use it to check the setting of PATH in your users' .profile files:

```
# paths - checks settings
#         of PATH in
#         users' .profiles

PATH=/bin:/usr/bin

trap ' ' 1 2 3
awk -F':' '$3 > 99 { \
 printf("%s %s\n",$1,$6)}' 3
 /etc/passwd \
| while read LOGIN LOGINDIR
do
    echo "$LOGIN\t\c"
    if [ -f $LOGINDIR/.profile ]
    then
        ( cd $LOGINDIR
        if grep 'PATH=' .profile
        then
           :     # do nothing
        else
           echo '$PATH not set'
        fi
        )
    else
        echo 'No .profile'
    fi
done
```

Referencing files

Interpreting a long pathname makes a significant demand on system resources, especially if individual directories are large. If you need to read (or write) many files in a directory, it's far better to cd to that directory first. For example, the following two scripts do the same job, but the second runs faster and demands fewer system resources:

```
for file in /usr/you/src/talk/*
do
    pr $file | lp -dpr1
done
```

```
cd /usr/you/src/talk
for file in *
do
    pr $file
done | lp -dpr1
```

Note that in the second script, the pipe to lp is also outside the loop.

Pipelines

In pipelines, it's common sense to put reducing filters before non-reducing filters. Aim to reduce the amount of data being processed at each stage. For example, these two pipelines are equivalent, but the second is more efficient:

```
$ ps -ef | sort -t' ' +1n | \
  grep tony

$ ps -ef | grep tony | \
  sort -t' ' +1n
```

Use shell built-ins wherever possible

Use shell built-in commands wherever possible. They are faster, and there is no process creation overhead. Also, in the UNIX System V Release 2 shell, more commands - such as echo and test - have become built-ins.

Similarly, if you need to call a shell script from within a shell script, if possible 'source' it with the . command.

Shell functions

Shell functions are new in UNIX System V Release 2. Unfortunately, apart from a brief paragraph under sh in the manual, there's very little in the way of documentation on either syntax or how to use them.

So, how are shell functions used? So far as I can see, there are two ways of using shell functions: first, to add commands or alias existing commands in your login shell - much as you would use `alias` in `csh` or `ksh`; or second, as subroutines within a shell script - like the . built-in command or the `source` command in `csh`.

Within a shell function, you can use any shell construct or built-in command, call any UNIX utility, call any other shell script (though that's a bit silly), or call any other shell function - plus, of course, shell functions, like shell scripts, can be recursive. So what's the difference between a shell script and a shell function? Unlike shell scripts, shell functions are executed within the current shell, so you avoid the process creation overhead. Also, the syntax is different.

The basic format of a shell function is:

```
function_name ()
{
    commands ...
}
```

Using shell functions to add commands to your login shell

Here are some short examples of how you can use shell functions to add commands to your login shell. What you do is put them all into one file called, say, `.funcs` in your login directory and then 'dot' the file in your `.profile`, as in:

```
# Example .profile file
stty cr0 nl0 ofill ixany
stty kill '^x' intr '^c'
PATH=/bin:/usr/bin:$HOME/bin:.
MAILCHECK=60
CDPATH=.:..:../..:$HOME
export PATH MAILCHECK CDPATH
export PS1 PS2
# This is the important bit:
. $HOME/.funcs
#
trap '. $HOME/.logout' 0
```

Anyway, here are those example shell functions. The first is called `chdir` and was written specially for MS-DOS types:

```
chdir()
{
# chdir - change PS1 prompt to
#         echo working directory.
if [ $# -lt 1 ]
   then cd
else
   cd $1
fi

if [ "`pwd`" = "$HOME" ]
   then PS1='$ '
else
   PS1="`pwd | sed s?$HOME/??`>"
fi
}
```

This next one I actually use quite a lot. I call it `sw` because there's a VAX/VMS DCL command called that, and also because it's short for 'switch':

```
sw()
{
# switch directory
if test -z "$OLD_DIR"
then
    OLD_DIR=$HOME
fi
NEW_DIR=${1:-$OLD_DIR}
OLD_DIR=`pwd`
cd $NEW_DIR
pwd
}
```

The next one is very simple - it just aliases the shell built-in `exit`. I wrote it because I thought there was more consistency between `login:` and `logout` than with `exit` or `CTRL-d`:

```
logout()
{
# logout
exit
}
```

