ISSN 1035-7521

Australian UNIX systems User Group Newsletter

AUUGN

Volume 13, Number 1

February 1992

Registered by Australia Post, Publication Number NBG6524

The AUUG Incorporated Newsletter

Volume 13 Number 1

February 1992

CONTENTS

AUUGN is the journal of AUUG Incorporated, an organisation with the aim of promoting knowledge and understanding of Open Systems including but not restricted to the UNIX* system, networking, graphics, user interfaces and programming and development environments, and related standards.

* UNIX is a registered trademark of UNIX System Laboratories, Incorporated

AUUG General Information

Memberships and Subscriptions

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

Membership and General Correspondence

All correspondence for the AUUG should be addressed to:-

The AUUG Secretary, P.O. Box 366, Kensington, N.S.W. 2033. AUSTRALIA	Phone: Fax: Email:	(02) 361 5994 (02) 332 4066 auug@munnari.oz.au

AUUG Business Manager

Liz Fraumann, P.O. Box 366, Kensington, N.S.W. 2033. AUSTRALIA	Phone: Fax: Email:	(02) 953 3542 (02) 953 3542 eaf@softway.sw.oz.au

AUUG Executive

President	**Pat Duffy** *pzd30@juts.ccc.amdahl.com* Amdahl Pacific Services Pty. Ltd. 1 Pacific Highway North Sydney NSW 2000	Vice-President	**Chris Maltby** *chris@softway.sw.oz.au* Softway Pty. Ltd. 79 Myrtle Street Chippendale NSW 2008
Secretary	**Rolf Jester** *rolf.jester@sno.mts.dec.com* Digital Equipment Corporation (Australia) Pty. Ltd. P.O. Box 384 Concord West NSW 2138	Treasurer	**Frank Crawford** *frank@atom.ansto.gov.au* Australian Supercomputing Technology Private Mail Bag 1 Menai NSW 2234
Committee Members	**Andrew Gollan** *adjg@softway.sw.oz.au* Softway Pty. Ltd. 79 Myrtle Street Chippendale NSW 2008		**Glenn Huxtable** *glenn@cs.uwa.oz.au* University of Western Australia Computer Science Department Nedlands WA 6009
	Peter Karr Computer Magazine Publications 1/421 Cleveland Street Redfern NSW 2016		**Michael Tuke** *mjt@anl.oz.au* ANL Ltd. 432 St. Kilda Road Melbourne VIC 3004
	Scott Merrilees *Sm@bhpese.oz.au* BHP Information Technology P.O. Box 216 Hamilton NSW 2303		

AUUG General Information

Next AUUG Meeting

The AUUG'92 Conference and Exhibition will be held from the 8th to the 11th of September, 1992, at the World Congress Centre, Melbourne. See later in this issue for Preliminary Announcement and Call for Papers.

AUUG Newsletter

Editorial

Wellcome to AUUGN Volume 13 Number 1, the *first* for 1992.

By the time this reaches members, most of the AUUG summer technical conferences will be over. In this issue we have a report by Adrian Booth on the Perth meeting. Hopefully in the next issue we will have a report on most of the other summer meetings.

The next major activity is the Winter Conference. The call for papers has been included in this issue.

As most members will be aware, Liz Fraumann has been appointed AUUG's Business Manager. I have included the press release of her appointment.

Papers include, A New C compiler from the Plan 9 series, a paper by Andrew McRae on a UNIX Task Broker and two articles by Rolf Jester which originally appeared in DECUS News, but are just as useful to AUUG members (if not more so).

Jagoda Crawford

AUUGN Correspondence

All correspondence regarding the AUUGN should be addressed to:-

AUUGN Editor
PO Box 366
Kensington, NSW, 2033
AUSTRALIA

E-mail: *auugn@munnari.oz.au*

Phone: +61 2 543 3885
Fax: +61 2 543 5097

AUUGN Book Review Editor

The AUUGN Book Review Editor is Dave Newton (dave@teti.qhtours.oz.au). Contact him for more details.

Contributions

The Newsletter is published approximately every two months. The deadlines for contributions for the next issues are:

Volume 13 No 2	Friday 17th April
Volume 13 No 3	Friday 29th May
Volume 13 No 4	Friday 31st July
Volume 13 No 5	Friday 25th October
Volume 13 No 6	Friday 27th November

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, pic and grap preprocessors, but please note on your submission which macros and preprocessors you are using. If you can't use troff, then just plain text or postscript 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.

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 first 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 the AUUG secretariat, phone (02) 361 5994, fax (02) 332 4066.

Back Issues

Various back issues of the AUUGN are available. For availability and prices please contact the AUUG secretariat or write to:

AUUGN Inc.
Back Issues Department
PO Box 366
Kensington, NSW, 2033
AUSTRALIA

Acknowledgement

This Newsletter was produced with the kind assistance of and on equipment provided by the Australian Nuclear Science and Technology Organisation.

Disclaimer

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

AUUG Institutional Members as at 05/03/1992

A.N.U.
AAII
Adept Business Systems Pty Ltd
Adept Software
AIDC Ltd.
Alcatel Australia
Amdahl Pacific Services Pty Ltd
Andersen Consulting
ANSTO
ANZ Banking Group/Global
Technical Services
Apple Computer Australia
Apscore International Pty Ltd
Ausonics Pty Ltd
Australian Airlines Limited
Australian Bureau of Agricultural and
Resource Economics
Australian Bureau of Statistics
Australian Eagle Insurance Co. Ltd
Australian Electoral Commission
Australian Information Processing
Centre Pty Ltd
Australian National Parks &
Wildlife Service
Australian Taxation Office
Australian Technology Resources (ACT)
Australian Wool Corporation
Avid Systems Pty Ltd
Bain & Company
Ballarat Base Hospital
BHP CPD Research & Technology Centre
BHP Information Technology
BHP Minerals
BHP Research - Melbourne Laboratories
BICC Communications
Bond University
Burdett, Buckeridge & Young Ltd.
Bureau of Meteorology
Byrne & Davidson Holdings Pty Ltd
C.I.S.R.A.
Capricorn Coal Management Pty Ltd
CITEC
Co-Cam Computer Group
Codex Software Development Pty. Ltd.
Cognos Pty Ltd
Colonial Mutual
Com Tech Communications
Commercial Dynamics
Communica Software Consultants
Computechnics Pty Ltd
Computer Power Group
Computer Science of Australia Pty Ltd
Computer Software Packages
Corinthian Engineering Pty Ltd
CSIRO
Curtin University of Technology
Cyberscience Corporation Pty Ltd
DMP Software Pty Ltd
Data General Australia
Deakin University
Defence Housing Authority
Defence Service Homes
Dept. of Minerals & Energy (NSW)
Dept. of Transport
Dept. of Treasury & Finance
Dept. of Agricultural & Rural Affairs
Dept. of Conservation & Environment
Dept. of Defence
Dept. of Foreign Affairs & Trade
Dept. of I.T.R.
Dept. of The Premier & Cabinet
Dept. of Treasury
Digital Equipment Corp
(Australia) Pty Ltd
DMP Software Pty Ltd
Duesburys Information Technology
Pty Ltd
ESRI Australia Pty Ltd
Eastek Pty Ltd
EDS (Australia) Pty Ltd
Electronics Research Labs
Emulex Australia Pty Ltd
ESRI Australia Pty Ltd
Expert Solutions Australia
FGH Decision Support Systems Pty Ltd
Financial Network Services
First State Computing
Fremantle Port Authority
Fujitsu Australia Ltd
G. James Australia Pty Ltd
GEC Alsthom Australia
Geelong and District Water Board
Genasys II Pty Ltd
General Automation Pty Ltd
George Moss Ltd
GIO Australia
Hamersley Iron Pty. Limited
Harris & Sutherland Pty Ltd
Hermes Precisa Australia Pty Ltd
Highland Logic Pty Ltd
Honeywell Ltd
I.B.A.
IBM Australia Ltd
Iconix Pty Ltd
Infonetics
Information Technology Consultants
Insession Pty Ltd
Internode Systems Pty Ltd

AUUG Institutional Members as at 05/03/1992

Ipec Management Services
James Cook University of
North Queensland
Labtam Australia Pty Ltd
Land Information Centre
Leeds & Northrup Australia Pty. Ltd
Macquarie University
Mayne Nickless Courier Systems
McDonnell Douglas Information
Systems Pty Ltd
McIntosh Hamson Hoare Govett Ltd
Medical Benefits Funds of Australia Ltd
Metal Trades Industry Association
Mincom Pty Ltd
Minenco Pty Ltd
Ministry of Consumer Affairs
Ministry of Housing & Construction (VIC)
Mitsui Computer Limited
Motorola Computer Systems
Multibase Pty Ltd.
NEC Information Systems Australia Pty Ltd
Nucleus Business Systems
Office of the Director of
Public Prosecutions
Olivetti Australia Pty Ltd
OPSM
Oracle Systems Australia Pty Ltd
Parliament House
Paxus
Philips PTS
Port of Melbourne Authority
Prime Computer
Public Works Department
Pulse Club Computers Pty Ltd
Pyramid Technology
Q.H. Tours Limited
Queensland Department of Mines
Queensland University of Technology
Radio & Space Services
RMIT
SBC Dominguez Barry
Sculptor 4GL+SQL
SEQEB Control Centre
Shire of Eltham
Silicon Graphics Computer Systems
Snowy Mountains Hydro-electric Authority
Software Development International Pty Ltd
Softway Pty Ltd
Sony (Australia) Pty Ltd
South Australian Lands Dept.
Sphere Systems Pty Ltd
St Vincent's Private Hospital
Stallion Technologies Pty Ltd
Stamp Duties Office
Standards Australia
State Bank of NSW
Steedman Science and Engineering
Sugar Research Institute
Swinburne Institute of Technology
Sydney Ports Authority
Systems Union Pty Ltd
Tasmania Bank
Tattersall Sweep Consultation
Telecom Australia
Telecom Australia Corporate Customer
Telecom Network Engineering Computer
Support Service
Telectronics Pty Ltd
The Anti-Cancer Council of Victoria
The Far North Qld Electricity BOard
The Fulcrum Consulting Group
The Opus Group
The Preston Group
The Roads and Traffic Authority
The Southport School
The University of Western Australia
Toshiba International Corporation Pty Ltd
Tower Computing Services
Tower Technology Pty Ltd
Tradelink Plumbing Supplies Centres
Turbosoft Pty Ltd
TUSC Computer Systems
UCCQ
Unidata Australia
Unisys
University of Adelaide
University of Melbourne
University of New South Wales
University of Queensland
University of South Australia
University of Sydney
University of Tasmania
University of Technology
UNIX System Laboratories
Unixpac Pty Ltd
Vicomp
VME Systems Pty Ltd
Wacher Pty Ltd
Wang Australia Pty Ltd
Water Board
Westfield Limited
Wyse Technology Pty Ltd

--

AUUG President's Report

Dear AUUG Member,

The last report in AUUG'N, coupled with the AUUG column in Open Systems Review magazine, has resulted in three postings on the net, a disappointing result both by volume and lack of constructive criticism or suggestions.

On the other hand, there's been plenty of general criticism - invective even. One posting (from someone who is not a current member) hopes that "we" can "wrest control of AUUG back from the marketing types who dominate it..." and goes on to unearth a conspiracy by vendors to "get some of their staff involved, gradually change the focus of the group, and before too long you have created a marketing group that meets your requirements *and* that already has hundreds of members".

I don't believe it's any secret that I, along with a number of other committee members, am employed by a vendor. I don't think it's any secret, either, that a vendor organisation actually makes quite a sacrifice in terms of time and money to support an employee who is on the committee. I'm not sure what benefits accrue to a vendor organisation through this involvement; what I am certain of is that EVERY person who has been involved with the committee during the five years that I have has been 100% dedicated to the aims of the organisation and the needs of the members, and has put immense personal effort into AUUG.

The management committee regards AUUG's directions as its most serious piece of business and would like to hear the views of EVERY member, but suggestions for improvement would be much more useful than us/them criticism.

We've maintained that one organisation can suit the needs of all members. This doesn't necessarily imply that every event will suit every member, but that within the context of our aims, membership and financial means, we can provide a sufficient range of benefits from which members can choose.

The decision to make changes to AUUG, to become involved in marketing both the organisation and its events, and to try to attract new members, was based on the need to increase revenue so that member benefits could be increased. We've had lively discussions about the "chicken and egg" nature of this conundrum, but I think there's been clear consensus all along that what we were doing was in everyone's interests. Nothing's taken place without the membership being informed (admittedly, at times, through poor channels). And, remember, the composition of the committee is still very heavily weighted in favour of "traditional" members.

I believe (as previously stated) that the last AGM clearly gave us the mandate to continue the new directions. At that time, I reported to the membership exactly what we had been doing, planned to do, and why, and there was not only no disagreement, but what seemed to be an air of general approval.

We don't want to disenfranchise ANY AUUG member, and we are investing an immense amount of time, energy and emotion into this entire issue. Please, could we hear from the silent majority as well as what seems to be the vocal minority?

AUUG

AUSTRALIAN OPEN SYSTEMS USERS GROUP

February 25th, 1992

ELIZABETH FRAUMANN APPOINTED BUSINESS MANAGER AT AUUG

The Australian Open Systems Users Group, (AUUG Inc.) have appointed Elizabeth Fraumann as their Business Manager. The newly created position will see Fraumann established as the main contact point for AUUG and handling the general business of the organisation. Fraumann has over ten years experience in marketing, sales and human resources with Open Systems organisations in the United States. She brings this knowledge, plus a good knowledge of the Open Systems market and its key contirbutors, to her role with AUUG.

A major project for Fraumann will be AUUG '92, the major, annual exhibition and conference of the Australian Open Systems Users Group. This year to be held at the World Congress Centre in Melbourne, September 8 - 11.

For further information regarding this press release please contact:
Liz Fraumann
Business Manager, AUUG
Telephone or Fax: +61 2 953 3542

or

Ellen Gubbins
AUUG Media Liaison
Telephone: +61 2 973 1992

--

AUUG '92 World Congress Centre, Melbourne, Australia, September 8-11

1992 Preliminary Announcement and Call for Papers

AUUG, Inc., forum for Open Systems Users Presents:

"Maintaining Control in an Open World."

How do you "maintain control" with open systems?

...Stories From the Front...

"I've been dealing with company 'X' since I started this business.
How do I move to open systems and not have to completely retool my office?"

"I made Perth 'talk' TCP/IP to South East Asia!"

"Changing Corporate EDP Strategy to Open Systems"
"Changing Corporate EDP Strategy from Open Systems"

"WAN implementation overview... who holds the key?"

"Who is in control?... The Vendors?... The 'Standards' Organisations?... The Customers?"

"Who is steering 'SS Open Systems' ?"
"Who is driving computer PR?"

"Communications with two carriers... AOTC & OPTUS... and these are my experiences..."

Mission critical applications and environments, in particular system and
network management and high reliability/availability systems are at stake.
Everyone wants answers!

The management team on,"How is our bottom line going to be affected;

The System Administrator who has just been asked to keep 10 'Open Systems' strung together with no down time; and

Of course, the budding "Guru" who continues to amaze everyone with innovations and tools to go beyond our dreams for tomorrow.

* Quality,
* Impact of standards and standardization,
* Commercialisation of UNIX
* Analysis of network/host security issues and
* Protection of current and future investments.

AUUG '92 will explore "maintaining control with open systems" from all aspects.

Events:

AUUG '92 will be a four day conference, commencing September 8, 1992. The first day will be devoted to tutorial presentations, followed by three days of papers, work-in-progress sessions and BOFs.

Tutorials:

Provisions for two full-day tutorials and up to eight half-day tutorials have been made. These sessions, typically in a lecture format, are targeted to educate the audience and arm them with new and innovation "how to" lessons. The speakers selected to present the tutorials will receive 40% portion of the total tutorial fee their session draws, in addition to receiving a free conference registration. Please submit tutorial abstracts, along with preference for a half- or full-day slot to address below.

Papers:

AUUG '92 provides a dual Technical and Commercial track for the afternoon presentations. To share your new and innovative details of implementation, 'how to', and similar areas submit your abstract for the technical track. We are also interested in your experiences, 'why', 'so what', strategic issues, and the like. If your topic better fits these areas submit your abstract for the Commercial track. Many subjects are equally interesting and would benefit attendees being presented from different perspectives. If you feel your topic has both technical and commercial interest value and could be presented twice with differing emphasis, your paper will receive priority from the committee and a special recognition from AUUG if it is accepted. The above should not, of course, discourage papers which are either more specifically targeted or are appropriate for both audiences at once.

Prize for the Best Student Paper:

A cash prize of $500 will be awarded for the best paper submitted by a full-time student at an accredited tertiary education institution.

Work-in-Progress Sessions:

In order to schedule work-in-progress sessions we will need some idea of the number of people interested in making a 10 to 15 minute presentation. Please mail expressions of interest to the committee at the address below.

Birds-of-a-Feather Sessions (BOFs):

Are you interested in hearing side by side product comparison, the global affect of computing, AARNET, or other controversial topics? At the end of each presentation day, one hour time slots for BOFs will be available. We distinguish two types of BOF; general interest and vendor sponsored. Please contact the Program Committee if you would like to organise a Birds-of-a-Feather Session. There may be some facilities charge to vendor sponsored events.

Speaker Incentives:

Tutorial presenters will receive 40% of their total attendee draw and a free conference registration. Presenters of papers are afforded free conference registration.

Slide Preparation Offer:

We understand most presenters have access to slide/overhead generating equipment today. For those presenters who do not have this resource available, AUUG will again offer a slide production service. Final slide information will be required at least 4 weeks prior to the conference to partake in this service. Please note presenters meeting the 4 week deadline will be afforded a proof cycle before final slide generation.

Form of Submissions:

Please indicate whether your submission is relevant to the technical or commercial audiences, or both. In either case, submissions are required to be in the form of an abstract and an outline. Please provide sufficient detail to allow the committee to make a reasoned decision about the final paper; of course a full paper is also perfectly acceptable. A submission should be from 2-5 pages and include:

1. Author name(s), postal addresses, telephone numbers, bio, and e-mail addresses.

2. Abstract: 100 words

3. Outline: 1-4 pages giving details of the approach or algorithms pursued.

4. References to any relevant literature

5. Time needed for the presentation. Most presentations will be for 30 minutes including a 5 minute question/answer time, although 1 hour time slots may be made available.

6. Audio-visual requirements
   - 35 mm slides are preferred, however, overheads will be accepted.
   - Hand written or typewriter generated overheads will not be accepted.

Acceptance:

Authors whose submissions are accepted will receive instructions on the preparation of final papers for inclusion in the conference proceedings, and the format requirements for slides.

Relevant Dates:

Abstract and outlines due: April 30, 1992
Notifications to authors: May 15, 1992
Final Papers due: July 15, 1992

Please submit one hard copy and one electronic copy (if possible to the address below):

AUUG '92 Program
P.O. Box 366
Kensington, NSW 2033

e-mail: AUUG92@softway.sw.oz.au

Phone: +61 2 361-5994
Fax: +61 2 332-4066

Please be sure to include your postal code and electronic mail addresses in all correspondence.

Program Committee:

Chair: Peter Karr - CMP Publications
Ian Hoyle - BHP Research Labs
Robert Elz - Melbourne University
Liz Fraumann - AUUG

Open System Publications

As a service to members, AUUG will source Open System Publications from around the world. This includes various proceeding and other publications from such organisations as

AUUG,
Uniform,
USENIX,
EurOpen,
Sinix,
etc.

For example:

EurOpen Proceedings		USENIX Proceedings	
Dublin	Autumn'83	C++ Conference	Apr'91
Munich	Spring'90	UNIX and Supercomputers Workshop	Sept'88
Trosmo	Spring'90	Graphics Workshop IV	Oct'87

AUUG will provide these publications at cost (including freight), but with no handling charge. Delivery times will depend on method of freight which is at the discretion of AUUG and will be based on both freight times and cost.

To take advantage of this offer send, in writing, to the AUUG Secretariat, a list of the publications, making sure that you specify the organisation, an indication of the priority and the delivery address as well as the billing address (if different).

AUUG Inc.
Open System Publication Order
PO Box 366
Kensington, NSW, 2033
AUSTRALIA

Fax: (02) 332 4066

SESSPOOLE

SESSPOOLE is the South Eastern Suburbs Society for Programmers Or Other Local Enthusiasts. That's the South Eastern Suburbs of Melbourne, by the way.

SESSPOOLE is a group of programmers and friends who meet every six weeks or so for the purpose of discussing UNIX and open systems, drinking wines and ales (or fruit juices if alcohol is not their thing), and generally relaxing and socialising over dinner.

Anyone who subscribes to the aims of SESSPOOLE is welcome to attend SESSPOOLE meetings, even if they don't live or work in South Eastern Suburbs. The aims of SESSPOOLE are:

> To promote knowledge and understanding of Open System; and to promote knowledge and understanding of Open Bottles.

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 and Summer'91 Melbourne Meetings.

SESSPOOLE meetings are held in the Bistro of the Oakleigh Hotel, 1555 Dandenong Road, Oakleigh, starting at 6:30pm. Dates for the next few meetings are:

Thursay, 16 April 1992
Tuesday, 26 May 1992
Wednesday, 8 July 1992
Thursay, 20 August 1992

Hope we'll see you there!

