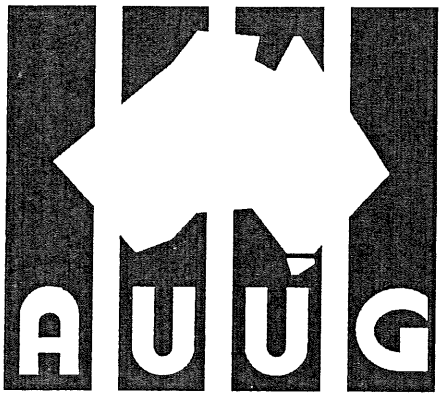


Australian UNIX systems User Group Newsletter

AUUGN

Volume 11, Number 2

April 1990

1990 Sumer Technical Meetings

Registered by Australia Post, Publication Number NBG6524

The Australian UNIX* systems User Group Newsletter

Volume 11 Number 2

April 1990

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

Next AUUG Meeting

The AUUG90 Conference and Exhibition will be held
from September 25th to September 28th 1990.
The venue will be the World Congress Centre, Melbourne.

Editorial

This issue features reports on the AUUG Summer'90 series of meetings. Meetings were held in Perth, Hobart, Melbourne, Canberra and Sydney, and were generally considered to be a great success.

This issue also features three of the papers presented at summer meetings, and some of the remaining papers will appear in the next issue. (Aside to people who presented papers but have not written them up for me - it is not too late.)

The success of Summer'90 shows that the concept of Summer Technical Meetings is a workable one. However, to make sure we have a successful program in 1991, planning should start now. In particular, there was no meetings in Adelaide, Brisbane or Darwin because organisers could not be found for those cities. So now is the time to start thinking about volunteering, either yourself or someone else, to organise a meeting.

People often ask me, or members of the committee, why they should join AUUG. At the moment the only visible benefits are a discount at the conference and summer meetings, and a copy of this illustrious publication. You will be glad to know, though, that the issue of member benefits is being hotly pusued by your management committee. The committee plans to issue a survey of the members to assess the perception of member benefits. In the meantime, if you have any ideas on how the group could better serve you, please write to the management committee, or to me.

The good news is that the management committee have decided to hold membership fees for the next financial year at their current levels while the issue of member benefits is sorted out.

AUUGN Correspondence

All correspondence reguarding the AUUGN should be addressed to:-

David Purdue
AUUGN Editor
PO Box 366
Kensington, NSW, 2033
AUSTRALIA

ACSnet: *auugn@munnari.oz*

Phone: +61 3 587 1444
Fax: +61 3 580 5581

Contributions

The Newsletter is published approximately every two months. The deadline for contributions for the next issue is Friday the 1st of June 1990.

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

I prefer documents to be e-mailed to me, and formatted with troff. I can process mm, me, ms and even man macros, and have tbl, eqn and pic preprocessors, but please note on your submission which macros and preprocessors you are using. If you can't use troff, then just plain text please.

Hardcopy submissions should be on A4 with 30 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.

Come on, everyone, contribute! If the muse is upon you, and you have to write, but you can't think of anything to write about, give me a call and I'll throw some ideas at you.

Advertising

Advertisements for the AUUG are welcome. They must be submitted on an A4 page. No partial page advertisements will be accepted. Advertising rates are $300 for the fisrt A4 page, $250 for a second

page, and $750 for the back cover. There is a 20% discount for bulk ordering (ie, when you pay for three issues or more in advance). Contact the editor for details.

Mailing Lists

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

Back Issues

Various back issues of the AUUGN are available, details are printed at the end of this issue.

Acknowledgement

This Newsletter was produced with the kind assistance of and on equipment provided by Labtam Information Systems Pty Ltd.

Disclaimer

Opinions expressed by authors and reviewers are not necessarily those of AUUG Incorporated, its Newsletter or its editorial committee.

AUUG Institutional Members

ACUS / UNISYS
ANL Limited
Adept Business Systems Pty Ltd
Alcatel STC Australia
Aldetec Pty Ltd
Alliance Computer Centre Pty. Ltd
Apple Computer Australia
Apscore International Pty Ltd
Australian Electoral Commision
Australian Nuclear Science &;Technology Organisation
Australian Wool Corporation
BHP Melbourne Research Labs
Ballarat Base Hospital
Basser Department of Computer Science
Bond University Library Service
Bond University;School of Information and Computing Science
CADAD Support Section - SECWA
Capricorn Coal Management Pty. Ltd.
Civil Aviation Authority
Colonial Mutual
Commodore Business Machines Pty. Ltd.
Comperex (NSW) Pty Ltd
Computer Power IR+D, NSW Branch
Computer Power Today IR+D
Computer Software Packages
Computerscene International
Corinthian Engineering Pty Ltd
Cybergraphic Systems Pty Ltd
DBA Limited
DMR Group
Data General
Davey Products Pty Ltd
Dept of Agricultural + Rural Affairs
Dept of Industry, Technology and;Resources, Victoria
Digital Equipment Corporation; (Australia) Pty. Limited
Earth Resource Mapping Pty Ltd
Elxsi Australia Ltd
Epson Australia Pty Ltd
Exicom Australia Pty Ltd
Flinders University;Discipline of Computer Science
Fremantle Port Authority
Genasys II Pty Ltd
Golden Casket Art Union
Gould Electronics Pty Ltd
Harris & Sutherland Pty Ltd
Hewlett Packard Australia Limited
Hewlett-Packard;Australian Software Operation
Honeywell Software Centre
IBM Australia Limited

AUUG Institutional Members

ICL Australia Pty. Ltd.
IPS Radio and Space Services
Ipec Transport Group
Labtam Information Systems Pty Ltd
Macquarie Bank Limited
Macquarie University
Main Roads Department, Queensland
Mincom Pty. Ltd.
Motorola Communications Australia
NEC Information Systems Australia Pty Ltd
NSW Parliament
National Engineering Information Services P/L
Nixdorf Computer Pty Limited
OPSM
Olivetti Australia Pty Ltd
Olympic Amusements P/L
Overseas Telecommunications Corporation
Performance Systems
Prentice Computer Centre
Prime Computer of Australia Ltd
Pyramid Technology Corporation
Q. H. Tours Limited
Queensland Department of Mines
RMIT Computer Centre
Roads and Traffic Authority NSW
SEQEB
Sigma Data Corporation Pty Ltd
Sola International Holdings Ltd
South Australian Institute of Technology
Sphere Systems Pty Ltd
State Bank Victoria
State Bank of NSW
State Library of Tasmania
Sugar Research Institute
Sun Microsystems Australia
Swinburne Institute of Technology
Sybase Australia Pty Ltd
Tattersall Sweep Consultation
Telecom Network Engineering - C.S.S
The Australian Artificial Intelligence Institute
The Australian National University
The Mathematics and Computing Department - BCAE (Kelvin Grove Campus)
The University of Adelaide
The University of Wollongong
University of Technology Sydney;Computing Services Division
Vicomp
Wang Australia Pty Ltd
Wyse Technology Pty. Ltd.
Yartout Pty Ltd

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**

SESSPOOLE

Who is SESSPOOLE?

SESSPOOLE is the South Eastern Suburbs Society for Programmers Or Other Local Enthusiasts. Here we are talking about the South Eastern Suburbs of Melbourne.

What is SESSPOOLE?

SESSPOOLE is a group of programmers and friends who meet every six weeks or so for the purpose of discussing UNIX, drinking wines and ales, and just relaxing and socialising over dinner. Anyone who subscribes to the aims of SESSPOOLE is welcome to attend our meetings, whether they come from the South Eastern Suburbs or not. The aims of SESSPOOLE are:

> To promote knowledge and understanding of the UNIX system, and of similar or related computer systems; and to promote knowledge and understanding of red and white wines, and of similar or related wines.

SESSPOOLE is also the first Chapter of the AUUG to be formed, and its members were involved in the staging of the AUUG Summer'90 Melbourne Meeting.

When is SESSPOOLE?

The next meeting of SESSPOOLE is on Thursday, the 17 of May 1990, at 6:30pm.

Where is SESSPOOLE?

The next meeting of SESSPOOLE will be held in the Bistro of the Oakleigh Hotel, 1555 Dandenong Road, Oakleigh.

Want to know more?

To find out more about SESSPOOLE and SESSPOOLE activities, contact either David Purdue <*davidp@labtam.oz*> or John Carey <*john@labtam.oz*>. Their phone number is 587-1444 (bh). Or look for announcements in the newsgroup **aus.auug**.

Summary Report on AUUG - Summer '90 (Victoria)

The first Victorian AUUG Summer Conference was held on Tuesday 6th of February at the Rotunda Lecture Theatre of Monash University. The theme was *UNIX - The Universal Environment.* This theme attracted a wide range of topics such as data and network security, software/hardware design and implementation. multiprocessor simulation, and graphics. The keynote address was given by Greg Rose (President of AUUG Inc) on Public Key Privacy and Authentication. The programme was also made up of two invited talks; Dr Ken J. McDonell talking about *MUSBUS - What has Been Learnt* and Arnold N. Pears with a talk about *Shared Data Access Rates as a Metric for Distributed Algorithm Performance.*

The meeting was attended by 95 people including the speakers. Thanks to the hard work of the organising and programme committee, the meeting went smoothly and general opinion deemed it a great success. Following the conference was a SESSPOOLE meeting, which most people attended.

Following is a copy of the abstracts submitted by the speakers, some of these full papers appear in this issue.

Public Key Privacy and Authentication

Greg Rose
Softway Pty. Ltd.
greg@softway.sw.oz

Lucy Chubb and Greg Rose, of Softway, have been working on and off for about eight months on a public key privacy and authentication system based on the RSA (Rivest, Shamir and Adelman) system.

This talk will examine some of the interesting technical problems to be solved, and how we solved them, as well as an overview of how the system is intended to work and integrate with networks.

One of the first problems to be solved was the selection of very large prime numbers without predictable characteristics. This involved high precision integer arithmetic, and some interesting numerical approaches. The issues to be addressed to take advantage of Softway's 6 CPU Sequent in this context are also interesting.

The RSA system is about to be adopted by the Internet; this talk will also examine how the internet intends to address user validation questions and public key interchange.

Unix Security Aspects in the US DoD

Dr Mark Anderson
Defence Science & Technology Organisation
South Australia
mxa@itd.dsto.oz

Various UNIX systems have introduced mandatory access controls in order to provide support for multilevel security. Much of the impetus for such implementations has come from the defence sector. However, most implementations focus exclusively on issues of confidentiality, typically variants of the Bell and La Padula model. As yet there are very few systems attempting to tackle integrity, as well as confidentiality issues. After a very brief review of mandatory confidentiality access control mechanisms, we go on to describe what might be expected of UNIX systems in the future concerning integrity. Two models will be reviewed, i.e. Clark and Wilson's commercial integrity model and Biba's dual of the Bell and La Padula model. In addition we give a short discussion on Cohen's access program which exploits the setuid feature. Some obvious benefits, e.g. generic resistance to viruses, are discussed as well as possible implementation strategies in the UNIX environment.

STELNET - A Secure Remote Login Frontend and Shell

Lawrence Brown
Department of Computer Science
University College, UNSW,
Australian Defence Force Academy
Canberra ACT 2600. Australia.
lpb@csadfa.cs.adfa.oz

This paper discusses the development of stelnet, a frontend for telnet, and an associated shell for use on the remote system. These provide an automated challenge/response login procedure, session key exchange, and encryption of all data exchanged for the subsequent session. The encryption algorithm may be chosen from DES, LOKI, or a stream cipher; depending on the security/speed tradeoff desired. Initially stelnet has been implemented using BSD

pseudo-ttys. The reasons for their choice, advantages, and limitations, are discussed. A future version, to be incorporated into telnetd is then proposed.

MUSBUS – What Has Been Learnt

Dr Ken J. McDonell
Pyramid Technology Corporation
Melbourne, Australia
kenj@yarra.oz.au

As the eighth anniversary of the initial development of MUSBUS approaches, it seems appropriate to contemplate the lessons that have been learnt and the likely future evolution of multi-user system performance analysis and benchmarking.

General conclusions include,

- The overwhelming lure of a single "figure of merit", despite all the evidence testifying to the irrationality of the approach.
- General laziness and/or illiteracy amongst consumers of benchmark reports.
- Good benchmarks break systems that are delivered through sloppy "QA" procedures.
- The importance of packaging, marketing and folk lore as a defence against poor experimental engineering.
- Scaling of tests across two orders of magnitude in performance is technically very difficult, and often simpler to ignore.

The intention is to highlight the pitfalls associated with predicting system performance, so that end-users may either adopt a more critical, cynical and analytical attitude to competing performance claims, or conduct their own well-engineered performance testing.

Using UNIX as a Persistent Programming Environment

Dr A. J. Hurst
Monash University
ajh@bruce.cs.monash.oz

Persistent programming denotes a philosophy of programming in which all program objects have the same data rights. In particular, one important data right of a program object is its lifetime. Most languages control lifetimes of data objects by declaration and scoping rules, but one important exception is the data object *file*. Whereas all other objects normally are destroyed when a program terminates, a file may outlive the program that creates it. Programmers have exploited this mechanism in order to preserve important data, but it necessitates a tedious process of converting from input representations to internal representations, and then back again for output. It has been estimated that this activity accounts for some 30% of program code.

Persistent languages accord all objects the same lifetime rights, so that no conversion from external to internal form is required. This mechanism is akin to virtual memory, where the transfer of objects between internal and external stores is managed automatically. However, persistent systems usually give the programmer more explicit control of such movement, without requiring explicit conversions. Data objects can then be said to be *persistent*, meaning that they have the right to outlive their parent environment, and this right is conferred independently of the data object's other attributes.

In order to implement persistent systems, a form of virtual memory is required. This may be done by means of capability based systems, such as has been developed at Monash University. If persistent systems are to succeed however, implementations on conventional architectures are also required. The paper develops the important point of how the underlying persistent mechanism reduces to the problem of mapping names to objects, describes the processes involved in implementing persistence, and shows how the UNIX file system can be used as a tool for achieving this. In addition, some experiments with using a graphical interface to both a persistent system, and to the UNIX file system can be used as a tool for achieving this. In addition, some experiments with using a graphical interface to both a persistent system, and to the UNIX file system, are described.

Cexp – An Expression Parser for C

Douglas Ray
Melbourne University
ray@murdu.mu.oz

A syntax-based text filter is discussed. A tool of similar functionality to grep (a regular-expression-based text filter), **cexp** parses C programs, recording arbitrary strings of syntactic elements.

A crude precursor is outlined, and then the complete version discussed in more detail. Algorithms, a developmental strategy, and applications are discussed, as well as the history of the project.

Bring your thinking caps – your opinions will be sought concerning appropriate facilities for inclusion, and the user interface.

Execution Driven Multiprocesser Simulation

Dr Rhys Francis
Concurrency Research Group
Department of Computer Science
La Trobe University, Bundoora, Australia 3083
rhys@latcs1.oz

The THREADS project models the interaction of application, system and architectural structures for shared memory multiprocessor by simulation. The simulator consists of a compiler, a library of machine modules, and kernel and run-time libraries. The heart of the system is actually the compiler. It takes a test application and builds a special purpose simulator to analyze the performance of that program. To do this, the compiler parses the program, analyzes and then divides the parse tree representation into a linear form comprising large numbers of small fragments, called slices, and then compiles the sequence of slices twice. The first compilation targets an abstract model processor and annotates each slice with the list of code and data accesses that would be emitted by the modeled CPU when executing that slice. The second compilation emits a C program from the annotated slices which interacts with the machine module in such a way as to measure the programs' execution on the modeled architecture and to update the simulator's state space to track memory and cache contents. The simulation proceeds by executing concurrent slices so that the precision and overhead varies with the grain of the slicing.

The final C program is linked with libraries containing a similarly compiled run-time system and kernel and the result is loaded into an image to give a simulator for the application running under a particular kernel and architecture. Architectural features such as the number of processors or memory modules, cache algorithms and cache sizes can be set by switches to that image. More radical differences in machine models can be constructed and compiler switches used to load the appropriate libraries into the simulation. The use of simulation allows any amount of performance data to be gathered and analyzed on the fly without disturbing the history of the execution.

Shared Data Access as a Metric for Distributed Algorithm Performance

A. N. Pears
Concurrency Research Group
Department of Computer Science
La Trobe University, Bundoora, Australia 3083
pears@latcs1.oz

Continuing increases in the computing power required for industry, and commercial applications is placing a heavy burden on the bus based machine model. Though increases in speed and power for these machines can still be achieved through the application of new technologies, it is apparent that the physical limitations of this architecture will soon be reached.

One model that offers both increased power and scalability is Distributed Computing. This model postulates a connected topology of processing nodes each with local memory and cache. It is vital in such a machine that message traffic throughout the network be minimised.

One aspect of algorithmic construction which can lead to high message traffic in such an environment, is shared access to commonly used data objects. Thus some measure of the suitability of programs for dynamically scheduled execution on such machines is the ratio of shared data accesses, particularly writes as they must be propagated to ensure memory and cache coherence.

This talk presents the preliminary results of an experiment conducted on three applications, namely; Matrix multiplication, Matrix Inversion, and a Simulated Annealing solution to the Traveling Salesman problem.

Access figures will be presented for varying sized problems, in each of the algorithm areas. Data access counts for Shared Read, Shared Write, and Cached Shared Read will be presented and commented on. These results will then be used to draw preliminary conclusions on the suitability of these programs for execution in a distributed environment.

Design and Implementation of a Parallel Unix Kernel

Lim O. Sim
Department of Computer Science
Monash University
sim@bruce.cs.monash.oz

In this talk, we describe the design and implementation of a fully parallel Unix kernel on the multiprocessor machine. This kernel is based on 4.3BSD with partial POSIX conformance. As it is built on top of a small capability-based OS, the services provided by the kernel of this OS, as well as the hardware itself will be discussed. A high-level view of this Unix kernel structure and the resource (e.g. inodes, process entries) allocation strategy used will be presented. Finally, problems encountered when building this kernel will also be discussed.

Characterizing Graphics Hardware & Software

Michael A. Gigante
Director, RMIT Advanced Computer Graphics Centre
mg@cidam.me.rmit.oz

Computer Graphics is many things to many people, resulting in an enormous range of compute and display requirements for what is generically called "Computer Graphics Workstations".

In this paper, I will characterize the various sub-domains of Computer Graphics and examine the necessary hardware characteristics to achieve various levels of performance.

I will also examine on the many graphics software systems that are now available and comment on 1) their applicability to each subdomain and 2) how well they map onto different hardware designs.

Design Of The Labtam Xengine†

David Purdue
davidp@labtam.oz

Tim Roper
timr@labtam.oz

Michael Podhorodecki
michael@labtam.oz

Graeme Gill
graeme@labtam.oz

Labtam Information Systems Pty Ltd
Braeside, Victoria, 3195

In most projects, software is designed to run on the hardware given. In ideal projects hardware and software are designed together. The Labtam Xengine is a hardware architecture designed to make use of existing software, namely the X Window System‡ Version 11 sample server. In this talk we discuss the architecture of the Xengine in its three existing implementations.

This architecture incorporates an Intel 80960KB RISC processor driving a dumb frame buffer. We will highlight features of this architecture and particularly of the 80960 that suit it to implementation of an optimised X11 server based on the sample server.

---

†Xengine is a trademark of Labtam Information Systems Pty Ltd.

‡ The X Window System is a trademark of the Massachusetts Institute of Technology.

Tasmanian Unix Conference Report

Monday, 5 February 1990

The Australian Unix Users Group (AUUG) and the University of Tasmania Computing Centre hosted a regional one-day technical conference at the University of Tasmania. An unexpectedly large turnout of 80 people attended the conference, with over half the participants coming from outside the University.

The conference provided an opportunity for Unix users in Tasmania to get to know each other and find out what others were doing. All conference participants received a newly compiled Tasmanian Unix User & Site Directory (20 pages), which provides details about each Unix site: names, phone and fax numbers, hardware configurations, applications software and network configuration.

Silicon Graphics and Sun Unix workstations were on display throughout the day.

Programme

Experiences with Macintosh A/UX
Nigel Williams, Desktop Power, Hobart. A/UX is Apple's version of Unix. Apple has attempted to hide some of the complexities of Unix, to make things simpler for naive users.

Macintosh Oracle with A/UX Oracle database server. *Gavin Orr, Desktop Power, Hobart.* Several applications have been developed, using Macintosh Hypercard/Oracle as a front-end and A/UX-Oracle as a database server.

Bringing Network Services to the Desktop. *Rod Bilson, Uni. Tasmania Computing Centre.* The University has integrated ASCII and IBM terminal networks with Ethernet and AppleTalk, allowing desktop access to all host computers.

Unix and DOS Integration
Ross Parish and David Strong, St. Audit Dept. The Audit Department uses a Prime EXL Unix machine as a server for client MS-DOS PCs, using the PC-Interface software package. The EXL runs both DOS and Unix with Merge386.

Unix File System Security
Tony Grainger, University of Tasmania Computing Centre. A tutorial covering process and Inode security issues, as well as clarification of levels of trust implicit in overall Unix security.

Using 2.3GB Exabyte tape with Unix systems. *Steve Clift, CSIRO Marine Lab.*Experiences in developing interface software to connect a 'bare-bones' Exabyte tape drive to a Labtam system.

New Features in SunOS 4.1
Roger Desalis, Sun Microsystems, Melbourne. SunOS 4.1 prepares for a forthcoming major release which will combine SunOS and AT&T's SystemV Release 4.

MUSBUS Benchmarking—What Has Been Learnt. *Ken McDonell, Pyramid Technology, Melbourne.* The MUSBUS benchmark is widely recognised throughout Australia and internationally as a multi-user Unix benchmark. A new version is now being developed to incorporate new features and address recognised inadequacies.

OLTP Performance -Behind the Smoke and Mirrors. *Ken McDonell, Pyramid Technology, Melbourne.* Competition is fierce as Unix vendors battle to establish reputations as serious, high-performance On-Line Transaction Processing (OLTP) platforms. Experience gained in benchmarking Unix database products highlights some of the pitfalls involved in OLTP benchmarks.

Unix Performance Analysis Tools
John Parry, Uni. Tasmania Central Science Lab. Unix provides only a few software tools which can be used to measure and tune system performance. Examples from the Sun workstation environment highlight suggestions about analysing and improving performance.

Procuring Unix Workstations
Steven Bittinger, Uni. Tasmania Computing Centre. What key issues should decision makers consider when making choices about purchasing workstations? What makes one workstation better than another? What trends seem likely to develop as a result of anticipated future changes in the workstation arena?

Acknowledgements

In order to minimise administrative efforts and encourage attendance, the conference was organised on a 'come as you please' basis, without registration or any fees. This approach was obviously successful. Conference participants and organisers greatly appreciated the generous sponsorship from:
- University of Tasmania Computing Centre
- Australian Unix User Group
- Australian Computer Society
- Sun Microsystems
- Silicon Graphics

—Steven Bittinger, Conference Organiser

Abstracts from AUUG - Summer '90 (WA)

AUUG Inc. and WAUG would like to thank Pyramid Technology Corporation for making available the keynote speaker, Ken McDonell, for the AUUG Summer 1990 Technical Meetings throughout Australia.

OLTP Performance - What's Behind The Smoke And Mirrors

Ken J. McDonell
Pyramid Technology Corporation
kenj@yarra.oz.au

As the speed and capacity of UNIX systems approaches that of conventional mainframes, the database vendors and the UNIX vendors are embroiled in a battle to establish reputations as serious, high performance OLTP platforms.

These efforts are characterised by:

1. Leap-frog performance as a consequence of database vendors engineering their products — a process that should have started many years ago! We shall consider issues such as recovery, server architectures, memory demands and scalability.
2. TP1 "sleaze wars" — we shal quickly review the TP1 Cheater's Guide, thereby convincing the reader that numbers quoted as "TP1 transactions per second" are of almost no value in predicting real OLTP performance. As a by-product, this discussion will lead to some suggestions for better engineered database benchmarks. Some attention will be given to the Transaction Processing Performance Council (TPC) and the emerging TPC-A Benchmark as a serious TP1 and Debit-Credit replacement.
3. Better marriages between the operating system and database management system. In particular, UNIX is being extended in ways that preserve SVID/POSIX conformance, but provide additional system services that database implementors need. Examples include fast mutual exclusion operations, asynchronous but guaranteed disk I/O, finer control over CPU scheduling policies, mirrored disks, special macine instructions, etc.

Throughout, examples will be drawn from experiances gained from using the major UNIX database products in benchmarking and performance analysis.

The presentation will conclude with some comments of future trends, in particular System V Release 4, product management issues in the database development and maintenance houses and the IEEE working group P1003.1 (Transaction Processing Application Environment Profile).

Session Management For ASCII Terminals

Steve Landers
Functional Software Pty Ltd
landers@wacsvax.cs.uwa.oz.au

The paper provides an overview of the following, as they relate to session management:

- System V virtual terminals (sxt's),
- Pseudo terminals (pty's),
- System V STREAMS and
- BSD job control.

If sufficient time, POSIX, and therefore System V.4 and BSD4.4, job control and System V.4 session management will also be addressed.

Spoilt For Choice - Or Are We?

Tim McGraith and Ian Crawford
tim@poh.fpa.oz.au, ian@pooh.fpa.oz.au

With the continuing increase in the market for UNIX based computer systems, the selection of the most appropriate hardware becomes more and more difficult.

How do you rationalise the masses of glossy brochures, technical specs, benchmarks, standards and emotional salesmen to arrive at a realistic comparison? Sometimes the effort is not worth the cost of the machine!

This paper presents a methodology for selecting the most suitable hardware for your UNIX applications, based on our experiences at the Fremantle Port Authority.

Techniques are offered that will assist you to -

- Recognise the need,
- develop the justification,
- design the solution,
- categorise requirements,
- draw up a purchase contract,
- negotiate a maintenance contract,
- carry out benchmarks,

- perform final evaluation and selection and
- manage installation and acceptance.

Whilst none of these ideas are new, it is their practical application to the diverse range of UNIX based systems that makes this a significant issue for UNIX system users.

Choosing An RDBMS - Do Your Homework!

Michael Selig
Functional Software Pty Ltd

This talk will provide a checklist for anyone choosing a Relational Database Management System running on UNIX-based hardware. Topics covered include:

- performance features,
- distributed databases,
- adherence to standards and
- benchmarking.

Armed with information you should be better able to cut through the retoric that most vendors bombard us with and make a more objective decision.

Public Key Privacy And Authentication

Greg Rose
Softway Pty Ltd
greg@softway.sw.oz.au

We have been working on and off for eight months on a public key privacy and authentication system based on the RSA (Rivest, Shamir and Adelman) system.

This talk will examine some of the interesting technical problems that have been addressed and solved as well as providing an overview of how the system is intended to work and integrate with networks.

One of the first problems to be solved was the selection of very large prime numbers without predictable characteristics. This involved high precision integer arithmetic, and some interesting numerical approaches. The issues to be addressed to take advantage of Softway's 6 CPU Sequent in this context are also interesting.

The RSA system is about to be adopted by the Internet; this talk will also examine how the Internet intends to address user validation questions and public key interchange.

Oh, What A Tangled Thread We Weave!

Chris McDonald and James Pinakis
Department of Computer Science
University of Western Australia
chris@budgie.cs.uwa.oz.au,
james@bison.cs.uwa.oz.au

Of emerging popularity in the UNIX community has been the use of *lightweight processes*, or *threads*, in user-level programs to simulate concurrency on uniprocessor architectures. Lightweight processes comprise multiple execution paths and contexts within the single address space of a "standard", or *heavyweight*, UNIX process. In many applications they are favoured over standard processes as they exhibit the properties of inexpensive creation, termination and context switching and permit fast, efficient interprocess communication using their single, shared, address space. However these advantages are not provided without some cost — their use in preference to standard processes precludes rigorous fault containment, normally provided by the operating system, and user-level code must assume the responsibility of coordinating access to shared resources. In particular, input and output and memory allocation must be restructured to be used within lightweight processes. This talk will provide an overview of lightweight processes, their theoretical grounding, implementation and the availability of libraries in the public domain. Our experiences in developing a lighweight process library at The University of Western Australia will also be discussed.

MUSBUS – What Has Been Learnt

Dr Ken J. McDonell
Pyramid Technology Corporation
Melbourne, Australia
kenj@yarra.oz.au

As the eighth anniversary of the initial development of MUSBUS approaches, it seems appropriate to contemplate the lessons that have been learnt and the likely future evolution of multi-user system performance analysis and benchmarking.

General conclusions include,

- The overwhelming lure of a single "figure of merit", despite all the evidence testifying to the irrationality of the approach.
- General laziness and/or illiteracy amongst consumers of benchmark reports.
- Good benchmarks break systems that are delivered through sloppy "QA" procedures.

- The importance of packaging, marketing and folk lore as a defence against poor experimental engineering.
- Scaling of tests across two orders of magnitude in performance is technically very difficult, and often simpler to ignore.

The intention is to highlight the pitfalls associated with predicting system performance, so that end-users may either adopt a more critical, cynical and analytical attitude to competing performance claims, or conduct their own well-engineered performance testing.

SECLOG

A Secure Remote Login Frontend and Shell[1]

Lawrence Brown

Department of Computer Science
University College, UNSW, Australian Defence Force Academy
Canberra ACT 2600. Australia.

Abstract

This paper discusses the development of **seclog**, a frontend for **rlogin**, and an associated shell for use on the remote system. These provide a secure remote terminal facility, using an automated challenge/response login procedure, session key exchange, and encryption of all data exchanged for the subsequent session. The encryption algorithm may be chosen from DES, LOKI, or a stream cipher; depending on the security/speed tradeoff desired. Initially **seclog** has been implemented using BSD pseudo-ttys. The reasons for their choice, advantages, and limitations, are discussed. A future version, to be incorporated into **telnet** and **telnetd**, rather than a shell around **rlogin**, is then proposed.

1. Introduction

With the growth of wide area networks linking together existing local area networks, there are increasing security problems with the existing network utilities traditionally supplied for such LANs. These utilities have assumed that the LAN was a secure communications service (even though it was recognised fairly early that it was not necessarily so), and that hence there was no need to provide additional security services for these utilities. However a wide-area internet certainly cannot be assumed to be secure[2]. Thus there is a desire to provide utilities with additional security. Some of the issues involved in providing such security are canvassed in [WoKo85 pp 189-198], [HeRo89] and [DaPr84].

This paper examines the remote terminal service provided by rlogin or telnet for TCP/IP systems. It considers the security implications of such services, and then describes an initial implementation of a utility to provide additional security for rlogin.

2. Login Over a Network - Some Issues

2.1. Identity of the User

When a user logs in to a host computer, they provide a username and password to identify themselves to the computer. On a hardwired line this is assumed to be secure, as it is difficult for someone to tap such a line and intercept the username password pair. However when logging in over a network, anyone with access to a workstation and packet monitoring software could intercept the pair, and then subsequently claim to be the original user (see Fig 1). In either case it is assumed that the password is well chosen, so that it cannot be guessed [MoTh79], [Spaf88 Appendix A].

What is required is that the information sent over the network not be sufficient to allow an intruder to impersonate the user, without additional knowledge. There are a number of ways of doing this.

---

[1] Presented to AUUG Summer 90 (Victoria) Monash University, Victoria, Australia 6th February 1990

[2] for a fascinating account of the 1986 Internet breakin, see [Stol88], and for details on other breakins see [Reid87], [Spaf88].

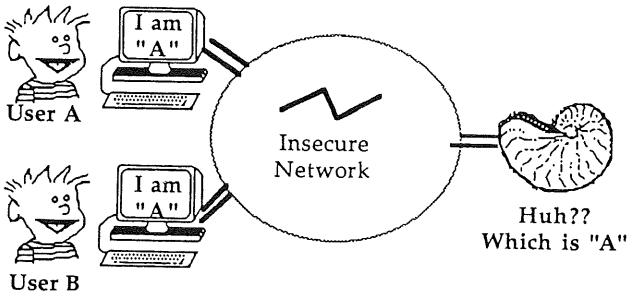


Fig 1 - Identity of User Uncertain

Challenge - Response

This involves the host computer sending a random number, a **challenge** vector, to the user. The user encrypts this using a **private** key, known to both the remote host and themselves, and sends this encrypted vector, the **response**, back to the host. The host encrypts the **challenge** vector using the user's **private** key, and compares it with the **response** returned. The user is accepted only if they are identical.

Since any intruder monitoring the network will see a different challenge and response each time, they will not be able to produce a correct response, unless they have obtained the user's key by some other means (which is a separate problem). This technique assumes that the user and remote host have already arranged a private key prior to wanting to establish a remote login session. Since, in this application, a user must have negotiated with the remote site to obtain a usercode and conventional password, obtaining a network private key simply becomes another piece of information exchanged during this negotiation. Thus in this case, I believe it is not a problem.

Digital Signature

This method uses a public-key authentication scheme [SePi88] to identify the user. To use it, the user signs an identity message with the private half of the key and sends this to the remote host. There it is verified by using the public half of the key to confirm that the identity message is unchanged. Since only the user knows the private half of the key, only they could have signed the identity message. There is still a problem however, of obtaining the correct public half of the key. If it is obtained from some remote database, that message could be faked instead to be that of the intruder. The problem is solved by providing the public key signed by a public key centre (or possibly a hierarchy of signed keys), whose public key is well known. This can be supplied by the user as part of the identity message which removes the need for a network key server to be accessed on every remote terminal session request. Whilst this method has a number of nice properties, including no need for prior key exchange, it is computationally slow. However if a public key scheme is to be used for encryption, then this would be the obvious counterpart.

Zero Knowledge Proof of Identity

This technique provides a way of authenticating a user, by using a series of challenges and responses to convince the remote site that the user is who they say they are. However it does not require the prior arrangement of a key. In this method, the user sends a certificate (Ident,j) originally obtained from a public key centre, to the remote host. The host then interacts with user to prove ownership of the certificate [FeFS88], [SePi88]. This method is much faster than a digital signature, though it involves a dialog between the user and the host. In the longer term, it will probably be the method of choice.

2.2. Privacy of Session

Having authenticated the user, the next issue to address is the provision of privacy for the user (see Fig 2).

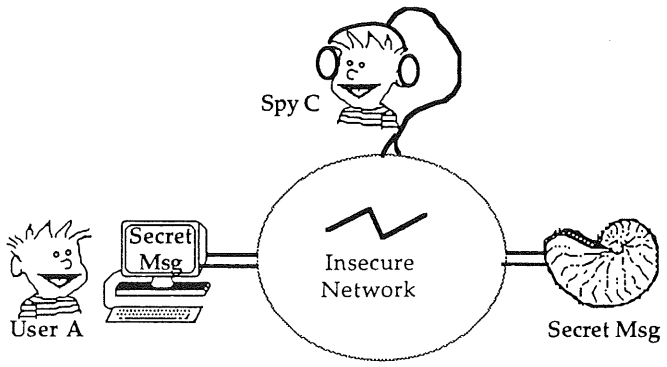


Fig 2 - Privacy of Session

This implies a need to encrypt the data transferred during the session. However, a choice has to be made as to which form of cipher is to be used to provide privacy. A Public-Key scheme has the nice property that the keys need not be exchanged prior to establishing a session, but current implementations are too slow. The alternative is to use a private-key scheme. These are considerably faster, but need the keys to have been exchanged in some secure fashion, either prior to the session, or via a public key exchange scheme. If a **challenge-response** scheme is being used, then a suitable key may be derived, for example, by encrypting the **response vector** to form the session key.

Having decided to use a private key scheme, there is still a range of ciphers to chose from, trading off speed for security. The choices include:

Simple Substitution Cipher

This is very fast, very simple to implement, but is very insecure. Guessing as many characters as there are in the key is sufficient to crack the scheme. However, if all that is required is to defeat casual scanning (or an automatic word scanner), then it may well suffice.

Stream Cipher

This is usually based on a pseudo-random number generator. They are reasonably fast, and given a suitable choice of generator (such as a non-linear feedback shift register, or multiplexed linear feedback shift registers), these schemes have a moderate to high level of security. They can be cracked by guessing a sufficiently large number of characters, but the number needed could be very large. Another possible precaution is to change keys before this number of characters is sent (using for example the challenge-response key again, and encrypting the previous session key again every period).

Block Cipher

The cipher would be used in a stream mode. Any of the modern block ciphers may be used, eg DES [ASA85], LOKI [BrPS90], FEAL8 [ShMi88] [Miya89]. These schemes all have a very high level of security, but are probably the slowest of the alternatives. Suitable modes include either Cipher Feedback (CFB), or Output Feedback (OFB) modes (see Fig 3). If the CFB mode is chosen, it uses the message contents in forming the new key (which can slow down parallel key stream generation), but which leads to different key streams for different messages. If the OFB mode is used, the key stream generated is independent of the message content (and thus the key stream can be precomputed). However OFB should not be used to encipher a number of different messages with the same keystream, as this creates a security hole. If different keys are used though, it is equivalent in security to CFB, with the advantage

of being able to precompute the key if desired.

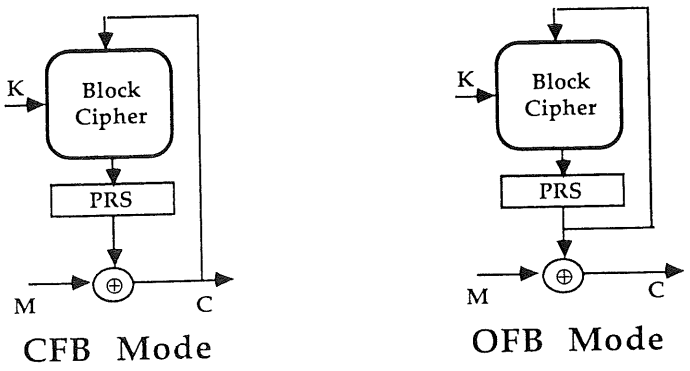


Fig 3 - Stream Modes for Block Ciphers

3. Where to Add Security?

Since there are a number of existing protocols for remote terminal sessions, eg rlogin or telnet, it would be preferable to adapt these to provide the security features detailed earlier, rather than write a new protocol. The issue is how to adapt these existing tools. There are two major options, as shown in Fig 4.

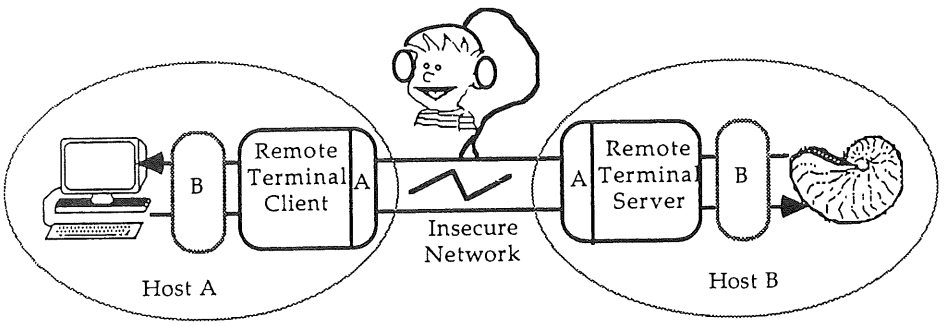


Fig 4 - Adding a Security Utility

(A) Integrated into the Utilities

Here, the programs are modified to include the required security features. It has the advantage of being able to properly merge the extensions with the existing protocols. However, access is required to the source to achieve this on all systems where it is required. In the longer term, it is the option of choice.