Finally, here is a rather longer example that implements command history. It also has a bit of history attached to it. A couple of years ago, I got very fed up with somebody on one of my UNIX courses who went on and on about how UNIX or the SVR2 shell didn't have command history (he was either an MS-DOS type or a `csh` fan - I forget which) so, after dinner, I sat down and wrote this:

```
history()
{
# shell command history
# Don't forget to export PS1 in
# your .profile
while echo "$PS1\c" ; read CMD
do
  if test "`echo $CMD | \
            cut -c1`" = "~"
  then
      NUM=`echo $CMD | \
           cut -c2-3`
      if test -n "$NUM"
      then
          CMD="`sed -n $NUM'p' \
               $HOME/.history`"
          eval $CMD
      else
          cat $HOME/.history | \
              nl
      fi
  elif test "$CMD" != ""
  then
     eval $CMD
     echo $CMD >> $HOME/.history
     tail $HOME/.history > \
            /usr/tmp/HIST.$$
     mv /usr/tmp/HIST.$$ \
            $HOME/.history
  fi
done
}
```

So there you have it: the SVR2 (Bourne) shell with history. As I remember, Brian Kernighan in the report on the 'new' `awk`,[1] shows how you can do something very similar but probably much better, with `awk`.

Using Shell functions as subroutines

Just enough space left to show how you can use shell functions as subroutines. There are two things to remember: first, shell functions must be defined before they are referenced; and, second, shell functions don't `exit`, they `return`.

As subroutines, a typical structure would be:

```
shell_function_1()
{
    ...
}

shell_function_2()
{
    ...
}

main_loop()
{
if test ...
then
    shell_function_1
else
    shell_function_2
fi
}
```

It would be nice to have an example, but there it is - perhaps another time. On the other hand, if anybody out there is `really` interested, I'll post the code in this article (plus even more examples, with subroutines!) to `eunet.sources`.

1. AT&T Bell Laboratories, *Computing Science Technical Report*, No.118 (1985). There is now a section on the new *awk* in Volume 1 of the UNIX System V Release 3 `Programmer's Guide`.

AUUG Management Committee Meeting

September 12, 1988

MINUTES

The meeting opened at 13:27 with the following committee members present: President Greg Rose (GR) in the chair, Secretary Tim Roper (TR), Treasurer Michael Tuke (MT), Chris Maltby (CM) and Rich Burridge (RB). Also present were the AUUGN Editor John Carey (JC), and the following members of the retiring committee: John Lions (JL), Robert Elz (KRE), Piers Lauder (PL) and Peter Wishart (PW). The President GR took the Chair. Retiring committee member, Chris Campbell (CC) arrived at 13:54 and committee member Frank Crawford (FC) arrived at 14:23.

1. Apologies
   There were no apologies.

2. Minutes of last meeting (May 13, 1988)
   There were no amendments.

   Moved CM/TR **That the minutes of the previous meeting be accepted.** Carried unanimously.

3. Business arising from Minutes
   50. KRE stated that the election and referendum has happened.
   17. CM stated that the post office box has not been redirected.

4. President's Report
   GR welcomed the new committee members.

5. Retiring Secretary's Report
   KRE reported current membership figures of 202 Ordinary, 41 Institutional, 4 Student, 0 Honorary Life, and 17 newsletter subscriptions.

   Moved RB/CM **That the retiring Secretary's report be accepted.** Carried unanimously.

6. Secretary's Report
   TR welcomed the new members of the committee. He tabled the following correspondence.
   (a) from **IDC Australia Pty Ltd, Compass Research Pty Ltd, Warhurst Brown** and **Sun Microsystems Australia** concerning their respective purchases of the mailing list.
   (b) from **/usr/group** regarding AUUG's entry in their *UNIX Products Directory*.
   (c) from **Sigma Data Corporation** regarding word processing software available for the Miniframe.
   (d) to and from **The British Council** regarding travel support for Sunil Das to attend AUUG88 (unsuccessful).
   (e) from **Moss Products Pty Ltd** requesting membership information.
   (f) to all members with a copy of the first issue of *Computer Systems*.
   (g) to all members giving notice of the 1988 AGM.
   (h) from James Watt requesting $250.00 travel subsidy to

present a paper at AUUG88.

(i) a proposal for $10000 Physical Loss and $1000000 Public Liability insurance at AUUG88 to **H M Bates Australia Pty Limited**; a quote from them for $792.00; a letter to them enclosing payment for that amount; a letter from them enclosing the Policy Document.