To find out more about SESSPOOLE and SESSPOOLE activities, contact either **Stephen Prince** (ph. (03) 608-0911, e-mail: *sp@labtam.oz.au*) or **John Carey** (ph. (03) 587-1444, e-mail: *john@labtam.oz.au*), or look for announcements in the news-group **aus.auug**.

The WAUG Column

Welcome to what I intend to be a regular column on the machinations of WAUG, the Western Australian UNIX systems Group. We have two major activities: a bimonthly newsletter and monthly meetings.

WAUG gets together at 6pm on the third Wednesday of each month. If we have a vendor-sponsored meeting it is usually at the vendor's premises; otherwise we meet at a pub or other venue that can provide drinks and nibbles. We have a 40-45 minute seminar, then we socialise until we get tired, which usually takes quite some time. 20 to 30 people usually attend - about a quarter of our members.

Cameron Ferstat <cferstat@sydvm1.vnet.ibm.com> currently has the unenviable job of Meeting Organiser. If you would like to speak at WAUG you can email him or phone him on (09) 327 6659. Talks must be UNIX-related and not thinly disguised sales pitches.

WAUG meetings attract a wide range of speakers and topics. In January the members did not throw tomatoes when I spoke on "Why I Don't Have a Filofax: time management tools using UNIX shells and `perl`" (mind you there have been some rude remarks about `perl`). In March, Tandem provided refreshments and Peter Lloyd to tell us about "Fault-Tolerant UNIX: Integrity and NonStop-UX".

In February our regular meeting was cancelled in favour of the AUUG Summer Technical Conference, which was a great success both technically and socially. Congratulations to Alan Main, the organiser, and Major, the program chair.

WAUG's newsletter is called YAUN (Yet Another UNIX Newsletter). Apart from the usual newsletter stuff, YAUN encourages members to write UNIX-related articles and to review our meetings. Phil Sutherland has written about the fun-and-games of carrying UNIX around the world on a portable machine. Adrian Booth has written several things, including a column on UNIX tricks and traps.

If you're interested in joining WAUG or contributing to YAUN, our address is PO Box 877, WEST PERTH WA 6005.

Janet Jackson <janet@cs.uwa.edu.au>

Canberra AUUG Chapter

Users meeting

The Canberra Chapter of AUUG (otherwise known as the Canberra Open Systems Users Group) will be holding a meeting to discuss the dialup facility that we are currently setting up.

The idea of this facility is to provide email, electronic news, and public-domain source access to local members. Many of the local members can't get access to these facilities (at work) due to work-imposed restrictions.

The facility will be described, methods of access shown, and functionality displayed. The system will be set up for 6 months initially, and if popular this will be extended. If you are from the Canberra region, and are interested in this system but can not make the meeting please RSVP your interest.

Topic:	Canberra Dialup Unix Facility
When:	Thursday April 30th
Time:	7:30 for 8pm
Where:	Huxley Lecture Theatre Huxley Building Mills Road ANU Campus Canberra

RSVP: John Barlow, Secretary, work: (06) 2492930, fax: (06) 2492930, email: John.Barlow@anu.edu.au
(If you are not familiar with the ANU ask for a fax of the campus map, or a copy sent via the post).

John Barlow <JohnD.Barlow@anu.edu.au>

AUUG Summer '92 Technical Conference - Perth†

Adrian Booth

The Perth AUUG Summer '92 Technical Conference ('the Conference') was a great success, and attracted approximately 70 delegates.

The keynote talk, which was very well received, was '*RTE and other tools for Software Quality Assurance*' by Ken McDonell from Pyramid Technology. Many of the other talks related to different aspects of UNIX network and/or system management.

The talks, in the order of presentation, were:

Ken McDonell	*RTE and other tools for Software Quality Assurance*
Hooman Samini	*The role of UNIX in the management of distributed networks*
Toivo Pedaste	*A quick look at SNMP*
Craig Farrel	*Network Management Software*
Hugh Irvine	*TCP/IP over ISDN*
Ian Crawford	*An Information Strategy for the Network Manager*
Janet Jackson	*Automating user administration with* **perl**
Steve Landers	*A technology for UNIX data center management*
Major	*Project GNU*

I have not attempted to cover everything in each talk, but to extract enough of a summary of the interesting points to give the flavour of each talk. Any errors in the summaries are, of course, mine and not the speaker's.

RTE and other tools for Software Quality Assurance
Ken McDonell, Pyramid Technology

(Ken's talk stemmed from his interest (passion?) in performing meaningful benchmarks of multiuser platforms. Ken was responsible for developing Pyramid Technology's *sscript* product - which allows realistic simulations of multiple concurrent users for benchmarking purposes - and has never lived down writing MUSBUS. Ken's presentation related to extending *sscript's* mechanism - Remote Terminal Emulation (RTE) - to Software Quality Assurance (SQA)).

The central aspects of SQA are functional correctness, robustness, and performance. Related techniques that provide an *illusion* of quality include software 'science' metrics, code coverage analysis, proofs of program correctness, and beta-release programs.

However, to do real SQA, the computer's humanoid data input peripherals need to be modelled - batch jobs and shell scripts do *not* emulate real software usage.

Ken defined the common SQA testing modes - unit and module tests, single- and multi-user performance measurements, and capacity planning experiments - before discussing ways to perform cost-effective SQA.

† *Copyright (c) 1992, Adrian Booth Computing Consultants*

Ken suggests that the cost of and delays in test *development* are a barrier to the *coverage* of the testing procedures. Similarly, the cost of and delays in test *execution* are a deterrent to the *frequency* and *scope* of validation.

The solution to this is to automate the process, which can be done in a 'tool-rich' environment that maximises productivity. This leads to Ken's claim:

UNIX-like environments and sophisticated RTE form the basis of extensive, cost-effective SQA.

What is wrong with traditional, batch-oriented SQA? For a start, some applications won't work at all in a batch environment: many of those that have a full-screen user interface, for example. Also, the usefulness of batch-oriented SQA is limited due to input device independence and typeahead and input buffering problems. The output may be date- and/or time-dependent. Exhaustive testing may require non-determinism (variation) in input - this is difficult, and changes the output. The response time of the application cannot be measured. It is difficult to generalise to multiple concurrent instances of the same test. Finally, batch-oriented tests don't use the concept of a *typical* workload, making them useless for capacity planning.

RTE-based SQA, on the other hand, provides a complete hardware and software environment that simulates the activities of multiple concurrent users. The simulation includes synchronisation between 'keyboard' input and system response, real-time delays for 'think time' and realistic typing rates, and no artificial temporal correlation.

Aspects of the 'Master Script' that can be customised include the method used to establish the virtual circuit (serial line, rlogin, telnet, ...), think times, loop control (iteration) and the input rate.

In conclusion, Ken posed the question:

'Quality software - can we get there from here?'

and answered 'Yes' - but this requires:

• A dramatic reduction in the cost of test suite development, with special purpose tools built on general purpose tools.
• Integration of robustness into the SQA process - non-determinism.
• Acceptance of performance as a quality measure (which leads to a need for performance and capacity planning information in SQA).
• User knowledge and usage patterns should be 'learnt' in SQA tests.

Ken's talk - like his keynote talk last year on benchmarking - was one of the best received talks at any of the three Perth conferences to date. Everyone I spoke to would definitely like to see him return next year.

The role of UNIX in the management of distributed networks
Hooman Samini, Oscom International

(Hooman's brief talk was a pleasant break for those who were overwhelmed by Ken's. Hooman gave an introductory presentation on the OSI system management model, which fitted in well as the first of a series of talks on network management).

The OSI system management model is a distributed application model, with a communication architecture based upon the full seven OSI layers. Its data model comprises two objects: the rules for defining OSI management information (the SMI: **Structure of Management Information**), and the set of standard definitions which can be used and/or extended to form the **Management Information Base (MIB)**.

OSI splits system management applications into several fundamental types:

- Configuration
- Faults - receiving alarms and identifying alarm causes
- Accounting - charging and billing
- Performance
- Security

(Work on the last two types is ongoing).

Hooman then described how the functionality of several vendors' network management software related to the OSI model.

Hooman's conclusions were that UNIX-based systems would be the network management systems of the future, and that it was very important for network management to be performed on a *distributed* management system, not a *centralised* one.

A quick look at SNMP
Toivo Pedaste, Winthrop Technologies

(Toivo gave another brief but informative introductory talk, this time on the Simple Network Management Protocol).

SNMP is part of the Internet Management Framework, and is basically the equivalent of OSI's Common Management Information Protocol (CMIP), or, specifically, CMIP over TCP/IP (CMOT).

The main goals in the design of SNMP - unlike most other networking standards coming into vogue today - were that it should be simple and implementable.

SNMP sees the world as a set of variables. In particular, a network entity is basically just a sack of these variables. SNMP provides both **polling** (by the network manager) to get variable values and to confirm that a node is up, and **traps** generated asynchronously by the network node and sent to the management node.

This combination is referred to as **trap-directed polling.**

There are two types of variables: the standard set, which all TCP/IP nodes must support, and enterprise-specific variables specific to a vendor (e.g. Cisco). All variables are effectively ASN.1 objects.

The initial MIB definition was extended to MIB2. This MIB supports these standard variable groups:

SYSTEM	Describes the particular system (node).
INTERFACES	Describes each network interface on the node.
AT	Describes address translation being performed by the node.
IP	IP protocol specifics, such as routing tables.
ICMP	ICMP protocol specifics
TCP	TCP protocol specifics
UDP	UDP protocol specifics
EGP	EGP protocol specifics
TRANSMISSION	media-dependent items
SNMP	Related to SNMP itself

SNMP is usually implemented as a synchronous send/request protocol using UDP over IP. However, definitions also exist for IPX (Novell) and over OSI.

There are five SNMP Message Types:

GET	Manager requests specified variable(s) from agent
GET-NEXT	Manager requests the variable that follows the previously specified variable(s)
SET	Manager sets variables in agent
TRAP	Agent sends an unsolicited message to the manager
GET-RESPONSE	Agent's reply to a GET or GET-NEXT

Toivo's talk was ideal for those who, like me, are interested in network management and its protocols, but don't want to spend up to $50 on a book that tells us how the protocol is implemented. For those who want more information on SNMP, Toivo recommends *The Simple Book* by Marshall Rose, who was the head of the committee that designed SNMP.

Network Management Software
Craig Farrel, Curtin University

Craig's talk detailed the typical network management problems he is faced with managing the Computer Science network at Curtin, and described some software implemented at Curtin to help cope with these problems.

The problem that Craig and his team needed to solve was that they could not answer any of the following questions:

- Who is talking to who?
- How much talking is being done?
- Where is my network bandwidth going?
- Why is the network slow?
- Which machines should I subnet off in order to improve network response time?

Initially, they evaluated several network management packages. They found most PC-based software to be cheap, but inflexible. Three other UNIX-based packages they evaluated were:

- Net Visualiser from Silicon Graphics - excellent (especially the user interface), but requires a Silicon Graphics workstation, and costs $10,000 for universities (i.e. too much).

- Sniffer - excellent, especially the packet decomposition. Far more expensive than Net Visualiser.
- SunNet Manager - understood SNMP.

Craig's team didn't need all of this functionality, however, and certainly didn't want to pay these prices. They determined that what they needed was a package that could show and analyse LAT, DECnet, AppleTalk and various IP packets, the connections on the network (i.e. 'Who?'), and the amount of traffic on each connection ('How much?'). Of course, they also wanted this information presented through a good, visual interface that made it immediately and intuitively obvious what was happening.

Craig demonstrated that they had largely achieved this goal with some overhead slides of the software in action.

Finally, Craig presented a list of futures:

- A complete Ethernet packet analyzer
- An individual host analyser (who has it been talking to, and with what protocols)
- A network monitor (large FTP transfers, errors/collisions, alarms) - SNMP?
- An SNMP tool providing management and visualisation of local and remote networks and hosts
- Report generation (daily, weekly, monthly) detailing hosts, protocols, and times in a presentable hardcopy form
- Automatically generating a physical network map

Craig's talks are well known for both their technical content and entertainment value. I won't mention any of the comments he made of the cuff, but it is probably a good thing that a prominent 'Open Systems' vendor chose not to send any representatives to the Perth conference this year.

TCP/IP over ISDN
Hugh Irvine, Hugh Irvine and Associates

I probably won't do this talk justice as I had already attended it at the AUUG'91 Winter Conference. Hugh described the Port of Melbourne Authority's state of the art TCP/IP wide area network, which was implemented using ISDN.

Hugh started the talk by describing the computing history of the Port of Melbourne Authority (PMA). About 18 months ago, the PMA embarked on an ambitious project to totally upgrade their computing facilities. Approximately six sites throughout Victoria were to be fully interconnected, with UNIX, TCP/IP, X Windows and Ingres as the base technologies.

The connections between the remote sites and the central head office have been implemented using Telecom ISDN lines. Hugh demonstrated that ISDN lines - whether MicroLink (2 x 64K data channels) or MacroLink (20 x 64K data channels) - offer substantial price/performance advantages over traditional leased line technology.

Hugh also pointed out that with the low cost of ISDN for the home - $300 installation, plus $90 per year - 'every home should have at least one'. Two 64K data lines make an X terminal a very feasible home-based alternative to a complete computer system.

An Information Strategy for the Network Manager
Ian Crawford, Department of Community Services

I always enjoy Ian's talks because I never feel like he is trying to sell or convert me to something. Instead, they offer a refreshing breath of pragmatism in a marketplace where common sense has been all but displaced by three-letter acronyms.

Ian's talk wasn't a technical one - it was a down to earth summary of the steps that a struggling network administrator (from a fictitious organisation) could take to encourage users and management to support and effectively use network facilities.

Ian started by describing the familiar situation where, after a new network has been installed, an evaluation is done of its effectiveness. The evaluation shows that no files have been created on the network server for three weeks. Most users have days-old mail that hasn't been read, and sneakernet is still the prevailing technology for getting printouts. User confidence is at an ebb because of initial network teething problems. Management have allocated a computer network budget of (number of computers *x* unit price).

What can the network manager do to turn around such a situation, and make the network a success?

Ian suggests that there are four layers to the network:

- Management
- Technical
- Operational
- Users

The distinction between the upper three layers is how much information they need about the status of the network 'now'. Operational staff need immediate notification of any network problem. Technical staff need notification of problems that operational staff can't handle. Management need notification of disaster or very long-lasting network problems.

The key questions that the network manager must ask are:

- What are you going to manage?
- What factors are critical for the success of the network in this organisation?

To make the network a success for **Users**, the network must become easier to use, and it should provide information of value to the users. User confidence in the network should also be increased.

>From an **Operations** perspective, the requirements are for real-time event reporting, and access to every device's status. Both of these are easily provided by a network management package.

The management applications required are neatly summed up by the OSI network management model: fault management, performance, configuration, security and accounting.

The **Technical** layer needs access to detailed information, as contrasted with **Management**, who need tactical reports, regular service level reports, and fault escalation.

In summary, Ian suggested that the first step that the network manager should take is to ask the question 'What is an information strategy for network management?'. (The answer is usually to identify the players who influence the success of the network, and to define the critical information required by each of these players). The second step is to repeatedly go back to the first step.

Automating user administration with **perl**

Janet Jackson, Department of Computer Science, University of Western Australia

Janet works as a UNIX system administrator at UWA. She has developed a suite of *perl* scripts that automate much of her user account management tasks, especially temporary accounts for visitors, and the bulk addition/removal of student accounts.

Janet began with a brief description of UWA's Computer Science network - around 30 Sun workstations running NFS and NIS.

The first part of Janet's talk described her account expiry system. In it, when a temporary account must be added, she runs a script **newexp** that automatically adds the user and records the details of the account (including its expiry date) in a data file.

A related script, **check_accexp**, is run nightly by *cron*. It checks the data file for accounts about to expire and mails the details to Janet. When Janet receives the mail, she ensures that the account is no longer needed and wipes it.

Janet then described the **mkstudents** and **newuser** scripts. Janet had some much-hacked scripts which had outlived their usefulness. She wanted them to provide automated, non-interactive, bullet proof, site-specific, NIS-friendly and maintainable user account management. Of the available tools, none matched UWA's specific requirements.

It comes as no surprise to those who know Janet (1) that she decided to write the tools herself; and (2) that she decided to write them in *perl*.

Why perl? Janet cited its text handling capabilities, built in arithmetic expression evaluation, access to system calls, and error handling. Besides, she **likes** *perl*'s syntax.

Janet described the implementation of these tools, and showed some *perl* examples. I was surprised, after my first few looks at *perl*, that it looks quite bearable. (I especially liked the idea of
print STDERR "Message" unless $quiet;
- maybe I will learn *perl*, after all).

Janet then described potential improvements/extensions to her scripts. These mainly involved fully automating all those little things we forget to do when someone leaves, such as deleting their mail alias.

Janet concluded by posing the rhetorical question, 'Was *perl* a good choice?'. She answered 'Yes', citing:

- Availability of *system()*
- Speed
- Complex processing
- Associative arrays
- Powerful regular expressions
- Ease of maintenance

A technology for UNIX data center management

Steve Landers, Functional Software

Functional Software are well known for their *Admin Manager* product, which automates the tedious tasks involved in managing a UNIX system or network. The quality and usability of this software can be gauged by its adoption as a standard product throughout the Australasian region by Pyramid Technology. Functional Software have now carried this work further with the introduction of the *Data Center Manager*, which provides management at a level previously only available on mainframes. Steve's talk described the object-oriented database technology that Functional Software have implemented as a building block of the *Data Center Manager*.

Steve started by defining a *data center* as any site, comprising a number of central hosts, which utilised segregation of responsibility, emphasis on standards and procedures, and formal audit reviews as mechanisms to perform effective center management. The primary difference between traditional and UNIX data centers is that UNIX data centers typically comprise heterogenous systems.

The *Data Center Manager* was designed around the following implementation framework:

- Support existing practices, and lead to better ones
- Accomodate change with consistency and control
- Be flexible and extensible
- Facilitate portability of operations staff
- Encourage innovation and creativity
- Accomodate different management styles

This led to the application structure:

$$\begin{tikzcd}
\text{User Interface} & \text{Functional Toolset} \\
\text{Application procedural code} \arrow[u] & \text{Bourne/Korn shell} \arrow[u] \\
\text{Data management} \arrow[u] & \text{Functional Database} \arrow[u]
\end{tikzcd}$$

Much of the *Data Center Manager* is implemented using the *Functional Toolset*, which comprises user interface tools, X Windows support, an object management framework, an interpretive 4GL (i.e.: the Bourne shell), a data dictionary, and a relational database system.

Problems in developing a management package include supporting different UNIX variants, supplying a consistent command-line interface, allowing distribution (delegation) of management tasks, and facilitating customisation. These problems were the causes of Functional Software moving towards an object-oriented paradigm.

Steve then went on to describe the implementation of object management within the *Data Center Manager*.

Steve's talk was both instructive and entertaining, and I will certainly be asking for a test copy of the *Data Center Manager* when I start managing a large UNIX site.

Project GNU
Major, Pyramid Technology

Major wrapped up the conference with a light discussion about Project GNU. As expected, Major spent much of the talk trying to convert the delegates to *elm*. Major's talk was an interesting historical look at the origins of Project GNU, plus an overview of the GNU software available (for free) today.

I could not possibly do Major's talk justice with a short summary, and hope that he will write up his talk in a separate article.

In summary, the Perth Summer Conference provided a marked contrast to AUUG'91 in that it consisted of consistently high quality talks presented by technically competent people who earn a living from producing real results from today's technology. I hope to see a similar set of talks in Perth next year, and wish I could expect to see the same at AUUG'92.

Finally, on behalf of all of the conference delegates, I would like to thank - in no particular order - all of the speakers; Major (Pyramid Technology), the programme chair; Alan and Lexie Main (Functional Software), who handled all of the conference administration, including registrations; Glenn Huxtable (UWA), the national coordinator of the AUUG Summer Technical Conferences; and of course AUUG, who floated the conference and sent along an AUUG banner (even if it did have 'Open Systems' written on it :-).

See you all again next year!

Adrian Booth
Adrian Booth Computing Consultants
abcc@DIALix.oz.au
(09) 385 1003

ACSnet Survey

1.1 Introduction

ACSnet is a computer network linking many UNIX hosts in Australia. It provides connections over various media and is linked to AARNet, Internet, USENET, CSnet and many other overseas networks. Until the formation of AARNet it was the only such network available in Australia, and is still the only network of its type available to commercial sites within Australia. The software used for these connections is usually either SUN III or SUN IV (or MHSnet). For the purposes of this survey other software such as UUCP or SLIP is also relevant.

At the AUUG Annual General Meeting held in Melbourne on September 27th, the members requested that the AUUG Executive investigate ways of making connection to ACSnet easier, especially for sites currently without connections. This survey is aimed at clearly defining what is available and what is needed.

Replies are invited both from sites requiring connections and sites that are willing to accept connections from new sites. Any other site that has relevant information is also welcome to reply (*e.g.* a site looking at reducing its *distance* from the *backbone*).