(B) Using Addon Processes

Here, some filters are added, in traditional Unix fashion, around the existing utilities. This is much easier to do, since there are no changes to the existing tools, but there are several disadvantages. It adds more processes and thus context switches, it blocks user interaction with the client program using escapes, and it requires a transparent (8-bit) path. Nonetheless, this was the option chosen for the initial implementation.

4. SECLOG - A Secure Remote Login Implementation

4.1. Overview

SECLOG is an initial implementation of some utilities to provide a basic secure login service. Structurally, it uses extra processes around rlogin (option B). Rlogin rather than telnet, was chosen as the basic remote terminal service because it provides a transparent (8-bit) mode (using the undocumented -8 flag). This made implementation a little easier. I intend to modify it to handle telnet in the near future, though this may involve sending every encrypted character as two hex characters, doubling the size of each message. SECLOG performs the following operations:

automatic login to a secure userid
challenge/response to verify user
negotiates encryption cipher to use
perform encryption of session data

It does not need to run as a a privileged (setuid) program, though it does require a small privileged utility to cleanup utmp/wtmp and reset the terminal permissions. It assumes that a network key (here it is an 8-char DES key) has been arranged previously (ie when the usercode is created).

The SECLOG processes are structured as shown in Fig 5. The client program **seclog** is run by the user to establish a secure remote terminal session. It then runs rlogin, which establishes a link to rlogind on the remote host. The **seclog** client then logs into the secure user on the remote system, which in turns runs the **seclog** server. These then negotiate the challenge/response and cipher to use, and if successful, the server runs login to actually login the user and start their shell. For the remainder of the session, the client and server encrypt all information being transferred between them.

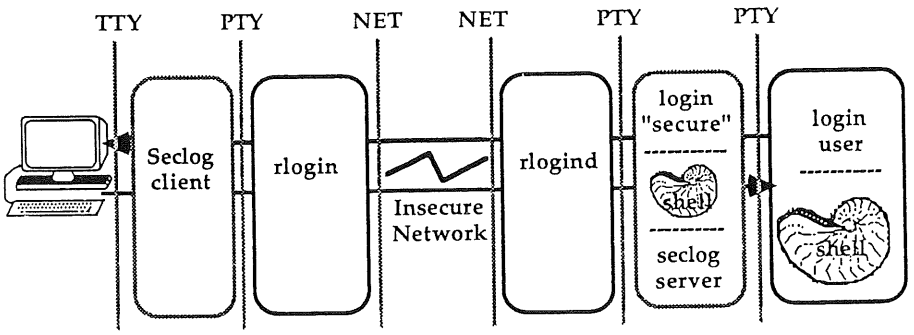


Fig 5 - SECLOG Processes

In more detail, the flow of control between the client and server, detailing the sequence of messages exchanged, is shown in Fig 6:

Client	Server
initialize	
grab pty pair	
fork & exec "rlogin" via pty	
lookup user key	
do remote login to "secure"	(rlogind execs server)
	initialize
	grab pty pair
	issue challenge
get challenge	lookup user key
calc & send response	get & check response
	prompt for scramble alg
accept suitable scramble alg	& init accepted alg
& init selected alg	fork & exec "login user" via pty
	prompt scramble start
accept scramble start	
copy & cipher session data	copy & cipher session data

Fig 6 - SECLOG Control Flow

4.2. SECLOG - Private Key File

As mentioned previously, keys must be arranged prior to establishing a secure remote login. It is assumed that these keys will be distributed at the time the account is created, along with the conventional password.

The SECLOG keys are stored in a file **rkeys**, which must NOT be readable by anyone other than by the **seclog** server and the system administrator. This can be arranged by making a special group to which this file belongs, and for which the systems administrator is a member. The SECLOG server program should then be setgid to this group.

The format of the **rkeys** file is as shown in Fig 7 below:

```
# format for rkeys file for seclog
# login_name   des_key_in_hex
lpb 0123456789abcdef
bvd 6789abcdef012345
gco 456789abcdef0123
```

Fig 7 - **rkeys** file format

4.3. Typical Session Script

A script for a typical secure remote login session, showing the interaction between the client and server is shown in Fig 8 (nb: italic text is entered by the user, bold text is returned from the remote system, plain text is from the local system).

Script	Comments
cssun2_11 *%seclog cssun0*	run program
seclog started, remote host cssun0, user lpb	
seclog got slave /dev/ttyp2 on file id 4	
Password:stelnet	rlogin "secure"
Last login: Wed Jan 24 15:55:28 from cssun2.cs.adfa.o	
Secure login:lpb	name user
Challenge: 5858585825bd35ae	does challenge
Response:11d4efd7d5554e83	gets response
Scramble Mode: DES : y	negotiate cipher
Remote User Validated	
Scramble Start:	start scrambling data
Password: *mypassword*	conventional login
Last login: Wed Jan 24 15:55:43 on ttyp1	nb all info scrambled
tenowatch: Maximum continuous work period 60 minutes	
work : rest ratio 1:1	
Terminal type is vt100 on ttyp1	
cssun0_11 *%who*	show who is on
secure ttyp0 Jan 24 15:57 (cssun2.cs.adfa.o)	seclogd user
lpb ttyp1 Jan 24 15:57	real user
wku ttyp7 Jan 24 14:10 (csadfa.cs.adfa.o)	someone else
cssun0_12 *%logout*	request logout
Connection closed.	
seclog done	seclog client exits
cssun2_12 %	

Fig 8 - Typical SECLOG Session Script

4.4. SECLOG - Problems Observed

Whilst writing and debugging **seclog**, a number of problems were found, including:

- The handling of prompts and replies during login. It was very easy to get out of sync between the client and server during the automated login. In particular, care is needed in the ordering of patterns matched (eg "Login" matched both the wanted "Login:" prompt, but also the "Last Login at ..." banner).
- Sync of encrypted data during session. This was a major problem. Apart from ensuring that the cipher algorithms themselves stay in sync, a more serious problem was the use of stderr by programs inside the encrypted data path, especially messages from **rlogin** and the **seclog** server. This was partly handled by bypassing rlogin's stderr round the client, but error messages from **seclogd** or **rlogind** had to be very carefully avoided, or suitable ciphered.
- Clearing out of utmp/wtmp entries, and resetting the terminal permissions for logged in users. Traditionally this is handled by one of **rlogind**, **telnetd**, or **init**. However, in the arrangement used here we have the following:

  tty1: "secure" utmp created by rlogind, utmp cleared by rlogind
  tty2: "user" utmp created by login, utmp cleared by ???

  This required writing a small setuid root program "fixutmp". This takes as an argument a tty device, and it clears the associated utmp and wtmp entries for it, and resets it permissions. It uses the standard Berkeley routines (as used in rlogind) for doing this. Obviously this program should not be available to normal users. It should also belong to the special group used to protect the rkeys file.

5. Future Versions

Having gained some experience with SECLOG as it is, I have some indicators for how it is to evolve in future. Most importantly, in a major rewrite, it should be integrated into the telnet and telnetd programs. This will bypass many of the problems found in the current implementation, due to the much closer merging of the features with the existing protocols. Despite requiring source access, I feel this is a needed improvement to the current scheme.

If this is done, I would like to get the security extensions approved, especially for the initial challenge/response and cipher negotiation protocol exchanges. This will greatly improve the chance of getting an agreed means of implementing these extensions.

Other option improvements include the use of a streams module for encryption of the session data, rather than having it buried in the telnet code; the use of a zero knowledge proof of identity rather than challenge/response; and public key exchange of session the key. These latter two are probably less important in this particular application, than in say electronic mail, because of the need to have established an account on the remote host prior to establishing communications with it.

It is likely that a private key cipher will continue to be used for privacy, since a massive speed improvement of the public key schemes appears unlikely.

6. Conclusions

In this paper I have detailed a trial of a secure remote login program. It provides authentication of the user and privacy of the session. It does require the use of prearranged keys, and suffers from fairly high additional resource usage due to its structure. Nonetheless it has shown that the problem can be solved, and has pointed to a future version which should overcome some of these problems.

Acknowledgements

To Mike and Cathy Newberry, Andrzej Goscinski, Chris Vance and Geoff Collins, for their help and suggestions. Thankyou.

This work is supported in part by ARC grant A48830241.

Bibliography

[ASA85] ASA, "*Electronics Funds Transfer - Requirements for Interfaces, Part 5, Data Encryption Algorithm*," AS2805.5-1985, Standards Association of Australia, Sydney, Australia, 1985.

[BrPS90] L. Brown, J. Pieprzyk and J. Seberry, "LOKI - A Cryptographic Primitive for Authentication and Secrecy Applications," in *Auscrypt'90 Abstracts - A Workshop on the Theory and Application of Cryptographic Techniques*, pp. 166-167, ADFA, Sydney, Australia, Jan. 1990. also issued as Tech. Rep. CS90/1 by Dept Computer Science.

[DaPr84] D. W. Davies and W. L. Price, *Security for Computer Networks*, John Wiley and Sons, New York, 1984.

[FeFS88] U. Feige, A. Fiat and A. Shamir, "Zero-knowledge proofs of identity," *Journal of Cryptology*, vol. 1, no. 2, pp. 77-94, 1988.

[HeRo89] M. Henning and A. Rohde, *UNIX File System Security*, CCSR Tutorial Series in Computer Security, no. 1, Centre for Computer Security Research, Dept Computer Science, UC ADFA, Canberra, ACT, Australia, 1989.

[Miya89] S. Miyaguchi, "The FEAL-8 Cryptosystem and a Call for Attack," in *Crypto '89 Abstracts*, p. Rump, IACR, UCSB, Aug. 20-24, 1989.

[MoTh79] R. Morris and K. Thompson, "Password Security: A Case History," *Comm. of the ACM*, vol. 22, no. 11, pp. 594-597, Nov. 1979.

[Reid87] B. Reid, "Reflections on Some Recent Widespread Computer Break-Ins," *Comm. of the ACM*, vol. 30, no. 2, pp. 103-105, Feb. 1987.

[SePi88] J. Seberry and J. Pieprzyk, *Cryptography: An Introduction to Computer Security*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[ShMi88] A. Shimizu and S. Miyaguchi, "Fast Data Encipherment Algorithm FEAL," *Eurocrypt 87 Abstracts* , pp. 11-14, 1988.

[Spaf88] E. H. Spafford, "*The Internet Worm Program: An Analysis*," Technical Report Computer Science Dpt.-Tech. Rep.-823, Dpt. Computer Sciences, Purdue University, West Lafayette, IN 47907-2004, USA, 8 Dec. 1988.

[Stol88] C. Stoll, "Stalking the Wily Hacker," *Comm. of the ACM*, vol. 31, no. 5, pp. 484-497, May 1988.

[WoKo85] P. H. Wood and S. G. Kochan, *UNIX System Security*, Hayden Book Company, 1985.

MUSBUS – What Has Been Learnt?

Ken J. McDonell
Pyramid Technology Corporation
Melbourne, Australia
kenj@yarra.oz.au

Abstract

As the eighth anniversary of the initial development of MUSBUS approaches, it seems appropriate to contemplate the lessons that have been learnt and the likely future evolution of multi-user system performance analysis and benchmarking.

General conclusions include,

- The overwhelming lure of a single ''figure of merit'', despite all the evidence testifying to the irrationality of the approach.
- General laziness and/or illiteracy amongst consumers of benchmark reports.
- Good benchmarks break systems that are delivered through sloppy ''QA'' procedures.
- Workload characterization is perceived to be a difficult task.
- Scaling of tests across two orders of magnitude in performance is technically very difficult, and often simpler to ignore.

The intention is to highlight the pitfalls associated with predicting system performance, so that end-users may either adopt a more critical, cynical and analytical attitude to competing performance claims, or conduct their own well-engineered performance testing.

1. Introduction

The Monash University Software for Benchmarking UNIX* Systems (MUSBUS) was developed in 1980 to assist in equipment acquisition decisions for the Computer Science Department at Monash University.

An overview of the architecture of the multi-user component of MUSBUS is shown in Figure 1.

---

* UNIX is a trademark of AT&T

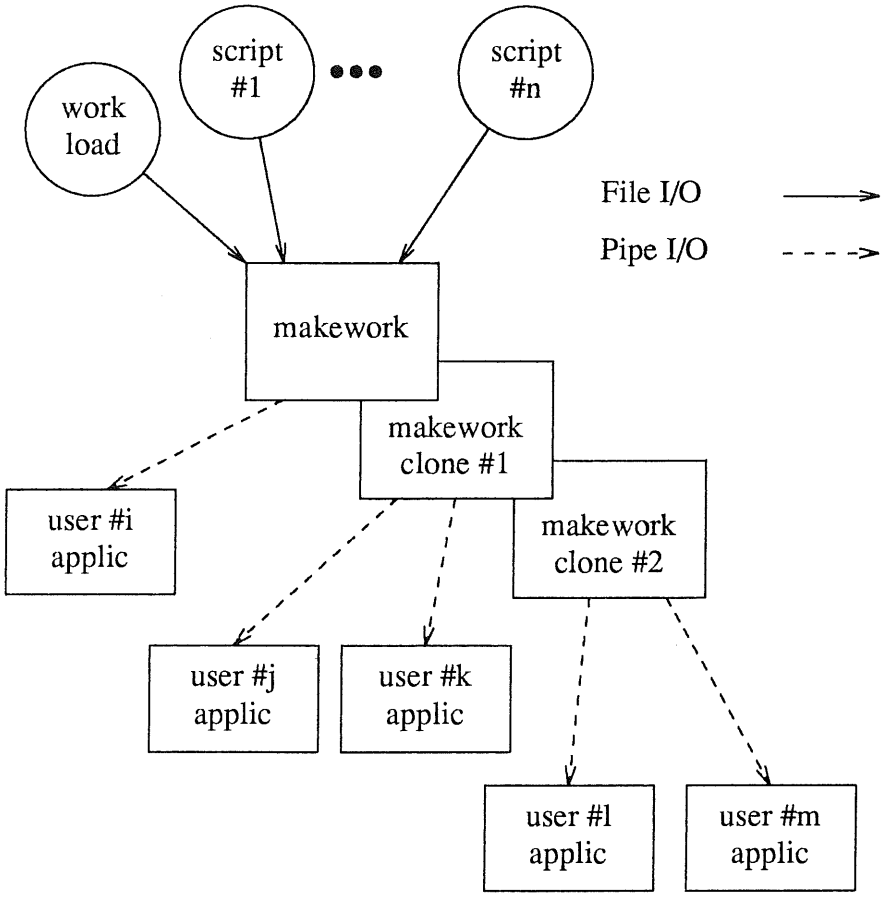


Figure 1: MUSBUS multi-user test architecture

The aim was to provide some plausible comparison of multi-user performance amongst competitive systems, and the suite of tests in MUSBUS evolved from a realization that no adequate commercially available benchmarks or performance profiles existed. At that time, all existing performance metrics suffered from at least one of the following problems.

1. The benchmark code was proprietary, leaving no way to verify that the implementation of the tests matched the description of what the tests were measuring.
2. Workload characterization was either not available (usually on the weak assumption that integer arithmetic speed of the processor was correlated with system performance), or involved assigning relative weights to load dimensions that had unknown relevance to our anticipated workload (e.g. what are the percentages of ''disk I/O'', ''memory access'' and ''integer computation''?).
3. The benchmarks were poorly engineered, with results being influenced by

a. statistical instability,

b. poor software engineering in which workload failures were not detected,

c. glaring lack of knowledge concerning common C coding practice and UNIX internals, and

d. sensitivity to factors that were unrelated to the aims of the benchmark developer (e.g. an "I/O Benchmark" that does **no** I/O due to file system buffer cacheing).

Unfortunately, most of these criticisms are still valid for much of the benchmarking and performance analysis that is being conducted eight years later.

2. The Lure of the Single Figure of Merit

Benchmarks such as Whetstone, Dhrystone, Dhampstone, Linpack, IOCALL, xfroot, "count to a million", "square root of 2 to 99 decimal places", etc. have gained considerable currency. This popularity is due to,

a. software availability,

b. software portability (less so in the case of Linpack),

c. ease of installation and execution, and

d. the test produces a single measure, or "figure of merit" that allows a one dimensional comparison with other systems.

The appeal of a single figure of merit is that **relative** system performance is reduce to a simple ratio.

Such a simplification ignores the multi-dimensional nature of system behaviour. More generally a system's performance (as judged by the end-user community) is a mixture of response-time and throughput during periods of dramatically different system loading. The variations in system load may be attributed to changes in the number of concurrent users, differing throughput demands, mix of batch and on-line processing, and the current time of day, or day in the business' financial cycle.

In this N-dimensional space, system performance is a complex "surface" with peaks (excellent performance) and troughs ("running like a snail on Valium"). In this context, the selection of a single representative point seems silly – but this is **exactly** what a single figure of merit is.

The simple single figure of merit tests are also based on the execution of one application, which makes them susceptible to benchmark-specific tuning or architectural anomalies (hardware and software) that produce uneven relative performance across a wider range of applications.

In objective terms, these simple measures can only be used as a crude filter to eliminate grossly inappropriate systems from consideration, e.g. to help you **not** consider a Cray when a 286 PC would do the job. In this respect, the tests are no more useful than list purchase price, system mass, power consumption or the much maligned MIPS rating of the vendor.

From the outset, MUSBUS discouraged this type of single dimension analysis by reporting several performance metrics, **none** of which could honestly be used as a single figure of merit.

3. Conjuring for Fun and Profit

Many commonly accepted benchmarks and performance metrics can be ''tricked'' into providing artificially high performance using techniques that are either technically or operationally infeasible in a production environment. The following examples are typical of a wider range of techniques that have been exposed.

1. Special compiler options, usually to permit optimizations that will not work in the general case, but are ''safe'' for a known benchmark, e.g. one that does not use pointers in C, or operates on strings of a fixed length. The best known examples appear in the context of the Dhrystone benchmark.
2. The ''Claytons'' system call, in which the *libc* interface to a kernel system call may ''fake'' the system call entirely in user mode, e.g. all calls to getpid() after the first call and up until a fork() occurs.
3. Very small program size, so that a large number of concurrent executions have very meagre memory demands, thereby allowing a file system buffer cache of 80% of the available memory.
4. Small programs that execute entirely from the hardware caches and avoid all memory references.
5. Badly engineered synthetic programs that perform long computations and do not use the result – current compilers may well delete the useless code fragment as part of their default optimization.
6. Inadequate control over the environment in which the tests are run, e.g. allowing different levels of compiler optimization or the maths library routines to be ''in-lined''.
7. **Pretend** to implement O_SYNC for file writes, and risk exposure to irrecoverable database corruption for the sake of fast transaction processing performance.

The best defence against this type of conjuring is to incorporate **real** applications in the testing, or to be very careful when crafting synthetic benchmark programs. MUSBUS opts for the former strategy in the multi-user test.

When the tests are run, it is also important to ensure that the environment in which they were run is fully documented.

4. It is Easier to Believe Than to Think

The following questions seem obvious, but asking them seems beyond many people embarking upon a performance analysis exercise.

1. Does this test really measure what it claims to measure?

2. Is the test relevant to the perceived workload, and if so how?
3. Is the reported number statistically reliable, and what is the magnitude of the expected error?
4. Will I be told the relative resource consumption (processor utilization, memory requirements, disk bandwidth) required to achieve the quoted performance?
5. Is the hardware and software configuration completely documented?
6. Is the system configured in a manner that is consistent with the required operational environment? For example, size of the file system buffer cache, enabling database transaction logging, realistic database to page cache size ratio.
7. What procedure will be adopted to resolve the contradictory results that will most likely arise if more than one test or metric is considered? For example, if Systems A and B are under consideration, and System A is faster on Test X but slower on Test Y, which system offers the better performance?
8. Are the results guaranteed to be reproducible in a production system installed at the customer site?

Less blind faith and more investment in thought about these types of issues would dramatically reduce the frustration, confusion, necessity for re-runs and recourse to irrational grounds for decision-making that surrounds much pre-sales performance analysis activity.

5. Benchmarking as a Quality Assurance Exercise

One of the interesting outcomes of the whole MUSBUS evolution is the observation that, especially early in a product's life cycle, many vendors follow QA procedures that do not adequately expose production systems to the stresses of realistic multi-user workloads.

This has been reinforced so many times that end-users are demanding MUSBUS results not only as a performance metric, but also as a demonstration of system robustness.

Internally, several vendors have incorporated MUSBUS or similar tests into their QA suites, and suspicion for a failed MUSBUS run is now initially directed towards the hardware or the UNIX port, rather than assuming MUSBUS is broken. In the early 80's just the opposite was the norm.

6. Workload Characterization

Initially MUSBUS was developed on the premise that accurate predictions would only come from measuring system performance under a multi-user workload that was a reasonable model of the anticipated processing profile.

The easiest way to achieve this accuracy was to build the workload using the same applications the users were using (the one change was to use tougher *ed* scripts to compensate for the fact that the editor of choice, *vi*, cannot be run from a script in any portable manner). The users were polled to obtain representative data files and edit session scripts, and the UNIX shell accounting records from the old system were used to identify and determine relative execution frequencies for those programs making the

largest contribution to the resource consumption.

The default MUSBUS workload is this same characterization, and it has been interesting to see how many organizations have been investing effort in running tests and gathering data whose predictive accuracy is limited to performance relative to the anticipated 1982 requirements for the Computer Science Department at Monash University.

Of course, the intention with MUSBUS from the outset was that the workload should be an **input parameter**, and the benchmark was engineered to make inclusion of new workloads as painless as possible. However the vast majority of MUSBUS users have ignored or avoided the facility and opted for the default workload. What is particularly depressing is that the alternative Work_text workload supplied in Version 5.2 simulates a full-screen editor, text processing and office automation like environment, and would be more appropriate for many commercial MUSBUS users, but to date I know of no organization that has used this alternative.

Throughout, I have consistently refused to publish tables of results (the workload is but one issue here), and argued that MUSBUS is a standard multi-user benchmark **environment**, not a standard benchmark. In all the cases I am aware of, when the effort has been invested in tailoring application-specific workloads to be run within the framework offered by MUSBUS, the results have been most satisfactory in respect of predictive accuracy and realistic comparisons between systems.

7. Performance as a Function of Increasing Load

One of the important design decisions in MUSBUS was to incorporate a parameterized user typing rate, and thereby constrain throughput at low levels of concurrency.

Compared to benchmark architectures in which the simulated user input is delivered to the applications at pipe or file system bandwidth rates, the MUSBUS approach has produced better predictive accuracy, because applications have longer residency and so the behaviour of the memory management subsystem, the hardware caches and the scheduler is much more realistic.

The addition of ''per user'' directory contexts, automatic script permutation tools, and automatic distribution of I/O across physical devices allows the demand for system resources per user to remain constant as the number of concurrent simulated users is increased. This allows the tabulation of both CPU utilization and elapsed time degradation as a function of increasing load. These derived metrics provide valuable information about concurrency level at saturation, which is useful in both sizing and comparing systems.

8. MUSBUS Strengths

The aspects of MUSBUS that are important and useful as criteria against which other multi-user benchmarks may be technically assessed, are as follows.

1. Benchmark software that is both portable and publicly available for scrutiny.
2. A software engineering philosophy in which all possible error conditions are checked and an error triggers a ''bells and whistles'' abort of the benchmark

(including killing off all other concurrent activity).

3. Statistically stable metrics, with repeated measurements and reported means and variances.
4. Workload as an input parameter.
5. Real-time delays in user input (via a parameterized simulated typing rate).
6. Terminal output being generated and sent to real physical devices.
7. Permutation of the workload to avoid ''lock-step'' synchrony between concurrent users.
8. A ''per user'' directory context to provide script invariance and avoid false file sharing.
9. Automatic distribution of user files and activity across multiple disk devices.

9. MUSBUS Weaknesses

The basic MUSBUS architecture has several weaknesses that either warrant special care when interpreting the results, or should be addressed in the evolution of multi-user benchmarks of this nature.

1. Self-adaptive scaling of the test duration and level of concurrency. MUSBUS has been used in systems with a raw processing power that differs by at least two orders of magnitude. The spread of UNIX both up and down the price-performance spectrum has placed MUSBUS in environments very different from the ones in which it was originally conceived and used. On very fast machines, the tests complete so fast that the times are not statistically stable, on very slow machines the hapless user may qualify for the pension before the test completes. In the multi-user test, the levels of concurrent activity are specified as an input parameter, whereas one would prefer the selection of the range of values for the number of concurrent simulated users to be made automatically based on the capacity of the system under test.
2. Lack of send-receive synchronization. The generation of simulated user input is constrained by the nominated typing rate. This works perfectly adequately on a lightly loaded system, but as the concurrent load increases, the user input may be submitted before the application is ready for it. This ''typeahead'' behaviour reduces the realism, because in real systems the user's typing rate actually drops under heavy load (more accurately, then think time between interactions is extended).

   There is another unfortunate side-effect, namely the benchmark architecture collapses altogether under heavy load when a customized workload has been developed for applications that periodically ''flush'' their typeahead buffer (e.g. screen-based applications built on top of *libcurses*).

   The lack of send-receive synchronization also means that no measurement of response time is possible.

Addressing these issues requires a much more sophisticated approach, as is evident in Pyramid's Remote Terminal Emulation (RTE) package *sscript* that represents a top-of-the-range extension of the MUSBUS philosophy.

3. Think time. Beyond the inter-keystroke delays from a constrained simulated typing rate, there is no mechanism within the MUSBUS architecture to include "user think" time in the workload scripts. Such a facility would allow more realistic modelling of the delays normally encountered between high level interactions (e.g. commands, transactions or applications).

   Additional delays would also enable simulated users to generate a volume of work per unit time that is closer to the expected productivity of real users, rather than the present situation in which one MUSBUS simulated user may represent the throughput of between 2 and 10 actual users.

   Although think times can be included in MUSBUS scripts as invocations of the shell function (or program) *sleep*, it is not very satisfactory because this often requires a "shell escape" (additional processes plus the */bin/sh* startup), and the delay must usually be a constant.

4. Logical contention. Concurrent MUSBUS users compete for system resources (CPUs, memory, disk bandwidth, etc.), but there is no mechanism for contention over logical resources such as locks or a database transaction log.

   Provision for this sort of contention can be made within a specific application environment, and hence a workload profile, but this is very difficult to do in a portable manner.

5. Benchmark overheads. When measuring system throughput, it must be remembered that in addition to the applications being run, the system must support the MUSBUS benchmark code. Typically this translates into a something of the order of 10%, due mostly to the additional processes that are required to generate the simulated user input.

6. Elapsed time versus mean elapsed time. The definition of the elapsed time for the multi-user benchmark covers the period from the launching of the first user until the last simulated user is finished. A statistically more stable metric would be the mean elapsed time across all of the concurrent simulated user sessions in a given experiment.

7. De-emphasize the raw speed tests. The raw speed tests purport to measure arithmetic speed, overheads associated with various system activities, file system throughput, etc. The intent was that these tests are purely diagnostic, and less effort was taken with their engineering. Consequently, the tests are susceptible to architecture and configuration changes (as described in Section 3), and this make them ill-suited for comparisons between heterogeneous systems.

   The **only** metric intended for comparison purposes is the system performance in the multi-user portion of MUSBUS.

10. MUSBUS Version 6.1

A forthcoming release of MUSBUS will preserve the basic architecture, but drop many of the raw speed tests from the default test suite.

An attempt will be made to make the testing self-scaling with result normalization occurring in the reported results so that comparisons may still be made between systems.

Think times drawn from uniform probability distributions will be supported at arbitrary points within the workload scripts.

The default workload for the multi-user test may be dropped entirely, or replaced by a large synthetic program that implements a parameter-driven model, better suited to applications in a commercial, and especially a database, environment. The following *yacc* grammar gives an outline of the application specification language that controls the behaviour of this application simulation program.

```
spec     : stmt
         | stmt spec
         ;

stmt     : TRANSACTION name optlist ';'
         | DATABASE name int RECORD int BYTE PER RECORD ';'
         ;

optlist  :
         | option optlist
         ;

option   : range READ conflict_opt
         | range WRITE conflict_opt
         | range SCREEN REFRESH delay_opt
         | range CHAR PER REFRESH
         | THINK TIME range SECOND
         ;

conflict_opt     :
                 | WITH LOCK CONFLICT number delay_opt
                 ;

delay_opt        :
                 | DELAY range SECOND
                 ;

range    : number
         | UNIFORM '(' number ',' number ')'
         | NEGEXP '(' number ')'
         ;
```

The application is modelled as a set of transactions against an arbitrarily sized file (the "database") of fixed length records. Each transaction consists of a number of random record reads, a number of random record writes, a number of screens to be displayed and some think time. All "numbers" may be constants, else drawn from either a uniform or a negative exponential probability distribution. For each record read or written there may be a possible lock conflict and attendant delay – the underlying mathematical model of lock conflict ensures that the probability of conflict actually occurring increases with the number of concurrent simulated users.

The hope is that this parameterization of the workload may prove to be more attractive, because the quantification of the parameters can be based upon the sizing calculations which would normally have been performed as part of the systems analysis, database design and specification phases of the application's development.

All "database" I/O is implemented via "backend" processes connected to the application simulation program via pipes.

The multi-user workload consists of multiple invocations of the application simulation program, each with its own transaction profile to be run.

11. Conclusions

Developing benchmarks that are capable of producing results with a high degree of predictive accuracy is a very difficult task. Using such a benchmark, and interpreting the results is also a skilled activity.

Improving the general quality of performance analysis requires a new generation of "standard" benchmarks that are well-engineered and in the public domain (note in particular the efforts of the Transaction Processing Performance Council (TPC), and the Systems Performance Evaluation Consortium (SPEC) in this regard). Further data processing management must be willing to invest more heavily in the specialized skills required to improve the predictive accuracy of performance analysis studies.

Design Of The Labtam Xengine[1]

David Purdue
davidp@labtam.oz

Tim Roper
timr@labtam.oz

Michael Podhorodecki
michael@labtam.oz

Graeme Gill
graeme@labtam.oz

Labtam Information Systems Pty Ltd
Braeside, Victoria, 3195

ABSTRACT

In most projects, software is designed to run on the hardware given. In ideal projects hardware and software are designed together. The Labtam Xengine is a hardware architecture designed to make use of existing software, namely the X Window System[2] Version 11 sample server. In this paper we discuss the architecture of the Xengine in its three existing implementations.

This architecture incorporates an Intel 80960KB RISC processor driving a dumb frame-buffer. We will highlight features of this architecture and particularly of the 80960 that suit it to implementation of an optimised X11 server based on the sample server.

1. Introduction

This paper describes the design of the Labtam Xengine, an architecture for an X Window System terminal constructed around the Intel 80960KB embedded processor and a dumb frame-buffer.

The architecture is a particularly simple one, and we will discuss how this simplicity gave us advantages in terms of a fast design track, a cheaper final product and a faster X terminal. We will also examine why the 80960 was a good choice of CPU for this project.

2. Overview Of X11

In order to understand the design of an X terminal it is necessary to have some understanding of the X Window System.

The X Window System was created with three goals in mind [NYE88]. First, it was to provide windows on bitmapped terminals in a portable manner while still allowing high performance. Second, it was to allow different types of machines to cooperate within a network. And third, it was to provide mechanism without dictating policy. To achieve these goals the designers of X created a protocol for communication between a server that provides display functions and clients that use these functions via windows.

[1]Xengine is a trademark of Labtam Information Systems Pty Ltd.
[2]The X Window System is a trademark of the Massachusetts Institute of Technology.

X works on a client/server model. Here it is important to realise that the server provides user interface services, so the server resides in the workstation on your desk, rather than in the large box next door that is often thought of as the file- or compute-server. (See Figure 1.) The services provided are graphics output onto the screen, and user input via keyboard and mouse. The server is responsible for separating output from different clients into appropriate windows, and for sending user input to the clients that expect it.

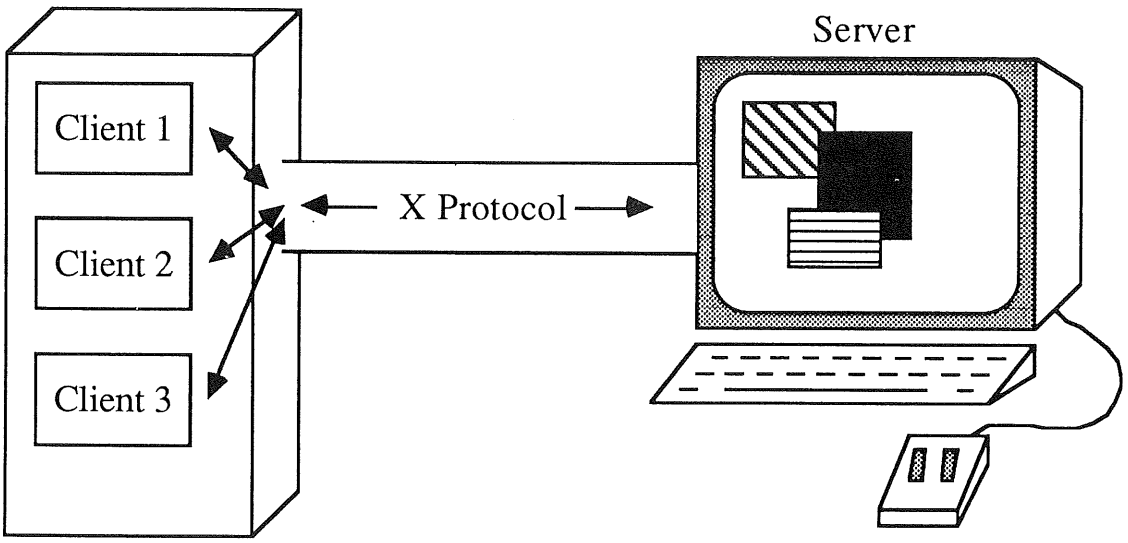


Figure 1 — The X Window System Client/Server Relationship.

Before any work was done on design of the server or the clients, the X Protocol was designed. The X Protocol [SCH88] dictates what makes up each packet of information sent from the server to the clients and vice versa. It was designed with the goal of high performance in mind, and so an effort was made to minimise the amount of data the server and clients have to exchange.

An implication of this design is that a lot of information is stored in the server. For example the *graphics context* stores information about drawing such as line and fill styles, what font to use when drawing text, clipping regions, etc. Because this information is stored in the server, drawing requests need only specify the type of object and its coordinates. Drawing attributes need only be specified when they change.

The X Window System distribution tape from MIT contains sample servers for several systems (including Sun and Apollo workstations), as well as the cfb (colour frame-buffer) and mfb (monochrome frame-buffer) code that is designed to be portable to any system that allows direct access to its frame-buffer. The cfb and mfb code sacrifices efficiency for portability.

3. Motivation For The Design

In early 1988 we started research on the replacement for our existing colour display controller. This controller was based on the Hitachi HD63484 ACRTC chip, and we were unhappy with both the performance and flexibility of this design.

We decided to use the X Window System because it seemed to be the upcoming standard, it was free, and being an open system it obviates the costs of porting application software to a proprietary interface.

We investigated a number of display controller architectures. The main evaluation criteria were:

- Ease of porting and tracking releases of the MIT sample server.
- Minimal development time for the initial product.
- A simple/low cost design.
- Does not require a high bandwidth interface with UNIX[3] (less than 1 MByte/sec).
- High bitblt bandwidth.

All the solutions investigated using graphics display processors used far too many components and required major porting effort for the MIT sample server.

We decided that a general purpose RISC processor and VRAM's for the frame-buffer with burst mode access would give us acceptable bitblt performance. After a two month evaluation period we decided on the Intel 80960KB as the display CPU. This would give us a design on which we could get the sample server running quickly while allowing us to achieve significant performance gains via optimisation.

4. Overview Of The Hardware

The Xengine is an architecture designed to run the X Window System sample server that is provided in the X distribution from MIT.

The Xengine provides an X Window System server, six serial ports, an Ethernet[4] interface and a keyboard, mouse, and bell. A block diagram of the major Xengine components is shown in Figure 2.

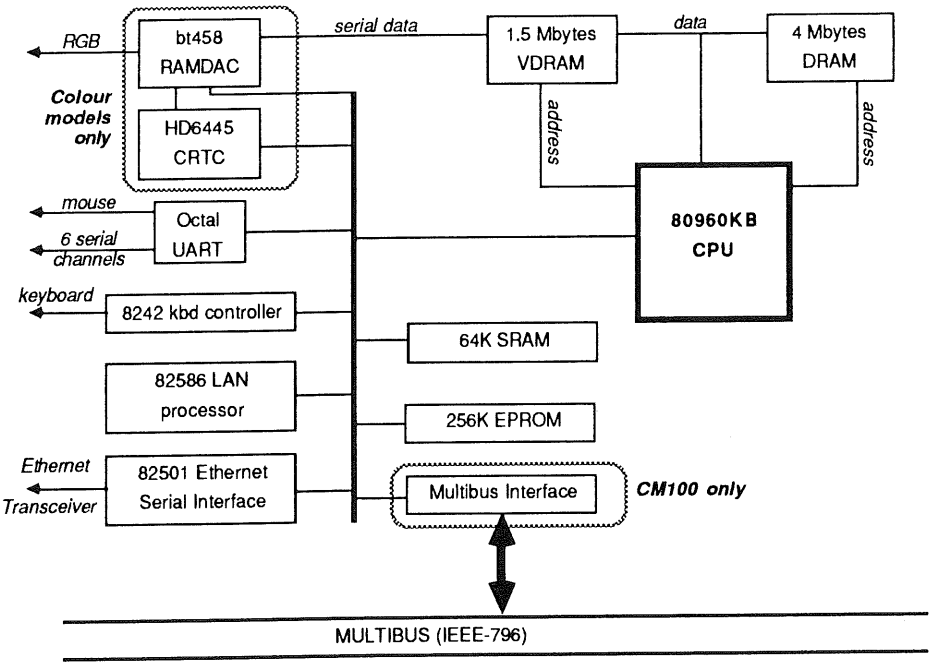


Figure 2 — Xengine block diagram.

[3]UNIX is a registered trademark of AT&T in the USA and other countries.
[4]Ethernet is a trademark of Xerox Corporation.

This architecture is available in three implementations: a colour server that resides on a system's Multibus[5] (CM100), a colour X terminal (CT100) and a monochrome X terminal (MT200).

The overall graphics model for the X server is that of a fast CPU driving a dumb frame-buffer. This architecture was chosen because it has a good fit for the code in the MIT sample server, and for performance and economic reasons (see below).

4.1. Intel 80960KB CPU

The 80960KB [INT88] was designed by the Intel Corporation as a CPU for embedded controller applications, and it is the heart of the Xengine. It is a RISC architecture incorporating a load/store design (i.e. the only register-to-memory operations are *load* and *store*; all other operations are register-to-register), and offers a sustained performance of 7.5 MIPS. It has a simple instruction format and a small number of addressing modes, allowing many instructions to be hard wired rather than microcoded. These instructions can execute in 1 cycle.