(j) from **UNIX PEOPLE** describing their offer of secretarial assistance; to them inviting Ken Preiss to present the proposal to the meeting at 15:00.

TR thanked KRE for performing membership functions whilst TR was involved in AUUG88 organisation.

Moved GR/CM **That the insurance contract be ratified.** Carried unanimously.

TR reported that due to the short time before AUUG88 at the time the request from James Watt was received, he and MT had approved the expenditure and Mr Watt had been advised of the approval by phone. TR pointed out that the notice for the 1988 AGM had been posted less than the required four weeks beforehand and that the AGM should move to ratify itself. Also, that the decision of the committee to unbundle conference dinner pricing as reported to the 1987 AGM had been subsequently reversed and that this should be discussed at the 1988 AGM.

Moved GR/RB **That the Secretary's reported be accepted.** Carried unanimously.

7. Retiring Treasurer's Report

CM presented a balance sheet showing an overall loss of $3550.94 for the period 1/7/87 to 31/5/88 with a final balance of $32490.21. He pointed out that the income from the NSWIT conference was not included on this sheet but the expenses were, as the surplus of approximately $12000 had not been extracted from the NSWIT organisers until after 31/05/88. He also tabled a statement from the accountant D. Howes indicating that the balance sheet looked reasonable but that there were insufficient records to perform an audit. In response to a question, CM advised that the donation to Wollongong Hospital had not yet been made.

Moved TR/RB **That the Retiring Treasurer's report be accepted.** Carried unanimously.

8. Treasurer's Report

MT reported that he had taken possession of the books that day and had previously contacted an accountant, Stuart Nicol of **Nicol&Nicol** regarding keeping and auditing the books. He indicated that he intended to have the books kept by an accountant.

Moved TR/CM **That the Treasurer be authorised to engage an accountant to keep and audit the books and to spend an amount of not more than $1000 per annum in doing so.** Carried unanimously.

9. AUUGN Editor's Report

JC noted that his initial term of two years expired today and gave 12 months notice of resignation. He reported a

continuing problem with the post office delivering bundles of newsletters to the address on the top copy despite his having talked to the printer and postmaster about it before. Also that the printer is slow, taking six weeks in the worst case. There was discussion about engaging a new printer and/or direct mail house. JC reported that he was still storing large amounts of back issues at his house and would like to be rid of them. He expressed concern about the small size of the newsletter mailing list and the lack of contributions to the newsletter. He suggested running a competition in the off-conference season and/or appointing sub-editors to manage sections of the newsletter.

Moved GR/CM **That the Editor's resignation be accepted with regret.** Carried unanimously.

Moved TR/GR **That the Editor's term be extended for a further 12 months.** Carried unanimously.

JC stated that he had to get his float refreshed approximately every issue.

Moved TR/RB **That the Editor's float be increased to $500.** Carried unanimously.

15. Secretarial Assistance
This item was brought forward due to the expected arrival of Ken Preiss at 15:00. The proposal from **UNIX PEOPLE** previously circulated and tabled was discussed. Ken Preiss (KP) arrived at 15:00. He described the background of the company and the motivation for the proposal. There was some discussion with him on the issues of future dissolution of any such agreement and of confidentiality of AUUG correspondence. It was agreed that some agreement on the method of splitting up the joint assets (viz the mailing list) after separation would be required in advance. He gave an assurance that confidentiality would be respected. KP left at 15:30.

Moved CM/TR **That the meeting be adjourned for 10 minutes.** Carried unanimously.

The meeting resumed at 15:55 and continued discussion of Item 15.

15. Secretarial Assistance (cont.)
Each member present stated their opinion on the proposal. This item was then adjourned.

10. Appointment of Auditor
Moved TR/CM **That David Howes be appointed Auditor until the end of the 1988 AGM.** Carried unanimously.

Moved MT/GR **That Stuart Nicol of Nicol&Nicol be appointed Auditor as of the end of the 1988 AGM.** Carried.

11. 1988 Conference and Exhibition
TR presented a report on AUUG88:
(a) A cheque account had been opened entitled AUUG88.
(b) As at 8/9/88 247 registrations had been received.
(c) There were approximately 20 exhibitors.