Please send replies to:

Mail:	Attn: Network Survey AUUG Inc P.O. Box 366 Kensington N.S.W. 2033	*FAX:* *E-Mail:*	(02) 332 4066 auug@atom.lhrl.oz

Technical enquiries to:

Frank Crawford	(frank@atom.lhrl.oz)	(02) 543 9404
or		
Scott Merrilees	(Sm@bhpese.oz)	(049) 40 2132

Thank you

=====

1.2 Contact Details

Name: ____________________
Address: ____________________
____
____
Phone: ____________________
Fax: ____________________
E-Mail: ____________________

1.3 Site Details

Host Name: ____________________
Hardware Type: ____________________
Operating System Version: ____________________
Location: ____________________
____
____

New Connections

If you require a network connection please complete the following section.

Please circle your choice (circle more than one if appropriate).

A1.	Do you currently have networking software?	Yes	No	
A2.	If **no**, do you require assistance in selecting a package?	Yes	No	
A3.	Are you willing to pay for networking software? If yes, approximately how much?	Yes ____________________	No	
A4.	Do you require assistance in setting up your network software?	Yes	No	
A5.	Type of software:	SUNIII TCP/IP Other (Please specify): ____________	MHSnet SLIP	UUCP
A6.	Type of connection:	Direct X.25/Dialin Other (Please specify): ____________	Modem/Dialin X.25/Dialout	Modem/Dialout
A7.	If **modem**, connection type:	V21 (300 baud) V22bis (2400) Other (Please specify): ____________	V23 (1200/75) V32 (9600)	V22 (1200) Trailblazer
A8.	Estimated traffic volume (in KB/day): (not counting netnews)	< 1 > 100: estimated volume: ____________	1-10	10-100
A9.	Do you require a news feed?	Yes Limited (Please specify): ____________	No	
A10.	Any time restrictions on connection?	Please specify: ____________________		
A11.	If the connection requires STD charges (or equivalent) is this acceptable?	Yes	No	
A12.	Are you willing to pay for a connection (other than Telecom charges)? If yes, approximately how much (please also specify units, *e.g. $X/MB* or flat fee)?	Yes ____________________	No	
A13.	Once connected, are you willing to provide additional connections?	Yes	No	
A14.	Additional Comments:			

Existing Sites

If you are willing to accept a new network connection please complete the following section.

Please circle your choice (circle more than one if appropriate).

B1.	Type of software:	SUNIII	MHSnet	UUCP
		TCP/IP	SLIP	
		Other (Please specify): ____________________		
B2.	Type of connection:	Direct	Modem/Dialin	Modem/Dialout
		X.25/Dialin	X.25/Dialout	
		Other (Please specify): ____________________		
B3.	If **modem**, connection type:	V21 (300 baud)	V23 (1200/75)	V22 (1200)
		V22bis (2400)	V32 (9600)	Trailblazer
		Other (Please specify): ____________________		
B4.	Maximum traffic volume (in KB/day): (not counting netnews)	< 1	1-10	10-100
		> 100: acceptable volume: ____________________		
B5.	Will you supply a news feed?	Yes	No	
		Limited (Please specify): ____________________		
B6.	Any time restrictions on connection?	Please specify: ____________________		
B7.	If the connection requires STD charges (or equivalent) is this acceptable?	Yes	No	
B8.	Do you charge for connection? If yes, approximately how much (please also specify units, *e.g.* *$X/MB* or flat fee)?	Yes	No	
		____________________		
B9.	Any other restrictions (*e.g.* educational connections only).?			
B10.	Additional Comments:			

A New C Compiler [†]

Ken Thompson

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

This paper describes yet another series of C compilers. These compilers were developed over the last several years and are now in use on Plan 9. These compilers are experimental in nature and were developed to try out new ideas. Some of the ideas were good and some not so good.

1. Introduction

Most C compilers consist of a multitude of passes with numerous interfaces. A typical C compiler has the following passes – pre-processing, lexical analysis and parsing, code generation, optional assembly optimisation, assembly (which itself is usually multiple passes), and loading. [Joh79a]

If one profiles what is going on in this whole process, it becomes clear that I/O dominates. Of the cpu cycles expended, most go into conversion to and from intermediate file formats. Even with these many passes, the code generated is mostly line-at-a-time and not very efficient. With these conventional compilers as benchmarks, it seemed easy to make a new compiler that could execute much faster and still produce better code.

The first three compilers built were for the National 32000, Western 32100, and an internal computer called a Crisp. These compilers have drifted into disuse. Currently there are active compilers for the Motorola 68020 and MIPS 2000/3000 computers. [Mot85, Kan88]

2. Structure

The compiler is a single program that produces an object file. Combined in the compiler are the traditional roles of pre-processor, compiler, code generator, local optimiser, and first half of the assembler. The object files are binary forms of assembly language, similar to what might be passed between the first and second passes of an assembler.

Object files and libraries are combined and loaded by a second program to produce the executable binary. The loader combines the roles of second half of the assembler, global optimiser, and loader. There is a third small program that serves as an assembler. It takes an assembler-like input and performs a simple translation into the object format.

3. The Language

The compiler implements ANSI C with some restrictions and extensions. [Ker88] If this had been a product-oriented project rather than a research vehicle, the compiler would have implemented exact ANSI C. Several of the poorer features were left out. Also, several extensions were added to help in the implementation of Plan 9. [Pik90] There are many more departures from the standard, particularly in the libraries, that are beyond the scope of this paper.

† This paper was originally published in Proc. of the Summer 1990 UKUUG Conf., London, July, 1990, pp. 41-51
Editor: Sunil K Das, City University London.

3.1. Register, volatile, const

The keywords `register`, `volatile`, and `const`, are recognised syntactically but are semantically ignored. `Volatile` seems to have no meaning, so it is hard to tell if ignoring it is a departure from the standard. `Const` only confuses library interfaces with the hope of catching some rare errors.

`Register` is a holdover from the past. Registers should be assigned over the individual lives of a variable, not on the whole variable name. By allocating registers over the life of a variable, rather than by pre-allocating registers at declaration, it is usually possible to get the effect of about twice as many registers. The compiler is also in a much better position to judge the allocation of a register variable than the programmer. It is extremely hard for a programmer to place register variables wisely. When one does, the code is usually optimised to a particular compiler or computer. The portability of the performance of a program with register declarations is poor.

There is a semantic feature of a declared register variable in ANSI C – it is illegal to take its address. This compiler does not catch this "mistake." It would be easy to carry a flag in the symbol table to rectify this, but that seems fussy.

3.2. The pre-processor

The C pre-processor is probably the biggest departure from the ANSI standard. Most of differences are protests about common usage. Some of the difference is due to the generally poor specification of the existing pre-processors prior to the ANSI report.

This compiler does not support `#if`, though it does handle `#ifdef`. In practice, `#if` is almost always followed by a variable like "pdp11." What it means is that the programmer has buried some old code that will no longer compile. Another common usage is to write "portable" code by expanding all possibilities in a jumble of left-justified chicken scratches.

As an alternate, the compiler will compile very efficient normal `if` statements with constant expressions. This is usually enough to rewrite old `#if`-laden code.

If all else fails, the compiler can be run with any of the existing pre-processors that are still maintained as separate passes.

3.3. Unnamed substructures

The most important and most heavily used of the extensions is the declaration of an unnamed substructure or subunion. For example:

```
struct      lock
{
      int    locked;
} *lock;

struct      node
{
      int   type;
      union
      {
            double dval;
            float  fval;
            long   lval;
      };
      struct lock;
} *node;
```

This is a declaration with an unnamed substructure, `lock`, and an unnamed subunion. This shows the two major usages of this feature. The first allows references to elements of the subunit to be accessed as if they were in the outer structure. Thus `node->dval` and `node->locked` are legitimate references. In C, the name of a union is almost always a non-entity that is mechanically declared and used with no purpose.

The second usage is poor man's classes. When a pointer to the outer structure is used in a context that is only legal for an unnamed substructure, the compiler promotes the type. This happens in assignment statements and in argument passing where prototypes have been declared. Thus, continuing with the example,

```
        lock = node;
```

would assign a pointer to the unnamed lock in the node to the variable `lock`. Another example,

```
        extern void lock(struct lock*);
        func(...)
        {
                ...
                lock(node);
                ...
        }
```

will pass a pointer to the lock substructure.

It would be nice to add casts to the implicit conversions to unnamed substructures, but this would conflict with existing C practice. The problem comes about from the almost ambiguous dual meaning of the cast operator. One usage is conversion; for example `(double)5` is a conversion, but `(struct lock*)node` is a PL/1 ``unspec.''

3.4. Structure displays

A structure cast followed by a list of expressions in braces is an expression with the type of the structure and elements assigned from the corresponding list. Structures are now almost first-class citizens of the language. It is common to see code like this:

```
        r = (Rectangle){point1, (Point){x,y+2}}.
```

3.5. Initialisation indexes

In initialisers of arrays, one may place a constant expression in square brackets before an initialiser. This causes the next initialiser to go in that indicated element. This feature comes from the expanded use of `enum` declarations. Example:

```
        enum   errors
        {
               Etoobig,
               Ealarm,
               Egreg,
        };
        char* errstrings[] =
        {
               [Etoobig]    "Arg list too long",
               [Ealarm]     "Alarm call",
               [Egreg]      "Panic out of mbufs",
        };
```

This example also shows a micro-extension – it is legal to place a comma on the last `enum` in a list. (Wow! What were they thinking?)

3.6. External register

The declaration `extern register` will dedicate a register to a variable on a global basis. It can only be used under special circumstances. External register variables must be identically declared in all modules and libraries. The declaration is not for efficiency, although it is efficient, but rather it represents a unique storage class that would be hard to get any other way. On a shared-memory multi-processor, an external register is one-per-machine and neither one-per-procedure (automatic) or one-per-system (external). It is used for two variables in the Plan 9 kernel, *u* and *m*. *U* is a pointer to the structure representing the currently running process and *m* is a pointer to the per-machine data structure.

3.7. Goto case, goto default

The last extension has not been used, so is probably not a good idea. In a switch it is legal to say `goto case 5` or `goto default` with the obvious meaning.

4. Object module conventions

The overall conventions of the runtime environment are very important to runtime efficiency. In this section, several of these conventions are discussed.

4.1. Register saving

In most compilers, the called program saves the exposed registers. This compiler has the caller save the registers. There are arguments both ways. With caller-saves, the leaf subroutines can use all the registers and never save them. If you spend a lot of time at the leaves, this seems preferable. In called-saves, the saving of the registers is done in the single point of entry and return. If you are interested in space, this seems preferable. In both, there is a degree of uncertainty about what registers need to be saved. The convincing argument is that with caller-saves, the decision to registerise a variable can include the cost of saving the register across calls.

Perhaps the best method, especially on computers with many registers, would to have both caller-saved registers and called-saved registers.

In the Plan 9 operating system, calls to the kernel look like normal subroutine calls. As such the caller has saved the registers and the system entry does not have to. This makes system call considerably faster. Since this is a potential security hole, and can lead to non-determinisms, the system may eventually save the registers on entry, or more likely clear the registers on return.

4.2. Calling convention

Rule: "It is a mistake to use the manufacturer's special call instruction." The relationship between the (virtual) frame pointer and the stack pointer is known by the compiler. It is just extra work to mark this known point with a real register. If the stack grows towards lower addresses, then there is no need for an argument pointer. It is also at a known offset from the stack pointer. If the convention is that the caller saves the registers, then the entry point saves no registers. There is therefore no advantage to a special call instruction.

On the National 32100 computer programs compiled with the simple "jsr" instruction would run in about half the time of programs compiled with the "call" instruction.

4.3. Floating stack pointer

On computers like the VAX and the 68020, there is a short, fast addressing mode to push and pop the top of stack. In a sequence of subroutine calls within a basic block, arguments may be pushed and popped many times. Pushing arguments is, to some extent, a useful activity, but popping is just overhead. If the arguments of the first call are left on the stack for the second call, a single pop of both sets of arguments (usually an "add" instruction) will suffice for both calls. This optimisation is worth several percent in both space and runtime of object modules.

The only penalty comes in debugging, when the distance between the stack pointer and the frame pointer must be communicated as a program counter-dependent variable rather than a single variable for an entire subroutine.

4.4. Functions returning structures

Structures longer than one word are awkward to implement since they do not fit in registers and must be passed around in memory. Functions that return structures are particularly clumsy. These compilers pass the return address of a structure as the first argument of a function that has a structure return value. Thus

```
        x = f(...)
```

is rewritten as

```
        f(&x, ...).
```

This saves a copy and makes the compilation much less clumsy. A disadvantage is that if you call this function without an assignment, a dummy location must be invented. An earlier version of the compiler passed a null pointer in such cases, but was changed to pass a dummy argument after measuring some running programs.

There is also a danger of calling a function that returns a structure without declaring it as such. Before ANSI C function prototypes, this would probably be enough consideration to find some other way of returning structures. These compilers have an option that complains every time that a subroutine is compiled that has not been fully specified by a prototype, which catches this and many other errors. This is now the default and is highly recommended for all ANSI C compilers.

5. Implementation

The compiler is divided internally into four machine-independent passes, four machine-dependent passes, and an output pass. The next nine sections describe each pass in order.

5.1. Parsing

The first pass is a YACC-based parser. [Joh79b] All code is put into a parse tree and collected, without interpretation, for the body of a function. The later passes then walk this tree.

The input stream of the parser is a pushdown list of input activations. The preprocessor expansions of `#define` and `#include` are implemented as pushdowns. Thus there is no separate pass for preprocessing.

Even though it is just one pass of many, the parsing take 50% of the execution time of the whole compiler. Most of this (75%) is due to the inefficiencies of YACC. The remaining 25% of the parse time is due to the low level character handling. The flexibility of YACC was very important in the initial writing of the compiler, but it would probably be worth the effort to write a custom recursive descent parser.

5.2. Typing

The next pass distributes typing information to every node of the tree. Implicit operations on the tree are added, such as type promotions and taking the address of arrays and functions.

5.3. Machine-independent optimisation

The next pass performs optimisations and transformations of the tree. Typical of the transforms: `&*x` and `*&x` are converted into `x`. Constant expressions are converted to constants in this pass.

5.4. Arithmetic rewrites

This is another machine-independent optimisation. Subtrees of add, subtract, and multiply of integers are rewritten for easier compilation. The major transformation is factoring; `4+8*a+16*b+5` is transformed into `9+8*(a+2*b)`. Such expressions arise from address manipulation and array indexing.

5.5. Addressability

This is the first of the machine-dependent passes. The addressability of a computer is defined as the expression that is legal in the address field of a machine language instruction. The addressability of different computers varies widely. At one end of the spectrum are the 68020 and VAX, which allow a complex array of incrementing, decrementing, indexing and relative addressing. At the other end is the MIPS, which allows registers and constant offsets from the contents of a register. The addressability can be different for different instructions within the same computer.

It is important to the code generator to know when a subtree represents an address of a particular type. This is done with a bottom-up walk of the tree. In this pass, the leaves are labelled with small integers. When an internal node is encountered, it is labelled by consulting a table indexed by the labels on the left and right subtrees. For example, on the 68020 computer, it is possible to address an offset from a named location. In C, this is represented by the expression `*(&name+constant)`. This is marked addressable by the following table. In the table, a node represented by the left column is marked with a small integer from the right column. Marks of the form `A1` are addressable while marks of the form `N1` are not addressable.

Node	**Marked**
`name`	`A1`
`const`	`A2`
`&A1`	`A3`
`A3+A1`	`N1` (note this is not addressable)
`*N1`	`A4`

Here there is a distinction between a node marked `A1` and a node marked `A4` because the address operator of an `A4` node is not addressable. So to extend the table:

Node	**Marked**
`&A4`	`N2`
`N2+N1`	`N1`

The full addressability of the 68020 is expressed in 18 rules like this. When one ports the compiler, this table is usually initialised so that leaves are labelled as addressable and nothing else. The code produced is poor, but porting is easy. The table can be extended later.

In the same bottom-up pass of the tree, the nodes are labelled with a Sethi-Ullman complexity. [Set70] This number is roughly the number of registers required to compile the tree on an ideal machine. An addressable node is marked 0. A function call is marked infinite. A unary operator is marked as the maximum of 1 and the mark of its subtree. A binary operator with equal marks on its subtrees is marked with a subtree mark plus 1. A binary operator with unequal marks on its subtrees is marked with the maximum mark of its subtrees. The actual values of the marks are not too important, but the relative values are. The goal is to compile the harder (larger mark) subtree first.

5.6. Code generation

Code is generated by simple recursive descent. The Sethi-Ullman complexity completely guides the order. The addressability defines the leaves. The only difficult part is compiling a tree that has two infinite (function call) subtrees. In this case, one subtree is compiled into the return register (usually the most convenient place for a function call) and then stored on the stack. The other subtree is compiled into the return register and then the operation is compiled with operands from the stack and the return register.

There is a separate boolean code generator that compiles conditional jumps. This is fundamentally different than compiling an expression. The result of the boolean code generator is the position of the program counter and not an expression. The boolean code generator is an expanded version of that described in chapter 8 of Aho, Sethi, and Ullman. [Aho87]

There is a considerable amount of talk in the literature about automating this part of a compiler with a machine description. Since this code generator is so small (less than 500 lines of C) and easy, it hardly seems worth the effort.

5.7. Registerisation

Up to now, the compiler has operated on syntax trees that are roughly equivalent to the original source language. The previous pass has produced machine language in an internal format. The next two passes operate on the internal machine language structures. The purpose of the next pass is to reintroduce registers for heavily used variables.

All of the variables that can be potentially registerised within a routine are placed in a table. (Suitable variables are all automatic or external scalars that do not have their addresses extracted. Some constants that are hard to reference are also considered for registerisation.) Four separate data flow equations are evaluated over the routine on all of these variables. Two of the equations are the normal set-behind and used-ahead bits that define the life of a variable. The two new bits tell if a variable life crosses a function call ahead or behind. By examining a variable over its lifetime, it is possible to get a cost for registerising. Loops are detected and the costs are multiplied by three for every level of loop nesting. Costs are sorted and the variables are replaced by available registers on a greedy basis.

The 68020 has two different types of registers. For the 68020, two different costs are calculated for each variable life and the register type that affords the better cost is used. Ties are broken by counting the number of available registers of each type.

Note that externals are registerised together with automatics. This is done by evaluating the semantics of a "call" instruction differently for externals and automatics. Since a call goes outside the local procedure, it is assumed that a call references all externals. Similarly, externals are assumed to be set before an "entry" instruction and assumed to be referenced after a "return" instruction. This makes sure that externals are in memory across calls.

The overall results are very satisfying. It would be nice to be able to do this processing in a machine-independent way, but it is impossible to get all of the costs and side effects of different choices by examining the parse tree.

Most of the code in the registerisation pass is machine-independent. The major machine-dependency is in examining a machine instruction to ask if it sets or references a variable.

5.8. Machine code optimisation

The next pass walks the machine code for opportunistic optimisations. For the most part, this is highly specific to a particular computer. One optimisation that is performed on all of the computers is the removal of unnecessary "move" instructions. Ironically, most of these instructions were inserted by the previous pass. There are two patterns that are repetitively matched and replaced until no more matches are found. The first tries to remove "move" instructions by relabelling variables.

When a "move" instruction is encountered, if the destination variable is set before the source variable is referenced, then all of the references to the destination variable can be renamed to the source and the "move" can be deleted. This transformation uses the reverse data flow set up in the previous pass.

An example if this pattern is depicted in the following table. The pattern is in the left column and the replacement action is in the right column.

`MOVE a,b`	(remove)
(no use of `a`)	
`USE b`	`USE a`
(no use of `a`)	
`SET b`	`SET b`

Experiments have shown that it is marginally worth while to rename uses of the destination variable with uses of the source variable up to the first use of the source variable.

The second transform will do relabelling without deleting instructions. When a "move" instruction is encountered, if the source variable has been set prior to the use of the destination variable then all of the references to the source variable are replaced by the destination and the "move" is inverted. Typically, this transformation will alter two "move" instructions and allow the first transformation another chance to remove code. This transformation uses the forward data flow set up in the previous pass.

Again, the following is a depiction of the transformation where the pattern is in the left column and the rewrite is in the right column.

`SET a`	`SET b`
(no use of `b`)	
`USE a`	`USE b`
(no use of `b`)	
`MOVE a,b`	`MOVE b,a`

Iterating these transformations will usually get rid of all redundant "move" instructions.

A problem with this organisation is that the costs of registerisation calculated in the previous pass must depend on how well this pass can detect and remove redundant instructions. Often, a fine candidate for registerisation is rejected because of the cost of instructions that are later removed. Perhaps the registerisation pass should discount a large percentage of a ''move'' instruction anticipating the effectiveness of this pass.

5.9. Writing the object file

The last pass walks the internal assembly language and writes the object file. The object file is reduces in size by about a factor of three with simple compression techniques. The most important aspect of the object file format is that it is machine-independent. All integer and floating numbers in the object code are converted to known formats and byte orders. This is important for Plan 9 because the compiler might be run on different computers.

6. The loader

The loader is a multiple pass program that reads object files and libraries and produces an executable binary. The loader also does some minimal optimisations and code rewriting. Many of the operations performed by the loader are machine-dependent.

The first pass of the loader reads the object modules into an internal data structure that looks like binary assembly language. As the instructions are read, unconditional branch instructions are removed. Conditional branch instructions are inverted to prevent the insertion of unconditional branches. The loader will also make a copy of a few instructions to remove an unconditional branch. An example of this appears in a later section.