The 80960 has several features that lend it to use in an X server [LAB89] [COL89]:

- A 32 bit address bus gives a large (4 Gigabyte) address space. This makes peripheral chip decoding easy.
- A simple memory interface.
- A 32 bit data bus combined with burst mode instructions (double, triple and quad word load and store) lead to high memory bandwidth. With the 80960 running at 20 MHz a peak drawing rate of 32 million pixels per second is possible.
- Byte aligned operations make 8 bit/pixel graphics natural.
- A 512 byte instruction cache improves memory usage by using burst mode to fetch instructions, then freeing the bus for data transfer.
- A built-in floating point unit in the KB version gives good performance for certain X operations that use sub-pixel resolutions, and operations that use trigonometric function (e.g. arctan).
- An adequate, orthogonal instruction set simplifies the job of compiler writing. The GNU C Compiler was adapted to produce code for the 80960 in three person-months with an extra three person-months spent on improving the optimiser.

Other CPU's were considered for use in the Xengine: the AMD 29000 from Advanced Micro Devices and the TI 34020 from Texas Instruments. The AMD 29000 would have given comparable performance to the 80960, but at the expense of requiring costlier memory. For example, one scheme for improving the 2900's performance requires using VRAM for code storage, and VRAM is many times more expensive than ordinary RAM. The TI 34020 is a dedicated graphics processor, with built in line drawing and bitblt. However its general integer performance is about half that of the 80960 or 29000, resulting in slower overall server speed.

The 80960 was chosen because our evaluation indicated that with well coded routines it could support area operations at nearly the full memory bandwidth, and also provide high integer and floating point performance.

4.2. Memory

The colour Xengine has 4 Mbytes of fast page mode DRAM for storage of program and data.

In addition it has 1.5 Mbytes of fast page mode video DRAM for the frame-buffer. This VDRAM provides enough memory for a 1024x800 pixel, 8 bit/pixel graphics plane, and an overlaid 2 bit/pixel cursor plane. There is memory left over that could be used for backing store or off-screen bitmaps, but so far that has not been implemented. The VDRAM is equipped with a write mask which can be used to implement drawing operations that draw only on selected planes.

[5]Multibus is a trademark of Intel Corporation.

The monochrome Xengine has 4 or 8 Mbytes of DRAM, and 1 Mbit of VDRAM.

Both colour and monochrome models have 64 Kbytes of SRAM that is used for communications buffers. This memory is accessible by the on-board 80960, by the system's host CPU via Multibus (in the CM100 version), and by the 82586 Ethernet controller. This memory, combined with a set of interrupts, is how the resident and host CPU's communicate in the Multibus connected version.

Finally, there is 256 Kbytes of EPROM that contain code for bootstrapping, a terminal emulator and the Lindy debugger/monitor.

4.3. Graphics

In the colour versions of the Xengine, the video output is generated by two chips. A 6445 CRT controller generates vertical sync, horizontal sync and blanking. A Brooktree bt458 colour RAMDAC reads serial data from the VDRAM and translates pixel values into RGB video outputs. The bt458 provides a graphics plane (1024x800 pixels, with 8 bits/pixel) and an overlaid cursor plane (1024x800 pixels, with two bits per pixel). It also provides a 24 bit colour map with 256 entries for the graphics plane and 3 entries for the cursor plane

These two chips translate pixel values from the VDRAM into red, green and blue video output with a composite sync on green. A high refresh rate (70Hz) gives a flicker free display.

The monochrome versions convert the serial output from the VDRAM to ECL levels and feed these directly to a digital monitor. The refresh rate for the monochrome screen is 65Hz.

4.4. Serial/Keyboard/Ethernet Interfaces

An octal UART provides the Xengine with an interface for a serial mouse, and 6 general purpose serial channels. (One of the UART's channels is unused.) Software currently supports any serially connected, Mouse Systems compatible mouse.

An 8242 keyboard controller provides support for any standard PC-AT style keyboard.

The Xengine has a Intel 82586 LAN processor and an 82501 Ethernet serial interface. These allow connection to thin Ethernet, or thick Ethernet via an external transceiver. The 82586 is an intelligent device that relieves the CPU of some Ethernet packet processing.

In the Multibus connected version (CM100) all the above devices are accessible by the system's host CPU as well as by the on-board 80960.

5. Advantages Of This Architecture

The primary advantage of this architecture is its simplicity, as opposed to other architectures involving dedicated graphics processors or custom VLSI.

This simplicity meant that we had a faster design track. The hardware was relatively easy to design, and as the design matches well the graphics model used by the cfb and mfb code provided by MIT it did not take long to get a server running. In all, 27 person-months were devoted to hardware design, and 58 person-months devoted to the supporting software. Much of the software effort was devoted to optimisation of the server to get the best possible performance from the hardware. We were able to go from initial ideas to a marketable product in less than a year.

Also, the simple design enabled an inexpensive and high performance product, an important consideration in a competitive environment.

It is well known that the combination of a powerful processor with a simple frame-buffer and smart, carefully crafted software can give good performance for window oriented graphics [PIK85] [MCC89]. In the case of the Xengine, the choice of the 80960 has given particular performance advantages, especially after optimisations (refer to section 6) giving the Xengine a good price/performance ratio.

6. Performance

In this section, performance gains are measured in *xstones* as determined by the *xbench* benchmark. While this is not an ideal benchmark, it does give measurements over a good spread of operations and attempts to measure how fast a server would feel to a user. It has an added advantage that *xbench* results are easily available for other architectures, so we can compare the performance of the Xengine against them. In particular, *xbench* results are standardised so that a (monochrome) SUN 3/50 running the MIT sample X server gets 10,000 xstones [GIT89].

Table 1 shows a comparison between the Sun 3/50 running the MIT X11 release 3 monochrome sample server and various Xengine configurations. CT100u is a colour Xterminal running a server based on MIT's cfb code. CT100opt is the same terminal running a server with the optimisations we have done to date installed. Similarly, MT200u and MT200opt are monochrome terminals running MIT's mfb code and Labtam's optimised code respectively. For both the colour and monochrome servers further optimisations are possible.

All ratings are relative to a monochrome SUN 3/50 running X11R3 communicating via unix sockets. The ratings are scaled to give this reference machine an *xStone* rating of 10000. A machine with double performance will therefore get 20000 *xStones;* a machine with half performance will get 5000 *xStones.*

The value in the **line** field is the *lineStone* rating for solid-line, dashed-line, wide-line and rectangle performance.
The value in the **fill** field is the *fillStone* rating for solid-, tile- and stipple-rectangle-fill performance.
The value in the **blit** field is the *blitStone* rating for invrects, screencopy, scroll, and bitmapcopy performance.
The value in the **arc** field is the *arcStone* rating for arc- and filledarc performance.
The value in the **p** field is the number of colour/greyscale planes. Machines with a '1' in the **p** field are monochrome.

machine	p	comm	line	fill	blt	text	arc	cmplx	xstones
MT200opt	1	TCP/IP	21023	18846	33420	39531	122939	11437	24836
MT200u	1	TCP/IP	20433	17661	26599	34375	123860	8823	21862
CT100opt	8	TCP/IP	35056	10991	15000	29136	547122	12941	19166
Sun3/50 (R3)	1	unix-socket	10000	10000	10000	10000	10000	10000	10000
CT100u	8	TCP/IP	3367	835	2297	5843	101329	10915	2368

Table 1 — *xbench* results.

6.1. Unmodified Code

The choice of the 80960 was vindicated as soon as a server was running. Good performance, as measured by *xbench*, was gained using unoptimised mfb code. The xstones rating for the monchrome MT200 running unmodified mfb code was double that of the Sun 3/50 running the same code. This performance can be attributed to the power of the processor and the use of a good compiler (*gcc*, adapted to the 80960 and optimised by Michael Podhorodecki).

The colour server running unoptimised cfb did not give as good results, which was not unexpected as a colour server has, in one sense, 8 times the work to do. Also, the cfb code is very inefficient. However the bulk of the optimisation effort has gone into improving the colour server code.

6.2. Optimisations

Many optimisations were possible to produce a server that runs near to the capacity of the hardware [COL89] [GIL89]. Optimisations such as:

- Revision of general algorithms for a particular hardware. Code can be rewritten to take advantage of the values of certain parameters such as the number of bits per pixel and how pixels pack into machine words.
- Hand optimisation of routines for critical tasks (e.g. bitblt). This is not an easy task for a RISC machine, but is certainly worthwhile. An example of such an optimised routine is shown in Figure 3. The impact of careful coding of bitblt routines using burst mode instructions of the 80960 mean that for drawing large areas that Xengine outperforms SPARC and MIPS based workstations.
- Separate the handling of special cases. For example, horizontal, vertical and diagonal lines can be handled with routines that draw much faster than the general case. The important thing is to ensure that the savings gained from using special case code are not lost in deciding whether the drawing operation is a special case or not.
- General optimisations that are not necessarily graphics related. These include strategies such as loop unrolling, inverting loops and conditionals to keep the inner loops smaller (so that the loop code stays in the instruction cache), recoding to avoid memory accesses, and generally trading code size for performance.

The result is a relatively inexpensive high performance X server, and so we can confidently say that we have achieved the goals we set for ourselves (refer to section 3).

7. Future directions

7.1. X Terminals

The first model of the Xengine to be developed was the Multibus connected CM100. Software development was easier for this model as the host CPU could be used to provide operating system services for the X server, such as file access, client communication multiplexing and interpretation of serial mouse information.

The adaptation of this model to a stand-alone X terminal was not a difficult task. In fact, no change in the hardware was required, because the UART, keyboard controller and LAN processor were already made accessible to the on-board CPU in anticipation of the board being used in an X terminal. If you open an Labtam CT100 Xterminal you will find a CM100 card and a Multibus connector providing it power. The changes in software required were to build TCP/IP into the server for client communication (we used Berkeley TCP from the 4.3BSD-tahoe sources), and tftp for reading font and rgb database files. RARP and BOOTP are used to automatically bootstrap the server, these being built into the EPROM.

Creating a monochrome X terminal (MT200) of course required some changes to the hardware, but because of the simplicity of the architecture and the experience our design team had gained from the CT100 we were able to get a monochrome server running on prototype hardware very quickly.

7.2. An Upgrade Path

The choice of the 80960 was further vindicated in late 1989 when Intel launched the 80960CA, a version of the 80960 that can give burst rate performance of 66 native MIPS at 33 MHz, which equates to a sustained performance of around 30 VAX MIPS. This means that there is a very obvious upgrade path for the Xengine architecture.

In addition, new VDRAM's with built-in raster operations are available. These should speed up bitblt operations, because once the ALU in the VDRAM has been programmed, there is no need to read the pixels being operated on.

Of course with both these improvements there is a price/performance trade-off, as the new components cost much more than those used in the existing Xengines. (e.g. the raster-op VRAM's are approximately 10-20% more expensive, and supplies are limited.) However, the market for high

```
# bitblt code for the fill colour operation.

        # <<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>> #
        #
        .align 4
        .globl  _fillcolor
_fillcolor:
        lda     28(sp),sp    # reserve space for register save
        stq     g8,64(fp)    # save g8 - g11
        stt     g12,80(fp)   # save g12 - g14
        mov     g4,r12       # pre-load set value
        mov     g4,r13       # pre-load set value
        mov     g4,r14       # pre-load set value
        mov     g4,r15       # pre-load set value
        mulo    g3,g1,g9     # calculate height in bytes
        cmpobge 31,g2,.L01fillcolor
                             # branch if less than 32 bytes
        subo    g0,0,g10     # calculate bytes to next quad word boundary
        and     15,g10,g10
        cmpobe  0,g10,.L01fillcolor
                             # branch if we are at a quad word boundary
        # ------------------------------- #
        # call the appropriate header routine to process 1-15 bytes
        ld      _Tfillcolor-4[g10*4],g14
                           # load the routine address from the jump table
        mov     g0,g13       # make copy of Destination address
        addo    g9,g13,r3    # calculate the end address
        addo    g10,g0,g0    # adjust Destination base address for move
        subo    g10,g2,g2    # adjust width to be done for header
        balx    (g14),g14    # call the 1-15 byte routine

        cmpobge 15,g2,.L03fillcolor     # branch if less than 16 bytes
        # ------------------------------- #
        # Do the operation using quad words
        andnot  15,g2,g10    # calc. number of bytes to do
        mov     g0,g13       # copy Destination address for this line
        addo    g10,g13,r3   # calculate this lines end address
        addo    g9,r3,g11    # calculate this lines very end address
        addo    g10,g0,g0    # adjust Destination base address
        and     15,g2,g2     # will have this much remaining width
```

Figure 3a — An example hand-optimised bitblt routine (part 1).

performance colour workstations for engineering applications may justify production of a more expensive, high performance Xengine.

7.3. GCC

gcc is being optimised to better match X server requirements and to provide opportunities for further code optimisations, e.g. the ability to store 16 byte structures in registers.

```
        bbc     4,g10,.L3fillcolor  # branch if even no. of transfers
        b       .L9fillcolor        # branch to start of odd transfers
        stq     r12,(g13)         # start store of destination register A
        addo    16,g13,g13        # bump destination pointer
        stq     r12,(g13)         # start store of destination register A
        addo    16,g13,g13          # bump destination pointer
        cmpojne r3,g13,.L6fillcolor # loop if not end of line
        addo    g1,r3,r3            # bump destination end address
        cmpo    g11,r3              # test for end condition
        subo    g10,g13,g13         # back to start of this line
        addo    g1,g13,g13          # and then skip to the next line
        bne     .L2fillcolor        # branch back around outer loop
        cmpobe  0,g2,.L04fillcolor  # branch if there is no more to do
        # ------------------------------ #
        # call the appropriate tail routine to process 1-15 bytes
        ld      _Tfillcolor-4[g2*4],g14
                          # load the routine address from the jump table
        mov     g0,g13      # copy Destination address
        addo    g9,g13,r3   # calculate the end address
        balx    (g14),g14   # call the 1-15 byte routine

        ldq     64(fp),g8   # restore g8 - g11
        ldt     80(fp),g12  # restore g12 - g14
        ret                 # back to whoever called us

_Tfillcolor:              # jump table
        .word   _fillcolor_1
        .word   _fillcolor_2
        .word   _fillcolor_3
        .word   _fillcolor_4
        .word   _fillcolor_5
        .word   _fillcolor_6
        .word   _fillcolor_7
        .word   _fillcolor_8
        .word   _fillcolor_9
        .word   _fillcolor_10
        .word   _fillcolor_11
        .word   _fillcolor_12
        .word   _fillcolor_13
        .word   _fillcolor_14
        .word   _fillcolor_15
```

Figure 3b — An example hand-optimised bitblt routine (part 2).

8. Conclusions

Firstly, and most obviously, we have reinforced the findings of Pike [PIK85] and McCormack [MCC89] that the best architecture for running a window system is a powerful CPU equipped with smart code driving a dumb frame-buffer. In fact we were surprised to find that for the MIT X sample server raw CPU performance is more important than large bitblt performance. Obviously a fast CPU speeds up all areas of the server, and for small area operations setting up the drawing and dealing with the X protocol dominate performance.

```
        # <<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>> #
        #
        .align 4
        .globl  _fillcolor_1
_fillcolor_1:
        bbc     0,g3,.L13fillcolor_1  # branch if even no. of transfers
        b       .L19fillcolor_1       # branch to start of odd transfers
        stob    r12,(g13)         # start store of destination register A
        addo    g1,g13,g13        # increment destination pointer
        stob    r12,(g13)         # start store of destination register A
        addo    g1,g13,g13        # increment destination pointer
        cmpobne r3,g13,.L16fillcolor_1  # loop if not finished
        bx      (g14)             # return


        # <<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>> #
        #
        .align 4
        .globl  _fillcolor_2
_fillcolor_2:
        bbc     0,g3,.L13fillcolor_2  # branch if even no. of transfers
        b       .L19fillcolor_2       # branch to start of odd transfers
        stos    r12,(g13)         # start store of destination register A
        addo    g1,g13,g13        # increment destination pointer
        stos    r12,(g13)         # start store of destination register A
        addo    g1,g13,g13        # increment destination pointer
        cmpobne r3,g13,.L16fillcolor_2  # loop if not finished
        bx      (g14)             # return


        # <<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>> #
        #

# etc. etc. up to fillcolor_15

        # <<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>> #
```

Figure 3c — An example hand-optimised bitblt routine (part 3).

In addition we have found that the Intel 80960 is a good choice of CPU for such an architecture, as its burst mode instructions allow for very fast area drawing operations, its processing power makes for a very fast server, its floating point unit is useful for the Xserver and the release of new models gives the Xengine an obvious upgrade path.

By using this architecture, we have been able to get a competitive product onto the market within a year.

Acknowledgements

We would like to acknowledge the efforts of the Xengine design team.

The hardware was designed by David Yeung, who lead the hardware team, Chris Taylor, who also ported the NINDY debugger and wrote hardware diagnostics, Hiren Patel and Scott Colwell. Colin Hall

wire wrapped the prototype Xengine.

The software team was lead by Tim Roper and included John Carey, who ported the X server.

Michael Podhorodecki retargetted *gcc* to the 80960, and optimised the text drawing routines. Graeme Gill ported *as* and *ld* to produce 80960 code, and wrote and hand optimised the bitblt and line drawing routines.

References

[COL89] Scott Colwell, Graeme Gill, Timothy Roper, "The Design Of An X Terminal", unpublished paper, Labtam Information Systems Pty Ltd, 1989.

[GIL89] Graeme Gill, "Quad Step Incremental Generation Of Lines", unpublished paper, Labtam Information Systems Pty Ltd, 1989.

[GIT89] Claus Gittinger, "xbench - A Set Of Benchmark Tests For X Servers", distributed with the *xbench* software, written March 1989.

[INT88] Intel Corporation, **80960KB Hardware Designer's Reference Manual**, Intel Literature Sales, Santa Clara, California, 1988.

[LAB89] Labtam Information Systems Pty Ltd, "Xengine Card Hardware Summary", Labtam documentation, 1989.

[MCC89] Joel McCormack, "Smart Code, Stupid Memory: A Fast X Server For A Dumb Color Frame Buffer", Research Report, Digital Equipment Corporation, Western Research Laboratory, Palo Alto, California, September 1989.

[NYE88] Adrian Nye, **Xlib Programming Manual**, O'Reilly & Associates, Inc, Sebastopol, California, 1988.

[PIK85] Rob Pike, Bart Locanthi, John Reiser, "Hardware/Software Trade-offs For Bitmap Graphics On The Blit", *Software—Practice And Experience*, **15**, No. 2, pp. 131-135, February 1985.

[SCH88] Robert W. Scheifler, James Gettys, Ron Newman, **X Window System: C Library And Protocol Reference**, Digital Press, Bedford, Massachusetts, 1988.

Directory Services for ACSnet

R. J. Kummerfeld

Sydney University

1. Introduction

The global electronic mail network now reaches millions of people in all parts of the world. While this is an extremely useful system, using it is not always as easy as it should be. A major problem is that the address of an individual on the network is not easy to determine. Even if the individual's name and postal address are known it is still hard to deduce their electronic mail address.

Printed directories that map names to electronic addresses are not available and would probably be of limited usefulness since they would become out of date quickly. Printed directories would also be very expensive to produce and distribute.

The solution is to use the network itself to solve the problem.

2. Current facility: whois

The SUN3 and MHSnet software (the software that underlies ACSnet) include a very simple directory service. It consists of a program, called *whois* (sometimes acswhois or netwhois) that accepts a search pattern and a site address. The program sends a message containing the search pattern to a *handler* program at the site. The handler program uses the pattern to search a local database and returns any entries that match the pattern.

For example, a user may type the command:

whois "[KC]ummerfeld"@cluster.cs.su.oz.au

and the pattern "[KC]ummerfeld" will be sent to the machine "cluster.cs.su.oz.au". This pattern is then used with the command ''egrep'' to scan a file containing one line per user. Each line has the users full name and email address. All lines that match will be mailed to the originator of the query.

There are many problems with this system: the information in the database is normally a single line with no standard for the contents of the line, the search criteria is a simple unstructured pattern match and the site address must still be known or guessed. In its favour the approach is very simple and could be improved without changing the basic protocol.

The database could be any file structure, not limited to a line per entry, and the search program could be a general database search command. The search command to be used is specified during installation. The query can be any arbitrary string that can be understood by the database search program.

The main problem is that the user must know the site to send the query to. This is helped to some extent by the fact that less significant parts of the address can often be left off with the message being forwarded to the closest machine in the nominated domain where, presumably, a reasonable database is kept. For example, the query given above could be given as:

whois "[KC]ummerfeld"@cs.su.oz.au

and the message would be processed by the closest machine in the ''cs.su.oz.au'' domain. As long as machines at the ''edges'' of the cs.su.oz.au region have a user database, the query will be handled correctly. Even this is often not enough to help most people compose a query since they have to know not only the institution and department but also the network code names for these.

Another problem, common to all directory systems, is that the database is often (or even usually) out of date. Many major sites on ACSnet don't have a database installed at all!

3. A Name Resolver

Another approach to the whole area of directory services is not to have a conventional one at all. When we send a letter to someone and we're not sure of their address we will often make a good attempt at it

and then hope that the post office can work it out for us. We can do something similar with ACSnet and similar message networks.

As with the simple "whois" system we don't attempt to solve the problem of determining the site address except that an abbreviated form can be used and the domain names can often be deduced by looking at network information held by MHSnet. The user name is very difficult to guess however. Many groups use first names (eg "ken") others use initials (eg "dmr") and some groups use a mixture. Some people have meaningless strings of alphanumeric characters forced on them as a user name while others are allowed to use cute, but equally meaningless names such as "frodo".

A name resolver will help solve this problem by allowing the user name part of an address to be one of the forms: firstname.lastname or firstinitial.lastname. The mail delivery program first looks for a user name of this form then, if not found, looks up a simple database to try and map the given user name onto a real user name. If this succeeds with a single match the mail is delivered and a message sent to the originator saying who it was delivered to. If there are multiple matches a message is sent to the originator giving each of the database entries matched but the message is not delivered. Only if there are no matches is the message bounced in the normal way.

The database search has to be handled carefully and some form of inexact matching must be provided. Some people are commonly known by their middle rather than first name. So, for example, a message sent to "robert.elz@cs.mu.oz.au" should be correctly delivered to "kre@munnari.oz.au". A "soundex" algorithm could also be used to match names that *sound* alike as well as those that are spelt the same.

A simple extension to this scheme can make it much more useful. This is based on the fact that the "real user name" that is found during the database search may not in fact be the real address of the required user but rather the address of a place where the name can be resolved further.

For example, mail sent to the address "R.Kummerfeld@su.oz.au" would be processed at the first machine encountered inside the "su" (Sydney University) domain. This machine may be in the University Computing Centre and may have a reasonably complete database for the resolver but the entry for "R.Kummerfeld" may be mapped onto the address "R.Kummerfeld@cs.su.oz.au" and the message forwarded there. The machine at the edge of the "cs" (Computer Science) domain would then have the complete mapping to the final destination address.

This scheme has the great advantage that the database becomes distributed and can be maintained in a distributed way. If a person in the Computer Science department moves to a new machine, only the part of the resolver database held in Computer Science department needs to be updated.

4. A Distributed Directory Service

The final approach to directory services for ACSnet combines elements of the first two as well as drawing on ideas developed for the CCITT X500 standard for directory service.

The X500 directory service standard involves the use of a distributed database of directory information managed by cooperating *directory system agents* (DSAs) and consulted using a *directory user agent* (DUA). The database is arranged as a *Directory Information Tree* (DIT) with each DSA responsible for part of it. Requests for information from other parts of the tree are passed between the DSAs to answer the query. Basic functions offered by the DUA include name to attribute mapping (like the telephone white pages directory), attribute to set of names mapping (like the yellow pages) and name to set of names mapping (distribution lists). Other services include browsing, access control and authentication.

Entries in the DIT contain information stored as name/value pairs. For each entry, one such pair is called the *distinguished name* and is used to specify that entry.

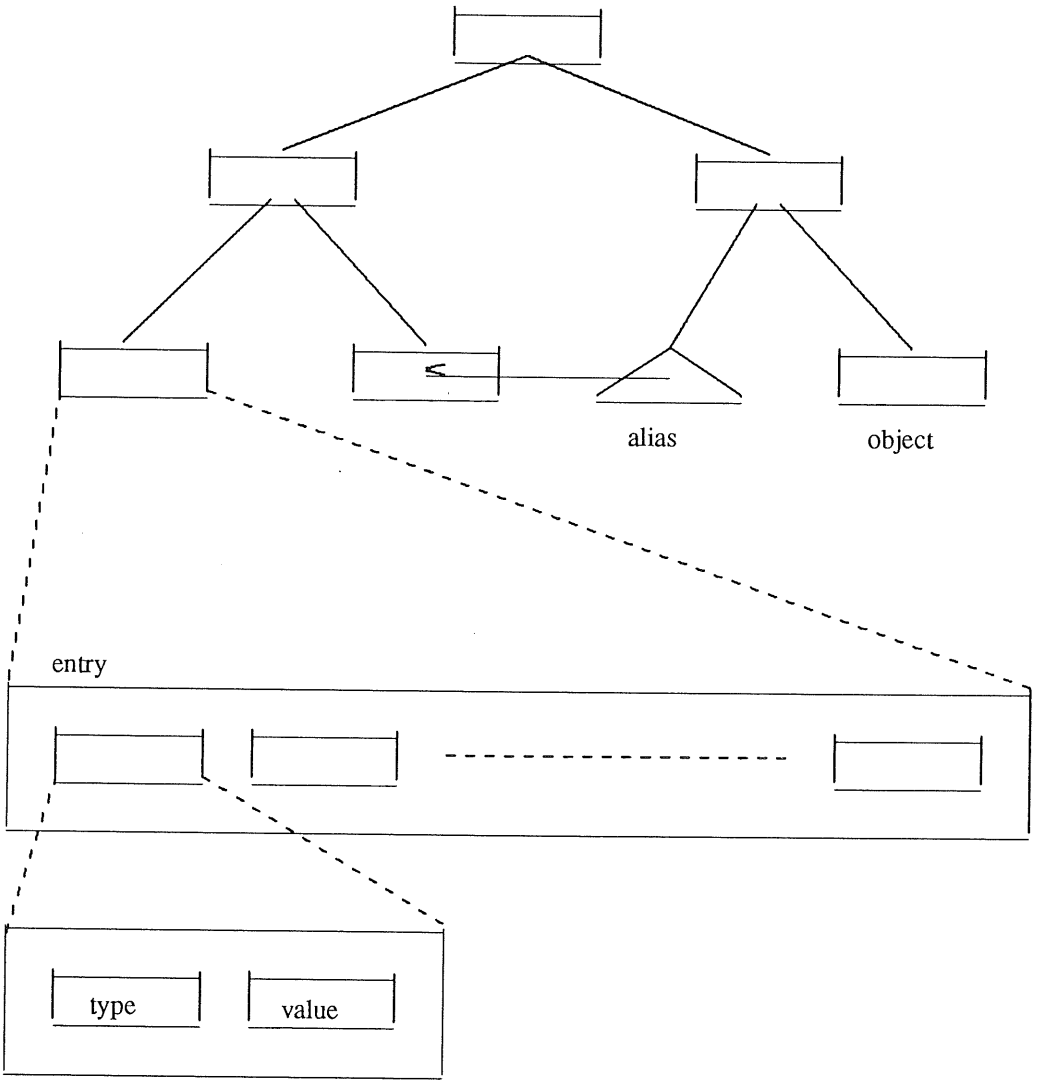


Figure 1. DIT Structure

The X500 system seems to offer all that is needed for ACSnet so why not use it? There are two reasons why this is not practical. The first is that the directory service protocols used are *real time* and are not easily adapted to the store and forward environment of ACSnet. The second reason is that the protocols require the use of other layers of the OSI model and the result is a high degree of complexity and a substantial amount of code to implement it all.

There are two ways to take advantage of the work done with X500: build a gateway from ACSnet to a real X500 system or use the basic ideas to build a similar DIT on ACSnet hosts.

4.1 X500 Gateway

The first approach is feasible and attractive given that an implementation of X500 (included in the ISODE package) will be used to provide an experimental directory service for the Australian Academic and Research Network (AARN) and that a large number of ACSnet sites will also be AARN sites. The gateway would involve a simple MHSnet *handler* that received a search request in a form similar to that required by the X500 implementation (called Quipu). This handler would then invoke Quipu to do the X500 directory search returning the result to the originator of the query. Users would formulate their search requests on any ACSnet host using a frontend program similar to the Directory User Agent of X500 with the request being sent to the X500 gateway handler on a remote host. The addresses of gateway hosts could be configured "by hand" or a database of hosts providing this service maintained automatically. The frontend program would send the request to the "closest" gateway.

One problem to be solved is that a search of the DIT using a real X500 DUA is an iterative, interactive process and may not map well onto what is essentially a "batch processing" approach. However, initial investigation of Quipu indicates that this approach is feasible and it is planned to implement this during 1990.

4.2 ACSnet Directory Information Tree

The second approach is to build a DIT among ACSnet hosts and send queries between hosts in a similar way to X500 Directory Service Agents.

The directory information tree of X500 is distributed across a number of directory service agents with each DSA holding part of the tree. Each node in the tree contains information about that node and information about successor nodes unless the node is a leaf of the tree. Information in a node is stored as a set of type/value pairs or *attributes*.

Leaf nodes would typically contain information about users but can contain any information at all. A leaf node might contain, for example:

CN=Bob Kummerfeld
surname=Kummerfeld
firstname=Robert

where "CN" denotes "Common Name". Other nodes contain information about the organisational unit or department ("OU"), the organisation ("O") and country ("C"). For example, the entry for the Computer Science Department at the University of Sydney might contain:

OU=CS
name=Basser Department of Computer Science
address=Madsen Building

The complete DIT may be split over a large number of machines in many countries.

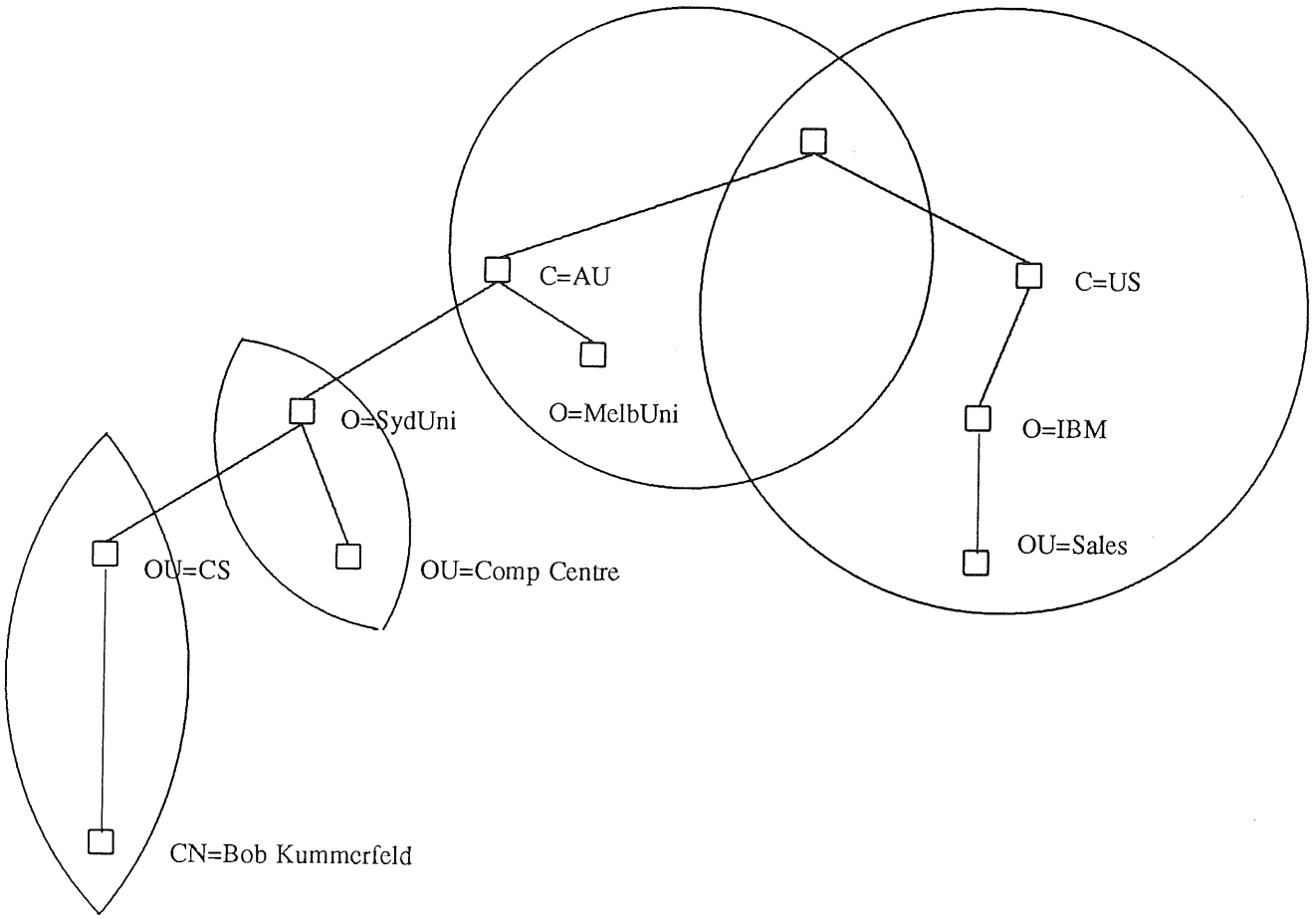


Figure 2. Example DIT

In a store and forward message network like ACSnet a directory service can be built using a distributed database similar to the X500 tree structure. Throughout the network certain nodes hold information about their *region*. For example, a node at the Computing Centre at the University of Sydney would hold information about people, departments and resources at the University of Sydney. It would also hold pointers to other machines on the campus that contain information about their region.

When a user formulates a directory search request they structure the query using components like country, organisation, department etc. Using the local database, the directory system matches as much of the query as possible. If the complete query still cannot be answered it is forwarded to the machine responsible for the part of the query that was matched. For example, a user at Melbourne University may want to find out the address of the author at the University of Sydney. They formulate the following request:

```
C=Australia
(O=University of Sydney)|(O=University of NSW)
CN=[KC]ummerfeld
```

The directory service is able to match the first two lines of the query against data it holds in the local database but not the last. It then uses the entries for "O=University of Sydney" and "O=University of NSW" to find the addresses of the machines holding directory information on each of those campuses and forwards the complete request to them. This process repeats with each host passing the request on to others or returning matching nodes of the tree.

This approach is under investigation at present. One of the main problems is the possible explosion of the search. Another problem is that since results will be returned over a period from hosts where a match has been found it is difficult to know when to declare the search finished.

5. Conclusion

Four approaches to the problem of providing directory services for a store and forward network such as ACSnet have been outlined. The existing system, ''whois'' is simple but not very effective. A ''name resolver'' is proposed as a useful alternative to a full directory service and two approaches to using the X500 directory standard have been suggested.

Stop Press
20% discount on Nutshell Handbooks

The Addison-Wesley Publishing Company is now importing the series of Nutshell Handbooks from O'Reilly & Associates, Inc. AUUG Inc. has negogiated a 20% on the books in this series for AUUG members. The series includes:

Nutshell Handbooks

- "Learning The UNIX Operating System", 75 pages, rrp $18.95
- "Learning The *vi* Editor", 144 pages, rrp $30.95
- "UNIX In A Nutshell" (BSD and System V editions), 264/270 pages, rrp $39.95
- "DOS Meets UNIX", 135 pages, rrp $30.95
- "Managing UUCP And Usenet", 256 pages, rrp $43.95
- "Using UUCP And Usenet", 185 pages, rrp $35.95
- "UNIX In A Nutshell For HyperCard", disk, rrp $99.95
- "Programming With Curses", 70 pages, rrp $18.95
- "*termcap* and *terminfo*", 248 pages, rrp $43.95
- "Using C On The UNIX System", 218 pages, rrp $49.95
- "Checking C Programs With *lint*", 82 pages, rrp $25.95
- "Understanding And Using COFF", 194 pages, $43.95
- "Managing Projects With *make*", 83 pages, rrp $18.95
- "!@%:: A Directory Of Electronic Mail Addressing & Networks", 194 pages, $43.95

X Window System Series

Vol 0: "X Protocol Reference Manual For X Version 11", 414 pages, $60.95

Vol 1: "Xlib Programming Manual", 659 pages, $69.95

Vol 2: "Xlib Reference Manual", 723 pages, $69.95

Vol 3: "X Windoe System User's Guide", 576 pages, $53.95

Vol 4: "X Toolkit Intrinsics Programming Manual", $60.95

Vol 5: "X Toolkit Instrinsics Reference Manual", $60.95

Vol 7: "XView Programming Manual", 586 pages, $60.95

To order, contact:

Addison-Wesley Publishing Co
6 Byfield St, North Ryde NSW 2113
Telephone (02) 888 2733
Fax (02) 888 9404

Don't forget to mention that you are an AUUG member, and that you saw this notice in AUUGN. I strongly encourage members to take advantage of this offer. After all, the more of you who use these offers, the more offers like this AUUG will be able to get!

David Purdue
AUUGN Editor

USENIX Association News for AUUG Members

Donnalyn Frey
donnalyn@frey.com

Frey Communications

Donnalyn is the USENIX Association Press Liaison. She provides members of the press, USENIX Association members, and AUUG and EUUG members with information on the activities of the USENIX Association.

The 1990 Summer USENIX Association Conference

Dennis Ritchie, of AT&T Bell Laboratories and co-author of the UNIX operating system, will present the keynote address at the USENIX Association 1990 Summer Technical Conference and Exhibition on June 11 - 15 at the Anaheim Marriott Hotel and Convention Center in Anaheim, California. Dr. Ritchie will be reflecting on "What Happens When Your Kid Turns 21?," and will show what has been described as "a completely different home video." This conference marks the fifteenth anniversary of the USENIX Association technical conferences.

Technical Exhibition

The Technical Exhibition will include over 65 hardware and software companies who will be displaying their latest technical innovations to a high focused end user community. Some of the participating exhibitors to date include IBM, Data General, AT&T, Intergraph, Sequent, Hewlett-Packard/Apollo, Digital Equipment Corp., Sequoia, Amdahl, Sun Microsystems, UUNET Communications, UNIX International, Open Software Foundation, NeXT, and HCR Corp. The Association is again sponsoring an Ethernet network, allowing exhibitors to display the networking capabilities of their products. The exhibitors will also have access to an FDDI network. The exhibition will be open Tuesday afternoon, Wednesday, and Thursday.

Technical Program

The upcoming Technical Program will emphasize retrospectives, analyses of tradeoffs, and critical thinking in today's UNIX environment. Papers presented at the conference will discuss new approaches in distributed systems, operating systems, file systems, applications, languages, lessons learned in computing, performance, windowing, and shared libraries. Conference sessions run on Wednesday, Thursday, and Friday. The tutorial program on Monday and Tuesday will feature new tutorials on AT&T's Open Look graphical user interface and the TCP/IP networking protocols. Popular repeat tutorials will include Mach, 4.3BSD UNIX, OSF/Motif, C++, Postscript programming, OSI, MIT X Window System, X Toolkit intrinsics, and more.

Concurrent Sessions

A second track of the conference sessions will once again feature informal talks on subjects such as computer generated music -- Peter Langston of Bell Communications Research and Mike Hawley of MIT Media Lab on how computers make music, and how are they being used in music production, arrangements, and composition. The concurrent sessions will also include Andrew Hume repeating his popular talks on regular expressions and make; Craig Hunt, of the National Institute of Standards and Technology discussing TCP/IP system administration; and Rob Kolstad of Sun Microsystems moderating a system administration problem solving panel.