(d) There were three major sponsors, namely **IBM Australia Limited** (brochure), **Nixdorf Computer Ltd** (folder), **Pyramid Technology Australia Pty Ltd** (dinner).
(e) The programme included 7 presentations by the 4 guest speakers (Lesk, Karels, Mashey, Thompson), 14 presentations reviewed by the Programme Committee, 6 presentations invited by TR, opening remarks by Prof Peter Poole, 1 panel session, 3 business meetings (AUUG 1988 AGM, ACSnetSIG, SLUG), 2 social events (cocktail reception and conference dinner), 1 other event (presentation of Sigma Data postgraduate awards).
(f) The publicity had included mailing the brochure to the complete mailing list; press advertisements and two press releases costing $4240, including an invitation to the press to attend the Wednesday morning session and the following lunch.
(g) New features included a membership desk and the publication of proceedings as an issue of AUUGN.

Moved GR **That TR be thanked.** Carried by acclamation.

12. 1989 Summer Meetings

Moved TR/RB **That a sub-committee be appointed to organise regional, technical meetings in early 1989, as many as one per state and territory; that the sub-committee be authorised to arrange one or more guest speakers; that the sub-committee be authorised to co-opt AUUG members and to appoint local organisers; that the meetings should be organised on a revenue-neutral basis but that AUUG should underwrite such meetings to the extent of $10000 in total.** Carried unanimously.

GR volunteered to chair the sub-committee. PL was co-opted on the spot. Some discussion followed.

13. 1989 Winter Conference and Exhibition

TR reported that he had asked Wael Foda to tentatively book the Sydney Hilton for the week 7th-11th August, 1989, which he has done. PL reported that he had investigated the new Gazebo convention centre but that its exhibition space was too small. CC stated that as an exhibitor he did not like the Hilton.

Moved TR/RB **That a sub-committee be appointed to organise a meeting known as "The AUUG Winter 1989 Conference and Exhibition" or "AUUG89"; that the sub-committee report to the next committee meeting with a timetable and budget; that ACMS be engaged as the exhibition organiser on a fee-for-service basis with AUUG underwriting the costs; that AUUG89 be held at the Sydney Hilton Hotel from Tuesday 8th to Thursday 10th August, 1989.** Amendment moved GR/RB **That the words "Thursday 10th August, 1989" be replaced with "Friday 11th August, 1989, with separately priced tutorials occurring on the Tuesday".** Amendment carried unanimously. Motion carried unanimously.

No sub-committee was actually appointed.

14. Incorporation

KRE reported that incorporation was complete and described some of the obligations that it entailed. He suggested that

at some stage the group may wish to replace him as Public Officer but indicated that he was happy to remain so for the time being.

Moved CM **That KRE be thanked for his efforts in bringing about incorporation.** Carried by acclamation.

16. Preparation of FY 88/89 Budget
MT agreed to prepare a budget for the 1988/99 financial year.

Moved RB/FC **That the signatories of the group's general accounts be GR, CM, MT and TR.** Carried unanimously.

Moved GR/RB **That the committee express its intention to accumulate less assets and to spend more on services to members.** Carried unanimously.

17. Benefits for Institutional Members
Moved GR/CM **That Institutional Members be provided with a copy of the /usr/group product directory.** Carried unanimously.

This item was adjourned until the next meeting.

18. Constitutional Changes
KRE foreshadowed some proposed changes and stated that he would circulate them.

19. Next Committee Meeting
This item was adjourned by the Chair.

20. Other Business
(a) JL stated that the group should plan further ahead and that planning for 1990 should start soon.
(b) JL asked whether AUUG had a representative on the editorial board of *Computer Systems*. It was mentioned that there was a member of AUUG on the board, namely KRE.
(c) GR said that he thought that the group was sufficiently large and active to warrant such things as business cards for committee members.

Moved GR/TR **That the Secretary be authorised to purchase the common seal, and have stationary showing the new name designed and printed.** Carried unanimously.

Moved TR/CM **That the policy on supplying the Mailing List to third parties provide that the procedure protect the group's interest in this asset by preventing unauthorised usage as far as practicable; that financial Members have the right to request that their entry be withheld from third parties and that such requests be honoured as soon as practicable; that the committee is of the opinion that this policy precludes the supply of the list itself to people other than agents of AUUG; that the terms for supply of the Mailing List be 50 cents per entry plus costs to be paid in advance; that the Secretary be authorised to determine the procedure and cost component from time to time, including the question of whether the list is supplied in full or in part.** Carried unanimously.