The next pass allocates addresses for all external data. Typical of computers is the 68020 which can reference ±32K from an address register. The loader allocates the address register `A6` as the static pointer. The value placed in `A6` is the base of the data segment plus 32K. It is then cheap to reference all data in the first 64K of the data segment. External variables are allocated to the data segment with the smallest variables allocated first. If all of the data cannot fit into the first 64K of the data segment, then usually only a few large arrays need more expensive addressing modes.

For the MIPS computer, the loader makes a pass over the internal structures exchanging instructions to try to fill ''delay slots'' with useful work. (A delay slot on the MIPS is a euphemism for a timing bug that must be avoided by the compiler.) If a useful instruction cannot be found to fill a delay slot, the loader will insert ''noop'' instructions. This pass is very expensive and does not do a good job. About 20% of all instructions are in delay slots. About 50% of these are useful instructions and 50% are ''noops.'' The vendor supplied assembler does this job much more effectively filling about 80% of the delay slots with useful instructions.

On the 68020 computer, branch instructions come in a variety of sizes depending on the relative distance of the branch. Thus the size of branch instructions can be mutually dependent on each other. The loader uses a multiple pass algorithm to resolve the branch lengths. [Szy78] Initially, all branches are assumed minimal length. On each subsequent pass, the branches are reassessed and expanded if necessary. When no more expansions occur, the locations of the instructions in the text segment are known.

On the MIPS computer, all instructions are one size. A single pass over the instructions will determine the locations of all addresses in the text segment.

The last pass of the loader produces the executable binary. A symbol table and other tables are produced to help the debugger to interpret the binary symbolically.

The loader has source line numbers at its disposal, but the interpretation of these numbers relative to `#include` files is not done. The loader is also in a good position to perform some global optimisations, but this has not been exploited.

7. Performance

The following is a table of the source size of the various components of the compilers.

lines	module
409	machine-independent compiler headers
975	machine-independent compiler Yacc
5161	machine-independent compiler C
819	68020 compiler headers
6574	68020 compiler C
223	68020 loader headers
4350	68020 loader C
461	MIPS compiler headers
4820	MIPS compiler C
263	MIPS loader headers
4035	MIPS loader C
3236	Crisp compiler headers
2526	Crisp compiler C
132	Crisp loader headers
2256	Crisp loader C

The following table is timing of a test program that does Quine-McClusky boolean function minimisation. The test program is a single file of 907 lines of C that is dominated by bit-picking and sorting. The execution time does not significantly depend on library implementation. Since no other compiler runs on Plan 9, these tests were run on a single-processor MIPS 3000 computer with vendor supplied software. The optimiser in the vendor supplied compiler is reputed to be extremely good. Another compiler, *lcc*, is compared in this list. *Lcc* is another new and highly portable compiler jointly written at Bell Labs and Princeton. None of the compilers were tuned on this test.

1.0s	new cc compile time
0.5s	new cc load time
90.4s	new cc run time
1.6s	vendor cc compile time
0.1s	vendor cc load time
138.8s	vendor cc run time
4.0s	vendor cc –O compile time
0.1s	vendor cc –O load time
84.7s	vendor cc –O run time
1.6s	vendor lcc compile time
0.1s	vendor lcc load time
96.3s	vendor lcc run time

Although it was not possible to directly compare *gcc* to the new compiler, *lcc* typically compiles in 50% of the time of *gcc* and the object runs in 75% of the time of *gcc*. The original *pcc* compiler is also not directly compared. It is too slow in both compilation and runtime to compete with the above compilers. Since *pcc* has not been updated to accept ANSI function prototypes, it is also hard to find test programs to form a comparison.

8. Example

Here is a small example of a fragment of C code to be compiled on the 68020 compiler.

```
int   a[10];
void
f(void)
{
      int i;

      for(i=0; i<10; i++)
            a[i] = i;
}
```

The following is the tree of the assignment statement after all machine-independent passes. The numbers in angle brackets are addressabilities. Numbers 10 or larger are addressable. The addressability, 9, for the `INDEX` operation means addressable if its second operand is placed in an index register. The number in parentheses is the Sethi-Ullman complexity. The typing information is at the end of each line.

```
ASSIGN (1) long
      INDEX <9> long
            ADDR <12> *long
                  NAME "a" 0 <10> long
            NAME "i" -4 <11> *long
      NAME "i" -4 <11> long
```

The following is the 68020 machine language generated before the registerisation pass. Note that there is no assembly language in this compiler; this is a print of the internal form in the same sense as the previous tree is a print of that internal form.

Here is some explanation of notation: `(SP)` denotes an automatic variable; `(SB)` denotes an external variable; `A7` is the stack pointer, `$4` is a constant.

```
f:    TEXT
      SUBL  $4,A7
      CLRL  i(SP)
loop: MOVL  $10,R0
      CMPL  R0,i(SP)
      BLE   ret
      MOVL  i(SP),R0
      MOVL  i(SP),a(SB)(R0.L*4)
      ADDL  $1,i(SP)
      JMP   loop
ret:  ADDL  $4,A7
      RTS
```

The following is the code after all compiling passes, but before loading:

```
f:    TEXT
      SUBL  $4,A7
      CLRL  R1
loop: MOVL  $10,R0
      CMPL  R0,R1
      BLE   ret
      MOVL  R1,a(SB)(R1.L*4)
      ADDL  $1,R1
      JMP   loop
ret:  ADDL  $4,A7
      RTS
```

The following is the code produced by the loader. The only real difference is the expansion and inversion of the loop condition to prevent an unconditional branch.

```
f:      TEXT
        CLRL  R1
loop:   MOVL  $10,R0
        CMPL  R0,R1
        BLE   ret
l1:     MOVL  R1,a(SB)(R1.L*4)
        ADDL  $1,R1
        MOVL  $10,R0
        CMPL  R0,R1
        BGT   l1
ret:    RTS
```

The compare sequence

```
        MOVL  $10,R0
        CMPL  R0,R1
```

was expanded from the single instruction

```
        CMPL  $10,R1
```

because the former is both shorter and faster. The relatively dumb loader made a second copy of the sequence without realising that the

```
        MOVL  $10,R0
```

is redundant.

9. Conclusions

The new compilers compile quickly, load slowly, and produce medium quality object code. The compilers are relatively portable, requiring but a couple weeks work to produce a compiler for a different computer. As a whole, the experiment is a success. For Plan 9, where we needed several compilers with specialised features and our own object formats, the project was indispensable.

Two problems have come up in retrospect. The first has to do with the division of labour between compiler and loader. Plan 9 runs on a multi-processor and as such compilations are often done in parallel. Unfortunately, all compilations must be complete before loading can begin. The load is then single-threaded. With this model, any shift of work from compile to load results in a significant increase in real time. The same is true of libraries that are compiled infrequently and loaded often. In the future, we will try to put some of the loader work back into the compiler.

The second problem comes from the various optimisations performed over several passes. Often optimisations in different passes depend on each other. Iterating the passes could compromise efficiency, or even loop. We see no real solution to this problem.

10. References

Aho87. Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman, *Compilers – Principles, Techniques, and Tools*, Addison Wesley, Reading, MA (1987).

Joh79a. S. C. Johnson, "A Tour Through the Portable C Compiler," in *UNIX Programmer's Manual, Seventh Ed., Vol. 2A*, AT&T Bell Laboratories, Murray Hill, NJ (1979).

Joh79b. S. C. Johnson, "YACC – Yet Another Compiler Compiler," in *UNIX Programmer's Manual, Seventh Ed., Vol. 2A*, AT&T Bell Laboratories, Murray Hill, NJ (1979).

Kan88. Gerry Kane, *MIPS RISC Architecture*, Prentice-Hall, Englewood Cliffs, NJ (1988).

Ker88. Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language, Second Edition*, Prentice-Hall, Englewood Cliffs, NJ (1988).

Mot85. Motorola, *MC68020 32-Bit Microprocessor User's Manual, Second Edition*, Prentice-Hall, Englewood Cliffs, NJ (1985).

Pik90. Rob Pike, Dave Presotto, Ken Thompson, and Howard Trickey, ''Plan 9 from Bell Labs,'' *Proc. UKUUG Conf.*, London, UK (July 1990).

Set70. R. Sethi and J. D. Ullman, ''The Generation of Optimal Code for Arithmetic Expressions,'' *J. ACM* **17**(4), pp. 715-728 (1970).

Szy78. T. G. Szymanski, ''Assembling Code for Machines with Span-dependent Instructions,'' *Comm. ACM* **21**(4), pp. 300-308 (1978).

A UNIX† Task Broker

Andrew McRae

Megadata Pty Ltd.
2/37 Waterloo Rd
North Ryde.
andrew@megadata.mega.oz.au

This paper is the first of two papers describing a redundant process manager and services lookup directory.

This paper describes a UNIX Task Broker, an application which provides redundant processing configurations using multiple hetrogeneous hosts connected via a network.

Firstly some background is provided to explain the context in which the Task Broker was developed; then the overall functionality and environment is discussed. Some closer detail is given of the various components that make up the system, and some real world results of delivered systems are reported.

1. Introduction.

The growth and development of the distributed computing environment has brought with it mixed blessings; on one hand cheap and affordable computing power has meant a major increase in the amount of CPU performance able to be applied to data processing; on the other hand it is often difficult (especially with a hetrogeneous distributed environment) to rationally apply the performance in a manner which guarantees high user availability and robustness, yet providing application invisibility and best use of resources.

To solve the general problem of high availability and robustness in the UNIX environment, Megadata has developed a distributed Task Broker system, comprising an application task manager and a services directory manager. The combination of these modules allow a group of diverse networked Unix systems to run a set of application services providing mutual backup in the event of host or network failure (for high reliability multiple physical networks remove any single point of failure). A Services Directory protocol allows dynamic discovery of available services for client applications, and automatic re-homing to a new server in the event of host or software failure. The Services Directory protocol is described in another paper.

One of the most interesting aspects of this system is the fact that systems of entirely different architecture (and systems from different vendors) may form a highly reliable network that is expandable and robust, where applications will 'fail-over' from one system to the other. This gives the customer independance from any one vendor's system, demonstrating that Open Systems are a reality in the real time world.

2. Background.

Traditional fault tolerant systems rely on proprietary hardware and software to ensure a guaranteed level of availability. Availability is measured as a percentage of up-time to down-time; typical real time systems must meet or exceed 99.9% availability under all operating conditions, which is equal to less than 9 hours down-time in a year. A goal is to achieve another order of magnitude of availability (99.99% - less than an hour of down-time per year). As an example, one system Megadata has had operational in the field since 1983 has experienced an accumulative total of 16 minutes downtime.

† UNIX is a trademark of Bell Laboratories.

To achieve this level of availability, real time systems usually relied on duplication (or triplication) of critical components such as CPUs, discs and other peripherals, all of which operated on proprietary bus interconnection, and relied on specialised inter-processor communication to detect host hardware or software failure (Figure 1). Because of the duplicated bus, generally only one CPU was online performing all necessary functions, with the other running in a monitoring standby mode.

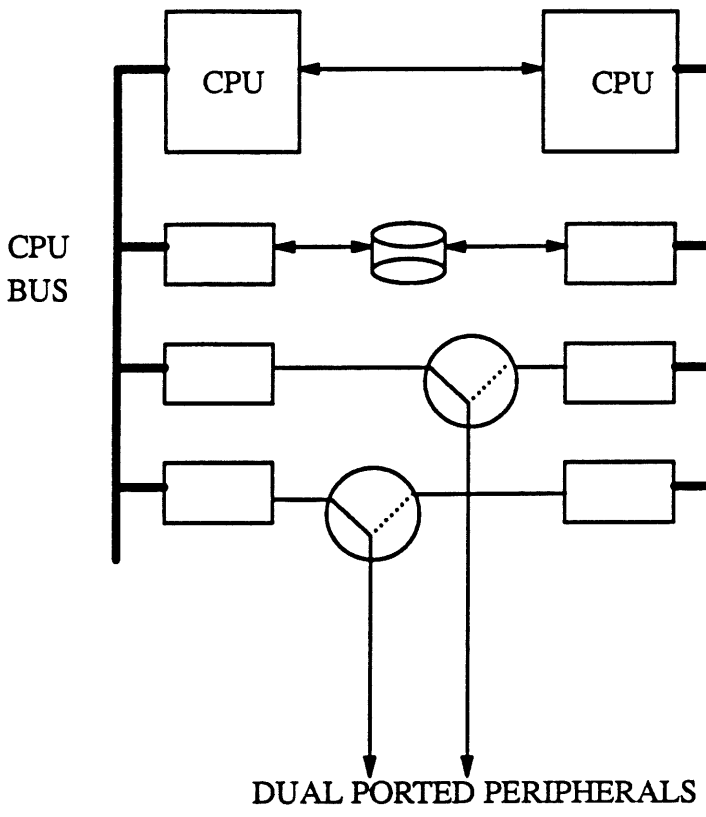


Figure 1

One side effect of this architecture is expense. It costs a lot of money both in actual hardware due to duplicated subsystems and peripherals, and in the design, support and commissioning of the specialised hardware and software. Another cost is the inability to keep up with new technology, as generally the proprietary nature of the products require a major investment in time to develop and prove. This is a valid point in favour of Open Systems in general, though especially relevant in the real time field which is traditionally slow to accept new technology.

Another problem is that the redundant systems must be in physical proximity because of the reliance on the bus hardware to share peripherals. More and more installations are requiring a greater degree of physical distribution, and yet maintain some degree of backup across the communication links connecting the control equipment.

Another major factor is that the implementor is at the mercy of the vendor supplying the major components, and cannot take advantage of any price/performance benefits offered by other vendors.

When Megadata started developing real time applications under UNIX instead of a proprietary architecture, it heralded a radical change in the approach to developing high availability, redundant real time systems. One point to highlight at this stage is that Megadata real time systems are not aimed at high speed laboratory data acquisition, but at high availability supervisory and control systems, in which the time scale is on the order of tens or hundreds of milliseconds, not microseconds. Specialised data acquisition hardware performs the *real* work of obtaining the telemetry data, which is then telemetered back to the UNIX host via slow speed FSK dedicated links, Wide Area Network virtual circuits or any other available communication means.

The problem that faced us was how to achieve high availability and fault tolerance using standard 'Open Systems' i.e. Unix workstations without specialised hardware.

After some evolution, the following diagram shows a new UNIX based architecture replacing the older style architecture (figure 2).

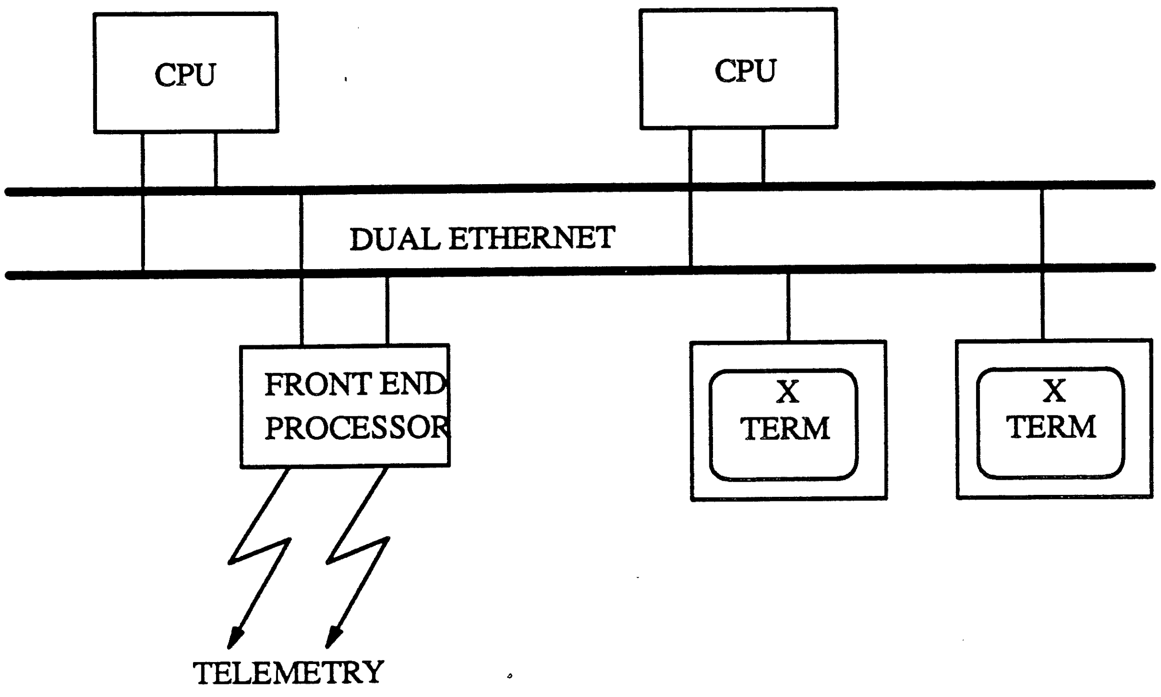


Figure 2

The new hardware architecture is orientated around a dual Ethernet LAN (for even greater levels of fault tolerance more LAN sub-nets or splitting segments with bridges may be considered), with two or more hosts acting as central processing nodes, and peripherals distributed along one or both LANs.

The new architecture has some major advantages over the older architecture:

- Being LAN based rather than CPU-bus based means that the peripherals need not be duplicated on a per CPU basis - this results in a major saving in dual peripheral switching hardware.
- The system may be upgraded much more easily, by replacing hosts or by simply adding new peripherals.
- A LAN configuration means that physical distribution of processing hardware is much more configurable, and the savings in cabling costs alone are high.
- The number of hosts available for operational duty is not limited to two. A corollary of having multiple hosts is providing a graceful degradation path in the event of multiple host failures.
- Because the peripherals are not closely associated with one particular host, applications may be distributed across different hosts.
- Different vendor hosts and peripherals may be used, since the common connection medium is a standard LAN. This also means that new technology such as FDDI may be integrated without major reconfiguration or replacement of hardware.
- By adhering to industry standards such as TCP/IP, it is easy to internetwork the system into a larger scale distributed system. By the same token it allows ready interconnection to customer MIS and vendor networks via gateways.

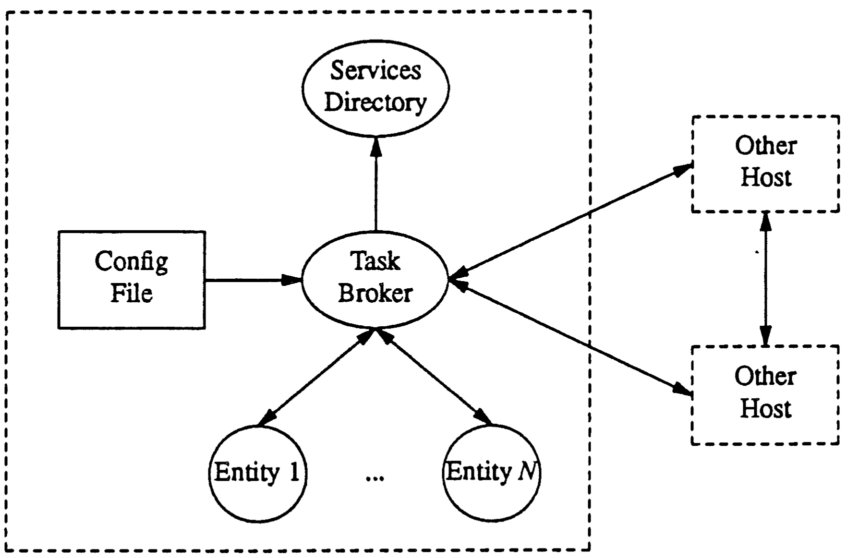
A critical component of this grand scheme is the software equivalent of the specialised hardware and software that controlled the software and host failure handling, which has evolved into a general software solution to maintaining high availability of services on multiple hosts on a network.

3. Overall Design.

The Megadata UNIX task broker basically provides host independent *service migration* (as opposed to *process* migration). True process migration operates by moving a complete process image (including file descriptors, memory image, process state etc.) transparently between different CPUs, either to load balance or to cater for processor failure. Very few systems provide this level of redundancy, and it requires a large degree of operating system support, high network bandwidth and network protocol support to achieve it. Standard UNIX (if such a beast ever really exists), does not provide any support for process migration.

Service migration, on the other hand, is the ability for clients to dynamically discover hosts which are providing required services, and then if some failure causes the host to stop providing the service, then another host will 'pick up' the service and initiate action whereby the client(s) of the failed host can transfer to the new host. In other words, the service *migrates* to a new host. Two elements work in conjunction to provide this action. The **Task Broker** performs the role of detecting host failures and ensuring that at least one host is providing the service by actively monitoring the operational state of processes running on other hosts via a task broker network protocol, and if need be running the processes locally; the **Services Directory** provides a dynamic database of services and hosts that clients query to discover a host that can provide a particular service. The Task Broker co-operates with the Services Directory by informing it when hosts or services are not available.

The Task Broker appears thus:



Each host that is involved in the system runs an instance of the Task Broker.

The tasks that the Task Broker controls are known as *entities*, where each entity is one or more UNIX processes. For an entity to be operational (or *online*), all processes in the entity group must be running. If any process dies, or is otherwise unrunnable, then the whole entity is considered *offline*.

Multiple instances of the Task Broker may run on the same host, and each is assigned a separate network port to communicate with other Task Brokers. Thus it is possible to have completely separate and distinct *systems* running on a group of hosts. Each system is uniquely identified via an defined name, known as a *System Identification*, or just System ID.

The hosts that co-operate in providing the services need not be on the same physical network, but may be scattered throughout a wide area network.