The Terminal Room and FaceSaver at the Conference

The USENIX Association will host a Terminal Room which has modems for a dialout connection. Conference attendees may log onto their home or work systems to read their mail and contact other UNIX users directly from the conference. Electronic mail should be sent to attendees with the address Your_Name@conference.usenix.org.

The FaceSaver will again return to the conference. Faces will be saved and attendees will get a page of sticky labels with their faces addressing information. The FaceSaver data will again go to the UUNET FaceServer.

1990 USENIX Workshops

Upcoming workshops include:

Mach on October 4 - 5 at the Radisson Hotel in Burlington, Vermont

Software Development Environments in UNIX on January 16 - 18, 1991 at Grand Kempinski Hotel in Dallas, Texas cosponsored with the SIGMA Project of Japan. Contact the USENIX conference office for information on these workshops.

Speakers Bureau

A Speakers Bureau was recently begun to provide a forum for people with expertise in various areas of UNIX and advanced computing to share their knowledge with educational groups, including high schools, colleges, universities, and local user groups. Potential speakers have been encouraged to contact the USENIX office for more information.

Further Information on Conferences and Workshops

If you need further information regarding USENIX conferences or workshops, contact the USENIX Conference Office at

22672 Lambert Street
Suite 613
El Toro
CA 92630
USA

Email to judy@usenix.org
or {uunet,ucbvax}!usenix!judy
Tel: +1 714 588 8649
FAX: +1 714 588 9706

Further Information about the USENIX Association

If you would like information on membership, or would like information on ordering USENIX publications (proceedings, manuals, the technical journal, *Computing Systems*, or the Association's newsletter, *;login:*, please contact the USENIX Association Executive Office at

2560 Ninth Street
Suite 215
Berkeley
CA 94710
USA

Email to office@usenix.org
Tel: +1 415 528 8649
FAX: +1 415 548 5738

From Unix To Open Systems

Cédric Thomas

Pierre Audoin Conseil
65 rue Desnouttes
75015 Paris
France

Cédric Thomas has zero years of Unix programming experience, he is not even a Unix user and not even a Data Processing specialist. That probably explains why, 10 years ago, he chose to become a "Strategy" consultant for the computer industry to avoid completing his PhD in Economics. As a director with PAC in Paris, Cédric has developed since 1984, amongst other things, an annual survey on Unix trends both in his country and worldwide.

Summary

Unix standardisation strategies are generating the market for open systems.

In this paper, we shall propose a general model for open systems including, apart from the application, three functional areas which we shall call environments: the application environment, the operating environment and the hardware environment.

This model makes it possible to place the importance of new technologies in relation to the market: standards and norms, client-server architectures, Distributed Network Computing (DNC), user interfaces, etc.

Introduction

Standardisation in the Unix industry has not stopped at the operating system. As in the rest of the data processing industry it has also concerned languages, and the task of POSIX working groups is gradually becoming more oriented towards the whole software environment. In the meantime, X/Open is probably the driving force behind the structuring of the market, with its proposal for a Common Application Environment (CAE).

The fact that Unix is the operating system which serves as the basis for work on norms is not sufficient to turn Unix itself into a norm. The norm which is emerging at present (POSIX) is not an operating system but an interface between an operating system and application software. There is nothing to prevent the development and marketing (i.e. implementation on machines) of operating systems other than Unix but which nevertheless conform to the norm.

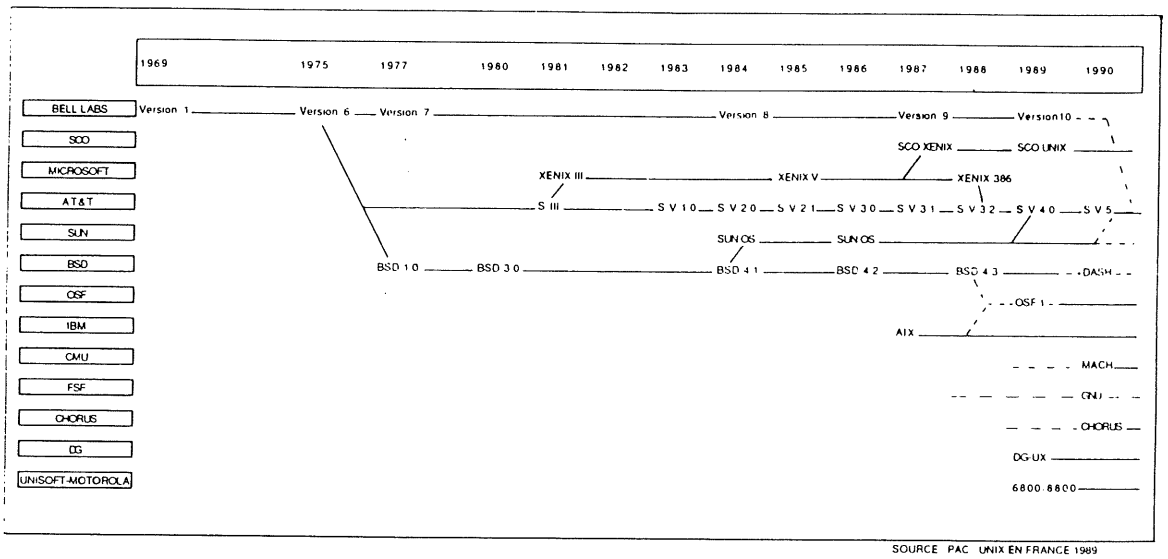
AT&T's entire Unix standardisation strategy has led to the industry becoming structured in such a way as to encourage the appearance of products in competition with Unix. The structuring of the industry thus leads to diversification. Unix was simply the first tree in a whole forest of open systems.

The new families being developed, such as the OSF and MACH families (one could also mention the AIX family, or even the GNU family) will take their share of the markets. The product which goes under the trademark of Unix will have contributed to the creation of a larger market, the market for "open systems"; it will subsequently be identified exclusively with the System V families and will be in competition with the other

families. Or, to use more familiar terms, the Unix operating system will be in competition with other "open" operating systems (see chart).

It is already becoming apparent that the open systems market of the mid-90s will be intensely competitive. In technological terms, standards will make it possible to substitute one vendor for another. Any loss of competitiveness will inevitably mean the withdrawal of one producer to the advantage of its immediate rival.

As with Hewlett Packard's takeover of Apollo, mergers will be made a great deal easier by standardisation, which will allow the fusion not only of technologies and product catalogues but also customer bases. Standardisation leads directly to the concentration of the industry.



SOURCE PAC UNIX EN FRANCE 1989

THE HISTORICAL DEVELOPMENT OF UNIX FAMILIES

From the traditional model...

The surge in technology which is a feature of the development of the Unix market has been paralleled by innovative concepts whose importance is not immediately clear in the context of conventional ideas about data processing. Renewal is necessary and, if a picture speaks a thousand words, a graphic description of this process will no doubt be more vivid than a painstaking demonstration. In such a way, what follows describes a rather complex concept through a simple model.

Of course, the model should not be taken for the reality. The point of the model is to give a general outline and to situate the importance of the innovations and the links between them. The model expresses the general rule. However, in an industry where the main forms of competition are the race for technological progress, the segmentation of markets and the differentiation of products, it should be borne in mind that all the players are jockeying for the most advantageous positions.

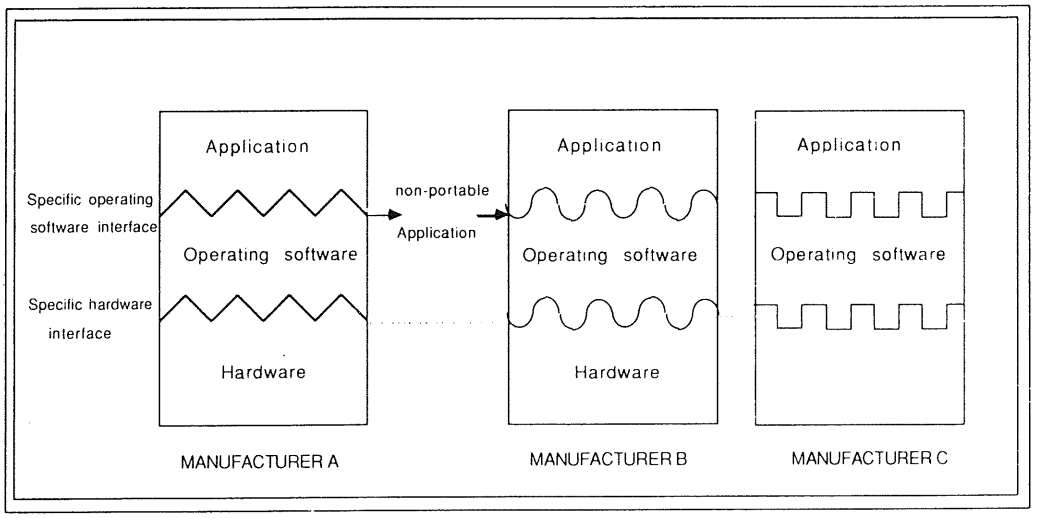
We have attempted to show in diagram form what a data processing configuration might look like and to highlight the significant innovations of the Unix market. Our model is directed towards this end and is based on layer diagrams (like those of the OSI reference model). These have the advantage of clarity as they have a single dimension, a single thrust and thus expose the essential connections between foundation and superstructure.

The traditional model represents a data processing configuration in three parts: hardware, operating software, application (see diagram). The operating software includes the operating system and the various utilities that go with it and that are sold by the computer manufacturer or by strategically attached specialised vendors. In the traditional model the emphasis is on the link between hardware and operating software because, in the development process, computer architecture comes first. As a result, the operating software is geared to the architecture and its specific features. The application which is developed on this basic setup in turn incorporates those specific features

and hence cannot be ported from one system to another. That, briefly, is the inherent condition of an industry of proprietary systems.

.... to the open system model

The fundamental change wrought by Unix was to turn the traditional process for developing operating systems on its head. The architectural restrictions which for manufacturers normally determined the development of an operating system did not apply to Unix. As Unix was not a "product" and hence was not bound by the limits of an industrial strategy, the only specifications it was required to meet were self-imposed by its designers.



SOURCE PAC - UNIX EN FRANCE 1989

A VIEW OF NON-PORTABILITY OF APPLICATIONS

Traditional operating software is bulky, as it takes into account the computers on which it runs and has to be capable from the outset of meeting all users' likely expectations. Unix, on the other hand, is less complicated and does not seek to meet all needs in advance. This approach is manifest in the system design which retains only essential functions in the kernel. The remainder is transferred to the periphery. The investments made by software developers targeting the Unix market tend to be in this periphery: development tools, file management tools, text processing and editing tools, communication, windowing, configuration management tools, etc. In the periphery, a distinction can be made between products which are systems-oriented (shell utilities, system commands, real-time or fault tolerant extensions, etc.) and those which are application-oriented (user interface, data management, programming tools, communication).

Above the operating system proper there is no longer that broad mass of programs known as the application. It has given way to something more structured in which it is possible to distinguish a "solution" part of the application (the application proper) and a "tools" part, built up on a "system" part.

Using this distinction, we can now consider the application as topping off a three-level structure made up of two levels of software and one level of hardware which we shall call environments (see diagram):

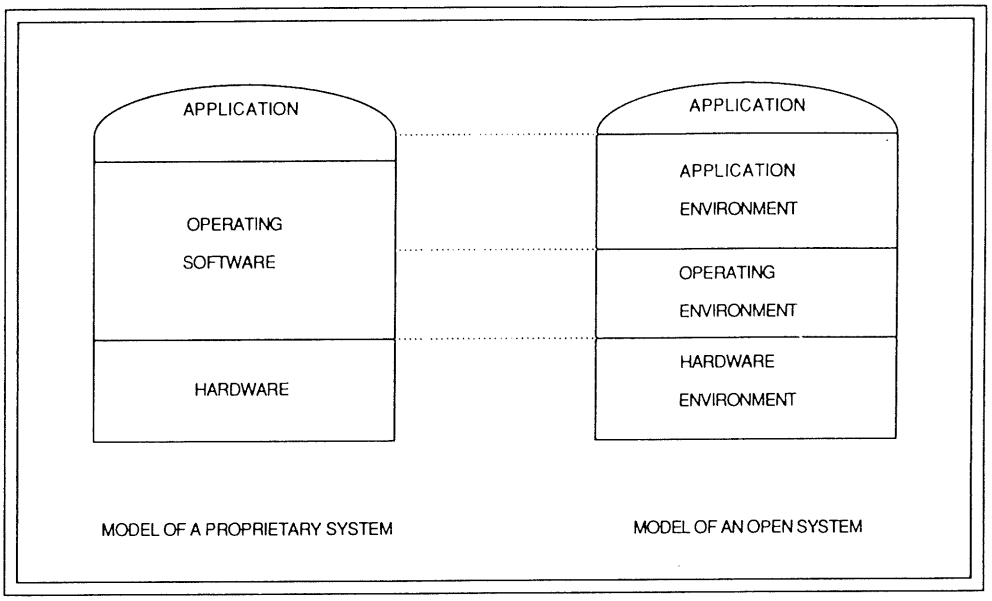
— application environment

— operating environment

— hardware environment

From an industrial point of view, the application environment and operating environment layers are made up of products offered on the market by manufacturers or increasingly (and this is something new) by specialised producers. These products mean that application developers, either at users or in software houses, now work only on the "solution" part of the application. They build this part "above" the application environment using the tools available at that level.

By paying close attention to the industrial and technical reality it is possible to determine a structure for each of these layers and hence to envisage a general model of an open system (see diagram).



STANDARD 1 SOURCE : PAC - UNIX EN FRANCE 1989

TWO BASIC MODELS

The layer structure of the hardware environment appears if the complete configuration is considered as a set of concentric rings: 1) processor, 2) bus structure (one or more buses), 3) peripherals.

Following the same pattern the next level, the operating environment, includes closely linked software such as 1) the system kernel, 2) the shell and its utilities, 3) the system's integrated extensions.

The application environment has the most complex structure. In terms of the approach used in our model it has four elements: 1) programming languages and tools, 2) the user interface, 3) data and file management, 4) communication.

The complete model (which can be refined) proposes a five-layer structure for the application environment, in relation to which each of its parts can be constructed. There are thus several specific models: one for programming tools, a second for the user interface, a third for data management and a fourth for communication.

The upper level of the general model is the application. It includes only the "solution" part of the traditional application layer, of which all of the "tools" part is now incorporated into the application environment.

Developing a model such as this one is not gratuitous. It makes it possible to situate the importance of the technical innovations and strategic behaviour which are structuring the open systems market.

Standardisation, the work being undertaken by consortia, user interface strategies, Distributed Network Computing (DNC) are all determined in relation to different sets of technologies within the operating environment and the application environment.

Open systems

The evolution of the concrete forms of Unix is a continuous historical and technical phenomenon. The upsurge in new product announcements indicates that new ways of writing and implementing Unix are transforming the market.

At a purely technical level, the most significant innovations toady are perhaps:

— rewriting the kernel in object-oriented language

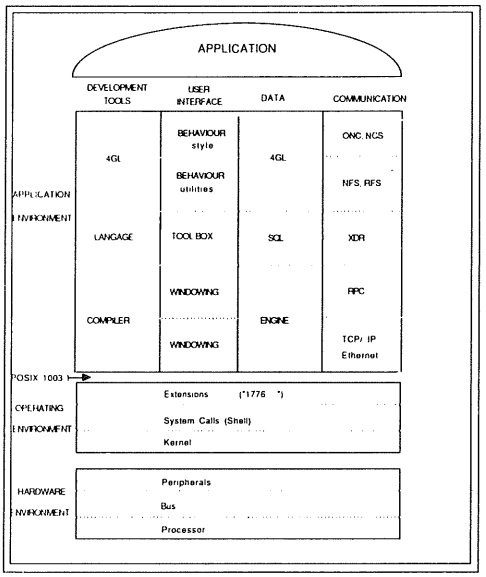
— implementing the system on multiprocessor architectures

— extending virtual memory capacity.

At a more structural or industrial level, systems-independent means of certification or validation are being seen as increasingly important. This independence guarantees a possible harmonisation on the basis of which differentiated but nevertheless compatible products can be developed.

What is the definition of an open system? It is a system which conforms to POSIX and to the rules on openness laid down by one of the standards authorities recognised by the market. As there are several of these authorities there can be several conceptions of open systems without the idea of openness being brought into question.

Rules on openness are themselves of strategic importance. This explains on the one hand why associations and consortia are competing to impose their "legitimate" conception of openness and on the other hand why producers are joining forces. Operating systems themselves become an area of competition, forming a specific market with more or less specialised producers, and products with different functions corresponding to different needs.



COMPREHENSIVE MODEL OF AN OPEN SYSTEM

Norms and standards

In the Unix industry it is important to distinguish between norms (i.e. the result of official procedure) and standardisation (i.e. de facto standards), as the two approaches do not apply to the same object. This distinction sheds light on how the market might develop in the long term and on producers' strategies.

The major distinction to remember is that the object to which a norm applies is an interface, i.e. a level of services rendered by one technological domain to another (by the operating environment to the application environment, for example). Standardisation, however, gives only an implicit description of the interface, being based on products which are characteristic of the technological domain concerned and which offer the required level of service.

It should be borne in mind that no official body is at present working on norms for the product Unix. Unless there is a sensational, but unlikely, turn of events, Unix is not and never will be a norm. However what is becoming an international norm (POSIX) is an interface between an operating

environment, whatever the products of which it is made up, and the application environment which it supports.

Besides norm procedures, strategic manoeuvres are taking place around standardisation. That is the reason for the competition between Unix International and OSF.

The position of X/Open is more complex but also more interesting. The members of this group of manufacturers recognise each other on the basis of POSIX but they are not bound to adopt exactly the same operating system. The group's strategy consists in defining and adopting a Common Application Environment. In this way the specific features of each architecture are respected and the individual advantages of each vendor are preserved.

Unlike POSIX working groups, which look at specific, precise subjects, X/Open looks at the whole application environment. What characteristics must a data processing system have in order to be standard? The definition of CAE attempts to answer this question.

CAE is not a standard itself: it is a super-set identified with the application environment layer of the open system model. It includes norms once they have been defined and products (de facto standards) if norms do not yet exist.

User interface

The graphic user interface carries out extremely important functions in present day data processing. Advanced interfaces are genuine graphic application environments which act as a framework for several functions:

— man-machine communication

— communication between applications

— communication between systems

The technical and economic importance inherent in the choice of such an interface becomes apparent. The model used here, with three main layers, is based on the X-Window model: 1) windowing, 2) tool box, 3) behaviour. Choosing similar technologies in one layer must not prevent dissimilarities in another layer and vice-versa. Given this basis it is possible to imagine multi-vendor, multi-architecture environments. The foundation of the interface is the window manager, or the windowing system. Identifying a layer with the windowing system supposes that the latter has a certain degree of autonomy. That means that there can be a windowing system independent of the graphic style.

In the next layer up, the tool box offers developers higher level commands than the primary functions of the windowing system proper. These commands are the building blocks of a graphic application. Also referred to as "widgets" or graphic objects, they include buttons, menus, command bars etc. All these building blocks depend on a basic layer of the tool box called "intrinsics" whose role is to ensure the connection between the widgets and the windowing system.

Lastly comes the behaviour layer. This is what the user sees, and it too includes two distinct sets of elements: utilities, which ensure that applications can work together, and style, the particular look and feel of an interface.

The emergence of user interfaces and of a reference model offers precise opportunities to developers of packages. The most opportunities are to be found in the upper layers of the model. In the tool box layer there are widely varying needs for widgets and although widget libraries are available in the public domain there will always be room for new tools such as "wysiwyg" widgets for text applications, graphic widgets for diagrams, or specific widgets for telecommunications, development applications, process control, etc.

Distributed Network Computing (DNC)

Distributed Network Computing is the shape which data processing seems to be taking at present. The concept of Distributed Network Computing is based on the division of applications between "client" workstations, for everything that concerns interactivity with the user, and "servers", for processing applications wherever they may be located. Whether it is a marketing concept or a technological reality, Distributed Network Computing is an element which structures the open systems market. DNC is the result of identifiable technological advances in both software and hardware and, an unavoidable condition, is based on standards.

In software terms, the basis of the Distributed Network Computing concept is the client-server

arrangement already integrated into advanced DBMS. The main foundations for the emergence of DNC are the spread of the X-Window windowing system, the maturing of major distributed environments such as ONC or NCS, and the development of targeted "client environments" for office automation workstations.

The client-server arrangement is also to be found in hardware. Here, micro-computers or workstations are the typical clients. Client systems are tending to be RISC machines in scientific and technical fields, while business and OA machines tend to be 32-bit workstations (this must be the market for 386 platforms) or the X-terminal, a new category of hardware based on X-Window technology.

Servers are genuine processing "engines" specialising in the various types of tasks which such machines are required to carry out (calculation, archiving, database, communication, etc.). DNC encourages the entry of new producers and the appearance of differentiated systems intended as servers. Lastly, and above all, DNC is based on standards: communication standards such as OSI, TCP/IP or, for LANs, Ethernet, Netware, Token Ring, Starlan etc.; file sharing or integrated environment standards such as NCS, NFS, RFS, ONC, etc.; and user interfaces, themselves based on X-Window technology, the two standards being Open Look and Motif. It should also be pointed out that the development of DNC, although in theory independent of machines and operating systems, in the end involves only machine bases sufficiently widespread to be considered standards, such as MVS-370, VMS VAX and MS/DOS-INTEL architectures, and Unix machines. Apart from IBM which, with SAA, is implementing its own conception of DNC, Unix appears to be the driving force in this field today.

The concept of DNC sheds new light on the importance of standards. DNC can only develop on the basis of standards and it is not possible to imagine assemblies of dissimilar configurations with systems and software from different vendors unless they speak a common language. The investment is so great as to be out of reach for a single producer unless it has a monopoly or a quasi-monopoly position.

By imposing a certain degree of technical harmonisation, standards protect users and manufacturers against innovations which destabilize the market. In helping to reduce technological uncertainty, norms and standards are shaping the data processing of the 90s. The client-server arrangement is also to be found in hardware. The client-server arrangement is also to be found in hardware.

USENIX Association News for EUUG Members

Donnalyn Frey
donnalyn@frey.com

Frey Communications
Fairfax, VA USA

Donnalyn 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.

1990 Winter USENIX Association Conference

The 1990 Winter USENIX Conference in Washington, DC drew 1,517 attendees.

Dr. James E. Tomayko, of the Software Engineering Institute, Carnegie Mellon University, delivered an entertaining and informative keynote address on "NASA's Manned Spacecraft Computers." Tracing NASA's sometimes hilarious meanderings through 13-bit and 32-bit words; from Burroughs to IBM; and concluding with the information that NASA had selected PS/2s running AIX and the DACS Ada compiler for the space station project, Dr. Tomayko amused over a thousand USENIX conference attendees. He concluded his talk by remarking that with NASA's decision in favor of AIX, UNIX was now truly "out of this world."

Concurrent Sessions

These new experimental sessions track provided attendees with a venue to exchange information in an informal setting. Talks were given by Andrew Hume on make and regular expressions; Mary Seabrook on getting the most from support; John Quarterman on surviving in networkland; Richard Stevens on NAWK – a new version of AWK; Tom Christiansen on PERL - a system administration language; and Eric Allman, Evi Nemeth and Mike O'Dell on submitting and presenting papers at USENIX.

Ethics Session

The conference also featured a special session on Ethics in the Computer Industry, moderated by Rob Kolstad. The panel included a communications attorney, a Chief Executive Officer, and an ethicist.

Best Student Paper

"Disk Scheduling Revisited" by Margo Seltzer, Peter Chon and John Ousterhout of the University of California at Berkeley received the USENIX Association's Best Student Paper Award.

The Terminal Room at the Conference

The USENIX Association hosted a Terminal Room which had modems for a dialout connection, as well as a T-1 Internet connection, provided by UUNET Communications Services. Conference attendees could log onto their home or work systems to read their mail and contact other UNIX users directly from the conference. All equipment was donated by various sponsors.

Facilities were also available to create cartridge tapes of GNU and public domain software. During the conference, electronic mail was sent to attendees with the address Your_Name@conference.usenix.org. The terminal room was staffed each day of the conference by USENIX Association volunteers and ran almost around the clock.

The 1990 Summer USENIX Association Conference

The 1990 Summer conference will be held on June 11-15, 1990 at the Marriott Hotel in Anaheim, California, home of Disneyland. The conference will emphasise retrospectives, analyses of tradeoffs, and critical thinking, with the theme of: "Beyond Mere Data: Perspective, Insight, Understanding" Papers at the conference will cover subjects such as:

- software release systems
- user interfaces, windowing, and graphics
- compilers, debuggers, tools, and runtime issues
- file systems
- distributed systems
- UNIX kernel approaches
- fault-tolerancy, reliability, and security
- computer architectures that stretch UNIX.

1990 C++ Conference

The 1990 C++ Conference will be held in San Francisco, California on April 9-11. It will be devoted exclusively to C++ and will offer an intensive three day program bringing together in-depth tutorials along with technical sessions covering a broad spectrum of work.

1990 USENIX Workshops

Definite dates have been selected for workshops in UNIX Security and Mach (please refer to the Calls for Papers published in this issue of the EUUGN). Workshops are also planned for Large Installation Systems Administration, Software Development Environments in UNIX, and Standards.

Further Information on Conferences and Workshops

If you need further information on registering for upcoming USENIX Association conferences or workshops, contact the USENIX Conference Office at:

22672 Lambert Street
Suite 613
El Toro
CA 92630
USA

Email to:

judy@usenix.org or
{uunet,ucbvax}!usenix!judy

Tel: +1 714 588 8649
Fax: +1 714 588 9706

Setting up a USSR UUG

Dr Vladas Leonas

Interquadro
4, 2-nd Novopodmoscovny per.,
Moscow, 125130
USSR

V. V Leonas was born on 23rd June 1956 in Moscow. After graduating from school in 1973 he entered the Moscow Aviation Institute, Facility of Applied Mathematics. In 1980 he graduated with a specialisation in systems software and entered a postgraduate course at the Institute of Mathematics and Cybernetics of the Lithuanian AS. He presented a Candidate Thesis in 1984 (a Candidate degree is equivalent to a PhD). From then until 1988 he was Head of the Operating Systems Laboratory at the Program Systems Institute of the AS of the USSR. In 1987 he received the Academic Status of a Senior Researcher.

Since 1988 at the Soviet-French-Italian joint Venture Interquadro, he was at first Portable Operating Systems Department Manager, later Scientific Research Director.

He is the author of approximately 50 papers and articles, and translator (from English into Russian) of several monographs.

He is married with one daughter (14 years old), and has a big Newfoundland dog.

UNIX™ in the USSR: 1980-1990

The first time that the word UNIX was uttered in the USSR was somewhere in 1979-1980. By the end of 1980 this word had become well known to a small group of specialists from different institutions, who formed a semi-formal/semi-informal Portable Operating Systems SIG (or more correctly - Machine Independent Operating Systems SIG) under the aegis of the State Committee of the USSR for Science and Technology. This group included mainly those who used SM-4 computers (a Soviet made mini-computer, fully compatible with the DEC PDP-11/40). This was not some form of alliance between institutions, but rather cooperation for information exchange between programmers.

From rather stochastic meetings and contacts during 1980-1981 this SIG managed, by 1982, to organise regular (twice monthly) seminars at the Advances Training Institute of the Ministry of the Automobile Industry of the USSR. By 1983 two different operating systems, both compatible with UNIX Version 6, had been implemented; this was a result of close cooperation and information exchange in the USSR. One of these two operating systems was called INMOS (a Russian acronym for Instrumental Portable Operating System) and was implemented by a team from the Institute of Electronic Control Computers, which later moved to the newly founded (in 1983) Institute for problems of Informatics of the AS of the USSR. The other version was called NMOS (a Russian acronym for Machine Independent Operating System) and was implemented at the above mentioned Advanced Training Institute for the Ministry of the Automobile Industry of the USSR.

In 1983 the first UNIX training courses in the USSR were started at the Advanced Training

™ UNIX is trademark of AT&T in the USA and other countries.

Institute of the Ministry of the Automobile Industry of the USSR. During the first teaching year there were two streams of students (2 and 3 groups respectively), who received an intensive full day 6 week MNOS course (including a course on C language programming). These training courses are still running, although the students receive a slightly different course, which lasts 8 weeks and is based on another implementation of the operating system. This is called DEMOS (a Russian acronym for Dialogin Common Portable Operating System). DEMOS is compatible with UNIX version 7 and is the result of joint development between the following institutions:

- Institute of Atomic Energy named after I. V. Kurchatov,
- Advanced Training Institute of the Ministry of the Automobile Industry of the USSR
- Scientific-Production Union Tsentrprogrammsystem.

The current version of INMOS is also compatible with UNIX version 7.

Between 1984 and 1987 a lot of work was completed. For example, in the Program Systems Institute of the AS of the USSR alone the following were designed and implemented in this period:

- MicroPROLOG Programming System for the MNOS environment.
- Real-Time version of MNOS.
- Experimental version of the operating system MICROS (for a LSI-11 compatible computer, equipped with two 8" floppy disks and 1Mbyte "electronic" (semiconductor) disk – ie a RAM disk but no hard disk).
- New methods of increasing operating systems portability.

Meanwhile the number of users of UNIX-like operating systems was growing, and first publications on, for example, use of CAD systems in the UNIX-like environment for the design of agricultural machines, or for office automation, or process control began to appear.

As regards Russian language publications connected with the UNIX operating system and the C programming language, it is necessary to admit, that such publications first appeared in 1982-1983 and began to grow in quantity rather rapidly. 1984 was the year when the first western monograph (connected with UNIX and C) was published in Russian:

1984 H Lorin, H M Deitel "Operating Systems" (Addison-Wesley, 1981), 20,000 copies.

1985 M Dahmke "Microcomputer Operating Systems" (McGraw-Hill, 1982), 20,000 copies.

1985 P Calingaert "Operating Systems Elements" (Prentice-Hall, 1982), 25,000 copies.

1985 B W Kernighan, D M Ritchie "The C Programming Language" (Prentice-Hall, 1978) under one cover with A R Feurer "The C Puzzle Book" (Prentice-Hall, 1982), 15,000 copies.

1985 R Fauthier "Using the UNIX System" (Prentice-Hall, 1981), 30,000 copies.

1985 K Christian "The UNIX Operating System" (John Wiley & Sons, 1983), 24,000 copies.

1985 H L Helms "Computer Language Reference Guide" (Sams, 1984), 15,000 copies.

1986 M Banahan, A Rutter "UNIX the Book" (John Wiley & Sons, 1982), 15,000 copies.

1987 R Thomas, J Yates "A User Guide to the UNIX System" (McGraw-Hill, 1982), 11,000 copies.

1986 S H Kaiser "The Design of Operating Systems for Small Computer Systems" (John Wiley & Sons, 1983), 50,000 copies.

1986 S R Bourne "The UNIX System" (Addison-Wesley, 1983), 25,000 copies.

1987 H M Deitel "An Introduction to Operating Systems" (Addison-Wesley, 1984), 30,000 copies.

1987 P J Brown "Starting with UNIX" (Addison-Wesley, 1984), 20,000 copies.

1988 M I Bolsky "The C Programmer's Handbook" (Prentice-Hall, 1985), 140,000 copies.

1988 D W Topham, H V Truong "UNIX and Xenix. A Step-by-Step Guide" (Prentice-Hall, 1985), 40,000 copies.

1988 M Waite, S Prata, D Martin "C Primer Plus" (Sams, 1984), 75,000 copies.

1988 R E Berry, B A E Meekings "A Book on C" (Macmillan, 1984), 12,000 copies.

1989 A R Feurer, N H Gehani "Comparing and Assessing Programming Languages Ada, C, and Pascal" (Prentice-Hall, 1984), 50,000 copies.

1989 M R M Dunsmuir, G J Davies "Programming the UNIX System" (Macmillan, 1985), 30,000 copies.

UNIX Today

It is not a simple task to give today's picture of UNIX in the USSR because of the large number of places where there are experienced UNIX users or even local informal UNIX User Groups. Such places are, for example, Moscow, Leningrad, Kiev, Riga, Novosibirsk, Kalinin, Zhaporozhje, Odessa, Kazan, to name but a few.

In order to give the reader just a small impression of the current picture the last part of this paper is devoted to the description of the work of the main (but not all!) well known teams in the USSR.

At the Interbranch Scientific-Production Union Electronmash (which includes the Institute of Electronic Control Computers) much attention is devoted to the development of portable software. As a result of this all models of the SM line of computers are equipped with the UNIX-like operating system (specially designed and implemented) and the rather wide spectrum of application software. At the present time the problem of creation of a unified portable operating environment for all models of the SM line computers (those already existing, and those designed for future production) is under heavily discussion.

The Scientific-Production Union Tsentrprogrammsystem is involved in the development and porting of UNIX-like operating systems for Soviet microcomputers based on the Intel 8088/86/286/386 family of microprocessors, including development and implementation of device drivers and different application software packages (software tools, database management systems, network software).

The organisation Mobilnost (which means Portability) was founded in 1989. This is the leading organisation in the USSR for the implementation of the Special Purpose State Program for Machine Indepedent Operating Systems and Portable Software, whose aim is to lead and coordinate all the works in the framework of the above mentioned Special Purpose State Program, including development, implementation, training and maintenance of the systems and applications software.

Specialists at the Institute for Problems of Cybernetics of the AS of the USSR are developing a portable operating system CLOS, based on the idea closely connected with the concept of object oriented programming. In order to provide UNIX compatibility most of the UNIX system calls are emulated in CLOS. For the same reason the standard I/O library and the command language interpreter (the shell) are implemented under CLOS. The first version of CLOS was implemented for a 16 bit DEC compatible computer. At the present time the first version of CLOS is being ported on to a MC86020 based workstation.

A lot of interesting work is being done at the Institute for Problems of Informatics of the AS of the USSR. Among this is work on removing device drivers from kernel address space (for 16 bit computers with a small address space), a bi-processor version of the kernel of the UNIX-like operating system (for a bi-processor computer, in which one of the processors works as a specialised file system processor), relation database management system, and many others, such as; building of modern user interface systems with PC like X Windows Terminals instead of normal terminals. At the present time a new POSIX conforming operating system for a Intel 80386 based computer is under development.

The Interquadro joint venture specialises in building turn-key systems for different area, such as office automation, CAD/CAM, process control, etc. As a component for such systems Interquadro uses the UTEC-32 family of MC68010 based computers with the QUIX operating system which was jointly developed by Interquadro and the French company Aniral UTEC Informatique International SA. QUIX is compatible with UNIX version 7, but is much faster. Different applications software packages for QUIX were also developed and implemented at Interquadro.

In concluding this paper it is necessary to admit that the interest in UNIX and UNIX-like operating systems in the USSR is growing rapidly.

Call Doc Strange

Colston Sanger
doc.strange@olibc1.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. He is very sorry that he wasn't able to do a column for the last issue...

FMLI — the Forms and Menu Language Interpreter

FMLI, the Forms and Menu Language Interpreter, is new in UNIX and UNIX/386 System V Release 3.2. FMLI syntax is a bit like shell, but also object-oriented. FACE, the Framed Access Command Environment, that I mentioned in my last column is actually an application built with FMLI.

FMLI provides a framework, a consistent 'look and feel', for applications that use menus and forms. It controls many aspects of screen management for you — with the result that you don't have to concern yourself with the low-level details of creation or placement of menus and forms on the screen, or of providing users with a means of navigating between or within them. Also, FMLI is terminal-independent. It will work on any character terminal: in colour if the terminal supports colour, otherwise in monochrome. In fact, the whole look and feel of FMLI is designed to be compatible with OPEN LOOK, the AT&T/Sun Microsystems graphical user interface for intelligent workstations.

'Object-Oriented'

I said that FMLI is object-oriented. An *object* in FMLI is either a form, menu or text frame and the items those frames contain. When you define a form or menu, you are defining an object. An *object operation* is an action that can be performed on an object. Object operations can be ordinary UNIX commands, which the FMLI interpreter passes to the shell for execution, but are much more likely to be FMLI built-in commands or keywords. For example, here is a fragment of FMLI code:

```
action=OPEN FORM Form.ph.lookup
```

It specifies the action to take when a selection is made from a menu. In the example, `OPEN` is recognised as a keyword, a FMLI command that forces an object operation to occur. The type of object to `OPEN` is a `FORM` whose name is `Form.ph.lookup`. Here I'm using both the `FORM` type-cast and the FMLI naming convention for forms (*i.e.*, `Form.*`), but it's really only necessary to use one or the other.

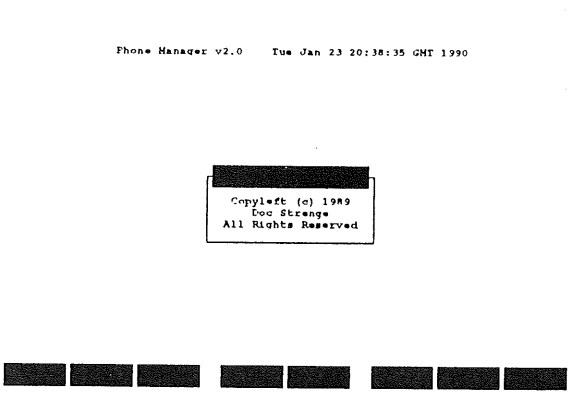
A Sample Application

The best way to introduce you to FMLI is probably to build a sample application. What I've done is built yet another version of the classic *phone_mgr* script for managing a list of names, addresses and telephone numbers. (See the 'UNIX Clinic' column in Vol.8 No 2 — Summer 1988.)

Anyway, here goes.

Typically, the scripts for a FMLI application consist of a set of frame definition files, each defining a single menu, form or text frame. In addition, applications can include three (optional) definition files: an initialisation file, a commands file and an alias file. You can use these three files to define global features of your application. I've only used one of these optional files in the *phone_mgr* application: the initialisation file. Here it is:

```
# Init.ph
#
# phone_mgr introductory object
title="Phone Manager v2.0"
text="\n  Copyleft (c) 1989\n     Doc Strange\n  All Rights Reserved."
rows=4
columns=25

# banner line
banner="Phone Manager v2.0    `date`"
bancol=center

# Colour Attributes
screen=black
banner_text=white
window_text=white
active_border=cyan
inactive_border=red
active_title_text=black
active_title_bar=cyan
inactive_title_text=black
inactive_title_bar=red
highlight_bar=cyan
highlight_bar_text=black
```



If you use an initialisation file, you can supply its name as an argument to the *-i* option to the FMLI interpreter, as in

```
$ fmli -i initialisation_file
```

or as I've done in packaging the *phone_mgr* application as a shellscript:

```
# phone_mgr - Phone Manager v2.0
# (FMLI version)
tput init
exec fmli -i Init.ph Menu.ph
```

The initialisation file above first defines a transient introductory frame (with the application name and a bogus copyright message) that is displayed when the application is invoked and is then cleared and replaced by the initial menu frame. This introductory frame is displayed again briefly when you exit from the application. The `title` descriptor defines the title that appears in the title bar of the introductory frame; the `text` descriptor defines the text that appears in the frame; `columns` defines how wide it is; and `rows` defines how high it is.