At 17:30 the Chair adjourned the meeting to the Southern Cross Hotel where it resumed at 18:25 with item 11 and Wael Foda (WF) present.

11. 1988 Conference and Exhibition (cont).
WF presented a report of current registrations an incomplete financial statement.

Moved TR/GR **That the report be accepted with thanks.** Carried unanimously.

The Chair adjourned the meeting until 15/9/88 at 17:00. The meeting resumed on 15/9/88 at 17:20 without PL, PW and CC but with committee member Tim Segall (TS) present. Steve Jenkin (SJ) was also present. Item 11 was continued.

11. 1988 Conference and Exhibition (cont.)
There was considerable discussion of AUUG88 which had just closed. GR compiled and took away a list of things that were wrong or things that could be done better next time. MT left at 17:50. TR stated that it would be a pity if the talents of CC for novel marketing were lost to the group.

19. Next Meeting
Moved TR/CM **That the next meeting of the AUUG Management Committee be on October 28, 1988 at Scripture Union House, 120 Chalmers Street, Surry Hills, NSW at 10:00.** Carried unanimously.

15. Secretarial Assistance (cont.)
GR reported that **Informix** had agreed to provide a licence for Informix for the Miniframe in exchanged for one year's Institutional Membership. He stated on behalf of **Softway Pty Ltd** that they would be willing to develop software using Informix at no cost to handle the membership data and then make clerical assistance available at break-even rates.

Moved GR/CM **That this proposal be accepted subject to a review in 3 months time.** Carried with GR abstaining.

Moved CM/GR **That UNIX PEOPLE be thanked for their offer and informed that we are proceeding with an alternative proposal.** Carried with TR abstaining.

20. Other Business (cont.)
The Chair reconvened this item.
(a) There was a general discussion of the benefits of Chapters to the group. It was apparent that there was some confusion between WAUG and AUUG regarding the procedure for WAUG becoming a Chapter of AUUG. It was pointed out that the next step was for WAUG to petition AUUG in the manner prescribed in the AUUG constitution. There was discussion on the question of whether Chapter members had to be AUUG members. TR read out the relvant section of the constitution and the rules governing Chapters passed at a previous meeting. It appeared that such membership was not required. GR reported that he and Ken Thompson were to attend a meeting of WAUUG and stated that he attempt to solicit their feelings. Someone suggested that the Adelaide UNIX Users Group was

defunct. SJ offered to convene a potential Chapter in NSW and the ACT. TR stated that he had been talking to a potential convener of a potential Chapter in QLD.

(b) There was discussion on the protection of the group's trademarks.

Moved TR/CM **That the Secretary be authorised to arrange an appropriate form of protection for the group's trademarks and to spend up to $500 in doing so.** Carried.

Moved TR/GR **That the meeting be closed.** Carried unanimously.

The meeting then closed at 18:55.

THIS PAGE INTENTIONALLY LEFT BLANK

AUUG

Membership Categories

Once again a reminder for all ''members'' of AUUG to check that you are, in fact, a member, and that you still will be for the next two months.

There are 4 membership types, plus a newsletter subscription, any of which might be just right for you.

The membership categories are:

Institutional Member
Ordinary Member
Student Member
Honorary Life Member

Institutional memberships are primarily intended for university departments, companies, etc. This is a voting membership (one vote), which receives two copies of the newsletter. Institutional members can also delegate 2 representatives to attend AUUG meetings at members rates. AUUG is also keeping track of the licence status of institutional members. If, at some future date, we are able to offer a software tape distribution service, this would be available only to institutional members, whose relevant licences can be verified.

If your institution is not an institutional member, isn't it about time it became one?

Ordinary memberships are for individuals. This is also a voting membership (one vote), which receives a single copy of the newsletter. A primary difference from Institutional Membership is that the benefits of Ordinary Membership apply to the named member only. That is, only the member can obtain discounts on attendance at AUUG meetings, etc, sending a representative isn't permitted.

Are you an AUUG member?

Student Memberships are for full time students at recognised academic institutions. This is a non voting membership which receives a single copy of the newsletter. Otherwise the benefits are as for Ordinary Members.

Honorary Life Memberships are a category that isn't relevant yet. This membership you can't apply for, you must be elected to it. What's more, you must have been a member for at least 5 years before being elected. Since AUUG is only just approaching 3 years old, there is no-one eligible for this membership category yet.