A configuration file is used by the Task Broker to describe the hosts upon which entities are to be run, and each entity is defined (specifying the programs to run, arguments, home directory etc.). Each Task Broker in the same system on different hosts shares the same configuration file (making it easy to maintain).

4. Host Failure.

When several Task Brokers on different hosts are operational in a system, they continually monitor each other by communicating to the other hosts their state (including the list of entities that are currently online on this host). The transmission interval is configurable, and determines the length of time that elapses before host failures are detected. If two or more watchdog packets are missed from one host than the other hosts poll the (possibly failed) host, and if no reply is obtained then the host is considered failed. The entities that the host was running are re-arbitrated amongst the remaining hosts.

Dual physical LANs may be provided to remove the possibility of some kind of network problem, and the Task Broker will always attempt to use both physical paths to communicate to Task Brokers on other hosts.

Using a watchdog interval of 1 second (the lowest allowed) host failure can be detected within 4 seconds, and depending on the time it takes for entities to initialize, clients can be informed of the new server and be re-homed to the host providing the service within 5 to 10 seconds of the failure of the original hosts. A interval of one second does not generate excessive network traffic, nor does it load the hosts excessively (setting slower rates will proportionally relax the timing constraints).

5. Task Arbitration.

When the Task Brokers are started on the target hosts, the configuration file will specify which hosts are allowed to run the list of entities. Two types of host lists are attached to each entity. A *preferred* host list specifies a list of hosts for running a particular entity. The order of hosts is important, as they are assumed to be in priority order. Lower priority hosts will only start the entities if (and only if) the higher priority hosts are deemed to be inoperative. If a lower priority host is running an entity, and a host that is marked (for this entity) as being a more preferred host becomes available, then the lower priority host will yield up the entity to the preferred host.

Another host list may be specified either instead of, or appended to, the preferred list. This list places no priority onto the hosts, so arbitration becomes somewhat more random.

If an entity has no clear priority indicating which host to run on, the Task Broker on each host arbitrates by sending a *claim entity* message to all the other hosts that the entity can operate on. Attached is a psuedo-random number, and if two Task Brokers collide when arbitrating the random number is used as a magic token to select between the two.

A number of rules govern the arbitration process, ensuring that the same entity cannot be accidently started on different hosts at the same time. Basically every host that starts an entity **must** have obtained a go-ahead token from every host capable of running this entity. Of course if a host is broken it is considered *offline*, and does not take part in the arbitration process.

The only way possible that an enitity will be online on two hosts at the same time is if the hosts cannot communicate at all e.g all communication paths between the two hosts are broken. If this occurs, and communications are re-established, the task brokers on each host will detect the duplicated entity, and immediately abort its own running copy and attempt a new arbitration cycle.

6. Status Indications.

The task broker may be configured to listen on a UNIX domain socket. Other processes may *connect* to this socket and obtain information concerning the status of entities and hosts. When the socket is first connected, the task broker will report the current state of all entities as being `ONLINE` or `OFFLINE` depending on whether this host is running the entity or not. The state of each host is also reported, whether it is UP or DOWN. If during the connection the status of the entities or the hosts change, the new status for that entity or hosts will be reported. The status monitoring connection may then be a once-off reporting of current status, then changes being reported as they occur.

7. Configuration File

When the task broker is executed, it expects the name of a configuration file as one of its arguments. If no name is given, then it defaults to **/etc/system-config**. This configuration file must be the same for all task brokers operating as part of the same online system.

The file contains task broker parameters such as the network port number for peer to peer communication, defaults for the watchdogs timers, etc. Each entity along with the entity's parameters is defined as part of the configuration file.

Similar to many other configuration files under UNIX, the task broker's configuration file is ASCII based, with free form input. Line continuations using backslash ('\') are supported, and tokens may be separated by spaces or tabs for readability. The character '#' will cause the rest of the line to be ignored. For strings that contain blanks or tabs, quotation marks may be used to quote the string. Strings quoted in this way are considered a single token. Indentation may be used to make the file more readable, especially when multiple attributes pertain to an entity.

When the task broker is sent a HANGUP signal it will re-read the configuration file. If the new configuration indicates that certain entities are no longer runnable by this task broker, it will kill those entities; new entities and hosts are integrated smoothly into the currently active list of hosts and entities without affecting any running entities.

Each entity may be marked using one of three keywords to indicate how or when this entity should run. The ONCE flag indicates that the entity should be run locally by the task broker, and that if any of the processes terminate than no attempt is to be made to restart or rearbitrate for the entity. CONTINUOUS indicates that the entity should be started on the local host, and that if any of the processes should terminate the task broker should attempt to restart the entity. ARBITRATE is the default, which indicates that the task broker must arbitrate with the hosts on the HOSTS and PREFERRED lists to determine who should run this entity. When no hosts appear on either list when the ONCE or CONTINUOUS options are specified, it is assumed that the entity must run on all hosts.

Following is a sample configuration file:

```
#
#    Configuration file for task broker example
#
#   Set the system ID
#
id C999
#
#   Set up UDP port for task brokers to communicate
#
port sysman-2
#
#   Use this name for connections from processes wishing
#   to discover the status of entities
#
status /dev/sysman_sock
#
#   Send watchdogs every 3 seconds.
#
broadcast 3
#
```

```
#  Enable System logging.
#
syslog all
#
#  No network A extension
#  Net B extension is -2
#
netb -2
#
#  Entities to be controlled.
#
# Run CRTPROC on a few hosts. (continuous means restart it if
# it crashes).
#
entity CRTPROC continuous
    hosts cpu1 cpu2 mmi1 mmi2      # will run CRTPROC on all these.
    process /usr/online/crtserver  # no arguments
    alternate /backup/online/crtserver
    direct /usr/database/pictures  # cd's to this directory first
#
#  Dataproc - runs program /usr/online/dataproc on
#  cpu1 (preferred) or cpu2
#
entity DATA-PROCESSOR
    preferred cpu1 cpu2
    process /usr/online/dataproc # no arguments
    alternate /backup/online/dataproc
    process /usr/online/history
    alternate /backup/online/history
    direct /usr/database         # Has to run in this directory
    log system
#
#  EMS - must run on ems machines, but can also run on
#  others if the EMS machines are down.
#  Runs both opf and nmb
#
entity ENERGY-MANAGEMENT
   preferred ems1 ems2
   hosts cpu1 cpu2 mmi1 mmi2
   process /usr/online/nmb init-nmb -debug
   directory /usr/database/ems/nmb
   log system
#
   process /usr/online/opf init-file output-file
   directory /usr/database/ems/opf
   log file opf-output
```

The above file should exist on the hosts *cpu1*, *cpu2*, *mmi1*, *mmi2*, *ems1*, *ems2*. When the hosts are powered up, the task brokers will arbitrate according to the configuration rules in the file, and then begin execution. During operation the task broker will respond to changes in host's availability by checking periodically that the hosts running the entities are still active, and failing over (re-arbitrating) in the event of a host failure.

8. Process Execution.

The task broker executes the process(es) for each entity via *fork/exec* system calls. All file descriptors are closed except for *stdin*, *stdout* and *stderr*.

Each process may have arbitrary arguments, and in the event of a process's pathname being unrunnable, a backup process name be be declared. Each process may also have a current directory associated with it, and when that process is executed it will be started in that directory.

The LOG command allows logging from the *stderr* stream of the process. Two types of logging are supported, to a file, and to the system log. When logging to the system log is enabled, any lines sent to the *stderr* stream will be logged via the *syslog* library call. This normally causes a message to be printed on the console of the *loghost* machine. The SYSLOG command may be used to redirect log output to a file, or to turn it off entirely.

When the process is first started by the task broker, the process group number is changed to the process ID number. Any child processes subsequently started by the process will inherit the same process group number. When the task broker wishes to kill the process it sends a terminate signal to all processes belonging to that particular process group, ensuring that no processes are accidently left running.

When a process exits, the task broker will examine the status and log the reason why the process terminated. If any other host is flagged as being able to run this entity, and is seen to be operational, the task broker will not attempt to restart the entity, but will assume the other host will attempt to run it.

If the entity is flagged as running locally using the CONTINUOUS option, the task broker will attempt to restart the entity immediately.

9. Field Results and Experiences.

Because of the vendor-independant nature of the task broker protocol, it is possible to mix different makes of workstations and still provide redundant processing. In one experiment, a Hewlett-Packard host was shadowing a Sun workstation across a dual Ethernet, and upon power-cycling the Sun, the HP host smoothly took up the services that were operating on the Sun. Upon restoration of the Sun, the services were again transferred back to the original machine. This achieved our goal of creating a truly open system of real time high availability.

The task broker has been successfully installed and is operational in a number of sites. Each of these sites is considered a high reliability site, and must maintain greater availability then 99.9%. It has been ported to three common workstation architectures (Sun, Hewlett-Packard and DEC) with little effort. At this stage all sites have been designed as dual local area networks, but several planned installations are designed as wider area networks, where the task broker will be operating on hosts communicating locally on Ethernet and also communicating with remote hosts across a range of communication media.

Experience has shown that the UNIX hosts available at this time are generally very reliable, with virtually all failures due to either operator error, network failures (e.g unplugged LANs) or software error (too often OS crashes). In all cases the task broker has managed to reliably transfer the online services from the failed machine to another host, or operate correctly in the event of partial network failure. Of course we must keep in mind Jackson's Law:

"The chances of getting eaten by a lion on Main St. are one in a million, but it only takes once."

Thus it won't be for another 8 years or so that we can truly say that we have reproduced a similar level of availability by using non-proprietary network based redundancy as the older hardware-based redundant architectures.

The Hunting of the Open System†

by Rolf Jester

Open Systems Marketing Manager,
Digital Equipment Corporation, South Pacific Region;
Secretary, Australian Open Systems Users Group

"We have sailed many months, we have sailed many weeks
(Four weeks to the month you may mark),
But never as yet ('tis your Captain who speaks)
Have we caught the least glimpse of a Snark!"

From The Hunting of the Snark, Fit the Second.
by Lewis Carroll

So what is an Open System anyway? How shall we recognise one if we see it? And will we like it?

I think that it is possible now to answer these questions, or at least to lay some foundations for an emerging understanding of a new information systems discipline called Open Systems. There isn't a single one-word answer to what an open system is. It isn't something you can go out and buy at your corner store, not even at Digital or any other vendor.

In Graham Honeywill's thought-provoking editorial in DECUS news [vol. 12 number 4 November 1991], he describes an open system as "the implementation of a vision of the future", adding that "the hardware and software vendors can help us get there, but it is up to us to decide upon the vision first and then to establish the road-map to get there." That is one of the most perceptive things I have heard on this topic.

He had bought a large map representing the sea,
Without the least vestige of land:
And the crew were much pleased when they found it to be
A map they could all understand.

"What's the good of Mercator's North Poles and Equators,
Tropics, Zones and Meridian Lines?"
So the Bellman would cry: and the crew would reply,
"They are merely conventional signs!

"Other maps are such shapes, with their islands and capes!
But we've got our brave Captain to thank"
(So the crew would protest) "that he's bought us the best -
A perfect and absolute blank!"

† A slightly modified version of an article that appeared in DECUS news Volume 13 Number 1

The map of the open systems world may indeed seem to be unhelpful, or totally confusing. In what follows I shall sketch in some of the main features so that we can go on like the intrepid hunters in pursuit of the Snark.

UNIX?

In the DECUS News editorial referred to above, Graham Honeywill says that "UNIX ... is not an 'Open System'." That may be technically correct. In itself UNIX is just an operating system, and not sufficient to build a complete open system. But UNIX did give us the vision that open systems might be possible. It is a key component of many if not most open systems today, and will remain an important part of future open systems architectures.

Definition

"Come, listen, my men, while I tell you again
The five unmistakable marks
By which you may know, wheresoever you go,
The warranted genuine Snarks."

So let's start with a definition of an open system. This is the official definition of the POSIX 1003.0 committee of the U.S. Institute of Electrical and Electronics Engineers (IEEE), worked out by a thorough consensus process among professional engineers from vendors, users and governments.

".. a system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered applications software to be ported across a wide range of systems with minimal changes, to inter-operate with other applications on local and remote systems, and to interact with users in a style which facilitates user portability."

Leaving aside the mind-boggling thought of "user portability" (down-sizing gone mad maybe?), this is a good definition. It focuses on the purposes of open systems, namely the portability of applications, the inter-operability of applications, and the ease of use by end-users. It says that to achieve these things we need open specifications for interfaces. It doesn't talk about which brand name of computer, or which brand name of operating system or of data base we need to achieve it. It does recognise that only "properly engineered" applications will benefit from open systems. And that is something that is often overlooked by those pushing their particular flavour of UNIX.

The definition goes on to define what an "open specification" is:

".. a public specification that is maintained by an open, public consensus process to accommodate new technology over time and that is consistent with standards."

Such standards exist. They are independent of vendors and of particular products. It is possible today to plan an evolution from today's systems to an open systems architecture.

Standards

Far too many people use the word "Standard" to mean exactly what they want it to mean, much to the confusion of everyone else. So let's distinguish between various sorts of standards that are relevant to this topic.

Formal standards

Formal standards are made by official standards organisations at the international and national level. They are sometimes called "de jure" standards, by contrast with "de facto". Standards Australia and the Standards Association of New Zealand are two examples of formal standards organisations. The American National Standards Institute (ANSI) is important because of the leading role of U.S. vendors in our industry. At the international level, ISO and CCITT are the main organisations.

Formal standards are best in the sense that they are secure. Users and vendors can depend on them to be stable and agreed upon by everyone. They are also totally independent of particular proprietary products.

A good example is IS 9945, the ISO standard for a portable operating system interface. This is essentially the POSIX standard internationalised. Locally, Standards Australia has recently published the same specification as AS 3976.1-1991.

Formal standards are always interface definitions. They rigorously define the external behaviour of a product, not how it works inside. That leaves vendors free to compete for users' business with various products that all comply strictly but that differ internally and may differ on such criteria as price, performance, robustness, maintainability, packaging, marketing/distribution and service availability.

Consider the Open Systems Interconnection (OSI) set of standards for networking published by the ISO. Each of the component standards of OSI is one standard, but is implemented in many different products from different vendors. Assuming that they implement the same options of the same standard, they are still different products.

After all, if the "open" in "Open Systems" means anything it must mean open to competition.

De facto standards

Formal national and international standards should take precedence in a standards-based architecture. But they are not sufficient. The consensus process is slow and technology moves faster than ever. So de facto standards of various kinds are also needed. The de facto standards should supplement formal standards in such ways that compliance with the formal standards is maintained.

The best sort of de facto standard is one which has essentially universal support. A good example of this is the X Portability Guide (XPG) of the X/Open group - see below. Because the formal "POSIX" standard (IS 9945) covers only the interface to the operating system at this stage, the XPG provides a more comprehensive framework, a "Common Application Environment" (CAE), in which applications written strictly to the XPG standard are guaranteed to be portable by a re-compilation to any hardware/operating-system platform that has been verified and branded as conforming to the XPG.

De facto to formal standard

Some de facto standards go on to become formal standards, as has happened with the X Window System. This actually started life as a public-domain product, actual software, developed at MIT. A de facto standard emerged after the product became very popular, managed by the X Consortium, and adopted by all suppliers. Finally the ANSI committee X3H3.6 selected the X Window System interfaces as the basis for a formal user interface standard.

Proprietary standards

Major vendors may produce specifications that have considerable success in the market and are thus widely adopted by users and other vendors, as with IBM's SNA. Proprietary standards are useful for users of that vendors' products, but are still controlled by the vendor.

Proprietary standard to vendor-neutral standard

Some standards developed by a vendor or a group of vendors may then become accepted formally. Ethernet, originally developed by Digital, Intel and Xerox, evolved into an IEEE standard and an international standard IS 8802.3.

Proprietary products that lead to standards

Proprietary products may also end up as the basis for a standard. It was, after all, the UNIX operating system (trade marked and owned by a division of the American Telephone and Telegraph Company) that led to the standardisation efforts of UniForum (then called /usr/group) and eventually to POSIX, IS 9945 and AS 3976.

Another good example is C-ISAM, a product of Informix Software, Inc. The X/Open group, seeing the need for an ISAM standard for the UNIX environment, used this popular product as the basis for its definition because no other formal or de facto standard was available.

Products considered standard

Some proprietary products become so pervasive that they are frequently considered to be a standard. Microsoft's MS-DOS is the obvious example.

Products from major vendors, even if less universal than MS-DOS, may well serve users well as key components in their information systems architectures. But we must never lose sight of the essentially proprietary nature of such products, whose definition, evolution and licensing is in the hands of a vendor.

Profiles

A suite of standards may be so complex that subsets of it are necessary for realisable implementations in particular user environments. This is the case with the OSI network standards, where various major groups of users have defined paths through the range of options to create a workable set of standards for their needs. The Australian Government Open Systems Interconnection Profile (GOSIP) is an example that was developed by the Information Exchange Steering Committee (IESC) in Canberra, and has been formally adopted by the Federal Government.

"For, although common Snarks do no manner of harm,
Yet I feel it my duty to say,
Some are Boojums - " The Bellman broke off in alarm,
For the Baker had fainted away.

In the next article, we shall examine some of the major organisations, standards bodies, consortia, standards and products that are prominent in the open systems market, and try to produce some further clarity.

TRADEMARK ACKNOWLEDGEMENTS

IBM and SNA are trademarks of International Business Machines Corporation.
C-ISAM is a trademark of Informix Software, Inc.
Ethernet is a trademark of Xerox Corporation
MS-DOS is a trademark of Microsoft Corporation.
UNIX, System V and SVID are trademarks of UNIX Systems Laboratories.
The X Window System and X11 are trademarks of the Massachusetts Institute of Technology.

Life, the Universe and Open Systems†

by Rolf Jester

Open Systems Marketing Manager,
Digital Equipment Corporation, South Pacific Region;
Secretary, Australian Open Systems Users Group

The world of open systems can appear confusing and bewildering. It is certainly complex.

DON'T PANIC.

Since any discussion of this topic will involve use and abuse of various standards, consortia and numerous acronyms, a useful place to begin would be to clarify the roles of these various entities and place them in context.

I am going to deal with the major organisations, standards and products. And I shall distinguish among the following categories.

* Formal standards organisations
* De facto standards groups
* Government standards bodies
* User groups
* Vendor consortia
* Software providers
* Standards - formal and de facto
* Products.

There are many more organisations, standards and acronyms that could be covered, but I believe that these are the ones that are most directly relevant to current open systems issues and also the ones that need the most clarification.

FORMAL STANDARDS ORGANISATIONS

Formal standards organisations are official bodies that create, maintain and publish standards at a national or international level. Some have national Government or U.N. endorsement. The actual development work is often done by specialist professional organisations associated with the industry, involving all interested parties such as users, vendors and Governments. The process involves free discussion, criticism, wide circulation of drafts and a working towards consensus among all participants. The national and international bodies will then usually adopt and promulgate these standards after verifying that they meet the appropriate criteria.

ANSI

The American National Standards Institute is the formal national standards body for the United States.

† A slightly modified version of an article that appeared in DECUS news Volume 13 Number 1

It co-ordinates the development of standards by accredited standards organisations such as the "X3" Accredited Standards Committee which is responsible for information systems standards. Standards are actually developed by professional organisations like the IEEE (see below) on behalf of ANSI X3.

The importance of the U.S. Market and of U.S. vendors ensures that ANSI standards are internationally used and frequently adopted by ISO (see below), of which ANSI is the U.S. member. The global nature of the information industry at the same time helps ensure that these standards are produced with the international user community in mind.

CCITT

The International Telegraph and Telephone Consultative Committee is a permanent subcommittee of the International Telecommunications Union, which is a United Nations treaty organisation. It specialises in communications standards (e.g. X.25) which have become important parts of the over-all open systems framework.

IEEE

The Institute of Electrical and Electronics Engineers is a U.S.-based international professional organisation whose members are qualified engineers. The IEEE Computer Society produces formal information technology standards which are then adopted by ANSI. The IEEE Local Area Network (LAN) standards such as 802.3 are well known.

One of the most important efforts of the IEEE is the POSIX group of committees creating standards for a complete open systems environment - see below.

ISO

The International Organisation for Standardisation (ISO) is the body that co-ordinates international formal standards. Its members are the national standards bodies like SANZ, Standards Australia and ANSI. ISO has formed a joint committee with the International Electrotechnical Commission (which is responsible for electrical standards) called the Joint Technical Committee 1 (JTC1), and this committee is the one that is responsible for international information technology standards.

A standard passes through the stages of Draft Proposal (DP), Draft International Standard (DIS) and then becomes an official International Standard (IS).

JTC1 has a structure of Subcommittees (SC) and Working Groups (WG). JTC1/SC22/WG15 is the group responsible for the international "POSIX" portable operating systems interface standard IS 9945, which was ratified in 1990.

Standards Australia

Standards Australia is Australia's national formal standards body and a member of ISO. The Australian Information Industry Association (AIIA) is the link between the industry and Standards Australia.

Recently, Standards Australia published Australian Standard AS 3976.1-1991, which is IS 9945-1:1990, otherwise known as POSIX.1, the international standard for a portable operating system interface.

DE FACTO STANDARDS GROUPS

Because the formal standards process takes a long time, and because information technology is advancing so rapidly, agreement on many things is needed beyond the scope of currently ratified formal standards. Thus groups of users and vendors may form to achieve agreement on de facto standards that may eventually form the basis of formal standards.