Next I define a `banner` line, a line that is displayed at the top of the screen the whole time the application is running. `bancol` says I want the banner centred (in fact, this is the default). I haven't defined the `working` descriptor — a string used to notify users that they must wait until FMLI completes an activity — so the default '`working...`' will be displayed at the righthand edge of the banner line.

Finally there are the colour attribute descriptors that let you define the colours of various elements

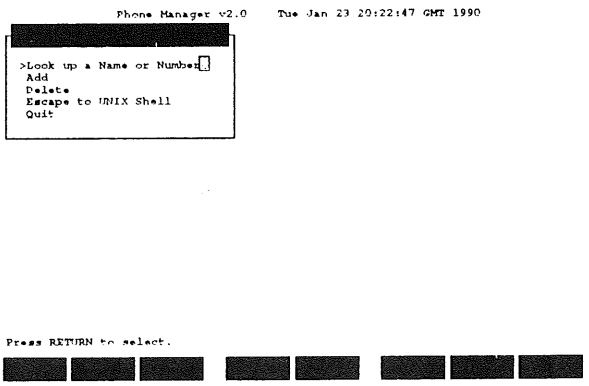
of the FMLI screen. These colour descriptors can only be defined in the initialisation file and, because of the nature of *curses*(3X), they must be set in pairs. In the initialisation file above, I've simply copied in the colour descriptors used in the FACE initialisation file. Obviously, if your terminal doesn't support colour, these descriptors are ignored.

```
# Menu.ph - Phone Manager main menu
#
menu="Menu"

# where to get help if the user
# presses the HELP key
help=OPEN TEXT Text.ph.mhlp

# menu items, with actions
name=Look up a Name or Number
itemmsg="Press RETURN to select."
action=OPEN FORM Form.ph.lookup

name=Add
itemmsg="Press RETURN to select."
action=OPEN FORM Form.ph.add

name=Delete
itemmsg="Press RETURN to select."
action=OPEN FORM Form.ph.del

name=Escape to UNIX Shell
itemmsg="Press RETURN to select."
action=unix

name=Quit
itemmsg="Press RETURN to select."
action=exit
```

There's not an awful lot to say about this. The `menu` descriptor defines a title to appear in the title bar of the menu frame; and `help` defines a help text frame to open if the user presses the HELP function key. The other descriptors are for menu items. `name` is what will appear on the menu; and `itemmsg` defines a message that will appear in the message line (the second line from the bottom of the screen) if you navigate to this menu item. (There are two ways of navigating to a menu item: you can either use the cursor keys or type the first part of the item's name in upper or lower case — it isn't case-sensitive.) The `action` descriptor defines what will happen if you select this menu item.

The Main Menu

Here is the main menu frame for the *phone_mgr* application:



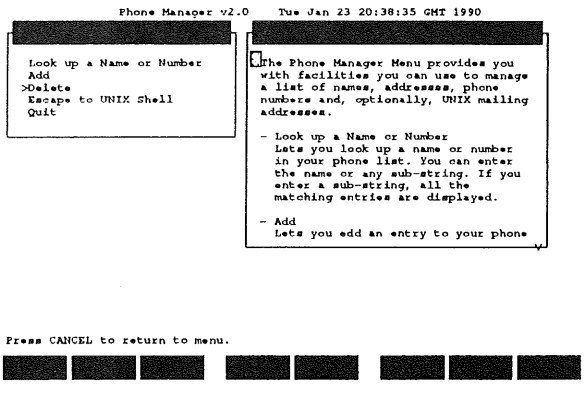
A Help Text Frame

It's always a good idea to provide help, and FMLI has a dedicated HELP function key. The help text frame below, `Text.ph.mhlp`, is invoked if you press HELP while in the main *phone_mgr* menu. It's pretty straightforward, see `Text.ph.mhlp`.

It has a `title`, and `columns` and `rows` like those you've seen before. The `text` descriptor just defines the text to be displayed. Notice the back-quoted `message` command near the end of the file. `message` is a FMLI built-in command used to display messages on the message line. I could have used `message -b` here to ring the terminal bell as well. The `lifetime` descriptor can be either 'shortterm', 'longterm', 'permanent' or 'immortal' (!). 'shortterm', as it is here, means

that the help text frame will be removed from the screen when another frame becomes current.

```
# Text.ph.mhlp - help for main menu
#
title="Help for Phone Manager Menu"

# Width of this screen
columns=40

# Height of this screen
rows=15

text=" The Phone Manager Menu
 provides you with facilities
 you can use to manage a list
 of names, addresses, phone
 numbers and, optionally, UNIX
 mailing addresses.

  - Look up a Name or Number
    Lets you look up a name or
    number in your phone list.
    You can enter the name or any
    sub-string. If you enter a
    sub-string, all the matching
    entries are displayed.

  - Add
    Lets you add an entry to your
    phone list.

  - Delete
    Lets you delete an entry from
    your phone list. You can enter
    the name or any sub-string. If
    the name or sub-string you
    enter matches more than one
    entry, you will be asked to
    choose which one you wish to
    delete.

  - Escape to UNIX Shell
    Lets you temporarily escape to
    the UNIX System shell. To
    return to this menu, press
    CTRL-d.

  - Quit
    To quit from this menu.
"
# do not wrap (re-format) text
wrap=FALSE

# Display message
`message "Press CANCEL to return \
 to menu."`

# frame will disappear when
# no longer current
lifetime=shortterm
```



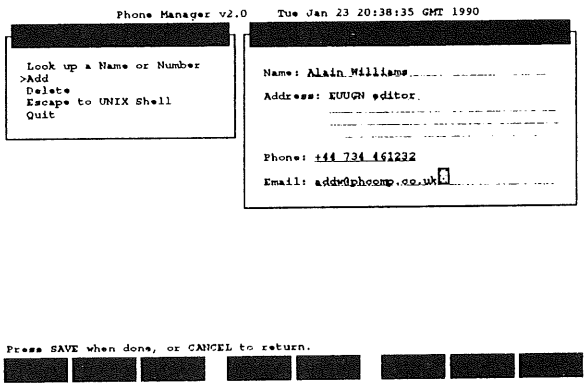
Forms

Forms are pretty easy too, up to a point. For example, here is the definition of the form used to add names and addresses, see `Form.ph.add`.

I've truncated this because the rest is just more field descriptors — essentially more of the same. The `title` and `help` descriptors you know all about by now. Skipping over the `done` descriptor for the moment, the field descriptors consist of the field `name` and its position within the form (`nrow` and `ncol`). These are followed by the input field position (`frow` and `fcol`) and its size (`rows` and `columns`). '`size=TRUE`' means that the field can actually be longer than its defined length and will scroll as required.

Now back to that `done` descriptor. This is where I began to find it difficult to do what I wanted to do with FMLI. `done` defines what happens when you press the SAVE key. `done` is a single-instance descriptor, one that can appear only once in a form, and it is of type *command*. The command here is `update`, which updates (refreshes) the screen. What I want 'done', however, is to have the contents of the input fields appended as a tab-separated line to my list of names, addresses and phone numbers (held in the file `phone.nos`), so I have to use a back-quoted expression that echoes (using the FMLI built-in command `echo`) the five input fields (`$F1` to `$F5`) to standard output.

```
# Form.ph.add
#
# Title
form="Add a Name and Number"
help=OPEN TEXT Text.ph.ahlp

# What to do when the user presses SAVE key
done=`echo "$F1 $F2     $F3     $F4     $F5     $F6     $F7" | putline` update

# Clear message line when frame is closed
close=`message -p ""`

# Form fields
name="Name: "
fieldmsg="`message -p Press SAVE when done, or CANCEL to return.`"
nrow=1
ncol=1
frow=1
fcol=7
rows=1
columns=30
scroll=TRUE

name="Address: "
nrow=3
ncol=1
frow=3
fcol=10
rows=1
columns=30
scroll=TRUE
.
.
.
```



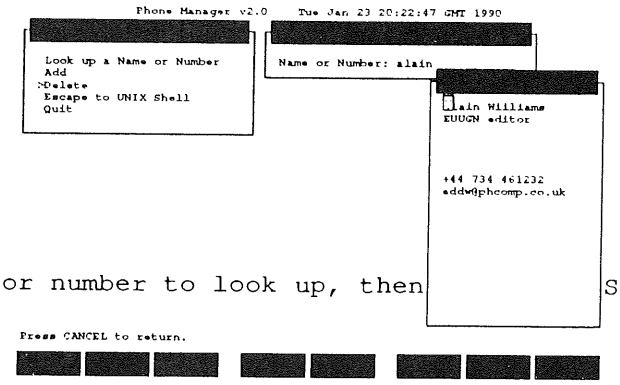
Then what? In fact, another difficulty. The current UNIX System V Release 3.2 version of FMLI has only a limited set of input/output redirection operators: < and >, but not >> or 2>. (It isn't really practicable to run a FMLI application from within a 'here document' (<<) because an explicit FMLI `exit` command is required to exit from an application.) The complete set (<, >, >>, 2> and 2>>) will be available in the UNIX System V Release 4.0 version of FMLI, but for now it's just plain irritating. To make up for this current limitation, the only way I can think of appending the input names and addresses to the `phone.nos` file is by piping standard output on to a stupid little shellscript called *putline*, the contents of which are:

```
# putline - stupid little
# shellscript
#
line >> phone.nos
```

OK, cool it. Let me show you the code for the 'Look up a Name or Number' form, see `Form.ph.lookup`.

A lot of this will be familiar to you — the input field descriptors, for example. By the way, in the `fieldmsg` descriptor, the `-p` option to the `message` command means make the message permanent, which is why I have to blank it out later with the `close` descriptor (`close` is evaluated whenever a frame is closed).

```
# Form.ph.lookup
#
# Title of this form
form="Look up a Name or Number"

# Where to get help
help=`message "Enter a name or number to look up, then SAVE or LIST.\
 Press CANCEL to return."`

# Check if name exists and display it, otherwise display a suitable message
done=`message "Looking for \"$F1\".";
        grep -i "$F1" phone.nos > /dev/null;
        regex -e -v "$RET"
                '0'     ''`set -l COMMAND="OPEN TEXT Text.ph.lookup \"$F1\"";
                        ``
                '1'     ''`message -b "Sorry, cannot find \"$F1\".";
                        set -l COMMAND="NOP";
                        ``
`$COMMAND

close=`message -p ""`

# Input field
name="Name or Number: "
nrow=1
ncol=1
frow=1
fcol=17
rows=1
columns=20
scroll=true
fieldmsg="`message -p Enter a name or number to look up, then
 Press CANCEL to return.`"

# Field validation
# Valid input is one or more NOT space or tabs only
valid=`regex -v "$F1" '^[       ]*$' false '^[^         ]+$' true`
invalidmsg="`message -b`Must be a name or number."


# Define FKey 8 as LIST
# NB. Fkey 3 (SAVE) does the same thing
name=LIST
button=8
action=done
```



What's new in this form are the field validation descriptors and the definition of an extra function key. I'd also like to discuss the definition of the done descriptor in this form in more detail.

Field Validation

The `valid` descriptor for the input field in the 'Look up' form above is defined to use the FMLI built-in command `regex` to validate the field.

`regex` uses the UNIX System V *regex*(3X) and *regcmp*(3X) functions, whose regular expressions are subtly different from those of *awk*, *ed*, *grep*, *sed*, *etc.* (See the 'UNIX Clinic' column in Vol.9 No.2 — Summer 1989) FMLI's `regex` command takes as input a stream of text and compares each line against one or more patterns.

These patterns represent regular expressions that are provided as arguments to the `regex` command line (so far so good: just like *grep*). However, a template — a string that is written to standard output if the corresponding pattern is matched — must appear after each pattern on the command line.

`regex` also provides ten registers, `$0-9` and `$m0-9`, in the pattern and template respectively. For example, the FMLI back-quoted expression

```
'readfile /etc/passwd | regex "^(fred:.*)\$0\$" '$m0' '
```

is a sort of *grep*. It scans `/etc/passwd` for a line starting with the login name 'fred' and writes the whole line to standard output. (The UNIX System V Release 4.0 version of FMLI will include as built-in commands a `fmlicut`, `fmliexpr` and `fmligrep`.)

As well as writing a template to standard output if a pattern is matched, `regex` also returns the string-value 'true', which is analogous to the way standard UNIX commands return a value of 0 on successful completion. If no pattern is matched, `regex` returns the string-value 'false'. The `valid` descriptor in the 'Look up' form illustrates this:

```
valid='regex -v "$F1" '^[          ]*$' false '^[^          ]+$' true'
```

as well as the use of the `-v` option to tell `regex` to use the argument that follows (rather than standard input) as input.

Yet More On `regex`

The `done` descriptor in the 'Look up' form uses `regex` with a `-e` option, which tells it to evaluate the corresponding template and write the result to standard output. To remind you:

```
# Check if name exists and display it, otherwise display a suitable message
done='message "Looking for \"$F1\".";
        grep -i "$F1" phone.nos > /dev/null;
        regex -e -v "$RET"
                '0'     ''set -l COMMAND="OPEN TEXT Text.ph.lookup \"$F1\"";
                        ''
                '1'     ''message -b "Sorry, cannot find \"$F1\".";
                        set -l COMMAND="NOP";
                        ''
'$COMMAND
```

Maybe I should explain what I'm trying to do here. What I'm trying to do is run a *grep*, then test the exit status (in *shell*, `$?`) to decide what to do next.

Now, I should be able to use `regex` instead of the *grep*. After all, I've just illustrated how to use `regex` to search for 'fred' in `/etc/passwd`. In the end I decided to use *grep* with the System V *-i* option (ignore case in searching) because the alternative seemed to be a horrendous regular expression. I then use `regex` in a kind of *shell* `case` statement to test the value of the built-in variable `$RET`, the exit status of the last executable (i.e., not built-in command?) run by the FMLI interpreter. Notice the multitude of quotes here.

Redefining Function Keys

FMLI provides two levels of screen-labelled function keys, known as SLKs (pronounced 'slicks'). Function keys F1-7 are defined by default as CANCEL, CHOICES, SAVE, PREV-FRM, NEXT-FRM, HELP and CMD-MENU. They may be disabled, but cannot be redefined.[1] (If your terminal doesn't have function keys, you can use the alternate keystrokes CTRL-f*n*, where *n* is a number.) F8 is undefined by default, but will be defined as CHG-KEYS if any of the alternate set F9-15 are defined. (F16 will also be defined as CHG-KEYS if this is the case.)

You can define which set of SLKs appear on the screen by setting the single-instance descriptor `altslks`. If `altslks` evaluates to `TRUE`, SLKs 9-16 will be displayed when a frame object is opened. `altslks` can appear in the initialisation file, or in menu, form or text frame definitions. You can also define the SLK layout, with the single-instance descriptor `slk_layout`. Two groupings of SLKs are supported: '3-2-3' and '4-4'. '3-2-3' is the default.

The 'Look up' form provides an example of the definition of a SLK:

```
# Define FKey 8 as LIST
# NB. Fkey 3 (SAVE) does the
# same thing
name=LIST
button=8
action=done
```

`name` is the screen label, `button` is which function key and `action` is what to do when this key is pressed.

Co-processing

A feature of FMLI that I wanted to use in the 'Look up' screen, but didn't in the end is co-processing. It consists of five built-in commands.

The `cocreate` command initialises a process and sets up pipes between it and a co-process. `cosend -n` sends information with 'no wait' down the pipe to the co-process. `cocheck` checks the incoming pipe for information; and `coreceive` does a 'no wait' read on the pipe. `codestroy` and the external UNIX command *vsig* clean up afterwards.

When I read about co-processing in the *FMLI Programmer's Guide* it seemed exactly what I needed to look up a name and then display the address and phone number. It could all be done in one screen I thought. Wrong. Or maybe I'm just not a programmer these days. Apart from more `regex` difficulties (separating the name, address and phone number fields — not insurmountable), there was the multiple match problem, which meant multiple output 'pages'.

Perhaps I'll have another try at it some time.

To Continue

`Text.ph.lookup` isn't all that interesting. It's the kludge I came up with after I gave up on co-processing:

```
# Text.ph.lookup
#
`grep -i "$ARG1" phone.nos |\
awk -F"\t" '{ printf "%s\n%s\n%s\n%s\n%s\n%s\n%s\n\n",\
$1, $2, $3, $4, $5, $6, $7 }' > /usr/tmp/lookup`

# Clear message line
`message -p ""`

title="Entries found:"

# frame will disappear when no longer current
lifetime=shortterm

done=`rm -f /usr/tmp/lookup`true

# Display list of names and addresses found
text="`readfile /usr/tmp/lookup`"
```

```
# Height of this screen
rows=15

# Width of this screen
columns=`longline`

# do not wrap (re-format) text
wrap=FALSE

# Display message
`message "Press CANCEL to return."`
```

It does introduce the FMLI built-in command `readfile`, and also `longline` which I use to set the width (number of `columns`) of the frame.

Moving on, the 'Delete a Name or Number' form is essentially the same as the 'Look up' form so I won't bore you with it here. Its `done` descriptor opens the menu `Menu.ph.del2` so that in the case of multiple matches against the name you type in, you can select which one you really want to delete. The code for `Menu.ph.del2` is:

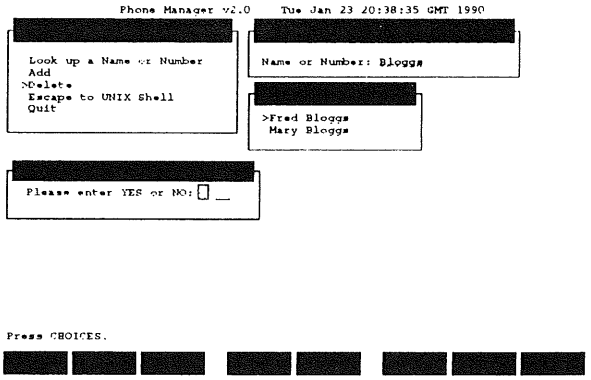
```
# Menu.ph.del2 - found possible entries to delete
#

menu="Matching Entries are:"

# where to get help if the user presses the HELP key
help="`message Press RETURN to select entry, else CANCEL`"

# Create menu items
`grep -i "$ARG1" phone.nos | regex '^([^          ]*)$0.*$' '
        name=$m0
        action=OPEN FORM Form.ph.dsure "$m0"
        itemmsg=Press RETURN to select.'`
```

It does something fancy with `regex` to dynamically create the menu items. I freely admit that I don't understand how this piece of code works, but it is modelled on an example in the *FMLI Programmer's Guide.*

Finally, I open the form `Form.ph.dsure` which asks you to confirm that you want to delete a name and address:

```
# Form.ph.dsure - Are you sure?
`message -p ""`

# Title of this form
form="Are you sure?"

# Where to get help
help=`message "Press CHOICES then DELETE, or CANCEL to return."`
```

```
done='set -l YESNO="$F1" ;
        shell "
                if [ \"$YESNO\" = \"YES\" ]
                then
                        grep -v \"$ARG1\" phone.nos > /usr/tmp/phone.nos
                        mv /usr/tmp/phone.nos phone.nos
                        echo \"Entry deleted. Press CANCEL to return.\"
                else
                        echo \" False alarm: entry has *not* been deleted.\"
                fi
        " | message 'update

# Input field
#
name="Please enter YES or NO: "
nrow=1
ncol=1
frow=1
fcol=25
rows=1
columns=3
fieldmsg="'message -b Press CHOICES.'"
rmenu={ YES NO }
choicemsg="Press SAVE or DELETE to delete this entry. CANCEL to return."
menuonly=TRUE
value=

# Define FKey 8 as DELETE
# NB. Fkey 3 (SAVE) does the same thing
name=DELETE
button=8
action=done
```



There's a couple of interesting things here. First, instead of going through all that `regex` stuff again in the `done` descriptor, I use the built-in `shell` command to invoke the standard UNIX *shell* to do an `if-then-else-fi`. The output of that is then piped into the FMLI built-in `message` command.

Second, I use `rmenu` in the definition of the input field to provide a restricted range of values (in fact, either 'YES' or 'NO') that can only be selected by pressing the CHOICES key.

The conclusion

I like FMLI. Sure I found it irritating, even exasperating at times — the currently incomplete set of input-output redirection operators, all those quotes, `regex` and the lack as yet of a `fmligrep`. But a lot of my short-tempered exasperation was the fairly natural consequence of simply not knowing *how* to do things — which is to be expected when you're learning a new language. FMLI does make the job of building form and menu-based applications on character terminals significantly easier. It also channels you implicitly towards a consistent 'look and feel' that is compatible with X.11 and the OPEN LOOK graphical user interface. That has got to be good news for users.

I'll post the code for *phone_mgr* to eunet.sources so if you're interested, look for it there.

Addendum

I've just remembered that I also meant to tackle the question of how you integrate home-grown FMLI applications with FACE. Briefly, yes it is possible and it is easy to do.

AT&T Column

Gill Mogg
gill@uel.uucp

Unix Europe Limited (UEL)
International House
Ealing Broadway
London W5 5DB

Gill Mogg is in Market Communications at Unix Europe Limited.

The guest writer this issue is Susan J picus.

Susan is Head pf Programming Languages Department, AT&T Unix Software Operation, USA.

C++ and Object Oriented Programming

Introduction to Data Abstraction

One of the major problems faced by programmers is that, in the software world, maintenance and modification is a fact of life. It has been estimated to account for as much as eighty percent of the total life cycle. Programmers are faced with changes of all descriptions - from requirements, to data representation, and to hardware. Traditional programming languages that focus on algorithms and building programs as a collection of functions do not always facilitate the implementation of such changes.

Data abstraction, on the other hand, focuses on the formation of data objects and the operations that operate on those objects. Object Oriented Programming is based on the model of building programs as a collection of data abstraction facilities. As a result, the time requirement for making changes to a program designed using object oriented design techniques has been shown to be substantially reduced.

What Are The Benefits Of An Object Oriented Approach?

Object Oriented Programming allows improved productivity primarily through software reuse. Using object oriented methods makes it significantly easier to reuse rather than rewrite programs, saving a tremendous amount of programming time not only in the development but also in the testing and proving of the reused code.

Secondly, Object Oriented Programming promotes better management of the development process. It encourages programming by the refinement rather than reinvention of existing data objects. In addition, it facilitates the development of more readable source code, through the use of overloaded functions and operators that allows developers to model the conceptual basis of data rather than traditional functional specifications, and through overt references to data objects.

Object Oriented Programming provides greatly enhanced portability characteristics by allowing developers to encapsulate the external description of objects. Finally, better tools can be provided to accelerate and ease the process of object oriented design.

A Note On C++

C++ was developed by Bjarne Stroustrup at AT&Ts Bell Laboratories in 1978. The language extends C by providing support for data abstraction facilities and object oriented programming. In 1984 AT&T released the so-called 'Release E' to educational institutions, and in 1985 made its first commercial release of the product. In 1986 Dr. Stroustrup published a C++ text which aroused wide general interest in the language. There were a number of OOPSLA conferences on Object Oriented Programming, Usenix introduced C++ workshops, and there was wide and increasing use of the language amongst a large number of commercial users. In Summer 1989, AT&T offered Release 2 of C++ as a fully supported source product.

The industry's acceptance of C++ is reflected in the large number of people interested in standardising the language. Interest has grown particularly in the last two years - ANSI for example formed the X3J16 subgroup whose first meeting was held in December 1989. The subgroup is using Bjarne Stroustrup's reference manual as the basis of their work on a C++ standard. ISO has also started formal standards activity, although initially it will monitor the ANSI X3J16 committee. UNIX International has also formed a special interest group, and X/Open has expressed interest, though for the moment it is also monitoring the other standards bodies.

Advantages of C++

C++ allows an easy transition for C programmers. Many C programmers start out using C++ merely as a 'better C' and later grow into using the data abstraction facility and some of the object oriented design facilities. C++ gives the programmer the option of using or not using these facilities as (s)he becomes more comfortable with them. C++ also preserves an investment in C software. C++ is link compatible with C, allowing applications to contain a mix of C and C++.

C++ is available on a wide variety of hardware systems, and most code is easily portable across variants. It supports multiple paradigms in the sense that application developers can opt to use more or less of its object oriented facilities.

C++ preserves the run time efficiency of C. In a number of applications that require very low level access similar to C, it is possible to achieve very high efficiency with C++.

C++ also provides strong type checking. When the ANSI C Committee introduced type checking for C, they borrowed most of their concepts straight from C++.

The language also supports inline functions, enabling the developer to avoid the overhead of a function call at the expense of some space trade offs. It provides for the specification of default arguments, and allows one to overload both functions and operators. In this way the developer is better able to model the conceptual data. At AT&T we have found that C programmers are able to start using the above features in almost no time.

Data abstraction facilities are provided through the notion of user defined objects or classes. Users define within a class the member functions or friend functions that are allowed to reference the data. There are also facilities for automatic initialisation and clean up of class objects.

Finally C++ offers object oriented facilities. Some of the object oriented facilities that are provided include inheritance and multiple inheritance, where data objects are derivable from existing objects, and the dynamic binding of virtual functions at run time.

To derive the full benefits of object oriented programming, a programmer must learn to approach a program in a different way from thinking of the algorithms. By allowing the programmer to ease into the world of OOP at their own pace, C++ has proved to be relatively easy to introduce, yet has measurable impact on productivity and maintainability, as seen in the following section.

C++ — Some Experiences Of The Language

C++ has been used on a wide variety of applications within AT&T, in such areas as telephone and networking products, systems software products, and the development of internally used tools. Currently there are over a hundred projects using C++, ranging in size from

a few people to about 75. These projects operate on a wide variety of platforms, from PCs to workstations, mini computers, Amdahls and Crays within the company.

One project, involving around 45 programmers, is a telephone Operation Support System. This project had been using C++ primarily as a 'better C' with some data abstraction, but had not originally followed object oriented design methodology. The programmers found that they were able to replace one commercial database system with another mid way through development with essentially no impact. Two senior members of the project redesigned the interfaces and the rest of the programmers had no impact on their productivity. Many of the implementation details were delayed until requirements were firmed up, but most programmers were nonetheless able to continue to design and develop their code from the interface specification.

Programmers on this particular project found also that minimal training was required. Three to four programmers were involved in designing the classes used by most of the rest of the project, so that the training for most of the programmers on the project was straightforward.

Users at AT&T have found the code to be much more modular and easily maintained. The use of data abstraction facilities in C++ allows fixes to be significantly more localised than in traditional programming languages. A second major benefit is that type checking has reduced syntactic errors significantly compared to C.

As mentioned above, C++ allows increased productivity. Because less code needs to be developed, programmers have found that projects can be completed within a significantly reduced time frame. The members of one particular project estimated that they had been able to save forty percent more of the code than would have been possible had they been using C.

Features of C++ Release 2.0

Release 2.0 has been available since June 1989. It offers a variety of enhanced features, including:

Multiple Inheritance
- a class can be derived from multiple ancestors.

Type Safe Linkage
- provides type checking across module boundaries.

Abstract Classes
other classes.
- an abstract definition of an object is used to derive

User Defined Freestore Management - gives the user greater control over data collection, and more compatibility with ANSI C.

Since ANSI evolved in parallel with C++ there were some differences that were syntactic rather than conceptual. When these differences were examined, anything that was not a required difference was changed to be compatible with ANSI C. C++ is not strictly upward compatible with ANSI C, but many of the features are very compatible. Type checking for example is almost identical.

Available with Release 2.0 is a reference manual which provides the de facto language definition. It is this definition to which more and more products are conforming, as reflected by announcements in the trade press.

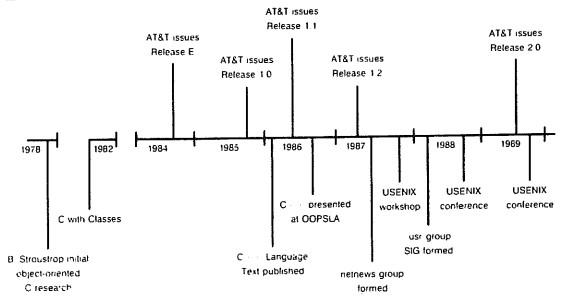
C++ Into The Nineties

C++ will develop in the 1990s into an industry standard product. The focus of activity will move from adding specific new language features to providing object oriented development environments as the language standardises and becomes stable. By the end of the decade, tools and integrated programming environments should be common place to provide for object oriented design.

Summary

C++ has evolved from a research prototype into a mature language suitable for production use. C++ has been shown to improve software and raise productivity, and use of the language is widespread and growing. Industry support of C++ is expanding, and the product should become fully standardised within this decade.

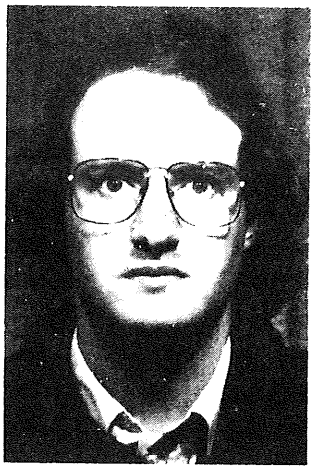
THE HISTORY OF C++



Windows Column

William Roberts
liam@cs.qmc.ac.uk

Queen Mary & Westfield College



William Roberts won the 1988 BCS Wilkes Award for a paper on formal specification. He works for Queen Mary & Westfield College, trying to find ways of managing 140 Macintoshes running A/UX. None of the above explains why he is also the author of the Window Systems column for the EUUG Newsletter.

This ''Window Systems Column'' is more a collection of small odds and ends I'm afraid, but here goes.

X11 Release 4 is out

Release 4 of the X11 window system is now available from various sources on this side of the Atlantic (see below).

For those unfamiliar with X distributions, ''X11 Release 4'' is the full source code to the ''sample server'' complete with the device specific parts needed to build it for a number of different systems.

What's in X11 release 4
The release notes for X11 release 4 say the following:

''This is the fourth release of the X Window System, Version 11 from MIT. Substantial progress has been made in optimising the sample server, window manager, and programming libraries. In addition, major improvements to the user interface of several of the key applications (in particular, xmh, twm, xman, and xterm) should make release noticeably nicer to use. Sample implementations of the various new Consortium Standards are included as well as prototype implementations of several efforts currently under development. No incompatible changes have been made to either the core Protocol or to the Xlib programming library. The Xt Intrinsics should be source compatible with the previous release. Changes have been made to the Xaw widget set, but a configuration option for providing backwards compatibility interfaces is available.

Several new sets of fonts have been added: a new fixed width family of fonts, a Kanji and Kana font, the Lucida family from Bigelow & Holmes and Sun Microsystems, a terminal emulator font from Digital Equipment Corporation, and 100 dots-per-inch (dpi) versions of all 75dpi fonts.

This release has been built on the following operating systems: Ultrix 3.1 (both VAX and RISC), SunOS 4.0.3, HP-UX 6.5, Domain/OS 10.1, A/UX 1.1, AIX RT-2.2 and PS/2-1.1, AOS-4.3, UTEK 4.0, NEWS-OS 3.2, UNICOS 5.0.1, and UNIX System V, Release 3.2 (AT&T 6386 WGS). It should work correctly, or with a minor amount of work, on a variety of other systems as well.''

There are some 17 pages or so of the release notes, so I'll stop quoting and start paraphrasing: The primary focus of this release has been optimisation of the server and improvements in the key applications; the sample server code has been made smaller, faster and more robust still

(they have been known to run for 3 months now without problems) and lots of bugs have been fixed in the various X libraries. Support has been added for System V (both with and without the STREAMS transport layers), and ANSI C function prototypes have been added to the Xlib and Xt header files; the include files should now also be usable from C++ without modification. The client libraries can be compiled as SunOS 4.0 shared libraries (good news for Sun filestores everywhere!)

A new Consortium standard extension has been added for non-rectangular windows, allowing oval buttons, round clock and all that sort of thing. There are two prototype extensions for handling multi-buffering (for animation) and alternate input devices.

The core (i.e. MIT supported) window manager has changed from the old Ultrix window manager (uwm) to the more popular twm. The xterm, xman and xmh utilities have been improved and supplemented by xditview for previewing the output of ditroff.

The user contributed tapes have swelled from 1 to 3 and include some new games, some new window managers and some user interface toolkits.

Where to get X11 release 4

The most convenient way to get X11 release 4 is to contact J. Watson who is again offering free distribution (as he did for X11 release 3). He has mailed me the following details, which I pass on to you:

> I believe it was you who mentioned in the EUUG newsletter last year that I was offering free distribution of the X Window System in Europe. You might be interested to know that your mention generated quite a bit of interest; I got quite a few requests for tapes from people who told me that they had read about my service in your column. I can't say exactly how many of these there were, but I sent a total of 109 tapes of V11R3, and I suspect that about half of them were requested after your column was printed.
>
> Now that V11R4 is out, I am offering the same service again. So, if you want to mention it in your column again, feel free. The conditions are the same as before; I will include here a brief description of the offer.

I offer distribution of the latest release of the X Window System, currently version 11 release 4, anywhere in Europe. I can only make cartridge tapes, either QIC-24/QIC-120/QIC-150 or TK-50/TK-70. No 9-track tapes, no HP cartridge tapes, no floppy disks (more than one person has asked for floppies over the years).

There is no charge for this distribution; the only requirement is that the tapes be returned. I do not require that tapes be sent in advance. I pay the postage to send the tapes from here; I expect the recipient to pay the postage to send them back. (One person sent back the tapes by Federal Express, *freight collect*. It takes all kinds, I guess.)

The distribution contains quite a bit more than just the MIT release:

— MIT X.V11R4, including core and all contributed tapes

— Official patches issued by MIT, if any

— Speedups/enhancements from reputable sources, if any (i.e. Purdue)

— Other interesting/useful software, and newer releases of software from the MIT contributed tapes, where appropriate. Currently this includes PBMplus and x11perf version 1.2. Sun has promised to send me a new release of XView which they call "R4+"; if it ever gets here, I will include it in the distribution.

— Selected postings from comp.sources.x.

— Source code for compress/uncompress, patch and tar.

The portion of this which is the MIT distribution is *identical* to what is sent out by MIT themselves. I read the MIT tapes and write my tapes with "dd", so I don't risk screwing things up trying to extract/archive everything. When I add newer releases of software that is already on the MIT tapes, I always put it in a separate tape record, rather than trying to merge it into the MIT distribution.

Anyone in Europe who wants to order a tape from me can contact me at:

J. Watson
Adasoft AG
Nesslerenweg 104
CH-3084 Wabern

Tel: +41 31 54.35.70
Fax: +41 62 61.41 30

mcsun!chx400!pan!jw
or jw@pan.uucp

For those of you in the UK with free access to the JANET network, try contacting the info-server at Imperial College, sending a message with no subject line and containing just the lines

request catalogue
topic xv11r4
request end

The distribution is MUCH TOO BIG to be sent by mail, so only people capable of using JANET NIFTP protocols should bother with the info server.

News of NeWS?

From the computing newspapers I read that AT&T UNIX System V.4 is now out, and that this includes X11/NeWS. I've seen X11/NeWS running on a Sun SparcStation 1 with a graphic accelerator and it was very nice indeed. However, if you have only got small Sun 3s then you can forget it (I'd welcome an article from anyone at Sun who'd like to comment on this).

On a smaller note, the MacNeWS implementation of NeWS 1.1 for A/UX is no longer available. Apparently the Grasshopper Group (who were selling it) sold so few in the 2-3 years they have been trying that they have given it up as a bad job.

Finally, Glenn Reid (the man who wrote the Adobe PostScript books) has left Adobe and gone to work for NeXT Inc., presumably on Display PostScript and the NeXTStep user interface.

Line Layout Manager delayed due to late running

When the Apple System 7.0 operating system is released later this year, it will not include the Line Layout Manager that I described in a previous edition of EUUGN (so Apple inform me). The Outline Fonts and new printer handling will be there, but the automatic handling of kerning, special ligatures and so on won't be available until "a future system software release".

A New X11 Users Group

The new European X User Group (EXUG) held its inaugural meeting at the Institute of Electrical Engineers in London, England on the 21st November 1989. Over 200 people attended and there was a related exhibition with 10 companies. The meeting elected Niall Mansfield of Unipalm Ltd as the Chairman of the group (I won a pound bet that Ray Anderson of IXI Ltd wouldn't stand) and Valerie Holt as the Secretary, plus deputy chairman, treasurer, newsletter editor and a committee of 4.

They have already produced a newsletter, and anyone interested in joining EXUG should contact:

The Secretary
European X User Group
Mitchell House
185 High Street
Cottenham
Cambridge
CB4 4RX
England

exug-committee@doc.ic.ac.uk

The proposed date for the next newsletter is 23rd March 1990, and there is talk of a Spring meeting but no date has yet been fixed.

It's All Writs being a Lawyer

More minor excitements over the long running Apple vs Microsoft legal battle about who owns what intellectual property rights in window systems. Apparently Xerox have entered the fray with a suit against Apple (recall that the Apple Lisa took a lot of ideas from work done at Xerox PARC, but that Apple claim to have some sort of licence from Xerox). Someone will get rich out of all this, but it won't be you or I, dear Reader.

A Threat

My working life at present has little to do with window systems; if this column is to continue then it will need considerable input from other people, so get writing.

Puzzle Corner

Mick Farmer
mick@cs.bbk.ac.uk

Mick is a lecturer at Birkbeck College (University of London) and the Secretary of the UKUUG. His interest is in all aspects of Distance Learning and he is the Senior Consultant (Software) for LIVE-NET, an interactive video network connecting London's colleges. He is also a member of the University's VLSI Consortium, mainly because the design tools draw such pretty pictures.

Hello peeps,

Solution to Puzzle Number 8

The reasoning needed to arrive at the unique solution is quite long so I will not reproduce it here. If you would like a copy, send me a mail message. Here is the solution. Note that I did not say that there were only seven sevens in the solution!

```
                        5 8 7 8 1
                    ---------------
1 2 5 4 7 3 )7 3 7 5 4 2 8 4 1 3
              6 2 7 3 6 5
              -----------
              1 1 0 1 7 7 8
              1 0 0 3 7 8 4
              -------------
                  9 7 9 9 4 4
                  8 7 8 3 1 1
                  -----------
                  1 0 1 6 3 3 1
                  1 0 0 3 7 8 4
                  -------------
                        1 2 5 4 7 3
                        1 2 5 4 7 3
                        -----------
                                  0
```

Solution to Puzzle Number 9

The basic number must be such that its square has seven digits and its cube eleven digits. This gives limits of 2155 and 3162 for the basic number (try 2154 and 3163 to see why). The second multiplication indicates that the penultimate digit of the square must be zero. Only numbers ending 00, 01, 02, 03, 47, 48, and 49 have squares with this property. The first multiplication indicates that the first three digits of the base number are below 5, while the last one must be 5 or more. This leads to a four-digit number beginning with the pair 22, 23, or 24 and ending with the pair 47, 48, or 49. Trial and error now yields the solution 2348!

Puzzle Number 10

One year (long ago :-) a number of my students failed their examinations. Two thirds of those failed on Compilers, three quarters on Graphics, and four fifths on Networks. Moreover, 26 failed on all three subjects. If this is the smallest number possible, how many students failed overall?