Its also possible to subscribe to the newsletter without being an AUUG member. This saves you nothing financially, that is, the subscription price is the same as the membership dues. However, it might be appropriate for libraries, etc, which simply want copies of AUUGN to help fill their shelves, and have no actual interest in the

contents, or the association.

Subscriptions are also available to members who have a need for more copies of AUUGN than their membership provides.

To find out if you are currently really an AUUG member, examine the mailing label of this AUUGN. In the lower right corner you will find information about your current membership status. The first letter is your membership type code, N for regular members, S for students, and I for institutions. Then follows your membership expiration date, in the format exp=MM/YY. The remaining information is for internal use.

Check that your membership isn't about to expire (or worse, hasn't expired already). Ask your colleagues if they received this issue of AUUGN, tell them that if not, it probably means that their membership has lapsed, or perhaps, they were never a member at all! Feel free to copy the membership forms, give one to everyone that you know.

If you want to join AUUG, or renew your membership, you will find forms in this issue of AUUGN. Send the appropriate form (with remittance) to the address indicated on it, and your membership will (re-)commence.

As a service to members, AUUG has arranged to accept payments via credit card. You can use your Bankcard (within Australia only), or your Mastercard by simply completing the authorisation on the application form.

AUUG

Application for Ordinary, or Student, Membership Australian UNIX* systems Users' Group.

*UNIX is a registered trademark of AT&T in the USA and other countries

To apply for membership of the AUUG, complete this form, and return it with payment in Australian Dollars, or credit card authorisation, to:

AUUG Membership Secretary
P O Box 366
Kensington NSW 2033
Australia

- Please don't send purchase orders — perhaps your purchasing department will consider this form to be an invoice.
- Foreign applicants please send a bank draft drawn on an Australian bank, or credit card authorisation, and remember to select either surface or air mail.

I, ... do hereby apply for

☐ Renewal/New* Membership of the AUUG	$65.00	
☐ Renewal/New* Student Membership	$40.00	(note certification on other side)
☐ International Surface Mail	$10.00	
☐ International Air Mail	$50.00	

Total remitted AUD$_____
(cheque, money order, credit card)

* Delete one.

I agree that this membership will be subject to the rules and by-laws of the AUUG as in force from time to time, and that this membership will run for 12 consecutive months commencing on the first day of the month following that during which this application is processed.

Date: __/__/__ Signed: _____

☐ Tick this box if you wish your name & address withheld from mailing lists made available to vendors.

For our mailing database - please type or print clearly:

Name: ..... Phone: ..... (bh)

Address: .... (ah)

..... Net Address: .....

.....

..... *Write "Unchanged" if details have not*

..... *altered and this is a renewal.*

Please charge $_____ to my ☐ Bankcard ☐ Visa ☐ Mastercard.

Account number: _____ _____ _____ _____. Expiry date: __/__.

Name on card: _____ Signed: _____

Office use only:

Chq: bank _____ *bsb* ____-____ *a/c* _____ *#* _____

Date: __/__/__ *$* *CC type* ___ *V#* _____

Who: _____ *Member#* _____

Student Member Certification *(to be completed by a member of the academic staff)*

I, ... certify that

..... *(name)*

is a full time student at ..... *(institution)*

and is expected to graduate approximately ___/___/___.

Title: _____ Signature: _____

AUUG

Application for Institutional Membership
Australian UNIX* systems Users' Group.

*UNIX is a registered trademark of AT&T in the USA and other countries.

To apply for institutional membership of the AUUG, complete this form, and return it with payment in Australian Dollars, or credit card authorisation, to:

AUUG Membership Secretary
P O Box 366
Kensington NSW 2033
Australia

• Foreign applicants please send a bank draft drawn on an Australian bank, or credit card authorisation, and remember to select either surface or air mail.

..... does hereby apply for

☐ New/Renewal* Institutional Membership of AUUG	$300.00
☐ International Surface Mail	$ 20.00
☐ International Air Mail	$100.00
Total remitted	AUD$_____

(cheque, money order, credit card)

* Delete one.

I/We agree that this membership will be subject to the rules and by-laws of the AUUG as in force from time to time, and that this membership will run for 12 consecutive months commencing on the first day of the month following that during which this application is processed.

I/We understand that I/we will receive two copies of the AUUG newsletter, and may send two representatives to AUUG sponsored events at member rates, though I/we will have only one vote in AUUG elections, and other ballots as required.