X Consortium

The X Consortium was formed by vendors and other parties interested in managing the standardisation and promotion of the X Window System, a network-transparent window system first developed at the Massachusetts Institute of Technology (MIT). The group has been successful in maintaining the X11 standard, which has now been adopted by ANSI committee X3H3.6 as the basis for a formal user interface standard.

X/Open

X/Open is the major unifying force in the open systems industry. It is a consortium of all the major hardware vendors, and is truly representative of the industry as a whole. Its role is to publish and promote standards for an open computing environment. Members commit themselves to implement those standards in their products.

The hardware members of X/Open include AT&T/NCR, Bull, Digital, Fujitsu/ICL, HP, Hitachi, IBM, NEC, Olivetti, Prime, Siemens-Nixdorf, Sun and Unisys. In other words, every large computer supplier is a member. Importantly, both the Open Software Foundation (see below) and UNIX International (see below) are committed to complying with the X/Open standard called the X Portability Guide (XPG).

X/Open also has an active User Council consisting of representatives of many major IT user organisations, and an Independent Software vendor Council. Both of these work to ensure that the evolution of the X/Open standards meet the real needs of users.

GOVERNMENT STANDARDS BODIES

National and regional Governments are probably the largest purchasers of information technology. Being accountable to the public and responsible for a large information infrastructure, they are now tending to define vendor-neutral standards profiles for their IT purchases. These standards profiles are intended to be comprehensive information systems architectures that allow for rational and effective investment and technology planning over the long term. The organisations responsible for these standards are effectively setting an important lead for all other sectors of the economy.

CCTA

The Central Computer and Telecommunications Authority of the U.K. Government issues advice and guidelines for other central Government authorities. It has strongly endorsed the X/Open's X Portability Guide as an appropriate standards profile. It has also developed an OSI networking standards profile - the U.K. GOSIP (see below).

IESC

The Australian Government's Information Exchange Steering Committee is a group of senior executives from major Federal departments who co-ordinate information systems policies for the Federal Government. In 1990, Federal Cabinet approved the Australian Government Open Systems Interconnection Profile (GOSIP) as developed by the IESC. It is now Government policy to use OSI networking standards wherever possible, as specified in the Australian GOSIP.

Currently the IESC is working on the more far-reaching task of developing a comprehensive open systems framework along the line of the work done by the NIST in the United States (see below).

NIST

The National Institute of Standards and Technology is a U.S. Government standards body. One of its divisions is the National Computer Systems Laboratory (NCSL), which is chartered with producing Federal Information Processing Standards (FIPS) for U.S. Government information technology purchases. It therefore controls the standards direction of what is by far the biggest purchaser of information systems in the world. As such, it is one of the most important players in the whole standards movement, because no vendor can afford to ignore the FIPS.

FIPS 151 defines an Application Portability Profile (APP). This is a definition by NIST of the suite of standards that systems, components and applications must adhere to in order to achieve the aim of software portability across all vendors' systems. The APP is a comprehensive profile, but it can be summarised as a layered architecture made up of pre-existing formal standards from national and international standards bodies.

Operating system	POSIX
User interface / graphics	X Window System GKS & CGM, IGES
Languages	(ANSI) C, COBOL, Fortran, Ada, Pascal
Data management	SQL, IRDS
Data interchange	SGML, ODA/ODIF

While the needs of other user organisations may not be identical to those of the U.S. Government, NIST has at least set an excellent example in creating a vendor-neutral, product-neutral standards framework.

USER GROUPS

AUUG

Formerly known as the Australian UNIX-Systems Users Group, AUUG is now called the Australian Open Systems Users Group. It is an association of Australian IS professionals who use UNIX and other open systems. AUUG conducts conferences and publishes technical newsletters particularly for UNIX users. Although AUUG has not had a direct role in standards, it is the Australian affiliate of UniForum (see below) and also has links with X/Open. It is thus a source of information about international standards and publications from these bodies.

UniForum

UniForum is the U.S.-based international association of open systems users. It also started life as a UNIX user group and was formerly known as /usr/group. It now has over 7,000 individual members world-wide, publishes technical and general journals, a UNIX industry newsletter, technical booklets and an important directory of UNIX-based hardware and software products.

UniForum started the standardisation effort that led eventually to the POSIX standard. The many users of UNIX were dissatisfied with the many incompatibilities that existed between the many UNIX "flavours" and set about creating a common standard many years ago. The task of writing a standard proved bigger than the resources of a user group could handle, and therefore the work was handed over to the IEEE (see above), who could also apply their experience with the drafting of formal standards. See 'POSIX' below.

Apart from actively promoting POSIX and its further development, UniForum also has a link with X/Open and endorses the XPG standard. Thus the annual UNIX product directory now indicates which products conform to POSIX and to the XPG. That, in turn, along with pressure from major users, will be a factor in encouraging application developers to conform to these standards.

VENDOR CONSORTIA

Although short-term interests of vendors might seem to be best served by totally proprietary systems, the industry has changed significantly and vendors are actually better off by co-operating with wide industry consortia on common standards. As long as the products of these consortia comply with industry-wide standards, that is a good thing for users.

Vendor consortia will act on behalf of vendors, and their statements should always be evaluated critically as you would those of a vendor.

ACE

The Advanced Computing Environment (ACE) is an initiative by a group of now over 230 companies to build systems to a set of common specifications. ACE sets out to create a clone-like market for advanced systems ranging from notebook size to super-computer. In doing that, it recognises the importance of the Intel-based PC environment (MS-DOS, Windows) as used on tens of millions of desk-tops. ACE offers two operating systems for those systems - Windows NT from Microsoft, the next generation of the successful MS-Windows; and a unified UNIX operating system - Open Desktop from The Santa Cruz Operation (SCO). More recently, UNIX Systems Laboratories has stated that it will supply a version of its product UNIX System V for ACE systems.

At the higher performance levels, ACE uses Reduced Instruction Set Computing (RISC) processors from MIPS Computer, Inc. Again, both the Windows NT system and Open Desktop (ODT) will run on these systems, as will UNIX System V. Because they are built to a common agreed specification, applications will be compatible at the binary level across all vendors ACE/ODT systems in the same way that all MS-DOS applications are strictly executable on all PC clones.

Thus ACE, supported by so many vendors, small and large, from so many countries and industry sectors, creates an enormous market for compatible hardware, shrink-wrapped software, add-on components and services. It is helping software developers because they now have to port to fewer platforms. And that means that many thousands of quality applications will be available for users.

ACE products from many vendors based on the MIPS R4000 processor, on Open Desktop and on Windows NT are expected to come on the market during 1992.

OSF

The Open Software Foundation is a software development company, and is therefore described further below under Software Providers. However the original sponsors were major hardware vendors including Bull, Digital, Hitachi, HP, IBM, and Siemens-Nixdorf, so it appropriate to mention it under the heading of vendor consortia as well. OSF now has over 300 members among hardware vendors, software vendors, research institutions, Governments and end-users. Technology selections are made by the

OSF's professional staff and consultants on the basis of technical issues and market requirements.

UNIX International

UNIX International (UI) is a group of vendors who are committed to using USL's product UNIX System V. UI has some 300 or so members and it acts like a user group in that it provides feed-back to the UNIX Systems Laboratories (USL) subsidiary of AT&T as to the desired features for future versions of the System V product. Since the members of UI are vendors of systems, UI also plays a major role in the promotion of USL's products.

SOFTWARE PROVIDERS

OSF

The Open Software Foundation is a not-for-profit company that develops system software in accordance with industry standards. It is not in itself a standards body. Rather, it develops actual products (source code) and licenses them to anyone, OSF members or not. Although the OSF does not produce formal standards, its products may become the basis of de facto standards.

The OSF's employees are predominantly software engineers. However they normally do not produce new products in-house. Rather, a process of tendering, called the "open process" is used, based on "requests for technology" to elicit the best available software technology from research institutions and from hardware and software vendors. Thus OSF acts as an integrator of components from multiple sources, producing fully standards-compliant products that are then readily and inexpensively available to the whole industry.

See below for various OSF products. See also "AES" below.

USL

UNIX Systems Laboratories (USL) is a subsidiary of the American Telephone and Telegraph Company (AT&T). It develops, licenses and markets software products, especially UNIX System V (see below), one of the major UNIX operating systems.

STANDARDS: formal, de facto, and future

AES

The Application Environment Specification (AES) is the Open Software Foundation's specification of the suite of industry standards to which its products conform. Like any software developer, the OSF needs a coherent architectural framework for its products, and the AES provides OSF's engineers with a consistent basis for development. More importantly, it provides OSF's customers and end-users with the security of a firm set of industry standards to which they know OSF products will conform.

Since the OSF offers software products on the open market, it makes good sense for it to adhere to recognised industry standards in order to maximise the usefulness and applicability of its products on the widest range of platforms. In general, OSF's direct customers are hardware vendors who implement OSF products on their machines. For those vendors to easily take advantage of the OSF software, it has to conform to a common set of standards to allow easy porting and integration. Vendor "standards" will not do.

The AES specifications are not OSF's own standards, but taken from formal and de facto industry standards. The complete set is too extensive to list in a short article, but includes the POSIX operating system interface, various ANSI language standards, the X Window System network windowing interface, TCP/IP, OSI networking protocols, and SQL.

CAE

The Common Application Environment (CAE) is the name for the standards architecture developed by the X/Open group and published as the X Portability Guide - XPG (q.v.)

GOSIP

A Government Open Systems Interconnection Profile (GOSIP) is a profile of standards from the Open Systems Interconnection (OSI) suite. It is concerned with interconnection between systems of different manufacturers - i.e. networking. Because OSI is such a comprehensive set of networking standards, it has proved necessary to define sub-sets or "profiles" for various application environments, where choices have been made for the various options at each level.

The U.S., U.K., and Australian Governments, among others, have adopted their own GOSIPs, the Australian one being developed by the IESC (see above).

Note that the systems that are inter-connected need not be "open" in any other sense. They may be fully "proprietary" systems, but with OSI protocols implemented to allow them to exchange data readily.

OSI

The Open Systems Interconnection (OSI) suite is a comprehensive suite of communications protocol standards designed by ISO (see above).

POSIX

The standard for a Portable Operating System Interface (POSIX) was ratified by the IEEE in 1988 as P1003.1. It is an ANSI standard, an ISO standard (IS 9945-1) and an Australian Standard (AS 3976.1-1991).

The original P1003.1 specification is simply the definition of the interface between an application program written in the C language to an operating system. It was of course based on an abstract version of UNIX, but is now quite independent of any trade-marked and licensable product. To the best of my knowledge all major UNIX systems vendors support it.

An application that uses only the POSIX interface is fully portable to any POSIX-compliant platform. Current UNIX applications in fact still make use of the many proprietary extensions in the historical UNIX "flavours" (AIX, SunOS, HP-UX, System V, ULTRIX etc.). But pressure from major users will ensure that applications software vendors too will begin to strictly conform to POSIX in their future versions. Otherwise they sacrifice easy portability and access to the huge market for portable and standards-compliant products.

Work continues in the IEEE POSIX committees. The next standard, nearly complete now, will be the

1003.2 standard for the "shell" commands and utilities. Other POSIX committees are working on test methods for verifying conformance, real-time extensions, interfaces from other programming languages, security, system administration and many other aspects of a complete open systems environment.

But the basic 1003.1 standard (IS 9945-1) is secure and usable now.

XPG

X/Open (see above) publishes the X Portability Guide (XPG), which specifies a Common Application Environment (CAE) - a suite of compatible standards aimed at total software portability. The standards that make up the XPG are not developed by X/Open itself. Rather, X/Open adopts formal standards where they exist (such as IS 9945 or POSIX), and de facto standards where there are no formal standards (e.g. the X Window System).

The XPG is a complete computing environment rather than just an operating system interface and is thus more complete than the currently ratified parts of POSIX. It is independent of any actual product, both hardware and software.

X/Open conducts verification tests of XPG compliance and "brands" compliant products.

Since all vendors are committed to it, the XPG is the safest choice for users. Many Government agencies, especially in Europe have mandated XPG compliance for their systems. The European Commission has done so, and the U.K. Central Computer and Telecommunications Agency (CCTA - see below) recommends it. Such user endorsement in turn is driving application vendors to comply also, so that their application software packages are portable across all XPG branded systems.

VENDOR STANDARDS

SVID

The System V Interface Definition (SVID) is a document published by AT&T/NCR's subsidiary UNIX Systems Laboratories as a specification of their product System V (see below). It is a specification of the external interface to the System V operating system. Its usefulness lies in the fact that it makes it possible for other UNIX operating systems to be compatible with applications that were written for the unique features of System V - i.e. those features that are outside of the industry standards POSIX and XPG.

FUTURE STANDARDS

ANDF

The OSF's Architecture Neutral Distribution Format (ANDF) is a means of solving the problem of how to create executable software for multiple hardware architectures. The source-level standards (POSIX, XPG) achieve true portability by re-compilation for any standards-compliant platform. Application Binary Interfaces (ABIs) achieve binary compatibility between computers from various vendors provided that they use the same microprocessor chip and operating system. But ANDF goes one step further. It allows the application to be compiled into a neutral intermediate format in which it can be distributed and then loaded on various different hardware platforms through a second "installer" stage. The software developers source code is protected, but easy access to many vendors' hardware is gained because the installer need only be written once for each hardware/operating system platform.

Large IT users with multiple vendors' systems will also benefit from ANDF, as a single copy of each application will be loadable on many platforms.

The Open Software Foundation has selected the technology for ANDF and a prototype version for three different hardware platforms has been successfully demonstrated. It promises to be a truly revolutionary technology that will significantly advance the cause of portability and open systems. It may well become a de facto standard.

DCE

The OSF has produced a set of software tools called the Distributed Computing Environment (DCE). DCE is easily portable to many different systems, UNIX and non-UNIX, and provides application developers with a means of readily allowing applications to take full advantage of all the computing resources on a network. It permits the application program to be split into components each of which can run on the best machine for that task, and yet ensures that the components work together as one application.

DCE has been released by the OSF and has been licensed by some 200 vendors, including many who are not OSF members, to make available on their systems. It was demonstrated at the AUUG'91 exhibition in Sydney in September 1991. Its expected near-universal adoption will probably make it into a kind of de facto standard.

PRODUCTS

Products may or may not comply with standards, but products are not in themselves standards.

Open Desktop (ODT)

The Santa Cruz Operation (SCO) has packaged its popular operating system SCO UNIX for Intel systems with a database, user interface, networking products and other components to produce a complete UNIX-based operating environment for Intel systems called The Open Desktop (ODT).

As a leading participant in the ACE Initiative (see above), SCO now plan to make a future version of ODT available for both of the ACE hardware platforms - Intel and MIPS-based RISC systems. That future version will use the same OSF/1 operating system kernel as used by Digital, HP, IBM, Bull and others.

As ACE becomes more prominent in the market, SCO's Open Desktop will become one of the unifying forces in the UNIX world.

OSF/1

OSF's operating system, OSF/1 was officially made available in 1990, and is now being built into many vendors' operating systems. Of course it conforms strictly to the relevant standards (POSIX, XPG). But while not differentiated on standards compliance from any of the UNIX systems, OSF/1 takes standard operating systems to a new generation of well-engineered software technology that results in greater robustness, maintainability, security and expandability.

OSF/1 is available now from Digital and Hewlett-Packard, and other commercial implementations of it are expected to become available from other vendors in the very near future.

OSF/Motif

OSF's first product OSF/Motif is already the de facto standard for the graphical user interface, being available for every workstation platform. It conforms to the X Window System standard.

SCO UNIX

The most popular version of UNIX for Intel-based small systems based on the numbers of licences, supplied by The Santa Cruz Operation. See "Open Desktop" above.

System V

The UNIX version developed and marketed by UNIX Systems Laboratories (USL). It is licensed by USL to various vendors and is an extremely version of the UNIX operating system. It now conforms to the POSIX and XPG standards. Recently, USL announced that System V would in future also conform to the OSF's Application Environment Specification (AES).

UNIX

UNIX is a trademark of UNIX Systems Laboratories (USL), a subsidiary of AT&T/NCR. USL's current version is called System V release 4 (see above).

FURTHER READING

Pamela Gray, Open Systems: a business strategy for the 1990s. New York: McGraw Hill, 1991.

TRADEMARK ACKNOWLEDGEMENTS

AIX is a trademark of International Business Machines Corporation.
HP-UX is a trademark of Hewlett-Packard Company.
MS-DOS, MS-Windows and Windows NT are trademarks of Microsoft Corporation.
OSF, OSF/1, OSF/Motif are trademarks of the Open Software Foundation.
SCO, Open Desktop and ODT are trademarks of The Santa Cruz Operation, Inc.
SunOS is a trademark of Sun Microsystems, Inc.
ULTRIX is a trademark of Digital Equipment Corporation.
UNIX, System V and SVID are trademarks of UNIX Systems Laboratories.
The X Window System and X11 are trademarks of the Massachusetts Institute of Technology.

An Update on UNIX–Related Standards Activities

Stephen R. Walli

Report Editor, USENIX Standards Watchdog Committee

The Five Great Myths of Open Systems Standards

I recently read a column where the author described computer people at cocktail parties as the doctors of the 90's. Instead of everyone wanting to discuss their aches and pains with some poor medical practitioner while they're trying to sip scotch and nibble hors d'oeuvres, computer people are plagued with the latest chat from computer literate business people.

No longer are you merely cornered by DOS know-it-alls, now you get to deal with the sweeping issues of GUI Wars, and whether UNIX will displace DOS on the desktop. Open systems are in vogue. Standards are "sexy".

With all of this comes the new "Open Systems" know-it-all. These are people who can spell POSIX, but can't pronounce it. They've all been taken to lunch recently by their favorite marketing rep from one of those lavish companies whose name is a regulation three letter acronym, let's call them TLA for short.

I started discerning certain patterns in all of this idle gossip and chatter, and now present to you the Five Great Myths of Open Systems Standards:

Myth #1

"Vendor TLA IS the standard." This is the traditional mix-up between *de jure* standards, and *de facto* standards. Or REAL standards and market share. De jure standards are built by accredited standards development bodies. There is a fair process involved to ensure that all points of view are heard. It is a consensus process, not a majority one.

De facto standards are mostly under the limited control of a single organization. They are often trademarked. If they are available at all outside of their controlling organization, the technology is often licensed. The holder of the license effectively controls where they want to take the technology. They accept input from some form of user constituency, but ultimately they run the show. I look at this as the difference between a POSIX standard interface, and a UNIX operating system.

Myth #2

"Vendor TLA is part of the standards development group, and they're donating this technology to the standard." Always a knee slapper. As if all it took to make a standard was for a vendor to donate part of its technology, obviously out of the goodness of its heart for mankind. These people have not participated in the excitement of Threads Wars, or the current painful GUI Wars.

Many vendors would love to have their specification as a standard. It gives them an instant product to sell into the hot "standards" market. They just have to get past the rest of the standards working group, made up of various backgrounds and biases.

Then comes a balloting group, a superset of the working group. These people haven't necessarily had the benefit of participating in the discussions that led to a decision. The popularity of publishing the rationale for decisions helps alleviate this problem, but not always. There will always be people in a balloting group that know their solution is the technically correct one. It's much easier to disagree with the committee, balloting a draft you didn't help make, than in the working group sessions where the talking is done face to face.

Other vendors don't *want* their technology to be a public-controlled standard. They lose control of their own specification. If they have a large market share, i.e. they're a de facto standard, they may want nothing to do with becoming a de jure standard.

Myth #3

"Vendor TLA sells a POSIX conforming system." Wrong. No one sells a "POSIX" conforming system. Indeed, POSIX conformance is the real myth here.

POSIX.3 is a standard which defines the test methodology used to measure conformance to POSIX. It has recently become a standard, IEEE 1003.3-1991. An accompanying document, still in the balloting process and therefore unstable, is POSIX.3.1. This document contains the test methods themselves for POSIX.1, (the base system interface standard), which everyone refers to as "POSIX".

By definition, POSIX.3.1 is not yet a standard, hence no POSIX.1 conformance test suite actually exists.

There is a United States government procurement profile of POSIX.1 called FIPS 151-1, or in today's open systems circus, simply "THE FIPS." FIPS 151-1 chooses certain options within the standard. It even defines certain behavior that in the standard is left as implementation defined. It was written against the original POSIX.1 standard, IEEE 1003.1-1988, not the current one, (IEEE 1003.1-1990.) In fact it was written prior to the completion of the standard.

In theory, nothing changed in POSIX.1, between 1988 and 1990, except for the reformatting to make it ISO acceptable, and "bug fixes". The removal of *cuserid()* was a "bug fix".

Because of the obvious buying power of the U.S. government, most major vendors are implementing FIPS 151-1. It is a profile or subset of POSIX.1.

Test suites exist to test conformance against FIPS 151-1. These must use the test methods described in POSIX.3.1 (still in ballot.) One of them was written to an early draft of POSIX.3.1. Another was written by using the AT&T UNIX System V Verification Suite (SVVS) as a base. SVVS dependencies are still being discovered and weeded out of this one. It is quite possible to implement something different from the FIPS, which would fail the FIPS test suites miserably, yet would technically conform to the standard. (If only there was a way to prove it.)