Puzzle Number 11

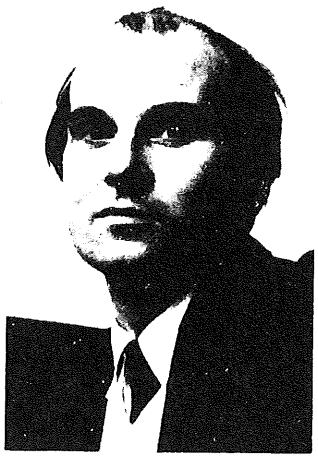
At a recent EUUG conference interpreters were hired for translating into Dutch, French, German, Hungarian, Italian, and Portugese. Furthermore, the interpreters had surnames corresponding to the six languages. Each of the six interpreters spoke two of the languages and no two of them spoke the same two languages. Each of the languages was spoken by just two people and none of them spoke the language of which they were the namesake.

Ms Hungarian could speak Dutch and German. Another interpreter spoke Dutch and Italian. Mr French and Mr Dutch, between them, spoke all of the four languages of which neither is the namesake. Of the two languages spoken by Mr Dutch, both the namesakes spoke French. Neither of the German-speaking interpreters had any knowledge of Italian. What two languages were offered by Ms Portugese?

A View of the Organisational Structure of POSIX Standards

Dominic Dunlop
domo@tsa.uucp

The Standard Answer Ltd



Equipped with an undergraduate degree in Electrical Engineering from the University of Bradford in England, Dominic sidled into the world of mini- and micro-computers. From there, he managed to effect an entry into the hallowed temples of Unix, and has hung around there ever since, writing the odd paper, contributing the odd standard, and starting the odd company. He became an independent consultant in January 1989, and his latest company, The Standard Answer Ltd, has just bought an Apple portable as the Unix machine is too heavy to carry around.

Just for a change, there has not been a meeting of the ISO POSIX working group to report for this issue of the newsletter, so I'm taking the opportunity to mount a small hobby-horse — a hobby-horse which, if harnessed, will help to distance POSIX away from its American roots.

This article provides an overview of the way in which the international standards community works, insofar as it affects POSIX and the incorporation into POSIX of internationalisation features. I'm not going to describe the technology underlying internationalisation other than to say that its aim is to make operating system and applications software independent of the user's spoken language and its representation (character sets, collation, text direction and so on). This done, localisations specific to each group of natural languages, users can tailor programs to their requirements without the need for expensive and legally-problematic hacking of source code. (If you want to know more, let me know, and I'll either expand on the topic, or give a few pointers.)

Figure 1 shows the relationship of standards bodies as far as POSIX is concerned. (The picture may look very different for other standards efforts, such as Open Systems Interconnection, but that need not concern us here.)

All standards must originate somewhere, whether in industry, in a professional association, in a national standards body, or in an international standards body. In the case of the POSIX family of standards, the Institute of Electrical and Electronic Engineers has assumed responsibility for the initial production of the documents. The IEEE is a professional association which is open to qualified engineers, no matter what their nationality. (It is not, as many people both inside and outside the U.S.A. believe, a solely North American organisation.) It has been involved for many years in the production of consensus standards — that is, standards arrived at through a formal process which gives ample opportunity for any interested party to comment and vote on proposals.

According to the standards procedures of the IEEE, the main group of interested parties is its membership, although non-members are also allowed to participate. Unusually among standards bodies, voting on IEEE standards is nominally "one member, one vote". (More typical standards bodies vote by corporation or by country.) The exception to the IEEE's individual voting scheme is that institutions can also participate, provided that they represent a broad constituency, rather than a single narrow commercial interest. Currently represented on the

POSIX effort are the Open Software Foundation, UniForum, UNIX International, Usenix and X/Open. None of these is an official standards body, although all are involved in the production of materials on which future standards may be based. In some cases, the organisations produce documents which look and smell like standards but which, because they are not produced by an open (and slow, and legalistic) consensus process, may well show some bias towards the interests of the originating organisation. Known broadly as industry standards, these documents appear before consensus standards, and must subsequently be brought into line if a consensus standard is to succeed.

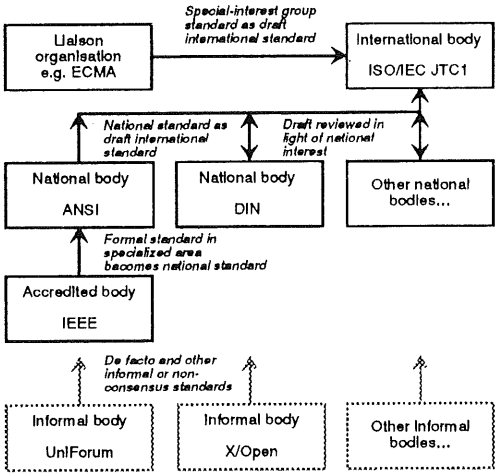


Figure 1

As figure 1 shows, in the hierarchy of standards organisations, the IEEE is near the bottom. Above it is firstly the national level, then the international. As the IEEE is based in the U.S.A., it has gained accreditation from the U.S. national standards body, ANSI (the American National Standards Institute). This means that ANSI considers the IEEE competent to produce national standards on behalf of ANSI. Of course, accreditation by ANSI gives rise to an anomaly: the IEEE, through a democratic process potentially involving an international membership, is creating national standards for the U.S.A. I shall return to this issue later.

ANSI, in turn, is a "member body" of Joint Technical Committee 1 (JTC1), an international standards body formed jointly by the International Organisation for Standardisation (ISO) and the International Electrotechnical Commission (IEC) to handle the standardisation of information technology. ANSI's role in JTC1 is nominally to represent U.S. interests in the "one nation, one vote" process by which international standards are ratified. Other member bodies such as DIN (West Germany), JISC (Japan) and IRISI (Iran), play a similar part, making sure that no standard conflicts with their own national interests.

Member bodies may sponsor draft standards at the JTC1 level. In the case of POSIX, ANSI is the sponsor. The international standards community expects that a draft standard sponsored by a national member body in this way is likely to show a bias towards the needs and culture of that member body, and so may require amendment and perhaps extension before it is suitable for adoption as an international standard. Certainly, both POSIX and the C language have come in for criticism at the international level for their lack of support for non-Roman alphabets.

In order to root out and correct any bias or omission in a draft standard sponsored by a particular member body, other member bodies are expected to pore over the proposal, and feed in changes which reflect their national needs. Obviously, this could take forever: approaching a hundred countries are represented on JTC1. Typically, the number of member bodies participating in a particular standards effort is limited, and of these few play a very active role. In the case of the POSIX effort, around a dozen member bodies are circulated with the working group's paperwork, and of these, perhaps half are regularly represented at its meetings. Even so, by the time a national standard has progressed to the level of becoming a JTC1 draft, it is rather late to begin making changes — particularly if, as is the case for POSIX and C, there is a pressing need for an international standard.

As presented so far, the standards world is strictly hierarchical: a standard such as POSIX progresses from an accredited special interest group within a country, firstly to national level, and finally to international status. Officially, it is not until the final stage that interests outside the originating country get to comment on it. The process could be made more efficient if interest groups outside the originating country had a means of commenting at an earlier stage, but the hierarchy seems to preclude such comment.

Interestingly, there is a "side door" at the international level which can be used to short-circuit the normal time-consuming process. The top level of figure 1 shows an organisation in

liaison with JTC1, the European Computer Manufacturers' Association (ECMA), which has gained the privilege of being allowed to propose and comment on standards at the international level. The process of obtaining liaison status is both difficult and lengthy, and is open only to international organisations with a valid claim to representing a specific broad area of interest. (Besides ECMA, the World Health Organisation and Mastercard International are among the sixty or so bodies in liaison with JTC1.) If the members of a liaison body can formulate a standard which is useful to them, liaison status allows that standard to be proposed for adoption as a formal international standard. Since all bodies with such status are themselves international (or at least regional), such proposals are likely to satisfy international needs without much need for amendment. (ECMA has sponsored several standards for magnetic media in JTC1; banking interests have been active in the standardisation of credit cards.) Indeed, JTC1 has developed a "fast track" approvals mechanism for use when member bodies agree that little review is necessary — although it has to be said that not every use of the fast track has resulted in a standard being approved.

The strict hierarchy imposed by ISO makes for easy and obvious management control, but is under some strain. Firstly, where emerging standards seek to accommodate international needs from their first drafting, the late review by national member bodies provided by ISO makes for unnecessary delay — delay which could be avoided if national bodies had an official means of providing input at an earlier stage. Secondly, regional standards organisations — most notably CEN, the European Standards Centre — are growing in importance, and do not fit well into a scheme which is set up according to strictly national guidelines.

These two problems combine to foster provincial attitudes on the part of standards makers — and politicians — involved with POSIX both inside and outside the U.S.A. Those inside reason that, since they are creating a U.S. national standard, international considerations are relatively unimportant, and can be left for later. Outside the U.S., standardisers reckon that it will be so long before they can mold a U.S.-produced standard to their own requirements that they might as well develop their own, probably incompatible, standards to fill their immediate needs. In Europe, a proposal to adopt issue 3 of X/Open's Portability Guide (XPG3) as a standard was strongly backed for a while, even though XPG3 is not wholly aligned with POSIX. (On the reasonable grounds that the 1003.1 standard had not been approved at the time of publication. XPG4 will be aligned with POSIX.) Interestingly, just as the IEEE is seen in Europe as representing U.S. interests, X/Open is seen by many U.S.-based observers as a European outfit, despite its many U.S. members.

Provincial attitudes among technical people and their managers outside the U.S.A. exacerbate the problems. Although the IEEE makes some effort to reach this constituency by holding one of the quarterly working group meetings outside U.S. every couple of years, the majority of attendees are always Americans. Europeans in particular seem, even if they have the inclination to attend, to find it difficult to justify the expense to their management. The interests of Arab countries and the Indian subcontinent are seldom represented at all. In contrast, delegates from Japan and other Pacific rim countries have been attending meetings in increasing numbers, even when lengthy and costly travel is involved.

Given the current structure of the international standardisation community, is it possible to work within it and yet overcome the two problems which face the POSIX effort: that of obtaining useful international input at an early stage; and the parallel problem of preventing divergence between POSIX and emerging industry, national and regional standards? Can the current structure accommodate formal mechanisms which provide for solutions, or will the problems remain unless the structure itself is changed?

Until now, practical international input to POSIX has come from two sources which are not a part of the formal hierarchy of international Standardisation: UniForum and X/Open. As I have already mentioned, X/Open is an international grouping seen by some as primarily European; its active membership has to date consisted of computer suppliers. UniForum, which was known as /usr/group until 1989, is a grouping of hardware suppliers, software authors, value-added resellers, and users. As with X/Open and other groupings, it is the suppliers which have played the largest part in the organisation — users have seldom made their voice heard. UniForum is U.S.-based, but has affiliates around the world.

These affiliates are largely autonomous, and, despite efforts to involve them, have played almost no part in Uniforum's standards activities — even when these are involved with internationalisation. (While UniForum's Technical Subcommittee on Internationalisation has active participation from outside the U.S.A., the people concerned became involved directly, rather than through their local UniForum affiliates.) USENIX, the other user grouping with institutional representation to the IEEE POSIX project has a better claim to providing a forum for users, but is almost exclusively North American, and, unlike UniForum, has no internal structures concerned with standardisation. The European UNIX systems User Group (EUUG) has a truly pan-European membership made up, like that of USENIX, primarily of computer programmers and technical users, but has not participated officially in any standards effort. Its involvement to date has been confined to the co-sponsorship with USENIX of a standards monitor service, which provides members with information about progress on POSIX and in related areas.

It is my view that, if international interests are to play a greater part in the drafting of POSIX standards, they must be represented formally within the IEEE. This is not to minimise the importance of the work done by UniForum, but rather to say that an official stamp of some sort is necessary in order that its importance receives a wider recognition both inside and outside the IEEE. Unlike other topics handled in the past by UniForum, real-time and transaction processing among them, internationalisation has never officially been incorporated into the POSIX effort because it cannot stand alone. There cannot usefully be such a thing as a standard for internationalisation: rather, internationalisation should be a consideration in the drafting of any standard for computer software.

The 1003.0 (POSIX Guide) working group is currently wrestling with the problem of handling internationalisation issues within POSIX. It may be possible to borrow a useful concept from ISO: that of the rapporteur group. Rapporteur groups cut across normal boundaries, bringing together those who are interested in some problem or activity which is common to a number of standards projects.

It is over-optimistic to hope that bringing internationalisation officially into the POSIX fold will result in immediate participation by those who currently wait until documents reach the ISO level before commenting through their national member bodies. One way to reach this audience might be to convince it that the IEEE is indeed an international, rather than strictly North American, grouping. A radical way of achieving this would be for the IEEE to seek liaison status with JTC1, so obtaining a means of submitting base documents directly, instead of through ANSI. To do this would involve the IEEE in the considerable expense and logistic complexity of sponsoring standards — a task for which resources are not currently in place in an organisation which seldom gives the appearance of being over-endowed with resources.

In any event, even if the IEEE were to apply for liaison status tomorrow, it would be a long time before it was granted. Unless or until this happens, it seems to me that it is the duty of user groups around the world to to encourage their members to play a part in the process through the IEEE. So that's what I've been doing in this article!

The Matrix: Computer Networks and Conferencing Systems Worldwide

The Matrix: Computer Networks and Conferencing Systems Worldwide, John S Quarterman, Digital Press, 1990, DP ISBN 1-55558-033-5, PH ISBN 0-13-565607-9. Price £53.95, Soft Back, 719 Pages pp, Size 25 cm x 18 cm.

This is a book which will be of interest to both new users and to those who think they know it all.

The first part of the book contains general background on networks, protocols, and most elements of communications between computers. If it doesn't tell you about what you want to know it usually has a reference to where to look for more information.

The index is fairly comprehensive and well cross referenced, for example UKC appears as UKC (University of Kent at Canterbury), University of Kent at Canterbury (UKC) and as Canterbury (University of Kent) The index also includes people, networks, protocols and standards bodies.

I would suggest it is the type of book to buy and put on your shelf to be dipped into as and when needed. It is described by John Quarterman in the Preface as "a random access book" and this is in fact an excellent description. It is also described (by Tracey L LaQuey in the foreword) as "a window to the world".

Thus if you wish to know about networks in Saudi Arabia you look in the index under Saudi Arabia and are directed to two pages in the book, one explaining that the PDN (Public Data Network) of this country is called IDAS, and the other directing you to the entry for GulfNet, seven of whose nodes are in Saudi Arabia. PDN, IDAS and GulfNet are also referenced in the index.

There are sections on etiquette on Computer Mediated Communication (CMC) and ethics which mentions viruses and worms.

The second half of the book describes the Matrix itself. This contains a list of networks in various countries and how they connect with other networks. Also in some cases why and when they were set up and how they have developed since then. There are maps of countries showing sites and connections, how to access various systems (ie who to contact), and many references to find more information about various items.

There is a chapter listing standards bodies, what they do and where to contact them.

The one problem I do see with this book is that the information it provides in the second half of the book could need to be updated fairly regularly, for instance it refers to mcvax (which has been replaced by mcsun) and gives Peter Collinson as the UKC contact (Peter Houlder is now the contact).

This has obviously been recognised by John Quarterman because included in the back of the book is a page detailing how to join a project which has been set up to provide an interactive relational database of information related to the Matrix. The prototype of this service was already being developed when the book went to press.

In conclusion, I would recommend this book as a reference for both naive and experienced users who want to get to grips with networking – both large and small. It is useful as a stepping stone to other books which describe specific areas in more detail but also for generally finding out about who else is out there.

UKUUG Winter Conference Abstracts

Here are the abstracts of the papers that were delivered at the UKUUG winter conference held in Cardiff.

Thanks are due to Robert Evans <robert@computing-maths.cardiff.ac.uk> who ran the conference and provided these abstracts.

The Development of an Internet Protocol Routing Gateway

Richard Almeida

The Computing Laboratory
The University
Canterbury, Kent CT2 7NF
England
rpa@ukc.ac.uk

UNIX networking is based around the TCP/IP protocol stack. To allow file transfer, login and NFS services across different network media, and between different sites using TCP/IP, an Internet Protocol Routing Gateway is required. This paper is split into two sections. The first section is an introduction to Internet Protocols, and describes the lower levels of the Internet Protocol stack, including IP, ICMP and UDP. The second section describes the reasons for development and the technical operation of the University of Kent's Internet Protocol Routing Gateway, a device which routes IP traffic between Cambridge Ring, X25 and Ethernet media. Also discussed are the features that are be provided by the gateway to prevent unauthorised access to the connected networks.

C News — Some Experience

Ian G Batten

BT Fulcrum
igb@fulcrum.bt.co.uk

C News is a re-implementation of the basic news software from Geoff Collyer and Henry Spencer of the University of Toronto. It provides most of the functionality news, like the difference between B and A News. It simply provides the mechanisms for unpacking and distributing news to sites running either B News or some package compatible with that.

C was actually written in parallel with the development of B 2.11. This means that much of the behaviour is documented in terms of the older B 2.10 News package[1], rather than the more widely used 2.11.

Document Authoring Tools in a Networked Workstation Environment

Martin D. Beer
Steven M. George
Roy Rada

Department of Computer Science
University of Liverpool
PO Box 147
Liverpool
L69 3BX

The availability of networks of UNIX-based graphical workstations has stimulated new developments in authoring software. This paper presents our experiences along several fronts. First, we discuss the lessons learnt from developing a simple authoring tool to run on the Atari-ST, using the GEM operating system. This was always intended to be used by a single author and was not tied to expensive computer networks. With the arrival of a large network of powerful graphical workstations in our department, developments have recently transferred to them. We discuss the development of software using 1) the X11 toolkit and one of the readily available widget sets, 2) a configurable editor (gnu-emacs) to develop prototype applications, and 3) the ANDREW toolkit to reimplement the original Atari authoring system, but this time providing a tool that will allow several authors to collaborate closely with each other. The practicalities of these approaches are discussed with reference to our own experiences.

A Model for Representing a University Organisational Structure in the X.500 Directory Service

Steve Benford

The University of Nottingham

Directory services will play a vital role in supporting future network users and applications. Services to be provided include:

- establishing a global namespace for humans, applications, devices and groups;
- mapping names to addresses ("white pages" service)
- providing a powerful information service for users ("yellow pages" service).

In order to establish a global Directory service for OSI applications, the ISO and CCITT jointly published the X.500 international Directory standard in 1988. Following this, October 1989 will see the start of a large scale Directory pilot experiment throughout the UK academic community. This experiment funded by the JNT, will involve twelve UK universities and will utilise the "Quipu" X.500 implementation which runs on UNIX (Quipu was developed at UCL). The aim of the pilot is to provide much needed experience with Directories and to pave the way for the JNT's migration to OSI protocols. The experiment therefore encourages as many interesting, large scale uses of the Directory as possible.

This paper presents some preliminary modelling work carried out at the University of Nottingham, prior to taking part in the

1. As run by cairn-el

experiment. This work shows how the detailed structure of the University might be represented in the Directory service. This includes support for a wide range of information, including:

- The structure of academic departments, research groups and service departments.
- University employees (names, addresses, telephone and email)
- University roles and occupants (e.g. department heads, union reps ...)
- Committees, meeting dates and locations.
- Halls of residence
- Departmental publications

The aim of the model is to provide a detailed breakdown of the University hierarchy. This might be used to provide a variety of novel services:

- Replacement of the current publication circulation lists by an on-line mechanism.
- Calendar facility for committee meetings, dates and management of membership.
- Role to address mapping (e.g. "who is the admissions tutor for the department of computer science?")

In order to develop this model, the paper introduces a number of new X.500 "object classes" and "attribute types". A description is then given of how these are arranged into a University naming tree. Arguments for the choice of this specific tree structure are presented.

Although specifically describing the University of Nottingham, it is intended that this model will be applicable to most universities and perhaps even, in a broader sense, to some companies. At a time when experience with Directories is urgently required, discussion of, and experimentation with, such models should provide useful insights into the possible applications of Directory services.

X/DeskMaster - Developing a Sophisticated UNIX Interface with X and OSF/Motif.

Paul Bentley
Gary Walsh

Siemens SDG
Woodley
Reading
Berkshire
RG5 3JP

X/DeskMaster is an object based, rules driven interface to UNIX, running together with X window managers. It offers many advantages and interesting features over other desktop managers.

X/DeskMaster is derived from the successful Collage windowing system, supplied for the last two years by Europe's largest UNIX vendor. The main requirements of the product were to retain the existing product functionality, but to use the OSF/Motif widget set.

This paper begins by briefly introducing the main features of X/DeskMaster. These are used as a framework to describe the experiences of porting this sophisticated application to X, using OSF/Motif widgets.

Techniques for adding a globally accessible menu to an application program, and for preventing desktop icons covering application windows are included. A `custom` widget was created so that icons may appear any shape the user desires and not in a rectangular box. Problems of interaction with window managers are covered.

This is not a critique of OSF/Motif. Much of the the content is independent of OSF/Motif, and is therefore relevant to any X application.

Naming Services rather than Machines

Piete Brooks

Computer Laboratory
Cambridge University
pb@uk.ac.cam.cl

Many sites and most users tend to think in terms of the host with which they are communicating, rather than the service that it is providing. People ask me for the X.121 address of my machine, rather than of an individual service it provides. The NRS has made a clear indication that an address is associated with a name, context, network triple. It also gave recommendations about the registration of suitable services at the site level, rather than the host level (notably mail).

The talk will extend these ideas and show how much simpler and more flexible they can make the provision of a more available system. This document provides some of the technical background to the talk, surveying the provisions made by various "name servers" in use in the UNIX environment.

The COSINE Project – An Introduction and the IXI Network

Ian Smith
Bob Cooper

Rutherford Appleton Laboratory

The EUREKA COSINE (Co-operation for Open Systems Interconnection Networking in Europe) Project has recently progressed from the specification phase to the implementation phase. The aim of the project is to create a pan-European data communications infrastructure based on OSI standards to serve the European Research Community. The user community, which spans both academic and industrial research, is expected to be large and diverse.

19 European countries, together with the Commission of the European Communities (CEC) are participating in the project. Funding will be derived from the CEC and national contributions. The UK national contribution will be funded by the Computer Board, DTI, ESRC, NERC and SERC.

The implementation phase will last for three years and is expected to create an ongoing set of services for the Research Community. RARE (Reseaux Associes pour la Recherche Europeene) will be responsible for managing the project.

The implementation phase comprises a wide range of activities and sub-projects, many directly concerned with the provision of services. An important sub-project, which will provide the foundation for all the other activities, is a private international X.25 network called IXI. This is scheduled to begin operation as a pilot service early in 1990 and will act as a backbone interconnecting many national academic and public networks.

The presentation will be divided into two parts. A brief introduction to the the COSINE Project will be followed by a presentation on the IXI network.

UNIX and Object Oriented Distributed Systems

Donal Daly
Vinny Cahill
Chris Horn

Distributed Systems Group
Department of Computer Science
Trinity College
Dublin 2
Ireland
daly@cs.tcd.ie
vjcahill@cs.tcd.ie
horn@cs.tcd.ie

UNIX is a well established system interface, as can be seen from the work of POSIX and X/Open. It has been gradually extended to support distribution and embrace concepts such as object orientation. Systems like Mach try to make the kernel smaller while providing increased support for distribution. Object oriented systems promise the potential for re-usable software, along with higher level data modelling. The Esprit COMANDOS project is supporting distribution and object orientation. It intends to provide an integrated platform for the development and online management of distributed applications. Placing a UNIX interface on top of such a distributed object orientated kernel is a possible approach to integrating UNIX and distributed object systems, which is explored in this paper. The motivation for supporting UNIX in an object oriented distributed environment is presented. We describe then, the main features of the COMANDOS kernel. Finally, an approach to supporting UNIX with an object oriented kernel is outlined. Such an approach would provide a migration path for existing UNIX users towards a fully object oriented system. It would also provide to UNIX users not interested in object orientation access to the increased functionality available in a distributed system.

Are Standards the Answer?

Dominic Dunlop

The Standard Answer Ltd

Moves are afoot to standardise every aspect of the UNIX® world in order that the benefits of open systems can be realised. But what *needs* to be standardised? What *are* the benefits? And who really cares anyway? The answers to these questions turn out to be rather vague, and are not always a good fit onto the standardisation activity which has taken place to date.

This paper examines the forces behind standardisation, reaching the conclusion that, while standardisation is a necessary process, it cannot and should not hope to have a significant effect on the diversity of ideas in the field of computer technology — or in any other field.

Designing an X.500 User Interface: The Early Stages

Andrew Findlay
Damanjit Mahl

Brunel University
Andrew.Findlay@brunel.ac.uk
Damanjit.Mahl@brunel.ac.uk

The X.500/ISO 9594 Directory is briefly described, and the early stages of the design of a user interface are detailed. Examples are included that give an idea of the appearance of the proposed interface under the X Window System.

An Idiots Guide to OSI Inter-Computer Cooperation

John Henshall BSc FSA Scot.

Senior Computing Officer
EUCS, University of Edinburgh

This tutorial paper is intended as an introduction to the application orientated layers of the ISO reference model. It will introduce each of the end to end related layers discussing what each uniquely provides for an application. The internal architecture of the model will be examined to give the feel of how the communication functions are realised. This is strictly a "beginners" guide and so any delegate with a basic knowledge of OSI could take the opportunity offered by this tutorial to catch up on some sleep/walk the dog/phone the spouse.

The X.500 Directory Service - Data Gathering

Julia M Hill

Computer Centre
Heriot-Watt University
Edinburgh

The technical requirement is for a system to interface with the X.400 Message Handling Systems, but there are a number of uses: the Directory Service will provide a means for humans to communicate with other humans.

The most obvious use is for finding electronic mail addresses of colleagues at home and abroad. Initial internal use may be very simple, but still valuable: after all, what use is a telephone without a directory? Users of electronic mail systems presently have to write to colleagues to find out their electronic mail name and address. The Directory will provide a central means of storing data about people whom others wish to contact for various reasons, and also data about administrative units. People spend a lot of time searching manually for names, addresses and mailnames. They ask Postmasters, Computer Centre Advisers, and Administrators. With current administrative policies tending towards the 'terminal on every desk' approach, many users who are not computer-literate will need this service. It may even provide the answer to the well-known question "Where can I get a list of all the Organic Chemists in Europe?".

Eventually it will become the master copy for much of the data it contains.

UKnet Overview, Accounting and Future plans

Peter Houlder

Computer Laboratory
University of Kent
Canterbury, UK
uknet@ukc.ac.uk

This paper is in three parts. The first part is essentially a repeat of old information, which can be skipped by regular attendees. It deals with the internal aspects of UKnet as a network; what it does, how it is administered, services offered costs etc. The second part tries to shed some light on the complexities of costing network services. The final part covers the UKC viewpoint on TCP/IP and OSI developments.

ISODE — Who, What and Where Next

Julian P Onions

Communications Research Group
Nottingham University

ISODE is the ISO development environment. It is a collection of libraries and application programs which together provide the upper layers of OSI. ISODE currently runs on most versions of UNIX. ISODE is openly available, which means it is usable by anyone for anything except that none of the authors takes responsibility for use or misuse of the product. It is available on a tape for a copying charge or can be retrieved directly by a suitable transfer protocol. The principal author of the ISODE is Marshall T Rose now with Nysernet.

However, since the first release, there have been many contributors to the code and it is hoped that this will continue in the future.

An Overview of Application-Level Services in the Internet Protocol Suite

Jim Reid

Department of Computer Science
Strathclyde University
Glasgow
Scotland
G1 1XH
jim@cs.strath.ac.uk
jim@strath-cs.uucp
.../uunet/mcvax/ukc/strath-cs/jim

The Internet Protocol family, often referred to as the TCP/IP protocols, have been widely adopted for local area and wide area networking. This has been a result of the evolution of the US ARPAnet and the development of protocol implementations for most of the commonly used operating systems. The increasing popularity of UNIX workstations has also meant an increased use of the Internet protocols, often in environments that have previously used proprietary protocols.

This paper presents a short introduction to the Internet Protocol architecture. A brief description of the main services that are built on top of the available transport service protocols is given. These include the obvious facilities like mail and file transfer (FTP and SMTP respectively) and a virtual terminal service (telnet). UNIX-specific services including system logging, remote command execution and network routing will also be discussed.

The Internet protocols are also used to specify experimental facilities that can be developed into fully-fledged production systems. Two common examples are Sun's Network File System (NFS) and Remote Procedure Call (RPC) mechanism. More recently, the Internet has switched to a domain based name service for address and name lookups.

In addition to providing these general network services, the Internet protocols are also used as the basis of both the X-windows and NeWS windowing systems. Other "interesting" facilities that can be provided include a network time-of-day service and a face server.

Speakers Bureau

The USENIX Association is seeking volunteers to participate in a Speakers Bureau. The purpose of the Speakers Bureau is twofold: to provide a forum for people with expertise in various areas of UNIX to share that knowledge with others; and, to provide a source of speakers for educational groups (high schools, colleges, universities, local user groups) who could discuss a variety of UNIX-related topics in a colloquium-type setting.

We would like the Speakers Bureau to have a local flavor where the volunteers and interested groups draw from within the local/regional community. In cases where round trip travel of greater than 50 miles is involved, USENIX would provide a mileage allowance.

Before you discard this idea, thinking you are too busy, please consider volunteering to speak just once or twice during the year. We are sure you will find this a rewarding experience and a great service to the audience.

When a request for a speaker is received, the Speakers Bureau will try to match the requirements with a particular volunteer's expertise. If you are contacted, and cannot speak, please feel free to decline. We want you to accept only those invitations which you truly want to do.

Should you decide to lend your expertise to this project, please fill out the Information Sheet and return it as soon as possible. We would like to promote this concept, but need a collection of speakers before we can announce its existence to potentially interested groups.

If you do not wish to volunteer, but know of someone within the UNIX community who might be interested in participating, please fill out the relevant part of the Information Sheet, and we shall contact that person directly. If you know of a group that would benefit from some of the resources that may become available, please let us know.

John L. Donnelly
Coordinator
USENIX Speakers Bureau

Speakers Bureau Information Sheet

Name: _____ Title: _____

Organization: _____

Address: _____

_____

Phone (work): _____ (home): _____

email address: _____

Do you have any logistical limitations where/when you can speak?

Geographical area: _____ Day of week: _____

Length of time: _____ Time of day: _____

Do you have any group preferences (high schools, colleges, user groups)?

_____

_____

How often would you make yourself available to speak?

_____

What are your particular areas of expertise?

_____

_____

Give a brief description of yourself (educational background, current position, previous speaking experience, etc.)

_____

_____

Do you know of any other people who might be interested in being involved in the Speakers Bureau (in any capacity – give name[s], phone number[s])?

_____

_____

Are you interested in using the resources of the Speakers Bureau for your group or organization? If so, what topics would you be interested in?

_____

_____

Please return this form to:

Speakers Bureau
USENIX Association
5398 Manhattan Circle, Suite 201
Boulder, CO 80303

303-499-2600
303-499-2608 (FAX)
johnd@usenix.org

UUCP Project Draws Strong Response

As discussed in the last issue of *;login:*, USENIX is studying *uucp* to see whether we can help promote better communication, in a literal sense, by activities which might range from standardization up to a possible sponsored implementation.

We have received more than 30 mail messages on this topic from Europe, Australia, and the U.S. Many of these were cheers and bravos, and most requested further information. A number were from people who were actively working on *uucp* itself, or on programs that were similar to, or "better" than *uucp*. In particular, there was relevant work going on at AT&T, GNU, in Australia with ACSnet, in Great Britain with UKUUCP, and at Prime Computer in Australia, with MYL. A program called PP, which supports numerous protocols, is in the beta stage in Great Britain, and will be "openly available." Finally, Rick Adams, has been doing advanced CPR on *uucp* for years to keep *uunet* running smoothly, and has suggested that he might make this available.

There appear to be three major decision areas (battlegrounds?). One is technical – what do we want, given that we can't have everything. Some people wrote and suggested that using anything other than streams and TLI was senseless and short-sighted; others wrote that the use of streams and TLI would lock us out of a large number of smaller and older machines, and should be avoided at all cost. I personally would like to have some graphic (X-ish) administration interface so I can add a phone number without screwing up the company-wide system for days; but this limits our communication scope.

The next set of decisions has to do with the distribution of the system; will it be free, or merely cheap. The majority think it should be distributed in source code; a vocal minority paint a picture of a totally snarled net created by enthusiastic hackery by hundreds of monkeys at their terminals. Some think it should be licensed, others totally free.

The related problem is how to get it done; should USENIX endorse, support, initiate, or purchase the work? There are a lot of touchy issues here, including what the cost would be, how it would be recovered (hold on to your wallets, members!), and how USENIX would ensure that it got its money's worth (ever try to manage a software project with volunteers?).

The USENIX board of directors will be taking this topic up again at its meeting in Washington, D.C. We shall report on the outcome from that meeting in the next issue of ;login:. I haven't really done justice in this summary to some of the lengthy, thoughtful comments that we have received; thank you for the encouragement and the information. Further comments and suggestions can be sent to *scj@usenix.org* or discussed with other USENIX board members.

Steve Johnson

Book Reviews

UNIX System Software Readings
AT&T UNIX Pacific Co., Ltd.
(Englewood Cliffs, NJ: Prentice-Hall, 1988, ISBN 0-13-938358-1)

Reviewed by George W. Leach
AT&T Paradyne

This book records the presentations made by six invited speakers from AT&T at a seminar held in Japan in the summer of 1986. The seminar was sponsored by AT&T UNIX Pacific (ATTUP) in order to disseminate information concerning new technologies for System V Release 3.0 to the Asian/Pacific UNIX community. The contents of the book are as follows:

1. Larry L. Crume, *Introduction*
2. Brian W. Kernighan, *Beyond UNIX* (Keynote Speech)
3. Gilbert J. McGrath, *Streams Technology*
4. Laurence M. Brown, *Networking Architecture & Protocol*
5. Arthur L. Sabsevitz, *Distributed UNIX System - Remote File Sharing*
6. Gary L. Lindgren, *Directions in Internationalization*
7. David G. Korn, *The Shell - Past, Present, and Future*

A great deal of the material presented here has not been readily available to the general public. For example, Kernighan's presentation covers the activities of the Computer Science Research Center at Bell Labs in Murray Hill. It has been several years since information has become available on much of the UNIX-related work going on there.[1,2] However, some of the current focus is shifting away from Research UNIX and the Blit towards Plan 9 and the Gnot.[3,4,5]

The papers by McGrath, Brown, and Sabsevitz describe the System V approaches to networking and distribution. Once again, very little has been written about Streams and the Transport Level Interface (TLI) for networking with System V.[6,7] Descriptions of RFS are hard to come by as well.[8] For someone new to System V these papers make an excellent, albeit brief, introduction to these facilities.

Some of the internationalization support that is described in Lindgren's paper will be incorporated into SVR4. At the time of the seminar, the Japanese Application Environment (JAE) 1.0 was available from ATTUP. However, today it is not even the current product offering, which will be superceded by SVR4. Still, the background material on internationalization is quite relevant and makes the paper worth reading. In fact, the existence of this paper in this collection is what attracted my attention to the book.

David Korn's paper on the UNIX Shell provides a nice history of the development of everyone's favorite command interpreter as well as an overview of its functionality. And, of course, the Korn Shell is featured in the paper. There are some comparisons of the capabilities of the Bourne, C, and Korn Shells as well, but they are minimal.

The viewgraphs from the seminar presentations are included as figures for each paper. This is a nice touch. Often folks will request copies of viewgraphs from a presenter despite the existence of a paper in the proceedings.

Overall, my impression of this collection is that for the experienced UNIX programmer the material is too elementary. The topic coverage is mostly at a conceptual level and of a tutorial nature. Furthermore, the bulk of the material is specific to System V. BSD fans will not be interested. And finally, the presentations are based upon a release of System V that is about to be eclipsed. Some of the areas covered in the book are due for some new features, for example an out-of-band communications capability will be added to Streams.[9]

There is, however, some worthwhile material in this collection for UNIX folks of all persuasions. The paper by Kernighan is probably the most concise description of the work

that the folks in Murray Hill have been involved in over the past several years. The paper on internationalization does present a nice overview of the problems involved with working on software with an eye towards the world marketplace. And David Korn's paper presents a unique perspective in the command interpreter arena. The papers are compact and to the point, much like the book (182 pages). And the list price of $21.95 won't break your budget.

References

1. AT&T, "The UNIX System," AT&T Bell Laboratories Technical Journal, Vol. 63, No. 8, Part 2, October 1984.

2. Members of CSRC, Various papers describing 8th Edition UNIX, Summer 1985 USENIX Conference Proceedings, Portland OR.

3. Presotto, Dave, "Plan 9 from Bell Labs – The Network," EUUG Spring 1988 Conference Proceedings, London UK.

4. Pike, Rob, and Thompson, Ken, "Position Paper for IEEE Workshop on Operating Systems," Workshop on Workstation Operating Systems, Cambridge MA, November 5-6, 1987.

5. Pike, Rob, "Why Workstations Don't Work," Summer 1989 USENIX Conference, Baltimore MD.

6. AT&T, "STREAMS Programmer's Guide," UNIX System V Release 3 Manual, Issue 1, Order Code 307-227, 1986.

7. AT&T, "Network Programmer's Guide," UNIX System V Release 3 Manual, Issue 1, Order Code 307-230, 1986.

8. Padovano, Michael, and Scheer, Michael, "UNIX System V Remote File Sharing (RFS)," Summer 1989 USENIX Conference Tutorial Notes, Baltimore MD.

9. Rago, Stephen, "Out-Of-Band Communications in STREAMS," Summer 1989 USENIX Conference Proceedings, Baltimore MD, pp. 29-37.

Elements of Computer Music

By F. Richard Moore

Prentice-Hall (1990) 546 pages
Foreword by Max Mathews

Reviewed by Mike Hawley
MIT Media Lab

In the spectrum of "Elements" books over the centuries, Moore's is rather far from the slim Strunk & White or Kernighan & Plauger end, and pushing decidedly in the direction of Euclid (13 volumes). Computer music is a highly synthetic field – making good music with computers necessarily requires competence in programming (of all kinds, including systems, numerical algorithms, and interfaces), mathematics, signal processing (with an audio bent), perhaps acoustics, probably psychology, and, from time to time, possibly even music. Considering all that, TECM is remarkably concise, yet still touches on a wealth of fine points. Because computing is a profession of mine, and music (including academic computer music) a consuming avocation, I tend to read books of these kinds more for new tidbits than old examples. But the book covers a balance of topics that should resonate well with novices, amateurs, and head-banging die-hard computer music groupies:

Introduction (Musical Data and Process; Musical Thought; Composing; Performing; Instruments; Rooms; Listening; Disciplinary Context; Prerequisites);

Digital Audio (Sound Representations; Sound Digitization; Spectrum Measurements; Digital Filters; Summary);

Instruments (Representing Instruments; cmusic; Additive Synthesis; Subtractive Synthesis and Physical Models; Nonlinear Synthesis; Summary)

Rooms (Concert Halls; Spatial Hearing; Early Echo Response; Reverberation; Sound Spatialization; Other Issues; Summary)

Composing (Computer-Mediated Composition; Music Representations; Random Numbers; Random Sieves; Markov Processes; Noise, Filters, Random Numbers, and Probability; Compositional Algorithms)

Appendices – Mathematics; Units of Measure; Tuning; CMUSIC

Index

Computer music has been attached to (some would say mired in) the domain of signal processing for years, so it's not surprising that the middle three chapters (digital audio, instruments, signal processing) involve this sort of theory and its applications and implementations. In fact, the book actually contains a good bit of the signal processing that every liberal-minded person ought to know and might almost make a workable introductory text for that field. There is a nice perspective on the streams of work done in the field, and extensive examples in the "cmusic" language (a dialect of C with add-ons for music and signal processing). For novices, the fact that so much working example code is presented through C and cmusic is likely to be a boon – so long as the book comes with a diskette, which it apparently does not. Moreover, a book about music without audio illustrations is like an architecture book without pictures of buildings (or a *Kama Sutra* without ...). There seems to be no demo record with this book. It is an expensive undertaking – if not an out-and-out financial loss – to print a computer music book, but an attached recording, particularly if it contained some demos by Prince of sampling synthesis applications or artificial ambience, say, would not only perk up the content, but might even boost the sale if it contained a new song. In fact, although the book does a nice service for the field, I suspect and hope that in five years, if not less, it will be preferable to embody this kind of survey in a computer ware, with running examples. My guess is that the advent of media-rich publishing systems, combined with the recent dramatic shift of attention away from signal processing and more towards content processing (e.g., MIDI), will make this text seem a bit quaint by 2000 A.D.