Date: __/__/__ Signed: _____

Title: _____

☐ Tick this box if you wish your name & address withheld from mailing lists made available to vendors.

For our mailing database - please type or print clearly:

Administrative contact, and formal representative:

Name: ..... Phone: ..... (bh)

Address: .... (ah)

..... Net Address: .....

.....

..... *Write "Unchanged" if details have not*

..... *altered and this is a renewal.*

Please charge $_____ to my/our ☐ Bankcard ☐ Visa ☐ Mastercard.

Account number: _____ _____ _____ _____. Expiry date: __/__.

Name on card: _____ Signed: _____

Office use only: **Please complete the other side.**

Chq: bank _____ *bsb* ____-____ *a/c* _____ *#* _____

Date: __/__/__ *$* *CC type* ___ *V#* _____

Who: _____ *Member#* _____

Please send newsletters to the following addresses:

Name: ..... Phone: ..... (bh)
Address: .... (ah)
..... Net Address: .....
.....
.....
.....

Name: ..... Phone: ..... (bh)
Address: .... (ah)
..... Net Address: .....
.....
.....
.....

Write "unchanged" if this is a renewal, and details are not to be altered.

Please indicate which Unix licences you hold, and include copies of the title and signature pages of each, if these have not been sent previously.

Note: Recent licences usally revoke earlier ones, please indicate only licences which are current, and indicate any which have been revoked since your last membership form was submitted.

Note: Most binary licensees will have a System III or System V (of one variant or another) binary licence, even if the system supplied by your vendor is based upon V7 or 4BSD. There is no such thing as a BSD binary licence, and V7 binary licences were very rare, and expensive.

☐ System V.3 source
☐ System V.2 source
☐ System V source
☐ System III source
☐ 4.2 or 4.3 BSD source
☐ 4.1 BSD source
☐ V7 source

☐ System V.3 binary
☐ System V.2 binary
☐ System V binary
☐ System III binary

☐ Other *(Indicate which)* .....

AUUG

Application for Newsletter Subscription
Australian UNIX* systems Users' Group.

*UNIX is a registered trademark of AT&T in the USA and other countries

Non members who wish to apply for a subscription to the Australian UNIX systems User Group Newsletter, or members who desire additional subscriptions, should complete this form and return it to:

AUUG Membership Secretary
P O Box 366
Kensington NSW 2033
Australia

- Please don't send purchase orders — perhaps your purchasing department will consider this form to be an invoice.
- Foreign applicants please send a bank draft drawn on an Australian bank, or credit card authorisation, and remember to select either surface or air mail.
- Use multiple copies of this form if copies of AUUGN are to be dispatched to differing addresses.

Please *enter / renew* my subscription for the Australian UNIX systems User Group Newsletter, as follows:

Name: ..... Phone: ..... (bh)

Address: .... (ah)

..... Net Address: .....

..... *Write "Unchanged" if address has*

..... *not altered and this is a renewal.*

For each copy requested, I enclose:

☐ Subscription to AUUGN	$ 65.00
☐ International Surface Mail	$ 10.00
☐ International Air Mail	$ 50.00

Copies requested (to above address) _____

Total remitted AUD$_____

(cheque, money order, credit card)

☐ Tick this box if you wish your name & address withheld from mailing lists made available to vendors.

Please charge $_____ to my ☐ Bankcard ☐ Visa ☐ Mastercard.

Account number: _____ _____ _____ _____. Expiry date: __/__.

Name on card: _____ Signed: _____

Office use only:

Chq: bank _____ *bsb* ____-____ *a/c* _____ *#* _____

Date: __/__/__ *$* *CC type* ___ *V#* _____

Who: _____ *Subscr#* _____

AUUG

Notification of Change of Address
Australian UNIX* systems Users' Group.

*UNIX is a registered trademark of AT&T in the USA and other countries.

If you have changed your mailing address, please complete this form, and return it to:

AUUG Membership Secretary
P O Box 366
Kensington NSW 2033
Australia

Please allow at least 4 weeks for the change of address to take effect.

Old address (or attach a mailing label)

Name: ..... Phone: ..... (bh)

Address: .... (ah)

..... Net Address: .....

.....

.....

.....

New address (leave unaltered details blank)

Name: ..... Phone: ..... (bh)

Address: .... (ah)

..... Net Address: .....

.....

.....

.....

Office use only:

Date: ___/___/___

Who: _____ *Memb#* _____