Myth #4

"POSIX isn't important — it's source code portability that's important." Well, no and yes. One vendor is notorious for this game.

Yes, absolutely, source code portability is what it's all about. This is typically one of the banners that's waved around in many people's definitions of open systems.

POSIX is a family of standards designed to provide source code portability. The interface was derived from the many UNIX system interfaces that existed. UNIX was/is a de facto operating system in many arenas. Many vendors are implementing the POSIX interface on their non-UNIX derivative proprietary operating systems.

No, POSIX is not UNIX. Many UNIX developers mourn and despise what has happened to the UNIX interface. They shouldn't. First of all. the base technology, which is close enough that they are already familiar with it, is becoming available on a huge installed base of technology. The demand will far outstrip the supply of technologists familiar with it. Second, nothing is preventing them from continuing on in their current preferred environment. It is different enough that they can continue developing software as they always have. It's just not as portable.

There are other software development environments which ensure software portability. VMS on a VAX architecture guarantees portability of source (and executables) across the entire line of VAX hardware. This is fine if that's where your business lays. Likewise, IBM's SAA will provide similar source portability benefits across disparate IBM architectures. They're really muddying the waters by also implementing some of the other "open system" interfaces on the SAA platforms. Again, it all depends where you, as a software developer, want to draw the portability line. POSIX is becoming the path to widest portability.

Myth #5

"Open systems technologies will revolutionize the way software is developed." Yet another silver bullet contestant. Does everyone remember the marketing hype around 4GLs? CASE? These are all good useful technologies. They simply need to be applied in their proper forum. They do not remove the responsibility of thought, i.e. creative design, careful development, and inventive testing of a problem's solution.

The current "promise" of open systems technologies has us living in a completely networked corporation of resources. Applications running where the optimal appropriate processing resource is. Information available everywhere at once, both properly protected and with its true location completely irrelevant. All of it interfaced via some wonderful intuitive graphic user interface.

I do believe this is where we're going. The technology is often commercially available already, but with some very real constraints on it. Often these constraints involve how new the technology is, and the lack of standardization.

It is a great vision, but before it's available in completely heterogeneous networked environments, the technology has to stabilize enough for standards to be created. No matter how dazzling the technology seems to be, a standard cannot be wrestled onto it too early, or it becomes a straight jacket on the creative forces shaping it.

Networked system administration at this level is in its infancy. A corporation's information and application architecture is often weighted down in a heavy history of legacy systems. (That's if the corporation can even draw its architectures!) These are a couple of the "minor" problems that need to be dealt with before marketing sells the "promise" too fully.

Conclusions

So there they are. My five favorite myths of open systems standards. I'm sure this is just the beginning. (I don't get to a lot of cocktail parties. I have small children.)

I'd love to hear other additions to this. No matter how outrageous.

POSIX.0 Guide to Open Systems Environment

Kevin Lewis <klewis@gucci.enet.dec.com> reports on the July 8–12, 1991 meeting in Santa Clara, CA:

The July meeting of POSIX.0 saw a different approach to the week's work. Instead of abiding by the draft agenda, the group trashed it and took what might be called a "fish or cut bait" approach. POSIX.0 looked at each major section and determined whether or not it was ready for mock ballot, or could be made ready by the October meeting.

Accomplishing the latter required individuals to step up to the task of editing sections during the meeting, with some degree of plenary review before the week's end. This required a commitment from the group at large to refrain from any super ethereal or journalistically-based editorial discussions. This has sometimes been hard to avoid in the past. The group stuck to its guns, however, and made a great deal of headway.

The sections within the guide that remain undecided for mock ballot are:

- networking,
- security,
- graphics (GKS, etc.),
- command user interface,
- system administration,
- fault management.

Should the group decide that a section is not ready, we will simply not include it in the mock ballot. It will be included in the formal ballot.

As it currently stands, the group plans to start the mock ballot early in November, bringing all ballot comments to the January meeting. This appears to be very feasible.

The POSIX.0 project was reviewed at this meeting by the TCOS-SS Project Management Committee. The review determined there was the need for other TCOS-SS working groups to better coordinate with and contribute to the POSIX.0 guide. This was mandated through an SEC resolution. The greatest concern among the other standards working groups is "how in the world are they going to find time to do that." The groups are already concerned about their current work loads.

I believe that once we go through the preparation at the October meeting, and get into the mock ballot, many of the loops that are still open will be closed. That is not to say that there will be no outstanding issues, but the major concerns should be laid to rest.

Report on POSIX.2: Shell and Utilities

David Rowley <david@mks.com> reports on the July 8-12, 1991 meeting in Santa Clara, CA:

Summary

POSIX.2(Shell and Utilities) Draft 11.1 closed its recirculation ballot on July 19. This draft was circulated as the 250 pages that had changed from Draft 11. Balloting a "changes-only" draft proved to be a challenge in itself. POSIX.2A (User portability extension) Draft 7 closed its recirculation ballot on August 19.

POSIX.2b has been approved after a number of recommendations from the Project Management Committee. The POSIX.2 group continued work on the new PAX archive format. Most of the time was again spent in a joint meeting with POSIX.3.2 (Test Methods for POSIX.2) creating test assertions for the document.

Background

A brief POSIX.2 project description:

- POSIX.2 is the base standard dealing with the basic shell programming language and a set of utilities required for the portability of shell scripts. It excludes most features that might be considered interactive. POSIX.2 also standardizes command-line and function interfaces related to certain POSIX.2 utilities (e.g., *popen()*, regular expressions, etc.). This part of POSIX.2, which was developed first, is sometimes known as "Dot 2 Classic."
- POSIX.2a, the User Portability Extension or UPE, is a supplement to the base standard. It standardizes commands, such as *vi*, that might not appear in shell scripts, but are important enough that users must learn them on any real system. It is essentially an interactive standard, and will eventually be an optional chapter to a future draft of the base document. This approach allows the adoption of the UPE to trail Dot 2 Classic without delaying it.

Some utilities have both interactive and non-interactive features. In such cases, the UPE defines extensions from the base POSIX.2 utility. Features used both interactively and in scripts tend to be defined in the base standard.

- POSIX.2b is a newly approved project which will cover extensions and new requests from other groups. such as utilities for the POSIX.4 (Realtime) and POSIX.6 (Security) documents.

Together, Dot 2 Classic and the UPE will make up the International Standards Organization's ISO 9945-2— the second volume of the proposed ISO three-volume POSIX standard.

POSIX.2 Status

Resolution of POSIX.2 Draft 11 ballot objections was completed, and a Draft 11.1 was recirculated. There were 900 objections received for Draft 11. The Draft 11.1 recirculation ballot closed July 19.

This draft was circulated as a 250 page "changes-only" document. This is created by printing the document and extracting all those pages containing change bars. Although this saves paper, it makes balloting extremely difficult. The context of the changes is lost. Since the page numbers (and even some section numbers) have changed since Draft 11, cross referencing old drafts doesn't help much.

The intent of this technique is to physically demonstrate the increase in consensus by the smaller size of the document. Even though balloting is made more difficult, I agree with the spirit of this approach, since most of the changes between Draft 11 and Draft 11.1 were fairly minor clarifications of the wording.

One advantage of the "changes-only" approach is that it helps to prevent balloters from commenting on those items that have not changed since the last draft. This is a restriction placed on recirculation ballots. You can't object to something you can't see!

The complete Draft 11.1 document is available from the IEEE for copying costs. Draft 11.2 is already in the works, and should appear sometime in September or October.

There have been a few requests lately to amend the POSIX.2 project's base documents list. This is a list of documents which may be referenced when discussing existing practice issues. The OSF's Application Environment Specification (AES) is one such candidate for addition.

Draft 9 of POSIX.2 is currently an ISO committee document. The ISO standards process sees a document move through three phases on its way to standardization — Committee Document, Draft International Standard, and finally International Standard. ISO has requested the U.S. Member Body to forward to them another draft once it has become more stable. Draft 11.2 has been recommended for this, when it becomes available.

Draft 11.3 should be out sometime in December. It should be complete from a technical standpoint. Hal Jespersen, the POSIX.2 Chair, reported that final IEEE approval of POSIX.2 as a full-use standard will be delayed until all ISO concerns have been addressed. This could mean postponing the IEEE POSIX.2 standard until the middle of 1992. I don't completely understand why the ISO concerns cannot be addressed now, through ISO responses to the Committee Documents sent to them. This will no doubt be discussed heavily in the months ahead.

POSIX.2a Status

Ballot resolution for POSIX.2A (UPE) Draft 6 was completed. There were only 400 objections. Draft 7 was produced and recirculated, and the ballot closed August 19. Ballot resolution is ongoing.

The list of POSIX.2a utilities is now stable. There should not be any additions or deletions. The technical content of the standard should be wrapped up in the first quarter of 1992. Draft 6 of POSIX.2 a was submitted to ISO as a proposed Committee Document/Proposed Draft Amendment (PDAM) for eventual balloting as ISO 9945–2, Amendment 1. Due to some procedural problems, it was changed to a Review and Comment draft. The next draft of POSIX.2 a will likely be Draft 8, a full draft. This will also be forwarded to the ISO, as a Proposed Draft Amendment, and will hopefully make it this time. Expect the approval of POSIX.2 a as a full-use standard anywhere from three to six months after POSIX.2.

Project Management Committee Review

Both POSIX.2 and POSIX.2 a are up for review by the Project Management Committee (PMC) in October. Each project will be examined to ensure that the work is fulfilling its mandate.

The PMC has recommended that the proposed project request (PAR) for POSIX.2B deal strictly with new utilities. The ISO timing and formatting issues originally included in the scope of POSIX.2B were thought to be unnecessary.

POSIX.2b will include utilities from the other POSIX working groups. These working groups may allocate chapters in the standard in a similar fashion to POSIX.2a. Each group retains control of its chapter. This is preferable to delegating the specification of the utilities to the existing POSIX.2 working group, which may not have the required expertise.

One question arose from this as the work of other groups is integrated into POSIX.2 should those other groups' base documents automatically be added to those of POSIX.2?

New PAX Archive Format

Work continued on the new PAX archive format. No new proposals were forthcoming, and the group continued working in its current direction. The intent is to build a new archive format on top of the ISO 1001 tape standard. The current new format specification does not draw a clear line between what is part of the ISO format, and what was added for PAX. This will be remedied in a subsequent draft.

I have reconsidered my earlier challenges to basing this new format on ISO 1001. It does have tangible benefits, and should make transferring tapes between non-traditional environments easier. The current proposal addresses both tape and non-tape based formats.

Unfortunately, the current POSIX.2 working group does not seem to have a great deal of enthusiasm for this project. Progress is slow. Unless someone champions this new format, it may well stall. Mark Brown (IBM) has volunteered to flesh out the current draft for distribution in the next POSIX.2 mailing.

Test Plans and Assertions

A test plan for POSIX.2 and POSIX.2 a was written, and submitted to POSIX.3.2 (Test Assertions for POSIX.2) for review. Lowell Johnson, POSIX.3.2 Chair, expressed some concerns over the linkage of the POSIX.2 and the POSIX.2 a test plans. It is important that each test plan cover the scope of one and only one project.

Tuesday to Friday were spent writing test assertions in a joint meeting between POSIX.2 and POSIX.3.2 . Confusion continues to reign when writing assertions. There are many different assertion styles, and it seems to be more art than science. Styles range from "you know what I mean", to precise, verbose, legalese. The group requested that the Chair (Lowell Johnson) and the Technical Editor (Andrew Twigger) produce a style guide for POSIX.3.2 assertions. The guide would be reviewed at the beginning of each joint meeting. This should greatly help the consistency of the assertions being produced.

Draft 5 of POSIX.3.2 is now 400 pages, and most of the POSIX.2 commands have assertions. The group is still intending to mock ballot the document after the October meeting. A few utilities are noticeably absent: **awk, lex,** and **yacc.** I'm sure donations of good assertions for these utilities would be most welcome.

The turnout for the joint meetings was disappointing. Writing test assertions is time consuming hard work. Ideally the joint meeting time should be spent *reviewing* assertions, and clarifying the implied interpretations of the standard. Unfortunately, it is difficult for members to find the time between meetings to write assertions.

Writing test assertions for POSIX.2A will likely start in January 1992. If you thought test assertions for make were difficult, wait until you try vi!

Report on POSIX.3: POSIX Test Methods and Conformance

Andrew Twigger <att@root.co.uk> reports on the July 8–12, 1991 meeting in Santa Clara, CA:

In many ways the Santa Clara meeting could be considered to be one of the less eventful of the recent POSIX.3 meetings.

Draft 5 of the POSIX.3.2 document was distributed at the meeting, with the majority of the test assertions having been aligned with the text of POSIX.2 (Shell and Utilities) Draft 11. This alignment was exquisitely timed to coincide with the production of Draft 11.1 of POSIX.2, immediately rendering parts of Draft 5 out of date! Perhaps the documents can be synchronized for the next meeting, or the one after that, or . . . .

The majority of the POSIX.3 working group spent most of the meeting writing assertions with POSIX.2. Having already dealt with the "simpler" utilities, some of the more complex utilities (**make, pax, ls**) were tackled during the week. The next draft should contain assertions for about 95% of the utilities, however the remaining 5% could take 95% of the time!

The ballot review group for POSIX.3.1 met briefly during the week to look over the objections received during the last ballot recirculation. Most of these had been resolved prior to the meeting and it was expected that the remaining items could be resolved by the end of August. Another brief recirculation ballot is expected in the Autumn, with possibly another standard being completed by the end of the year.

The Steering Committee for Conformance Testing (SCCT) met twice during the week and finally approved some of the test method development plans submitted by the other working groups. The rumour that this was only in response to moans from the other working groups that the SCCT had rejected every plan submitted in the previous nine months is not entirely without foundation! Most of the other working groups, however, are getting geared up to produce test methods with their documents.

Several members of POSIX.3 spent time in assisting other working groups to develop test methods for their standards. Much of this time was spent in helping the working groups to understand how significant a task this is and in helping the working groups to develop a reasonable strategy for test methods. Some time was also spent in reviewing the work that had already been done by work group members. There seems to be an increased awareness of the problems and an ever improving quality to the test methods that the working group are producing.

Report on POSIX.4, POSIX4a, POSIX4b, POSIX.13: Realtime POSIX

Bill O. Gallmeister <bog@lynx.com> reports on the July 8–12, 1991 meeting in Santa Clara, CA:

Summary

The working group continued work on Application Profiles, on the extended POSIX.4B Realtime Proposals, and on the thorny issues of IPC and synchronization mechanisms. Since both POSIX.4 and POSIX.4A are preparing for another ballot recirculation, there was little work done on these drafts.

Real-time Application Profiles

POSIX.4 has produced four different profiles, matching different scales of real-time endeavor. The Embedded profile is meant for small machines that may lack hardware for paging, disks, and terminals. As such, this profile is rather different than what is generally considered to be a UNIX® system. In particular, the threads work is called out, and some of the POSIX.1 file system, but *fork()* is not needed.

This requires subsets of POSIX.1. Its multi-process aspects and a lot of the extended file-system semantics are considered optional by the people working on the smaller Real-Time profiles. This subsetting work has to be sanctioned by POSIX.1. Getting them to agree to this work may be an interesting task.

Other profiles under development are a "Controller" profile, an "Avionics" profile, and the "Kitchen Sink" profile. The Kitchen Sink and the Embedded profiles define two endpoints of a spectrum of real-time practice. The Controller and Avionics profiles define particular points of practice within that spectrum. The Avionics profile reflects the current requirements of the Avionics industry. The Controller profile is a step up from the Embedded profile.

IPC Again

POSIX.4 inter-process communication (IPC) remains an issue. We had a liaison meeting with the POSIX.12 (Protocol Independent Interfaces) working group and presented our requirements for a Real-Time sockets mechanism. There were 28 possible requirements; we decided that 17 of these requirements were truly necessary for a socket-based mechanism for Real-Time IPC. The POSIX.12 group helped us refine these requirements into something they can use in defining a mechanism. These discussions will undoubtedly carry on for some time.

Meanwhile, the existing POSIX.4 IPC chapter is undergoing radical surgery. The recirculation draft that should come out this October should feature an IPC mechanism that more closely resembles the message-passing interfaces of small real-time kernels. The interaction of this message-passing mechanism and the future POSIX.12 real-time sockets mechanism is an open issue.

Synchronization Again

At the last meeting, it was the POSIX.4 proposal that needed guidance from the working group on its binary semaphores chapter. This meeting, the POSIX.4A proposal required guidance with regards to mutexes. (Mutexes are simple MUTually EXclusive locks.) Specifically, the priority ceiling protocols in the current draft ran into serious balloting problems. In response to this, a simplified version of the priority ceiling protocol, called Priority Ceiling Protocol Emulation, was proposed to replace the existing two mechanisms currently in POSIX.4A. The emulation protocol is much easier to understand, offers the same worst-case blocking behavior as the earlier proposals (although worse average-case behavior), and works with multiprocessor systems. The working group was torn whether any priority ceiling protocol should be in POSIX.4A at all. Assuming that one would be present, the group clearly preferred the emulation protocol.

The debates on priority ceiling featured a lively exchange between POSIX.4 and POSIX.14 (Multiprocessor Profile). This is the closest that POSIX.4 has come to its old glory days of large bloody group battles.

POSIX.4b

Some work was done on the timeout extensions of POSIX.4B. This work involves providing timeouts to all POSIX.4 calls that may block. An early draft of this proposal is available in the latest POSIX.4 mailing.

Future Drafts

The technical reviewers for POSIX.4 and POSIX.4A have been working hard towards new drafts of each of these documents. It is our current plan to recirculate them both at about the same time as the Fall meeting. If this happens, the next meeting will again focus on application profiles and continuing POSIX.4B.

Report on POSIX.6: POSIX Security Extensions

Ana Maria De Alvaré <anamaria@sgi.com> reports on the July 8–12, 1991 meeting in Santa Clara, CA:

Hello USENIX members!

This time my report will be very brief. It is brief because there were no big disagreements at the meeting, and because the whole week was spent in cleaning up the document for formal ballot.

This was the last meeting working in functional subgroups, addressing discretionary and mandatory access controls (DAC and MAC), audit, and privileges. At the next meeting the group will be divided into people helping with the balloting process, doing test assertions, and identifying areas that POSIX.6 has not covered. The ballot document should come out sometime after the September mailing (September 10, 1991).

POSIX.6 spent the whole week addressing all the mock ballot comments and objections. A small group of three people, including myself, began working on the first draft of the POSIX.6 test methods. The test methods draft will be brought to the next meeting and people from the disbanded subgroups will begin creating test methods for the functions defined in the POSIX.6 document. It will be a long week!

So what areas aren't covered in the current POSIX.6 draft? The three major areas that I know are not covered are:

- authentication,
- security system administration, and
- network security.

There are items in the subgroups which are also not addressed. A portable audit format has not been fully defined, and so is not going out for ballot. With mandatory access controls, we decided at this meeting to not enforce privileges on an implementation of multi-level directories. Except for some clean-up in Draft 11, discretionary access controls remain the same.

The data type issue still remains across the DAC, MAC, audit, and privileges subgroups. To interoperate between systems, opaque objects need to be stored and retrieved without concern for the implementation defined formats. An opaque object model also provides consistency across the interfaces. POSIX.6 subgroups have defined a number of security related objects. We cannot agree on a way to represent these, but have determined four possibilities:

- A Type 1 object is opaque, and is only valid for use by the process which gets the data, and only for the lifetime of the process.
- A Type 2 object is still opaque, but it must be self-contained and persistent.
- A Type 3 object is a text string with an undetermined format. MAC labels are represented as Type 3 data types.
- A Type 4 object is a text string with a defined format. Access Control Lists (ACLs) have a Type 4 representation.

One compromise was that the subgroups would define conversion routines for Type 2 and 3 data, which would return an opaque object and the length in bytes of the object.

We were still unable to agree upon a uniform type representation across the four subgroups in the July meeting. This issue will likely be a hot one in the balloted document. We will have to wait and see what the ballot brings to resolve this.

Well, that's all folks! Keep an eye out for the POSIX.6 ballot.

Report on POSIX.12: Protocol Independent Interfaces

Tim Kirby <trk@cray.com> reports on the July 8–12, 1991 meeting in Santa Clara, CA:

POSIX.12 is developing a set of protocol independent networking interfaces. There were no major changes in the group's direction this meeting. Two interfaces are proposed in language independent form — the simple network interface (SNI) and the detailed network interface (DNI). SNI is a proposal drawn from several sources, with no existing (de-facto) standard. DNI, however, is seen as a single language independent specification to which there are two valid C language bindings, one for BSD-style sockets and one for the X/Open Transport Interface (XTI).

The group once again reviewed the proposed changes to XTI option management from Gerhard Kieselman, following input from X/Open during the intervening three months.

A significant amount of time was spent in liaison with the Transaction Processing (POSIX.11) and Real-Time (POSIX.4) working groups as a result of the proposal from the last meeting to include their requirements in the POSIX.12 interface. POSIX.4 requirements are a direct result of ballot objections to the IPC interface proposed in their current draft. Given the announced intention of POSIX.4 to include a "simple" IPC interface regardless of the POSIX.12 efforts, there is some concern within our group that there are no advantages to the proposed POSIX.12 changes.

Work between the meetings continues on language independent versions of the interfaces, and the test methods without which the document may not become a standard.