Readers who would prefer to chaw on the gritty papers that populate the field can survey the *Computer Music Journal* over the years, or perhaps dive into a collection like "Foundations of Computer Music" (Curtis Roads and John Strawn, eds., MIT Press, 1985), which high-pass filters some of the esoterica. Although "great" computer music is tough to find, the field is definitely *not* short on printed matter. These disclaimers aside, though, Moore's book is nevertheless as clear a survey as one could wish for. He is, as Mathews points out in the foreword, not only a technical founder, but a fine teacher and writer. But perhaps better than that, he is also a bona-fide UNIX hacker. Many readers of *;login:* will enjoy tracking some of the familiar systems trails into somewhat new territory.

An Update on UNIX and C Standards Activity

Jeffrey S. Haemer
Report Editor, USENIX Standards Watchdog Committee

Report on IEEE 1003.0: POSIX Guide

Kevin Lewis <klewis@gucci.enet.dec.com> reports on the October 16-20, 1989 meeting in Brussels, Belgium:

Dot Zero's mission in Brussels was to step back and review where the group had been and where we needed to go. We learned that we are headed in the right basic direction but still need to make some course corrections.

There are two major contributors to this state of affairs. First, an honest review of the pre-Brussels document reveals that it still has significant holes. Also, its format is hard to follow. I must admit that it felt good to see unanimous consent on both the need to reorganize the document and on a new format. It does a co-chair's heart good to see two such rare events occur concurrently. The reformatting of the draft guide will be complete by the January meeting in New Orleans. The group will then review components of the document that are sufficiently complete section-by-section and line-by-line.

Second, Dot Zero faces a problem that is becoming widespread in 1003 and TCOS-SS: a serious dilution of effort. Little did Dot Zero realize, when it recommended the formation of a group to address a windows standard (now 1201), that we would lose people who had been shepherding key components of the Dot Zero guide. The new efforts have left us with no one to cover networking, graphics, or windows, though it's possible that new folks in these areas will join us in New Orleans. [Editor's note: Listen to this man. What are your ideas about open systems in these areas? If you have something useful to contribute, please contact someone on Dot Zero : Kevin. Don't wait until it's too late and then complain about the result.]

Regarding internationalization (for which the current buzzword is "I18N," because there are eighteen letters between the 'I' and the 'N'):

Everyone who attended the I18N study group meeting sponsored by Dot Zero found it interesting when the question regarding the group's future was posed. All those present tacitly agreed that it would not be in the best interests of I18N efforts for this study group to become a full-fledged working group. This study group would best serve the industry as a forum for issue flagellation, soap-boxing, and formation of proposals to the appropriate accredited bodies. At the appropriate time, the I18N group will declare that its time is up.

When the question of identifying the major contributors to the I18N efforts arose, I noticed an effort of OSF to remain at arm's distance from X/Open. In light of OSF's membership in X/Open, this might signify its desire to maintain its own identity.

That's enough negatives. Is there an upside to all this? Yes, absolutely. We have a reorganized document that will ease and streamline the review process. We now have the eyes of the industry and the press looking over our shoulders, eager to read our guide. We are reaching the point where fear of personal and professional embarrassment is motivating those who have an interest in this effort's succeeding. These will help us meet our goal of preparing a draft for review and comment by ISO by the Fall of 1990.

Report on IEEE 1003.1: System Services Interface

Mark Doran <md@inset.co.uk> reports on the October 16-20, 1989 meeting in Brussels, Belgium:

P1003.1 is now a full-use standard, so interest in attending the working group has waned somewhat. Attendance didn't get above fifteen or so all week and was nearer half a dozen most of the time. Even so, this was a bit low by comparison with recent meetings. So where was everyone?

[Editor's note – Notice that this is larger than the attendance at typical meetings of, for example, dot nine. "Low attendance" is relative. Author's additional note – And that's the frightening thing; standards are being established by as few as half a dozen *individuals.* This cannot be representative or balanced. Scary stuff, "...as we take you on a journey, into the Standards Zone..."]

We thought that meeting in Brussels was going to further the cause of international participation in the POSIX process. Several people I would normally expect to see, didn't show; Europe may be too far from home for a lot of the regulars. Unfortunately, I didn't see more than two or three Europeans (whom I would not normally expect to see at POSIX) all week either. Oh well, I'm sure it was a good idea really... So what did those that showed get up to?

ISO 9945. [Editor's note – ISO 9945 is, roughly, the ISO standard engendered by and corresponding to the POSIX work.]

It looks like 9945 is going to be split up into three major pieces, the first of which is founded upon the IEEE P1003.1-1988 standard. This piece is likely to include all the other system interfaces as well (notably, the real time stuff from P1003.4). The other two pieces will be based around utilities and system administration.

The basic IS9945-1:1989 will be just the same as the regular, ugly-green, dot-one book – well almost. ISO has yet another documentation format, and the book will have to be squeezed to fit it. This one doesn't allow line numbers either. We are assured that making the changes is not a major problem, but the working group has still requested a new disclaimer telling readers that all mistakes are the fault of ISO!

P1003.1a. [Editor's note – This document (supplement A) is supposed to contain clarifications of and corrections to P1003.1-1988, but no substantive changes.]

The meeting discussed resolution issues from the first ballot. Highlights included:

- the decision to withdraw the *cuserid()* interface; its loss will not be sadly mourned since one can use other interfaces to do the same job better.

- the addition of a new type name `ssize_t` to represent signed `size_t` values; this has all sorts of uses – for example, in the prototype for *read()*. Currently, the parameter specifying the number of bytes to be *read()* is given as a `size_t`, but *read()* has been specified to return an `int`, which may not be large enough to hold a `size_t` character count. Moreover, *read()* may return -1 for failure; or the number of characters read if the call was successful.

The recirculation ballot happened between November 10-20, 1989; if you care but didn't know that already, it doesn't matter because you (and many others, I suspect) have missed your chance. This all seems a bit fast but it does mean that P1003.1a will hit an ISO, six-month, letter-ballot window.

Transparent File Access. Isn't this a P1003.8 problem? Yes, but the chairperson of the TFA committee came to talk about TFA semantics as they relate to P1003.1.

The crux of the matter is that the six TFA folks seem to have decided that standardizing NFS will do nicely for TFA. Their chairperson wonders whether the rest of the world (or, more accurately, the balloting group for a TFA standard) will agree.

The answer from the dot one folks appears to be definitely not (thank goodness)! There appear to be several arguments against NFS as the TFA standard from dot one. These include:

- It is impossible to maintain full dot one semantics over a network using NFS. Consider the last-close semantics, for example, which can only be preserved over a network using a connection-oriented protocol, which NFS is not.

- Transparent File Access should be *transparent;* NFS isn't. It is possible for operations that are logically sound to fail because of network timeouts.

- NFS is a *de facto* standard; why should it get an official rubber stamp?

This appears to be a hot topic that many groups may have an interest in, so there will

be an "out-of-hours" meeting on TFA at the New Orleans POSIX – [Editor's note – If you do care, we suggest either writing directly to the TFA chair, Jason Zions <jason@hpcndr.cnd.hp.com>, or posting your opinions to comp.std.unix.]

Language-Independent Specification. It seems to have been decided that POSIX API standards should be written in a language-independent form, i.e. not expressed in C-language constructs.

My initial reaction was one of horror, but then someone pointed out that C is not the only programming language in the known universe. This I have to concede, along with the idea that bindings to POSIX APIs in other languages may not be such a bad idea after all. Indeed work is well underway to produce FORTRAN and Ada bindings.

But now it seems we have to express POSIX in a language-independent way. When you come to write the next set of actual language bindings, the semantics won't be clouded with language-dependent stuff; the idea is that you won't have to understand C in all its "glory" to write new language bindings.

So what will the language-independent specifications look like? Will I be able to understand those? The current proposal doesn't look like anything I recognize at all. Yes, that's right, we have to learn a whole NEW language (sigh). Why not use a more formal specification language that a few people know? (Like ASN.1 for example, which P1003.7 has decided to use.) Better yet, why not use constrained English? Since the FORTRAN and Ada bindings folks have managed without the aid of language-independent specifications, why can't everyone else? Is there more to this than a glorified job creation scheme? ("Wanted: expert in POSIX 'language-independent' language...") If there is, do we have to invent a new wheel to get the job done?

As you can tell, my opinion of this effort is somewhat jaundiced. Perhaps, I have missed the point. Maybe so, but if I have, I feel sure that some kind soul will be only too happy to correct me in "flaming" detail.

Messaging. The UniForum internationalization (I18N) folks brought forward a proposal for a messaging facility to be included in P1003.1b. The working group decided that it needs some more work but will go into the next draft.

[Editor's note – The problem being solved here is that internationalized applications store all user-visible strings in external files, so that vendors and users can change the language of an application without recompiling it. The UniForum I18N group is proposing a standard format for those files.]

P1003.1b. Work on production of the second supplement is still at a formative stage. The group is still accepting formal proposals for functionality to add to the document. Where P1003.1a has been drawn up as a purely corrective instrument, P1003.1b may add new functionality. Among the interesting things currently included are these:

- The messaging proposal described above.
- A set of interfaces to provide symbolic links. The basic idea is that *lstat()*, *readlink()* and *symlink()* operate on the link, and all other interfaces operate on the linked-to file.

Rationale will be added to explain that it is a unique directory, which is the parent directory in the same physical file system. This means that *cd* does not go back across symlinks to the directory you came from.

This is the same as the semantics on my Sun. For example:

```
(sunset) 33 % pwd
/usr/spare/ins.MC68020/md/train
(sunset) 34 % ls -ld ./MR_C++
lrwxrwxrwx  1 root  32 Sep 30  1988 MR_C++
   -> /usr/sunset/md/c++/trainset/c++/
(sunset) 35 % cd MR_C++
(sunset) 36 % pwd
/usr/sunset/md/c++/trainset/c++
(sunset) 37 % cd ..
(sunset) 38 % pwd
/usr/sunset/md/c++/trainset
```

The rationale is meant to help keep readers aware of what's really written in the standard and help them avoid misinterpreting it along lines of their own potential misconceptions.

• P1003.1b used to have two descriptions of Data Interchange formats. Now it has only one. The working group picked the one that remains because it more closely resembles existing standards in the area. In particular, the surviving proposal refers to X3.27.

Report on IEEE 1003.4: Real-time Extensions

John Gertwagen <jag@laidbak> reports on the November 13-17, 1989 meeting in Milpitas, CA:

Background

The P1003.4 Real-time Working Group, began as the /usr/group technical committee on real-time extensions. About two years ago, it was chartered by the IEEE to develop minimum extensions to POSIX to support real-time applications. Over time its scope has expanded, and P1003.4 is now more a set of general interfaces that extend P1003.1 than a specifically real-time standard. Its current work is intended to support not only real-time, but also database, transaction processing, Ada runtime, and networking environments. The group is trying to stay consistent with both the rest of POSIX and other common practice outside the real-time domain.

The work is moving quickly. Though we have only been working for two years, we are now on Draft 9 of the proposed standard, and expect to go out for ballot before the end of the year. To help keep up this aggressive schedule, P1003.4 made only a token appearance at the official P1003 meeting in Brussels. The goal of the Milpitas meeting was to get the draft standard ready for balloting.

Meeting Summary

Most of the interface proposals are now relatively mature, so there was a lot of i-dotting and t-crossing, and (fortunately) little substantive change. The "performance metrics" sections of several interface chapters still need attention, but there has been little initiative in the group to address them, so it looks like the issues will get resolved during balloting.

The biggest piece of substantive work was a failed attempt to make the recently introduced threads proposal clean enough to get into the ballot. The stumbling block is a controversy over how to deal with signals.

There are really two, related problems: how to send traditional UNIX/POSIX signals to a multi-threaded process, and how to asynchronously interrupt a thread.

Four options have been suggested: delivering signals to a (somehow) privileged thread, per Draft 8; delivering a signal to whichever thread is unlucky enough to run next, a la Mach; delivering the signal to each thread that declares an interest; and ducking the issue by leaving signal semantics undefined.

We haven't been able to decide among the options yet; the working group is now split evenly. About half think signal semantics should follow the principle of least surprise, and be fully extended to threads. The other half think each signal should be delivered to a single thread, and there should be a separate, explicit mechanism to let threads communicate with one another.

Personally, I think the full semantics of process signals is extra baggage in the "lightweight" context of threads. I favor delivering signals to a privileged thread – either the "first" thread or a designated "agent" – and providing a separate, lightweight interface for asynchronously interrupting threads. On the other hand, I would be happy to see threads signal one another in a way that looks, syntactically and semantically, like inter-process signals. In fact, I think the early, simpler versions of *signal()* look a lot like what's needed (around V6 or so). I don't care whether the implementation of "process" and "thread" signals is the same underneath or not. That decision should be left to the vendor.

Directions

Draft 9 of P1003.4 is being readied for ballot as this is being written and should be distributed by mid-December. With a little luck, balloting will be over by the Summer of '90.

Threads is the biggest area of interest in continued work. The threads chapter will be an appendix to Draft 9 and the balloting group will be asked to comment on the threads proposal as if it were being balloted. Unless there is a significant write-in effort, the threads interface will probably be treated as a new work item for P1003.4. Then, if the outstanding issues can be resolved expediently, threads could go to ballot as early as April '90.

With the real-time interfaces defined, it looks like the next task of this group will be to create one or more Real-time Application Portability Profiles (RAPPs?) that define how to use the interfaces in important real-time application models. Agreeing on the application models may be harder than agreeing on the interfaces, but we'll see.

Report on IEEE 1003.6: Security Extensions

Ana Maria de Alvare <anamaria@lll-lcc.llnl.gov> reports on the October 16-20, 1989 meeting in Brussels, Belgium:

The security working group worked the full week, discussing the draft. The meeting's primary goal was to approve the current draft for distribution to the international working groups. It was presented, at the EEC, to new members of the group from the European countries.

TRUSIX

A representative from TRUSIX, Charles Testa, gave a progress report on TRUSIX. [Editor's note – TRUSIX is an organization sponsored by the National computer security center (NCSC), developing a secure UNIX model. The participants are a number of vendors developing secure UNIX implementations.] Their modeling subcommittee has nearly completed a formal model describing the UNIX file system. They have accepted the "Ina Jo" model to describe Trusted UNIX System Interfaces. This model revolves around a MAC-read criterion, MAC-writes and a DAC constraint, and consists of simple security properties, confinement properties, and discretionary security properties representative of the Bell-LaPadula model. The TRUSIX ACL Rationale and Working Example Document has been approved by the NCSC and is being reviewed for publication under NCSC security publications.

One topic of interest to all security readers is prevention and/or detection of covert channels. TRUSIX is planning to include this under the Audit Rationale Document, which will include examples of typical covert channels and their implications. Issues such as bandwidth evaluation will be addressed by a separate white paper.

POSIX Conformance Testing

A representative from 1003.3, the POSIX Conformance Testing group, presented 1003.3's goal – to establish a series of specifications for testing the different POSIX standards. Although they have written the pseudo-code to test the conformance of a system to 1003.1, they feel they lack the staff and expertise to produce such tests for all the 1003 groups. Given this, their current plan is to draw upon each group for expertise and background knowledge of the subject at hand, and join those skills with their testing skills to produce a test bed for each 1003 standard.

Their ultimate goal is to allow testing of all elements of an open system for POSIX conformance by defining common test methods, which can then be implemented by private industries as test suites. They explained how to list the assertions, how to classify them, and what information they will need to produce a test method for 1003.6.

One factor mentioned was that the description has to address a single unit of behavior expected of a conformant system at a time. This implies dissecting interfaces into separate groups of assertions and generating assertions for both semantic and syntactic descriptions.

Discretionary Access Control (DAC)

The group focused on polishing and adding information to the draft. It was suggested the standard shouldn't define the behavior of *chmod* when old programs are being executed with the DAC mechanism.

It was noted that the current proposed Access Control List (ACL) might not be fully compatible with the current behavior of a *chmod* call. With the current, old-style behavior, when *chmod* is used to change owner, group and/or other permissions, the changed permissions represent the access modes of the file. In the current proposal for ACL, a *chmod* will provide the old behavior for the "owner" and "other" fields, but the "group" field permissions as set by *chmod* and shown by *stat*, wouldn't represent the actual access permissions of the group associated with that file; instead, it's a summary of all access permissions included under the ACL list for group entries. In other words, it would represent the maximum permissions allowed to the group entries included in the ACL list.

Some participants argued that *chmod* should be replaced by other system calls in a system conforming to 1003.6. In other words, if your system were to conform to 1003.6 the behavior of *chmod* would be unspecified and unpredictable.

I disagree. Although defining the behavior of *chmod* might restrict some implementations of ACLs, having a well-defined *chmod* behavior will provide backward compatibility and ease porting old programs to 1003.6-conformant systems. Otherwise old programs will have to be ported to platforms with system-dependent representations of *chmod* and *stat* information.

It was also proposed that the ACL list should allow entry types like timestamping. This would allow a policy that is more restrictive than the DOD orange-book policy to provide more granularity of file access.

Mandatory Access Control (MAC)

Kevin Murphy, of British Telecom, presented his views on electronic mail label usage and proposed that such a mechanism should be used as part of the standard. The electronic mail security labels consist of a generic format that includes four major components: security policy id, security classification, privacy mask, and security categories. This approach to labels is implemented on X.400 security services. One clear advantage of using such a format for labels is that it maximizes the potential synergy between operating system and electronic mail labels.

Chris Hughes from ICL presented British views on MAC information labels. Its main characteristics are these: object creation initializes the label, the label is implementation defined and changed by the owner, and the label is not used for access control. Chris proposed that the standard should provide a get/set mechanism for the object information label, and a way to merge and translate information labels, but should not standardize the labels' contents.

Information labels are useful because they provide added information on particular objects. We concluded that information labels should be in the scope of MAC as part of the standard, and requested that MITRE provide a presentation on information label use at the next meeting.

Privileges

The whole group heard a presentation of the internal draft of the privileges document. We decided that the wording had problems. The draft interface description is too obscure and needs a simpler description of privilege interfaces, before it can be included in the 1003.6 draft document.

Although the group argued considerably about the wording, they seemed to agree on the concepts. The main points are that privilege is associated with a process and privilege attributes can be attached to files.

I do not think I should burden the reader with the brainstorming ideas of the privilege group until a firmer position is taken at the next meeting. One thing I can say is that the process privilege concepts described in my last report (permitted, inheritable and effective) still stand, and a file still has three types of privilege attributes.

Audit

Kevin Brady from AT&T and Doug Steves from IBM presented a combined proposal, produced by them and Henry Hall from DEC, on how to define audit interfaces for 1003.6. Their proposal was meant to

contest the current audit stand, led by Olin Sibert from Oxford Systems.

The current audit definition is based on the token concept and on a pure procedural interface. In the procedural interface all data manipulation of the audit record is performed by function calls, with data passed explicitly through function parameters. Although this sounds attractive and clear, in practice it requires many function calls.

Another major point of controversy was the audit trail format. In Olin's proposal, conversion cost is minimal because writes and reads require an explicit specification of the format wanted. In Kevin, Doug, and Henry's proposal the conversion function is set to one of three conventional formats: little endian, big endian, or XDR. In other words, the information is stored in machine-dependent format while Olin's chooses a uniform format for all information stored.

One last contested feature was the ability to rewind audit trail information when viewing it. Kevin, Doug, and Henry's proposal does not allow a rewind, since information is manipulated at the data structure level.

Because of the heated discussion of procedural versus data structure interfaces for POSIX, both proposals will be formally written out, removed from the draft, and presented in the next meeting for a final vote.

Report on IEEE 1003.8/1: POSIX Transparent File Access

An anonymous correspondent reports on the July 10-14, 1989 meeting, in San Jose, California:

[Editor's note – Though this report came in substantially later than the other July reports, I think it's still useful, provocative, and well worth reading.]

Overview of New 1003.8 Developments

1. All standards produced by POSIX committees (with the exception of language-binding standards like 1003.5 and 1003.9) must be specified in a language-independent fashion, and must include at least one language-specific binding. Since P1003 will not have guidelines and rules for constructing a language-independent specification before April 1990, no POSIX networking group can possibly ballot a document before July 1990. "Mock" ballots (aka trial-use ballots) are unaffected by this, but their usefulness will be diminished.

2. Two new POSIX Networking working groups either have submitted or will soon submit PARs to begin work, bringing the total number of POSIX Networking working groups to six. These new groups are the Name Space and Directory Services Interface and the X.400 Mail Gateway Interface. [Editor's note – The SEC approved the PAR for the former, but decided that the latter transcends POSIX, and recommended that the IEEE form a separate, numbered group, analogous to 1003 or 1201, to handle X.400. See below.]

3. Due to the rules governing IEEE and IEEE-TCOS standards activities, as well as the logistical nightmare six sub-working groups cause, the hierarchical structure of the P1003.8 POSIX networking committee will be flattened out, with each current sub-group submitting PARs to cover their current work. Coordination will be provided by a "POSIX Networking Steering Committee," made up of the chairs and vice-chairs of each networking-related working group and the current chair and vice-chair of 1003.8. [Editor's note – This is still being debated by the SEC.]

4. Since each of the 1003.8 sub-groups will be submitting separate PARs, the P1003 Sponsor's Executive Committee (SEC) is taking the opportunity to evaluate the degree to which each PAR is intended to represent a part of the "POSIX Environment" or is a component which has no relationship to the rest of POSIX and should, hence, stand alone. The result of this is that several of the 1003.8 sub-groups may be issued project numbers outside of the P1003 family. There is some precedent for this; the X11 interface group was assigned project number P1201 for just this reason.

Activity in the TFA Subgroup, P1003.8/1

The group is making slow but steady progress towards the goals of a fully-specified programmatic interface for networked file

access and the semantics and suggested syntax for administrative interfaces for such a functionality. The group is dominated by a faction, apparently led by Sun Microsystems, with a goal of ensuring that NFS, in some form, is a sufficient mechanism to provide the service required by the "full TFA" interface. The balance of the committee is composed of people who simply want a standard they can use as an acquisition tool.

Achievements

- Enough consensus and material was obtained to permit development of a first draft of the programmatic interface part of the specification; this draft should be complete in time for the second mailing, due out on September 8.

- Liaison activities with 1003.7 (System Administration) continued; .7 indicated that all of the options for the TFA *mount / unmount* model were, in fact, of use in administering such a system. They also agreed that they owned responsibility for all file-system commands not completely unique in function to TFA, and that they would pursue input from the TFA group when the time was right.

- Further progress was made on identifying status and usage information which must be obtainable from the provider of a TFA service.

Problem Areas

1. Representation in the group is unbalanced; there is, as of this time [Editor's note – July, '89], no substantial representation of the "stateful" side of the semantic issues. The chairman has, so far, been unsuccessful in encouraging a more balanced group viewpoint so representation from the stateful camp has been solicited (with minimal success) for future meetings.

2. Several "grey areas" in the semantics of IEEE 1003.1-1988 have been identified by the TFA group over the last several months. The suggested "fixes" have been slanted in a way that would let the TFA interface avoid relaxing 1003.1 semantics while permitting a stateless implementation of the TFA service; i.e. rather than strengthening 1003.1 to define the actual behavior of a single stand-alone system, the proposals have been so weak that they under-constrain single-system behavior. It appears that the majority of the 1003.1 committee will not approve of this approach, and will require the "fix" to be of the proper strength to correctly specify single-system behavior.

Let me give an example. The original 1003.1 standard is silent on the issue of when the effects of a write are visible to a subsequent read of the same byte of the file. If process A writes byte 123 of file *foo*, then process B does a read of byte 123 of file *foo*, at what point is B guaranteed to see the byte A wrote?

Immediately? If so, stateless solutions employing read caches fail; if process B is remote on system "bsys" and reading the file via NFS, byte 123 might come out of the file cache on bsys and not from the file cache on the system where A lives.

Immediately if A and B are on the same system, and at some implementation-defined time otherwise? This requires 1003.1 to define what it means by "the same system," and doesn't adequately address multi-processor single systems with "interesting" caches. It also means a truly portable application that is interested in running in a distributed environment can **never** know when the byte written by A is visible to B.

Only in the presence of byte locking? In other words, A locks byte 123, writes it, unlocks it; if B then locks and reads 123, it is guaranteed to see what A wrote. Not a bad solution, but it breaks existing applications and in fact is a relaxation of the intended semantics of 1003.1.

Basically, the "intent" developing in 1003.1 is that the effect of the write must be seen immediately by *any other process with that file open,* without regard to process location, without recourse to O_SYNC mode opens, without the necessity for locking, and so on. 1003.1-1988 is silent on the issue; the proposed fix from TFA (basically a compromise I did not like and knew would fail) was that read-after-write be guaranteed only for co-located processes and in the presence of locking. This gave 1003.1 a chance to avoid relaxing their intended semantics while leaving TFA a "loophole" to change semantics

without having to indicate a change in wording from 1003.1.

This is what got rejected by 1003.1, which is getting pretty damned tired of TFA's trying to claim that the full TFA semantics are "just like" 1003.1 but with gaping differences that are introduced silently due to weak or weasel wording in 1003.1-1988.

3. 1003.7, System Administration, is making distressingly slow progress. If this continues, 1003.8 will have two choices with respect to client-side administrative commands:

- Do not standardize them; give feedback to 1003.7 and wait for them to complete their specification. This risks incompatible implementations.

- Standardize them insofar as they relate to TFA administration. This risks incompatibility between the TFA aspects of those commands and their more general aspects. An example is the *mount* command; if 1003.7 is unhappy with the form of the TFA mount command, they are under no constraint to remain compatible with it. If the group ballots far enough in advance of 1003.7, this sort of clash could be frequent.

4. Many of the contentious issues have been "resolved" through the various mechanisms POSIX provides for introducing optional behavior; most frequently, these involve either "implementation-defined" behavior, or the addition of path-specific attributes whose status can be determined through the *pathconf()* function. Several of these options should be viewed by the ballot group as being "gratuitous" in some sense; i.e., the TFA committee should take a stand one way or another, and be willing to be beaten up in ballot for it. The POSIX standards are wishy-washy enough without the addition of gratuitous options.

Report on IEEE 1201: User Interface

Eileen Coons <coons@osf.org> reports on the October 16-19, 1989 meeting in Brussels, Belgium:

"The time has come," the walrus said,
"To talk of many things:
Of shoes – and ships – and sealing wax –
Of cabbages – and kings –
And why the sea is boiling hot –
And whether pigs have wings."
– Lewis Carroll

The P1201 committee is on a divine mission to define standards for user interface technologies. Lewis Carroll would have loved P1201 meetings.

In keeping with the precedent set by previous P1201 meetings, this latest get-together was spirited. The quasi-good news is that, by the end of the session, not one, but three PARs had been defined, as the group split into 1201.1 (Application Programming Interface), 1201.2 (Drivability - Look & Feel), and 1201.3 (User Interface Definition Language). One participant aptly named the proceedings "PAR Wars."

There was agonized discussion over the various sub-groups' missions, and an equal amount of agonized, and at times agonizing, wordsmithing over the .1 and .2 PARs themselves. The .3 group thoughtfully elected to split off and define itself in private. The PARs will be submitted via proper official channels to be blessed at the January SEC meeting.

For anyone not familiar with the PAR process, PAR is an acronym for Project Authorization Request. An individual or group that believes some work should be done by an IEEE committee drafts a document describing the work, which is then submitted to the IEEE as a PAR. Usually the PAR is circulated to the IEEE membership.

The Standards Executive Committee (SEC) reviews the PAR during its next scheduled session, typically held during a POSIX meeting. The SEC votes on the PAR, and if it is approved by the SEC, it is presented to Technical Committee on Operating Systems (TCOS). TCOS decides in which committee the work will be done. In the case of the PAR for User Interface, TCOS elected to divorce the work from the core POSIX effort (1003), and created P1201.

The PAR becomes part of the statement of work and basic charter for the group doing the work.

Fortunately, at this meeting, the group finally created some real structure for itself. The group decided to define an agenda and resolved that all meeting attendees should receive minutes of the meeting. Jim Isaak, the chair of the 1003 SEC, helped with structural definition by supplying IEEE rules and charter information, explaining the balloting process, and listing action options open to the committee.

Seven ballot alternatives were proposed, ranging from submitting a proposal for immediate ballot, to disbanding 1201. A vote was called, and although there was no consensus the heavy favorite was a proposal to adopt Motif's API as the basis for a standard API specification, and to extend it to accommodate aspects of Open Look's style.

This general direction was unpopular with a vocal minority, and so the group discarded the vote and returned to its original, pre-poll path of action: defining a specification for an API based on neither Motif nor Open Look, but on some new API – probably a hybrid of the two.

[Editor's note: I've heard more than one person express ill-ease about the restricted range of choices being considered. Why is there no mention of NeXT/Step, for example? A noticeable feeling among people who aren't on the committee is that it's too early to try to standardize in this area, and that the answer to the question, "Motif or Open Look?" should be, "No thanks." The answer to the implied question, "Why is there a P1201 and why are we doing this now, anyway?" seems to be that NIST, the National Institute for Standards and Technology (the people who bring you FIPS), is pushing hard for rapid creation of a GUI standard.]

Two presentations were made: one by AT&T, in favor of the joint API concept, and one by OSF, arguing against its feasibility. This was followed by a critique of – some thought, attack on – the second presentation by one of the acting chairs, Clive Feather of X/OPEN. P1201 may be many things but, so far, staid isn't one of them...

On a more neutral note, several representatives from organizations working on UIDL technologies made presentations about what they were doing in that arena, and then went off to form P1201.3.

The rest of the group broke into the .1 and .2 sub-groups for working sessions during most of the remaining meeting time. Each group reviewed its newly drafted PAR. P1201.1 also spent time comparing Motif and Open Look, identifying and exploring the differences between the two API's, and looking for potential drivability issues that could be deferred to P1003.2. P1003.2 took a similar course of action, comparing the styles of the two technologies.

There was also a spirited discussion regarding when and where the next P1201 meetings should be held. After various alternatives were explored, the group decided to keep P1201 meetings in the same vicinity and timeframe as POSIX meetings, since many attendees need, or want, to participate in POSIX as well.

;login: 15:2

USENIX Online Library/Index

What Is It:

The USENIX online index is an electronically available list of papers published by the USENIX Association and related groups. It contains title, author, and related information about papers published in USENIX and UNIX-related conference and workshop proceedings, newsletters, journals, and the like.

The index is freely available, and is kept as a simple ASCII file, in refer/bib format, sorted by author. In some cases, electronically readable versions of full papers or abstracts are also available. If a paper is available online, this is indicated in its index entry.

How to Get the Index:

The index is available online from UUNET, either via a mail server or anonymous ftp. The index is about 200K, and available only in entirety. To get it via electronic mail:

```
$ echo send bibliography | \
  mail uunet!library
```

A (non-human) server will automatically break the index up into mailable chunks (if necessary), and return it to the sender of the mail.

Or, the index can be retrieved via anonymous ftp to *uunet.uu.net*:

```
ftp> get library/bibliography my_local_file
```

To get a help file:

```
$ echo help | mail uunet!library
```

To pick up the date the index was last changed:

```
$ echo send date | mail uunet!library
```

For those unable to reach UUNET, the index is also available in hardcopy format from the Association office.

Online Papers and Abstracts:

We are actively soliciting the donation of papers and abstracts to include in the library. If you have had a paper published in any of the publications listed below, and you wish to donate the paper, you must provide us with an electronic version and give us permission to distribute it. You or your employer may retain the copyright if you wish.

If you wish to donate an abstract, we are prepared to type it in for you – all we need is your permission.

Publications Indexed:

Currently we have indexed all available issues of the following:

USENIX:
- Conference proceedings
- Workshop proceedings
- *Computing Systems*
- *;login:*

European UNIX User Group:
- Conference proceedings
- Newsletters

Software Tools User Group:
- Conference proceedings

Australian UNIX User Group:
- Newsletters

UNIX Review periodical

We are in the process of incorporating Japanese UNIX Society publications to the index. Other sources (AFUU, GUUG, NZUSUGI, etc.) are being continually evaluated and will be included as deemed suitable.

More Information:

For additional information about the online index and library, and/or instructions for donating abstracts or papers, contact:

usenix!index (index@usenix.org)

Or contact the Association's executive office.

USENIX Supporting Members

Digital Equipment Corporation
Open Software Foundation
AT&T Information Systems
Quality Micro Systems
Convex Computer Corporation
The Aerospace Corporation
Sun Microsystems, Inc.
Sybase, Inc.
mt Xinu

;login: 15:2

Long-Term Calendar of UNIX Events†

1990 Apr 9	POSIX APP Workshop	NIST; Gaithersburg, MD
1990 Apr 9-11	USENIX C++ Conference	San Francisco, CA
1990 Apr 23-27	EUUG	Munich, Germany
1990 Apr 23-27	IEEE 1003	Salt Lake City, UT
1990 May 7-11	DECUS	New Orleans, LA
1990 May 17	U & Parallel Systems, NLUUG	Ede, Netherlands
1990 May 30-Jun 1	UNIX/90	/usr/group/cdn; Toronto, Ont.
1990 Jun 11-15	USENIX	Marriott Hotel, Anaheim, CA
1990 Jun 11-15	ISO WG15 (POSIX)	Paris, France
1990 Jul 9-11	15th JUS Symposium	Tokyo, Japan
1990 Jul 11-13	UKUUG	London, UK
1990 Jul 16-20	IEEE 1003	Danvers, MA
1990 Jul 17-19	UniForum	Washington, DC
1990 Aug 20-23	Interex	Boston, MA
1990 Aug 27-28	*Security	Portland, OR
1990 Sep 4-5	GUUG	Wiesbaden, Germany
1990 Sep 25-28	AUUG	World Congress Centre, Melbourne, Aust.
1990 Oct 4-5	*Mach	Burlington, VT
1990 Oct 17-19	*Large Installation Sys. Admin.	Colorado Springs, CO
1990 Oct 8-12	InterOp 90 ACE	San Jose, CA
1990 Oct 15-19	IEEE 1003	Seattle, WA
1990 Oct 22-26	EUUG	Nice, France
1990 Oct 31-Nov 1	UNIX Expo	New York, NY
1990 Nov 5-9	Computer Communication Conf.	ICCC; New Delhi, India
1990 Nov 8	Open Systems, NLUUG	Ede, Netherlands
1990 Nov 14-16	UNIX EXPO '90 UniForum	Stockholm, Sweden
1990 Nov 15	POSIX APP Workshop	NIST; Gaithersburg, MD
1990 Nov 15-16	16th JUS Symposium	Osaka, Japan
1990 Dec 4-5	JUS UNIX Fair '90	Tokyo, Japan
1990 Dec 10-12	UNIX Asia '90	Sinix, Singapore
1990 Dec 10-14	DECUS	Las Vegas, NV
1991 Jan 7-11	IEEE 1003	New Orleans, LA
1991 Jan 16-18	*Software Devel. Environments	Grand Kempinski, Dallas, TX
1991 Jan 21-25	USENIX	Grand Kempinski, Dallas, TX
1991 Jan 22-25	UniForum	Infomart, Dallas, TX
1991 Feb	UNIX in Government	Ottawa, Ont.
1991 Feb 18-22	DECUS	Ottawa, Ont.
1991 May	UNIX 8x/etc	/usr/group/cdn; Toronto, Ont.
1991 May 6-10	DECUS	Atlanta, GA
1991 May 20-24	EUUG	Tromso, Norway
1991 Jun 10-14	USENIX	Opryland, Nashville, TN
1991 Jun/Jul	UKUUG	Liverpool, UK
1991 Sep 16-20	EUUG	Budapest, Hungary
1991 Dec	UKUUG	Edinburgh, UK
1991 Dec 9-13	DECUS	Anaheim, CA
1992 Jan 20-24	USENIX	Hilton Square, San Francisco, CA
1992 Jan 21-24	UniForum	Moscone Center, San Francisco, CA
1992 Spring	EUUG	Jersey, UK
1992 May 4-8	DECUS	Atlanta, GA
1992 Jun 8-12	USENIX	Marriott, San Antonio, TX
1992 Autumn	EUUG	Amsterdam, Netherlands

† Compiled with the assistance of Alain Williams of the EUUG, Susanne Smith of Windsound Consulting and John Quarterman of Texas Internet Consulting.

* USENIX Workshops

Book Review

The Design and Implementation of the 4.3BSD UNIX Operating System

Leffler, McKusick, Karels, & Quarterman

(Addison-Wesley, 1989)

Reviewed by Sunil K. Das

City University London Computer Science Department

This learned volume presents knowledge about the Berkeley variant of the UNIX operating system previously unavailable in one text. The four authors are known within the UNIX community for their expert computing skills, articulate conference presentations, and the quality of their written technical papers. It is therefore, difficult to imagine the book being anything other than very good.

John Lions in Australia documented the internals of 6th Edition UNIX as long ago as 1977, but the 4.3BSD book will ultimately be considered alongside Maury Bach's account of System V.2 internals,[1] which is possibly unwise since their target audiences are different. By some quirk of fate both books are 471 pages long, with the V.2 account being aimed at introducing readers to operating systems at a first level, while the 4.3BSD text provides a natural progression and addresses a more knowledgeable, second level, advanced scholar.

The layout of this book provides a sensible way to document a process-based operating system. It commences with an excellent four page Preface giving a concise account of the material covered, its applicability to academic courses and the overall organization. It is structured into five parts and further subdivided into chapters:

1. Overview – History and Goals, Design Overview of 4.3BSD, Kernel Services

2. Processes – Process Management, Memory Management

3. I/O System – I/O System Overview, The File System, Device Drivers, Terminal Handling

4. Interprocess Communication – Interprocess Communication, Network Communication, Network Protocols

5. System Operation – System Startup

Graded exercises appear at the end of each chapter which range from testing the reader's knowledge of what has appeared in the text to presenting major design projects or open research questions. References for the chapter follow the exercises and there is an extensive Glossary and Index at the end of the book.

The book is enjoyable to read, with a very direct style. Periodically, the style does drift a little, which is not surprising with four authors, but this does not detract from its readability. There are lengthy descriptions of algorithms, and text with only a few sentences being difficult to understand. My suspicion is that these instances occur because the book is written by Americans for a US market or maybe I simply didn't understand the few sentences.

A useful perspective is gained from the discussion about the internals of early versions of UNIX. Since UNIX has evolved with the hardware, it is possible to see why some redundant code exists in UNIX, namely, for historical reasons and/or backwards compatibility. Moreover, as well as concentrating on an implementation description, the text discusses and details the reasoning behind the design of 4.3BSD. It not only gives an in-depth description of how things work, but more importantly, we are told why things are there.

My perception was that the 4.3BSD book, compared with the V.2 text, had fewer diagrams, fewer algorithms and concentrated on the written word. However, the information has been collected together into a coherent structure. In particular, having the details of the "fast file system" in one place, especially the part on file system data

[1] M. J. Bach, *The Design of the UNIX Operating System*, Prentice-Hall, Englewood Cliffs, NJ (1986).

fragmentation, is very helpful. I now understand something that gave me difficulty before. Conversely, 4.3BSD does not have a distributed file system and I believe an omission is that there is no discussion on Sun NFS which is so widely accepted. I appreciate that Sun NFS is not part of 4.3BSD, but it is difficult to separate them nowadays.

Networking (Part 4) is the least understood aspect of BSD UNIX, so a description of the ideas behind mapping protocols into the kernel would have proved useful. In fact, the way this part of the book is written gives the impression that networking evolved into the kernel. We all know that networking was "hacked" into the socket mechanism and not designed into the kernel. Furthermore, the Interprocess Communication chapter "dives in" at a very low level of detail rather than discussing the principles of networking. I found that what has been written did not relate to the kernel. This part of the text has not evolved into the book like other subjects, where principles and implementations have been discussed in parallel.

The account centers upon the VAX hardware so I was dubious about the usefulness of the memory management section. VAX memory management is quite dated which necessarily affects the discussion. The text will be read by kernel gurus and system programmers, among others, whose typical usage might be to read parts of the book followed by reading some source code. This is fine if you have "vanilla" 4.3BSD and a VAX, but the book won't cope so well for those with different hardware. However, you can't "please all of the people all of the time."

In summary, the text is comprehensive with one or two omissions – no Sun NFS discussion and no discussion on further memory management techniques expected to be found on different versions of BSD4.3. It is the memory management unit rather than the VAX that we need to know about. However, the book is invaluable to anyone working with operating systems. It enables him or her to see the design philosophies behind the construction of a real operating system. One can understand from reading this book how a real operating system works and how to "color" the operating system to suit the hardware.

USENIX Seeks an Editor

The USENIX Board of Directors, at its January meeting (see page 29), decided to begin publishing books, as a natural extension of the Association's already considerable publishing program (conference and workshop proceedings, manuals, *;login:*, and *Computing Systems*).

To that end the Association is seeking candidates to fill the position of Book Editor. The editor will have the support of an editorial board and a managing editor, and will establish, together with the publications committee of the USENIX Board of Directors, an editorial policy and a review process much as was done for the Journal. (The position of Editor will be remunerated.)

All parties interested in the position of Editor should send their resumes and a 500 word statement to the Executive Director as soon as possible, for circulation to the editor search committee. Deadline for submission of application is April 30, 1990.

Electronic Communications Privacy Act

Dan Appelman
Heller, Ehrman, White and McAuliffe

The last question asked at the panel on Ethics in the Computer Industry at the USENIX conference in Washington D.C. in January had to do with the relevance of the Electronic Communications Privacy Act to the discussion of ethical standards. The question did not get an adequate response. Dan Appelman, one of the panelists and legal counsel for the USENIX Association, briefly describes that Act below. If, as a result of this description, there is a response from the readers for a more detailed explanation of the Act, Dan has agreed to supply one in a future issue of *;login:*.

The Electronic Communications Privacy Act of 1986, signed into law on October 26th of that year, is an important piece of legislation for those in the computer industry. It was one of the first attempts of the Federal government to regulate acts interfering with electronic messaging. It is comprehensive and complex and impossible to summarize well in a few short paragraphs. Nevertheless, here are some of its features:

- The Act makes it illegal to intercept, review or disclose the contents of, log initiation and termination information about, or otherwise use certain kinds of electronic communications absent the consent of the sender or the recipient.

- The Act makes it illegal to gain unauthorized access to electronic communication services and to exceed an existing authorization to access such services.

- The Act places restrictions on communication service providers and gives them certain obligations to ensure the privacy of communications between senders and recipients.

- The Act creates both civil and criminal causes of action for violations. On the civil side, those injured may sue in Federal court. Remedies include injunctions, money damages, punitive damages, attorneys fees and costs of suit. The Act also enables the U.S. Justice Department to bring criminal actions against violators. Penalties range from six months to five years in jail and fines up to $250,000 depending on the nature of the violation.

Most of the cases citing the Act. have looked at whether the government exceeded its authority in intercepting communications by wire tap. A few others involved private suits against communications service providers for providing services in such a way that message content was available indiscriminately to third parties. In each of these, the court decided that the allegations of illegal conduct were beyond the scope of the Act, and the service provider was found to have operated within the law. The case law interpreting the Act does not yet include instances where hackers have been sued or indicted for illegally intercepting electronic communications.

The Act itself does address some of the activities which were mentioned during the panel discussion on Ethics in the Computer Industry in Washington D.C. However, as I said then, there is an important distinction between legal and ethical constraints. The law, this law included, describes certain minimum standards of behavior which society will tolerate and imposes sanctions for their violation. Ethical standards are imposed, usually self-imposed, on groups which are professional subsets of society. These professions use altogether different standards in describing acceptable behavior than do lawmakers, and there is often no penalty for their violation. The Electronic Communications Privacy Act of 1986 tells us what is illegal behavior, but it does not help us to define what is ethical behavior. The question posed by the panel remains, of course, open.

New Association President

Alan G. Nemeth

At the USENIX Board of Directors meeting in January, I resigned from the Presidency of the Association. A number of questions have since been raised about my resignation, and I would like to clarify what I did and why.

I have been President of the Association since June, 1984, and a director of the Association since June, 1982. This has been a period of extensive growth in the marketplace for the UNIX operating system (dare I say – the open systems market). I took the role of President with a desire to transition the Association from a forum for discussion of technical issues by a small number of dedicated technologists to a professional society with a more rigorous approach to evaluating and presenting the continuing technical work of its members. It was clear in 1984 that the number of individuals who would be participating was going to grow – the question was how to channel that growth into higher quality without losing the informal communication and debate that has been a hallmark of the UNIX community from the beginning.

Overall, I am highly satisfied with the way the Association has developed. I needed and received the assistance of a large number of talented people who have been generous with their time and their opinions. Without the energy and wisdom they provided, the Association would be a much less interesting organization, and could easily have lost the sense of community – the training and helping of others that is a distinguishing characteristic.

It is time for me to retire from the Presidency and give others with their own vision for the next stage of the Association's life a chance to again change the character of the organization. Accordingly, I decided not to run in the upcoming election. You are fortunate to have two excellent candidates for the office of the Presidency in Kirk McKusick and Steve Johnson – it is up to you as a group to decide whose version of the Association is more to your liking. You also have the good fortune to have an excellent slate of candidates for the other Board seats – listen carefully to their different views as you file your ballots.

During the entire term of my Presidency, one of the most valuable members of the Board was the Vice President, Ms. Deborah Scherrer. Debbie has been on the Board of the Association since June, 1980, and Vice President since June, 1984. She has shouldered more than her share of the tasks of running the Association. Debbie has given extensively of her time, her wisdom, and her joy to make the Association a special organization. Without her assistance, my role would have been much more difficult. Debbie has also decided that she will not be a candidate in the current election to allow others their own opportunity to drive the Association.

Once it was clear that neither Debbie nor I would be continuing on the Board past June, I felt that I would like to take the opportunity to honor Debbie's contribution in a way that only I could do. By resigning the Presidency, Debbie automatically became President by the terms of our bylaws for the remainder of her term on the Board. I saw no conflict with the current election since Debbie and I had both declared our intentions clearly. In January, I announced my decision – first privately to Debbie, then to the Board of Directors, and publicly at the Open Meeting with the Board. I retain my role as a director of the Association until the new Board takes office following the June conference.

I would like to ask you all to assist in thanking Debbie for the excellent work she has done for so long.

Survey of *Computing Systems*

Peter H. Salus
Managing Editor

There were several comments on *Computing Systems* listed in the 1989 Member Survey published in the last issue of *;login:*. I think it appropriate for me to respond to them.

First of all, though, Mike O'Dell and I would like to thank most of you for your plaudits and enthusiasm. Without your help and encouragement, we'd never have been able to make *Computing Systems* a success.

A volume devoted to a particular topic.

> Issue 3.1 of *Computing Systems* will be devoted to such a single topic – music. This has been in the works for over a year. More such issues will occur in the future, but there's a great deal of coordination which goes into this.

Articles that elaborate on papers from conferences.

> If we take this as "conferences and workshops," then issue 3.2 will be such an issue, containing elaborations of papers from the Distributed and Multiprocessor Systems Workshop held last autumn in Florida. Without an extraordinary effort on the part of Gene Spafford, this could not have come about.

More articles that include graphics/illustrations.

> There's lots of this in the next two issues, but we can only print what's submitted. See my general comment.

More how to/applied articles.

> There are some in the works.

More issues!

> It's really gratifying that folks want more. But one of the requirements for publishing more is having more. My computation for the first two years of the journal is that we have published under 20% of the submissions. This may seem very low, but it is a tribute to the volunteer readers and to the staff: we could print more, but the level of quality would drop.

General comment: As I said in both San Francisco and San Diego at the Conferences, USENIX is a volunteer organization. If you don't write and submit stuff, there will be no conference papers, no workshops, no *;login:*, no *Computing Systems*. And if the quality of what's submitted is low, the quality of the product drops or there are fewer pages printed. If you want a special issue of the journal on, say, X.500, then propose it and get a bunch of your buddies to commit to writing stuff on it. If you want more items containing graphics, generate submissions containing them. Mike O'Dell and I can't create contents. I think that the first 10 issues of *Computing Systems* have been extraordinary. If you want this to continue, it's up to you.

An Update on UNIX and C Standards Activity

Jeffrey S. Haemer
Report Editor, USENIX Standards Watchdog Committee

USENIX Standards Watchdog Committee

The reports that accompany this summary are for the Fall meeting of IEEE 1003 and IEEE 1201, conducted the week of October 16-20, 1989, in Brussels, Belgium. (This isn't really true of the 1003.4 and 1003.8/1 reports, but let's overlook that.)

The reports are done quarterly, for the USENIX Association, by volunteers from the individual standards committees. The volunteers are familiarly known as "snitches" and the reports as "snitch reports." The band of snitches and I make up the working committee of the USENIX Standards Watchdog Committee. The group also has a policy committee: John S. Quarterman (chair), Alan G. Nemeth, and Shane P. McCarron. Our job is to let you know about things going on in the standards arena that might affect your professional life – either now or down the road a ways.

More formally: The basic USENIX policy regarding standards is:

> to attempt to prevent standards from prohibiting innovation.

To do that, we

- Collect and publish contextual and technical information such as the snitch reports that otherwise would be lost in committee minutes or rationale appendices or would not be written down at all.
- Encourage appropriate people to get involved in the standards process.
- Hold forums such as Birds of a Feather (BOF) meetings at conferences. We sponsored one workshop on standards.
- Write and present proposals to standards bodies in specific areas.
- Occasionally sponsor White Papers in particularly problematical areas, such as IEEE 1003.7 (in 1989) and possibly IEEE 1201 (in 1990).
- Very occasionally lobby organizations that oversee standards bodies regarding new committees, documents, or balloting procedures.
- Starting in mid-1989, USENIX and EUUG (the European UNIX Users Group) began sponsoring a joint representative to the ISO/IEC JTC1 SC22 WG15 (ISO POSIX) standards committee.

There are some things we do *not* do:

- We do not form standards committees. It's the USENIX Standards Watchdog Committee, not the POSIX Watchdog Committee, not part of POSIX, and not limited to POSIX.
- We do not promote standards.
- We do not endorse standards.

Occasionally we may ask snitches to present proposals or argue positions on behalf of USENIX. They are not required to do so and cannot do so unless asked by the USENIX Standards Watchdog Policy Committee.

Snitches mostly report. We also encourage them to recommend actions for USENIX to take.

John S. Quarterman, Chair
USENIX Standards Watchdog Committee

We don't yet have active snitches for all the committees and sometimes have to beat the bushes for new snitches when old ones retire or can't make a meeting, but the number of groups with active snitches is growing steadily. This quarter, you've seen reports from .1, .4, .5, .6, .8/2, and a belated report of last quarter's .8/1 meeting, as well as a report from 1201. Reports from .2 and .7 are in the pipeline, and may get posted before this summary does. We have no reports from .3, .8/[3-6], .9, .10, or .11, even though we asked Santa for these reports for Christmas.

If you have comments or suggestions, or are interested in snitching for any group, please contact me (*jsh@usenix.org*) or John (*jsq@usenix.org*). If you want to make

suggestions in person, both of us go to the POSIX meetings. The next set will be January 8-12, at the Hotel Intercontinental in New Orleans, Louisiana. Meetings after that will be April 23-27, 1990 in Salt Lake City, Utah, and July 16-20, 1990 in Danvers (Boston), Massachusetts.

I've appended some editorial commentary on problems I see facing each group. I've emphasized non-technical problems, which are unlikely to appear in the official minutes and mailings of the committees. If the comments for a particular group move you to read a snitch report that you wouldn't have read otherwise, they've served their purpose. Be warned, however, that when you read the snitch report, you may discover that the snitch's opinion differs completely from mine.

Report on 1003.0

Outside of dot zero, this group is referred to as "the group that lets marketing participate in POSIX." Meetings seem to be dominated by representatives from upper management of large and influential organizations; there are plenty of tailor-made suits, and few of the jeans and T-shirts that abound in a dot one or dot two meeting. There's a good chance that reading this is making you nervous; that you're thinking, "Uh, oh. I'll bet the meetings have a lot of politics, positioning, and discussion about 'potential direction.'" Correct. This group carries all the baggage, good and bad, that you'd expect by looking at it.

For example, their official job is to produce the "POSIX Guide:" a document to help those seeking a path through the open-systems standards maze. Realistically, if the IEEE had just hired a standards expert who wrote well to produce the guide, it would be done, and both cleaner and shorter than the current draft.

Moreover, because dot zero can see the entire open systems standards activities as a whole, they have a lot of influence in what new areas POSIX addresses. Unfortunately, politics sometimes has a heavy hand. The last two groups whose creation dot zero recommended were 1201 and the internationalization study group. There's widespread sentiment, outside of each group (and, in the case of internationalization, inside of the group), that these groups were created at the wrong time, for the wrong reason, and should be dissolved, but won't be. And sometimes, you can find the group discussing areas about which they appear to have little technical expertise. Meeting before last, dot zero spent an uncomfortable amount of time arguing about graphics primitives.

That's the predictable bad side. The good side? Frankly, these folks provide immense credibility and widespread support for POSIX. If dot zero didn't exist, the only way for some of the most important people and organizations in the POSIX effort to participate would be in a more technical group, where the narrow focus would block the broad overview that these folks need, and which doing the guide provides.

In fact, from here it looks as though it would be beneficial to POSIX to have dot zero actually do more, not less, than it's doing. For example, if dot five is ever going to have much impact in the Ada community, someone's going to have to explain to that community why POSIX is important, and why they should pay more attention to it. That's not a job for the folks you find in dot five meetings (mostly language experts); it's a job for people who wear tailor-made suits; who understand the history, the direction, and the importance of the open systems effort; and who know industry politics. And there are members of dot zero who fit that description to a tee.

Report on 1003.1

Is dot one still doing anything, now that the ugly green book is in print? Absolutely.

First, it's moved into maintenance and bug-fix mode. It's working on a pair of extensions to dot 1 (A and B), on re-formatting the ugly green book to make the ISO happy, and on figuring out how to make the existing standard language-independent. (The developer, he works from sun to sun, but the maintainer's work is never done.) Second, it's advising other groups and helping arbitrate their disputes. An example is the recent flap over transparent file access, in which the group defining the standard (1003.8/1) was told, in no uncertain terms, that NFS wouldn't do, because it wasn't consistent with dot one

semantics. One wonders if things like the dot six *chmod* dispute will finally be resolved here as well.

A key to success will be keeping enough of the original dot one participants available and active to ensure consistency.

Report on 1003.2

Dot one standardized the UNIX section two and three commands. (Okay, okay. All together now: "It's not UNIX, it's POSIX. All resemblance to any real operating system, living or dead, explicit or implied, is purely coincidental.") Dot two is making a standard for UNIX section one commands. Sort of.

The dot two draft currently in ballot, "dot-two classic," is intended to standardize commands that you'd find in shell scripts. Unfortunately, if you look at dot-two classic you'll see things missing. In fact, you could have a strictly conforming system that would be awfully hard to develop software on or port software to. To solve this, NIST pressured dot two into drawing up a standard for a user portability extension (UPE). The distinction is supposed to be that dot-two classic standardizes commands necessary for shell script portability, while the UPE standardizes things that are primarily interactive, but aid user portability.

The two documents have some strategic problems.

- Many folks who developed dot-two classic say the UPE is outside of dot two's charter, and won't participate in the effort. This sort of behavior unquestionably harms the UPE. Since I predict that the outside world will make no distinction between the UPE and the rest of the standard, it will actually harm the entire dot-two effort.

- The classification criteria are unconvincing. *nm(1)* is in the UPE. Is it really primarily used interactively?

- *cc* has been renamed *c89*, and *lint* may become *lint89*. This is silly and annoying, but look on the bright side: at least we can see why *c89* wasn't put in the UPE. Had it been, it would have had to have a name users expected.

- Who died and left NIST in charge? POSIX seems constantly to be doing things that it didn't really want to do because it was afraid that if it didn't, NIST would strike out on its own. Others instances are the accelerated timetables of .1 and .2, and the creation of 1003.7 and 1201.)

- Crucial pieces of software are missing from dot two. The largest crevasse is the lack of any form of source-code control. People on the committee don't want to suffer through an SCCS-RCS debate. POSIX dealt with the *cpio-tar* debate. (It decided not to decide.) POSIX dealt with the *vi-emacs* debate. (The UPE provides a standard for *ex/vi*.) POSIX is working on the NFS-RFS debate, and a host of others. Such resolutions are a part of its responsibility and authority. POSIX is even working on the Motif-Open/Look debate (whether it should or not).

At the very least, the standard could require some sort of source code control, with an option specifying which flavor is available. Perhaps we could ask NIST to threaten to provide a specification.

As a final note, because dot two (collective) standardizes user-level commands, it really can provide practical portability across operating systems. Shell scripts written on a dot-two-conforming UNIX system should run just fine on an MS-DOS system under the MKS toolkit.

Report on 1003.3

Dot three is writing test assertions for standards. This means dot three is doing the most boring work in the POSIX arena. Oh, shoot, that just slipped out. But what's amazing is that the committee members don't see it as boring. In fact, Roger Martin, who, as senior representative of the NIST, is surely one of the single most influential people in the POSIX effort, actually chairs this committee. Maybe they know something I don't.

Dot three is balloting dot one assertions and working on dot two. The process is moving at standards-committee speed, but has the advantage of having prior testing art as a touchstone (existing MindCraft, IBM, and NIST test work). The dilemma confronting the

group is what to do about test work for other committees, which are proliferating like lagomorphs. Dot three is clearly outnumbered, and needs some administrative cavalry to come to its rescue. Unless it expands drastically (probably in the form of little subcommittees and a steering committee) or is allowed to delegate some of the responsibility of generating test assertions to the committees generating the standards, it will never finish. ("Whew, okay, dot five's done. Does anyone want to volunteer to be a liaison with dot thirty-seven?")

Report on 1003.4

Dot four is caught in a trap fashioned by evolution. It began as a real-time committee. Somehow, it's metamorphosed into a catch-all, "operating-system extensions" committee. Several problems have sprung from this.

- Some of the early proposed extensions were probably right for real-time, but aren't general enough to be the right approach at the OS level.
- Pieces of the dot-four document probably belong in the the dot one document instead of a separate document. Presumably, ISO will perform this merge down the road. Should the IEEE follow suit?
- Because the dot-four extensions aren't as firmly based in established UNIX practice as the functionality specified in dot one and two, debate over how to do things is more heated, and the likelihood that the eventual, official, standard solution will be an overly complex and messy compromise is far higher. For example, there is a currently active dispute about something as fundamental as how threads and signals should interact.

Unfortunately, all this change has diverted attention from a problem that has to be dealt with soon – how to guarantee consistency between dot four and dot five, the Ada-language-binding group. Tasks semantics are specified by the Ada language definition. In order to get an Ada binding to dot four's standard (which someone will have to do), dot four's threads will have to be consistent with the way dot five uses tasks in their current working document. With dot five's low numbers, the only practical way to ensure this seems to be to have dot four aggressively track the work of dot five.

Report on 1003.5

Dot five is creating an Ada-language binding for POSIX. What's "Ada-language binding" mean? Just that an Ada programmer should be able to get any functionality provided by 1003.1 from within an Ada program. (Right now, they're working on an Ada-language binding for the dot one standard, but eventually, they'll also address other interfaces, including those from dot four, dot six, and dot eight.) They face at least two kinds of technical problems and one social one.

The first technical problem is finding some way to express everything in 1003.1 in Ada. That's not always easy, since the section two and three commands standardized by dot one evolved in a C universe, and the semantics of C are sometimes hard to express in Ada, and vice-versa. Examples are Ada's insistence on strong typing, which makes things like *ioctl()* look pretty odd, and Ada's tasking semantics, which require careful thinking about *fork()*, *exec()*, and *kill()*. Luckily, dot five is populated by people who are Ada-language wizards, and seem to be able to solve these problems. One interesting difference between dot five and dot nine is that the FORTRAN group has chosen to retain the organization of the original dot one document so that their document can simply point into the ugly green book in many cases, whereas dot five chose to re-organize wherever it seemed to help the flow of their document. It will be interesting to see which decision ends up producing the most useful document.

The second technical problem is making the solutions look like Ada. For more discussion of this, see the dot-nine (FORTRAN bindings) summary. Again, this is a problem for Ada wizards, and dot five can handle it.

The social problem? Interest in dot five's work, outside of their committee, is low. Ada is out-of-favor with most UNIX programmers. ("Geez, 1201 is a mess. Their stuff's gonna look as ugly as Ada.") Conversely, most of the Ada community's not interested in UNIX. ("Huh? Another 'standard' operating

environment? How's it compare to, say, PCTE? No, never mind. Just let me know every few years how it's coming along.") The group that has the hardest problem – welding together two, well-developed, standardized, disparate universes – has the least help.

Despite all of this, the standard looks like it's coming close to ballot, which means people ought to start paying attention to it before they have no choice.

Report on 1003.6

Most of the UNIX community would still feel more at home at a Rainbow gathering than reading the DOD rainbow books. The unfamiliar-buzzword frequency at dot six (security) meetings is quite high. If you can get someone patient to explain some of the issues, though, they're pretty interesting. The technical problems they're solving each boil down to thinking through how to graft very foreign ideas onto UNIX without damaging it beyond recognition. (The recent posting about *chmod* and access control lists, in *comp.std.unix* by Ana Maria de Alvare and Mike Ressler, is a wonderful detailed example.)

Dot six's prominent non-technical problem is just as interesting. The government has made it clear that vendors who can supply a "secure UNIX" will make a lot of money. No fools, major vendors have been furiously working on implementations. The push to provide a POSIX security standard comes at a time when these vendors are already quite far along in their efforts, but still some way from releasing the products. Dot six attendees from such corporations can't say too much, because it will give away what they're doing (remember, too, that this is security), but must somehow ensure that the standard that emerges is compatible with their company's existing, secret implementation.

Report on 1003.7

There is no single standard body of practice for UNIX system administration, the area dot seven is standardizing. Rather than seek a compromise, dot seven has decided to re-invent system administration from scratch. This was probably necessary simply because there isn't enough existing practice to compromise on. Currently, their intent is to provide an object-oriented standard, with objects specified in ASN.1 and administration of a multi-machine networked system as a target. (This, incidentally, was the recommendation of a USENIX White Paper on system administration by Susanne Smith and John Quarterman.) The committee doesn't have a high proportion of full-time system administrators, or a large body of experience in object-oriented programming. It's essentially doing research by committee. Despite this, general sentiment outside the committee seems to be that it has chosen a reasonable approach, but that progress may be slow.

A big danger is that they'll end up with a fatally flawed solution: lacking good available implementations; distinct enough from existing practices, where they exist, to hamper adoption; and with no clear-cut advantage to be gained by replacing any ad hoc existing solutions except for standard adherence. The standard could be widely ignored.

What might prevent that from happening? Lots of implementations. Object-oriented programming and C++ are fashionable (at the 1988, Winter Usenix C++ conference, Andrew Koenig referred to C++ as a "strongly hyped language"); networked UNIX systems are ubiquitous in the research community; and system administration has the feeling of a user-level solvable problem. If dot seven (perhaps with the help of dot zero) can publicize their work in the object-oriented programming community, we can expect OOPSLA conferences and *comp.sources.unix* to overflow with high-quality, practical, field-tested, object-oriented system administration packages that conform to dot seven.

Report on 1003.8

There are two administrative problems facing dot eight, the network services group. Both stem directly from the nature of the subject. There is not yet agreement on how to solve either one.

The first is its continued growth. There is now serious talk of making each subgroup a full-fledged POSIX committee. Since there are

currently six groups (transparent file access, network IPC, remote procedure call, OSI/MAP services, X.400 mail gateway, and directory services), this would increase the number of POSIX committees by nearly 50%, and make networking the single largest aspect of the standards work. This, of course, is because standards are beneficial in operating systems, and single-machine applications, but indispensable in networking.

The second is intergroup coordination. Each of the subgroups is specialized enough that most dot eight members only know what's going on in their own subgroup. But because the issues are networking issues, it's important that someone knows enough about what each group is doing to prevent duplication of effort or glaring omissions. And that's only a piece of the problem. Topics like system administration and security are hard enough on a single stand-alone machine. In a networked world, they're even harder. Someone needs to be doing the system engineering required to ensure that all these areas of overlap are addressed, addressed exactly once, and completed in time frames that don't leave any group hanging, awaiting another group's work.

The SEC will have to sort out how to solve these problems. In the meantime, it would certainly help if we had snitches for each subgroup in dot eight. Any volunteers for .8/[3-6]?

Report on 1003.9

Dot nine, which is providing FORTRAN bindings, is really fun to watch. They're fairly unstructured, and consequently get things done at an incredible clip. They're also friendly; when someone new arrives, they actually stop, smile, and provide introductions all around. It helps that there are only half-a-dozen attendees or so, as opposed to the half-a-hundred you might see in some of the other groups. Meetings have sort of a "we're all in this together / defenders of the Alamo" atmosphere.

The group was formed after two separate companies independently implemented FORTRAN bindings for dot one and presented them to the UniForum technical committee on supercomputing. None of this, "Let's consider forming a study group to generate a PAR to form a committee to think about how we might do it," stuff. This was rapid prototyping at the standards level.

Except for the advantage of being able to build on prior art (the two implementations), dot nine has the same basic problems that dot five has. What did the prior art get them? The most interesting thing is that a correct FORTRAN binding isn't the same as a good FORTRAN binding. Both groups began by creating a binding that paralleled the original dot one standard fairly closely. Complaints about the occasional non-FORTRANness of the result have motivated the group to try to redesign the bindings to seem "normal" to typical FORTRAN programmers. As a simple example, FORTRAN-77 would certainly allow the declaration of a variable in common called ERRNO, to hold the error return code. Users, however, would find such name misleading; integer variables, by default and by convention, begin with "I" through "N."

It is worth noting that dot nine is actually providing FORTRAN-77 bindings, and simply ignoring FORTRAN-8x. (Who was it that said of 8x, "Looks like a nice language. Too bad it's not FORTRAN"?) Currently, 1003 intends to move to a language-independent specification by the time 8x is done, which, it is claimed, will ease the task of creating 8x bindings.

On the surface, it seems logical and appealing that documents like 1003.1 be rewritten as a language-independent standard, with a separate C-language binding, analogous to those of dot five and dot nine. But is it really?

First, it fosters the illusion that POSIX is divorced from, and unconstrained by its primary implementation language. Should the prohibition against null characters in filenames be a base-standard restriction or a C-binding restriction?

I've seen a dot five committee member argue that it's the former. Looked at in isolation, this is almost sensible. If Ada becomes the only language anyone wants to run, yet the government still mandates POSIX compliance, why should a POSIX implementation prohibit its filenames from containing characters that aren't special to Ada? At the same time, every

POSIX attendee outside of dot five seems repelled by the idea of filenames that contain nulls. (Quiz: Can you specify a C-language program or shell script that will create a filename containing a null?)

Second, C provides an existing, precise, widely-known language in which POSIX can be specified. If peculiarities of C make implementing some portions of a standard, specified in C, difficult in another language, then there are four clear solutions:

1. change the specification so that it's equally easy in C and in other languages,

2. change the specification so that it's difficult in every language,

3. change the specification so that it's easy in some other language but difficult in C, or

4. make the specification vague enough so that it can be done in incompatible (though equally easy) ways in each language.

Only the first option makes sense. Making the specification language-independent means either using an imprecise language, which risks four, or picking some little-known specification language (like VDL), which risks two and three. Declaring C the specification language does limit the useful lifetime of POSIX to the useful lifetime of C, but if we don't think we'll come up with good replacements for both in a few decades, we're facing worse problems than language-dependent specifications.

Last, if you think the standards process is slow now, wait until the IEEE tries to find committee volunteers who are fluent in both UNIX and some language-independent specification language. Not only will the useful lifetime of POSIX remain wedded to the useful lifetime of C, but both will expire before the language-independent version of dot one is complete.

It would be nice if this push for language-independent POSIX would go away quietly, but it won't.

Report on 1003.10

In July, at the San Jose meeting, John Caywood of Unisys caught me in the halls and said, accusingly, "I understand you think supercomputers don't need a batch facility." I didn't have the foggiest idea what he was talking about, but it seemed like as good a chance as any to get a tutorial on dot ten, the supercomputing group, so I grabbed it. (Pretty aggressively helpful folks in this supercomputing group. If only someone in it could be persuaded to file a snitch report.)

Here's the story:

Articles about software engineering like to point out that approaches and tools have changed from those used twenty years ago; computers and computing resources are now much cheaper than programmers and their time, while twenty years ago the reverse was true. These articles are written by people who've never used a Cray. A typical supercomputer application might run on a $25M, non-byte-addressable, non-virtual-memory machine, require 100 to 1000 Mbytes of memory, and run for 10 Ksecs. Expected running time for jobs can be greater than the machine's mean-time-to-failure. The same techniques that were common twenty years ago are still important on these machines, for the same reasons – we're working close to the limits of hardware art.

The card punches are gone, but users often still can't login to the machines directly, and must submit jobs through workstation or mainframe front ends. Resources are severely limited, and access to those resources need to be carefully controlled. The two needs that surface most often are checkpointing and a batch facility.

Checkpointing lets you re-start a job in the middle. If you've used five hours of Cray time, and need to continue your run for another hour but have temporarily run out of grant money, you don't want to start again from scratch when the money appears. If you've used six months of real time running a virus-cracking program and the machine goes down, you might be willing to lose a few hours, even days, of work, but can't afford to lose everything. Checkpointing is a hard problem, without a generally agreed-upon solution.

A batch facility is easier to provide. Both Convex and Cray currently support NQS, a public domain network queueing system. The product has enough known problems that the

group is re-working the facility, but the basic model is well-understood, and the committee members, both users and vendors, seem to want to adopt it. The goal is command-level and library-level support for batch queues that will provide effective resource management for really big jobs. Users will be able to do things like submit a job to a large machine through a wide-area network, specify the resources – memory, disk space, time, tape drives, etc. – that the job will need to run to completion, and then check back a week or two later to find out how far their job's progressed in the queue.

The group is determined to make rapid progress, and to that end is holding 6-7 meetings a year. One other thing: the group is actually doing an application profile, not a standards document. For an clarification of the distinction, see the discussion of dot eleven.

Report on 1003.11

Dot eleven has begun work on an application profile (AP) on transaction processing (TP). An AP is a set of pointers into the POSIX Open System Environment (OSE). For example, the TP AP might say, "For dot eleven conformance, you need to conform to dot one, dot four, sections 2.3.7 and 2.3.8 of dot 6, 1003.8 except for /2, and provide a batch facility as specified in the dot 10 AP." A group doing an AP will also look for holes or vague areas in the existing standards, as they relate to the application area, go point them out to the appropriate committee, and possibly chip in to help the committee solve them. If they find a gap that really doesn't fall into anyone else's area, they can write a PAR, requesting that the SEC (1003's oversight committee) charter them to write a standard to cover it.

Dot eleven is still in the crucial early stage of trying to figure out what it wants to do. Because of fundamental differences in philosophy of the members, the group seems to be thrashing a lot. There is a clear division between folks who want to pick a specific model of TP and write an AP to cover it, and folks who think a model is a far too detailed place to start. The latter group is small, but not easily dismissed.

It will be interesting to see how dot eleven breaks this log jam, and what the resolution is. As an aside, many of the modelers are from the X/OPEN and ISO TP groups, which are already pushing specific models of their own; this suggests what kinds of models we're likely to get if the modeling majority wins.

Report on X3J11

A single individual, Russell Hansberry, is blocking the official approval of the ANSI standard for C on procedural grounds. At some point, someone failed to comply with the letter of IEEE rules for ballot resolution. and Hansberry is using the irregularity to delay adoption of the standard.

This has had an odd effect in the 1003 committees. No one wants to see something like this inflicted on his or her group, so folks are being particularly careful to dot all i's and cross all t's. I say odd because it doesn't look as though Hansberry's objections will have any effect whatsoever on either the standard, or its effectiveness. Whether ANSI puts its stamp on it or not, every C compiler vendor is implementing the standard, and every book (even K&R) is writing to it. X3J11 has replaced one de-facto standard with another even stronger one.

Report on 1201.1

What's that you say, bunky? Uneasy about Xwindows? Well then, you won't care much for 1201.1, which is supposed to be "User Interface: Application Programming Interface," but is really "How much will the Motif majority have to compromise with the Open/Look minority before force-feeding us a thick standard full of 'Xm[A-Z]...' functions with long names and longer argument lists?"

Were this group to change its name to "Xwindows application programming interface," you might not hear nearly as much grousing from folks outside the working group. As it is, the most positive comments you hear are, "Well, X is pretty awful, but I guess we're stuck with it," and "What could they do? If POSIX hadn't undertaken it, NIST would have."

If 1201 is to continue to be called "User Interface," these aren't valid arguments for standardizing on X or toolkits derived from it. In what sense are we stuck with X? The number of X applications is still small, and if X and its toolkits aren't right for the job, it will stay small. Graphics hardware will continue to race ahead, someone smart will show us a better way to do graphics, and X will become a backwater. If they are right, some toolkit will become a de-facto standard, the toolkit will mature, and the IEEE can write a realistic standard based on it.

Moreover, if NIST wants to write a standard based on X, what's wrong with that? If they come up with something that's important in the POSIX world, good for them. ANSI or the IEEE can adopt it, the way ANSI's finally getting around to adopting C. If NIST fails, it's not the IEEE's problem.

If 1201.1 ignores X and NIST, can it do anything? Certainly. The real problem with the occasionally asked question, "are standards bad?" is that it omits the first word: "When." Asked properly, the answer is, "When they're at the wrong level." API's XVT is example of a toolkit that sits above libraries like Motif or the Mac toolbox, and provides programmers with much of the standard functionality necessary to write useful applications on a wide variety of window systems. Even if XVT isn't the answer, it provides proof by example that we can have a window-system-independent, application programming interface for windowing systems. 1201.1 could provide a useful standard at that level. Will it? Watch and see.

Applications Sought for Research Awards

The UniForum Association is now accepting applications for its new Research Award program. Two awards, each of up to two years' duration, and valued at $10,000 per year, will be granted annually to graduate students. The awards will aid students in researching problems concerning UNIX and open systems computing. One grant will apply to technical study in computer science and the other to management sciences as they affect information management.

UniForum will give preference to projects whose results can be demonstrated at UniForum Conferences and are of value to its sponsors.

Each recipient of an award must submit a one-page status report each quarter, and a four-to-five page progress report at the end of the first year, which must be approved for the award to continue. A formal paper is required at the end of the second year and must be presented at the UniForum Conference.

Candidates must apply through their university department chair. The deadline for departments to submit one or two nominations for next year is **May 1, 1990**. Winners will be notified on July 1 and their names announced at the Summer UniForum in Washington, D.C., later that month. Call the UniForum Association office at (408) 986-8840 for details and applications. Send applications to the UniForum Research Award Committee, 2901 Tasman Drive., #201, Santa Clara, CA 95054.

AUUGN Back Issues

Here are the details of back issues of which we still hold copies. All prices are in Australian dollars and include surface mail within Australia. For overseas surface mail add $2 per copy and for overseas airmail add $10 per copy.

pre 1984	Vol 1-4	various	$10 per copy
1984	Vol 5	Nos. 2,3,5,6	$10 per copy
		Nos. 1,4	unavailable
1985	Vol 6	Nos. 2,3,4,6	$10 per copy
		No. 1	unavailable
1986	Vol 7	Nos. 1,4-5,6	$10 per copy
		Nos. 2-3	unavailable
			(Note 2-3 and 4-5 are combined issues)
1987	Vol 8	Nos. 1-2,3-4	unavailable
		Nos. 5,6	$10 per copy
1988	Vol 9	Nos. 1,2,3	$10 per copy
		Nos. 4,5,6	$15 per copy
1989	Vol 10	Nos. 1-6	$15 per copy
1990	Vol 11	No. 1	$15 per copy

Please note that we do not accept purchase orders for back issues except from Institutional members. Orders enclosing payment in Australian dollars should be sent to:

AUUG Inc.
Back Issues Department
PO Box 366
Kensington NSW
Australia 2033

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 an 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 Membership is not a membership you can 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.

It's also possible to subscribe to the newsletter without being an AUUG member. This saves you nothing financially, that is, the subscription price is greater than 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 Visa or Mastercard by simply completing the authorisation on the application form.

AUUG Incorporated

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
PO 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.

This form is valid only until 31st May, 1991

..... does hereby apply for

☐ New/Renewal* Institutional Membership of AUUG $325.00

☐ International Surface Mail $ 40.00

☐ International Air Mail $120.00

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
- ☐ Other *(Indicate which)* .....

- ☐ System V.3 binary
- ☐ System V.2 binary
- ☐ System V binary
- ☐ System III binary

AUUG Incorporated

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
PO 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.

This form is valid only until 31st May, 1991

I, ... do hereby apply for

☐ Renewal/New* Membership of the AUUG	$78.00	
☐ Renewal/New* Student Membership	$45.00	(note certification on other side)
☐ International Surface Mail	$20.00	
☐ International Air Mail	$60.00	(note local zone rate available)

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 Incorporated

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
PO 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.

This form is valid only until 31st May, 1991

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	$ 90.00
☐ International Surface Mail	$ 20.00
☐ International Air Mail	$ 60.00

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
PO 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#* _____