A review of the test method requirements revealed a significantly larger amount of work than had originally been anticipated. This has resulted in a change to the test method schedule. The mock ballot of the POSIX.12 draft is not expected now before the second quarter of 1992, and the first real ballot not before the fourth quarter of 1992.

Report on POSIX.17 - Directory Services API

Mark Hazzard <markh@rsvl.unisys.com> reports on the July 8–12, 1991 meeting in Santa Clara, CA:

Summary

POSIX.17 made significant progress towards completing another draft in Santa Clara. The group is on track to mock ballot Draft 2.0 of the Directory Services API by the end of August. Key areas of progress were:

- test methods,
- language independence specification (LIS),
- Model text in Section 3,
- ds_gethostbyname() example,
- preparation for mock ballot.

Introduction

The POSIX.17 group is generating a user to directory API, e.g. an API to an X.500 Directory User Agent (DUA). We are using XAPIA — X/Open's XDS specification as a basis for work. The X/Open Directory Services API (XDS) is an object-oriented interface and requires a companion specification, X/Open's Object Management API (XOM), for managing the OSI objects as they pass through the directory API.

XOM is a stand-alone specification with general applicability beyond the directory services API. It will be used by IEEE 1224.1 (X.400 API) and possibly other POSIX groups. It is being standardized by IEEE 1224.

Status

Commitment within the group remains strong, with all Chicago attendees returning to Santa Clara, and completing homework assignments. We are committed to mock balloting our document between meeting cycles and have planned a special mailing for the end of August, (paid for by X/Open - Thanks!).

Once again, considerable time was spent examining POSIX.12 (Protocol Independent Interfaces) requirements for directory services. One of the requirements is a mechanism to protect existing applications from changes in how directory services are offered. We had decided that this was technically beyond the scope of our work, but that we would address this by providing a non-normative annex with coding examples, showing how it could be done.

The first example is a new function, *dsgethostbyname()*, which could be added to the existing practice API (BSD's *gethostbyname()* function). With it (or something similar) existing applications wouldn't need to be modified to work in a POSIX environment.

Another POSIX.12 requirement was that the underlying directory service provider be able to interoperate/co-exist with existing practice directory services (e.g. the Internet DNS). On the surface, impact to the API itself is minimal, requiring (at most) the use of an existing parameter which would allow the application to specify which (of many) services it wanted to use.

POSIX.17 and P1224 (XOM API) met in joint session to review the object management specification. Many corrections were made, and a new draft will be released in the first half of August (in time for our Mock ballot).

Mock Ballot

There were many homework assignments this time to get the mock ballot out between meetings. Significant progress was made towards producing a draft suitable for mock ballot. The technical editor completed his assignment to provide 25% of the LIS text. An estimated 25% of the test assertions were completed as well. Our plan is to go to mock ballot with this level of completeness in order to obtain feedback before we proceed further.

We plan to send it out before the end of August, so we'll be able to process the feedback at our next meeting. Hopefully, we'll get feedback on our LIS and test assertion work. The comments will help us determine our future direction and better estimate our completion date.

In Closing

The group made good solid progress in Santa Clara readying the document for mock ballot. We seem to uncover more requirements with each meeting but somehow we're managing to move forward. POSIX.17 will be mock balloted incomplete, needing more work on LIS, test methods and a few more examples.

The group will meet in October to process the input from our mock ballot, continue working on LIS and test methods, and determine where we go from there. As usual, there's a lot of work to do.

Report on P1224: X.400 API

Steve Trus <trus@osi.ncsl.nist.gov> reports on the July 8–12, 1991 meeting in Santa Clara, CA:

Summary

P1224 is producing two documents for standardization — the X.400 API and the X/Open Object Management API (XOM). At Santa Clara, the group continued work on the modifications required to the base documents. Specifically the group:

- reviewed the first draft of the Object Management API,
- worked on test methods,
- worked on IEEE balloting plans for P1224.

The IEEE Standards Board has approved the Project Authorization Requests (PARs) for P1224 (OSI Object Management API) and P1224.1 (X.400 API).

Report

The Santa Clara meeting was generally productive for the P1224 working group, and we are well under way to producing our draft documents for standardization. Progress wasn't what it could have been due to minimal participation by the X.400 API Association.

Review of Object Management Draft

The first draft of the Object Management API was distributed to the P1224 working group before the meeting. The group spent much of the week reviewing the API. The POSIX.17 (Directory Services) group joined the review for one day. Numerous changes were made to the document. When the changes have been incorporated into the document, it will again be sent out in the P1224 mailing.

Test Methods

Tony Cincotta, a test methods expert from NIST, spent much of the week reviewing the Object Management draft and test methods already developed. Tony provided many suggestions for improving the test methods developed thus far.

It has become apparent that the development effort required to build test methods for the X.400 and Object Management APIs could delay the completion of balloting of the APIs for years. To resolve this problem X/Open is considering funding a contractor to develop the test methods for these documents. This issue should be resolved by the next meeting. Additionally, Tony recommended that he train the X/Open contractor for the development of the test methods section of the documents.

Balloting Plans

Our current plans are to ballot the Object Management API after the October meeting, and to ballot the X.400 API after the January meeting. These ballots would not include the test methods; balloting cannot complete until the test methods are complete.

Currently we are developing the list of people who will be invited to ballot these documents. This list includes members of the IEEE TCOS, the X.400 API Association, and X/Open Limited. Invitations to join the balloting group will be sent out at the end of August by the IEEE. To be included in the balloting group, a person must return the invitation to the IEEE by October 10, 1991.

Iain Devine, the P1224 technical editor, will be the ballot resolution reviewer, assisted in technical matters by Enzo Signore.

In Closing

P1224 continues to make good progress. The primary focus of the Parsippany meeting will be to continue the review of our draft documents, work on our test methods, and prepare for balloting.

Report on ANSI and the X3 Committees

John Hill <hill@prc.unisys.com> discusses ANSI and the X3 committee:

Over the past few years information technology standards have become more important in the industry. One of the most prevalent areas for standardization is operating systems and allied services. This article discusses the largest formal standards development organization for information technology in the USA, X3, and its relationship to ANSI.

A brief background is useful in understanding how the USA develops standards. The premier standards body is the American National Standards Institute (ANSI). Its four main functions that are of interest here are:

- membership in the International Standards Organization (ISO) and the International Electrotechnical Commission (IEC), the worldwide standards bodies,
- accreditation of organizations to develop voluntary U.S. standards,
- publication of U.S. national standards,
- oversight of the standards development process to ensure due process and fairness.

That's right, ANSI does not develop standards; ANSI accredits other organizations to develop standards.

Standards are developed in one of three ways:

- by professional and trade organizations (e.g. Institute of Electrical and Electronic Engineers, IEEE, an organization which is involved in more than the development of standards),
- by accredited committee (X3),
- by canvass (e.g. Ada Joint Program Office, AJPO). The canvass method is intended for mature and non-controversial standards processing.

In a nutshell, ANSI accredits the standards development procedures of individual groups within some designated, but very broad scope, and according to one of the three ways. X3, while not a part of ANSI itself, is an example of a committee that ANSI has accredited to develop U.S. national standards.

X3 was established in 1961. Since its inception, the Computer and Business Equipment Manufacturers Association (CBEMA) has functioned as the secretariat. X3 has about 850 projects, 200 standards, and some 3000 volunteers.

The purpose of X3 is voluntary standardization in the areas of computers, information processing, and peripheral equipment and devices. This scope of activity includes standardization of subsystems in order to provide for hardware interoperability and software portability. As a result, many of the fundamental standards of the computer industry have been developed by X3.

The true, nuts-and-bolts work of X3 is done by its subgroups. There are two types of subgroups, advisory and technical. The three advisory committees are collectively empowered to ensure that the process of standards development is under control. These advisory committees are chartered to

- advise the secretariat in administrative activities,
- develop the X3 strategic plan,
- manage the technical activities of X3.

The forty technical committees of X3 actually develop the standards. There are committees for:

- media (both magnetic and optical),
- operating system services (database and graphics),
- programming languages (Fortran, C, COBOL),
- codes and character sets,
- vocabulary,
- data communications (OSI),
- systems technology (SCSI, security),
- and office systems (ODA/ODIF).

These technical committees frequently act as the U.S. technical advisory groups (TAGs) for the development of worldwide standards.

Worldwide standards are a major area of activity for X3. Over the past ten years, the X3 member organizations have expended more resources for development of the base OSI standards than on any other single functional area of standardization. These 175 largely anticipatory standards were developed for the most part in the international arena. The US delegates from the X3 technical committees actively worked in SC21, the ISO/IEC subcommittee responsible for the OSI standards, to ensure that US interests were met in the worldwide standards being developed.

Membership in X3 is open to anybody affected by its standards. Its current members, of about 41, include some of the most notable producers, users, and general interest groups involved in the information technology industry. Members have to participate in order to remain in good standing.

All members pay annual service fees to support X3 activities. The larger members pay more than the smaller. An additional fee is charged for participating in the subgroups.

If you have further questions concerning X3, you should contact the X3 secretariat (202-737-8888). These helpful people can send you several standing documents which expand upon this discussion.

Report on X3J16: C++

Mike Vilot <mjv@objects.mv.com> reports on the July 8-12, 1991 meeting in Santa Clara, CA:

Current Status

The ANSI X3J16 committee took a major step towards internationalization at its June meeting. This was the first joint meeting between X3J16 and ISO WG21. WG21 is the ISO C++ working group. X3J16 and WG21 are roughly peer organizations.

June meeting

The Lund Institute of Technology hosted the meeting in Sweden. The week's major activities focused on understanding the myriad details of producing a single, international C++ language standard.

The X3J16's sub-groups focus was on the key topics listed in the goals statement developed at the March, 1990 meeting. They worked by electronic mail between meetings, and reported their progress.

International Concerns

Steve Carter, of Bellcore, presented the major international concerns:

International cooperation is an explicit goal of X3J16, and the committee devoted the entire first day's discussion to international concerns. Most members want to avoid the difficulties encountered during the development of the C language standard.

Much of the work focused on attaining a smooth coordination with WG21 without losing technical effectiveness. The committee agreed to continue emphasizing informal discussions and informal "straw votes" before making formal decisions. All members of WG21 will be added to the email lists, and will receive X3J16 paper mailings.

On the other side, WG21 voted to hold its technical meetings jointly with X3J16. They appointed Jonathan Shopiro as their editor, which means both committees have the same editor.

X3J16 decided to conduct a letter ballot on the question of converting to a Type "I" process. This means developing an international standard, rather than developing a domestic standard followed by an international standard (as was the case for the C language). A straw vote indicated most members would vote in favor of the change.

The committee dissolved the International

Concerns working group, since it has served its purpose. Steve Carter, serving as Convener of WG21, will continue to address international C++ concerns.

Editorial

Jonathan Shopiro, of AT&T, presented the Editorial group's work:

The editorial change that simplified the treatment of names and name lookup, merging the concepts that had previously been treated under the topics of dominance and name hiding, remained in the document.

Much of the recent work on the document has been in clarifying or defining basic terms. For example, the basic unit of storage is a byte. In the C standard, it is a character, which confuses the notion of what type "char" is supposed to represent, especially in light of 8-bit and larger character sets. The process of resolving the definitions of the two base documents continues. (These are the Annotated C++ Reference Manual and the C standard.)

One minor change to the document format: the size is now suitable for A4 paper.

Formal Syntax

Reg Charney, of Program Conversions, presented the work of the Formal Syntax group:

The bulk of the discussion concerned the change that renamed most of the non-terminals in the grammar. There are still more proposed changes.

The conflict between the virtues of regularizing the naming versus the evils of gratuitous changes resurfaced. Bjarne Stroustrup made the strongest criticism, observing that the changes had been proposed and adopted without sufficient principles. He noted that the lack of such principles invited the kind of "random changes" that were presented at the June meeting. He also observed that the changes had not even been checked against the C standard's grammar.

Core Language

Andy Koenig, of AT&T, presented the Core Language group's work:

Most of the Core Language discussion centered on name resolution issues. These issues are highlighted by the interactions of nested classes, inline friend function definitions, and static class members.

This work has helped identify ambiguities in the present wording. Although there has been progress, open issues remain. For example, defining a friend function in a class causes the name of the friend to be made available in an "enclosing" scope. The cases involving nested and local classes still have to be resolved.

Environment

John Wilkinson, of Silicon Graphics, presented the work of the Environment group:

Discussion continued on static initialization order for objects in different translation units. The group proposed two new rules intended to provide correct initialization that still accommodated dynamic linking:

- Nonlocal static objects defined in a translation unit must be initialized in the order that the definitions appear in the translation unit.
- The nonlocal static objects defined in a translation unit must be initialized before any object or function defined in that unit is used by any other translation unit; the nonlocal objects defined in the translation unit containing main() must be initialized before control enters main().

Specifying translation limits in the standard was discussed, but seemed to generate more heat than light, and nothing was decided.

Libraries

Jerry Schwarz, of Lucid, presented the Library group's work:

There is an evolving proposal for a standard string class, and its interaction with internationalization concerns. The tradeoff involves generality (strings of both single- and multi-byte characters) versus efficient implementation. This discussion continues.

The group also worked on issues of conformance, and describing the options available to implementations that choose to extend the standard library. For example, the implementation may provide the standard classes by deriving them from base classes not mentioned in the standard, or as instances of templates not mentioned in the standard. As another example, an implementation may add members to a class definition, with the constraint that private virtual functions must be in the implementation name space.

Work also progressed on standard exceptions. One line of investigation is to use exceptions to clarify those aspects of the language that are vague or "undefined." For example, the default new_handler could throw a storage_error exception.

Language Extensions

Bjarne Stroustrup, of AT&T, presented the work of the Extensions group:

The group proposed a change that adds digraphs and new keywords as synonyms for certain characters. For example, '{' can be written as '<%', '&=' as 'and_eq', and so on. This allows expression of C++ programs in character sets that do not include certain of the ASCII characters. It is a proposal Bjarne has been working on with Keld Simonsen for over a year, and their work has been coordinated with the ISO WG14 (C language). The committee adopted this proposal.

The group is working through a long list of proposals for changes to the language. Some of the items are:

- adding 8-bit (i.e. international) characters in identifiers;
- allowing virtual functions in a derived class to use a more specific return type than the base class' version of the function;
- allowing overloading of operator . (dot);
- a name space control mechanism.

The largest issue currently lurking in the Extensions category is the addition of support for run-time type information. There will be much discussion on this topic over the next months.

C Compatibility

Tom Plum, of Plum-Hall, presented the work of the C Compatibility group:

The group continued its investigation of the vocabulary differences between C and C++. Only a few of the differences have been resolved, and Tom plans to meet with Jon Shapiro to decide which terms can be incorporated as C++ definitions.

Forthcoming events

The next three x3J16 1991 meetings (and their hosts) will be:

- November 11–15, Austin TX (TI)
- March 9–13 (or 16–20) 1992, London, UK (BSI and Zortech)
- July 13–17 Toronto Canada (IBM)

Membership on an X3 committee is open to any individual or organization with expertise and material interest in the topic addressed by the committee. The cost for voting or observer membership is $250. Contact the chair or vice chair for details.

Chair: Dmitry Lenkov
HP California Language Lab
19447 Pruneridge Avenue MS 47 LE
Cupertino, CA 95014
(408)447–5279
FAX (408)447–4924
email: dmitry%hpda@hplabs.hp.com

Vice Chair: William M. Miller
Glockenspiel, Ltd
P.O. Box 366
Sudbury, MA 01776–0003
(508)443–5779
email: wmm@world.std.com

Report on ANSI X3B11.1: WORM File Systems

Andrew Hume <andrew@research.att.com> reports on the July 8–12, 1991 meeting in Murray Hill, NJ:

Introduction

X3B11.1 is working on a standard for file interchange on random access, write-once media: a portable file system for WORMs. First let me apologize for tardy snitching (again); my excuse this time is that I am now technical editor of the working paper. I shall describe the results of the last two meetings, April (North Falmouth, RI) and July (Colorado Springs, CO), as a summary of where we are now. In brief, the current draft seems fairly stable and we expect to conduct a letter ballot after the next (October) meeting. There is also considerable international activity. I also discuss our method of electronically distributing our drafts.

International Activity

I am still a novice at international standards stuff so take the following with a larger than normal grain of salt.

The appropriate ISO committee, SC15, has been reconstituted and had its first meeting in several years in July in Tokyo. The meeting was mostly administrative in nature but there was a proposal for a volume structure standard submitted by Fujitsu. The motivation for this is that Fujitsu intends to introduce 3.5in media and drives that can be partially embossed (like CD-ROM) and partially WORM or rewriteable. Understandably, they would like to have a standard volume labelling scheme to enhance interchange. They figure that file system layout standards can come along later but we need the volume labelling scheme real soon now.

A common way for an ISO committee to do work is invite some recognized, accredited committee to submit a proposal and then vote on it. In particular, some committee with relatively little administrative procedure. This means ECMA, (European Computer Manufacturers Association,) and not ANSI.

Luckily, the relevant ECMA committee, TC15, has just been reconstituted with its next meeting in Geneva in early September. Ostensibly, TC15's first job is to consider and bless the work of the Frankfurt group, which has been working on an extension of ISO 9660 (CD-ROM) to handle CD-WO media. The very next thing will probably be a volume structure and file system standard, based on our (X3B11.1) work. (This has required significant changes to our working paper but more on that below.)

There is also a dark side to TC15. ECMA apparently has a hidden agenda for TC15 that includes the development of a general storage architecture. This is the storage equivalent of the OSI networking model with its 7 layers. The first guess at the layers are:

- physical layer,
- recording layer,
- formatting layer,
- volume structure layer,
- object management/file system layer (e.g. ISO 9660),
- file structure layer (e.g. ISAM),
- application layer.

Why does the international activity matter? (That this question needs to be raised is comment enough for our industry.) The goal of the standards game is to have a technically sound standard adopted as soon as possible. Assume for now that the X3B11.1 draft is technically sound. How do we get a standard? One way is to go through ANSI procedure, like the C standard did. Assuming no problems, hitches, objections and foulups, we could have an ANSI standard within two years. And then we would have to work within the ECMA/ISO committees to ensure that they adopt a technically equivalent standard (and thus avoid the prospect of an ANSI standard that conflicts with an ISO standard). The other way is to work within the ECMA committee and produce an ECMA standard which then, given the heavy European presence in ISO, would fairly automatically become an ISO standard. Astonishingly enough, it seems likely that we could get an ISO standard 6–9 months *sooner* than we could get an ANSI standard! (Yes, sadly, this means that often the quickest way to get an ANSI standard is to do the ISO standard via ECMA and have ANSI adopt the ISO standard.)

The Current Draft Working Paper

In order to facilitate adoption of the working paper by TC15, we have made several structural changes to the working paper. It is now in four parts. The first part is introductory. It specifies the scope and defines terms. The second part describes a volume labelling scheme. It specifies how volumes are labelled, how partitions are defined, how volumes are grouped into volume sets and a bad sector replacement scheme. The third part describes a file system layout that is independent of the details of part two. The fourth part is a short section detailing the (very few) changes needed to make part three suitable for rewriteable media. This restructuring was a significant labour, although it involved negligible technical change.

Volume/Partition Structure

This part is probably the most changed since my last report. It has become much simplified and made independent of the file system specification. It handles space allocation for the volume, recording of volume and partition descriptors, definitions of partitions, and bad-block mapping. Provision has been made for specifying the type of file system in a partition. Some of these will be predefined, such as ISO 9660 and 9223. Others can be registered, such as proprietary formats like SGI's EFS file system.

File System

This has been fairly stable although many details have been tweaked. The space management within a partition and integrity controls (essentially the dirty bit for a partition) have been moved out of the volume description and into the file system description as it was deemed too complicated to demand that everyone support it.

Technical Editing

Because of the previous technical editor's health problems, I was appointed technical editor. This has been quite entertaining for me; I have never been involved in such a complicated document production (and all of it my own doing!). A single processing pass produces a table of contents, an index, automatically generated data structure layouts, ANSI C declarations of all the data structures, an ANSI C program that tests that the declarations are correct on your system (the fields have the correct offsets and sizes), and last but not least, the *troff* output.

Each pass runs at least 8 *awks*, 4 *sorts*, 10 *seds*, 2 *uniq, spell, tbl, eqn, troff* and Kernighan and Van Wyck's balancing page makeup backend. It takes 6 minutes clock time on an 80 mips SGI multiprocessor. (This may not be a game for PDP-11s anymore but at least I know what to do with all those cycles!) We iterate until nothing changed since the last pass.

A more noteworthy accomplishment (other than writing more *awk* scripts than you can point an editor at) is that the current draft of the working paper is available online by both *ftp* and email (*netlib*). You can get either of two forms: PostScript and the *troff* input (minus all the formatting directives). This way you can print your standard and *grep* it too. The files are `x3b11.1-wp.ps` and `x3b11.1-wp.text` and are in the directory research/memo. For *ftp*, login as *netlib* on `research.att.com`.

Finale

What can, or should, you do? Well, the worst case is that a standard based closely on the current draft will become an ISO standard for interchange for all random access disk drives (optical and magnetic). You not only would have to support it; you may also have to boot off such a disk.

If you wish to comment on the draft, the best time would probably be in early November during the letter ballot. (You of course can't be in the letter ballot because you aren't a member, but you could give your comments to me.)

If you would like more details on X3B11.1's work, you should contact either me (`andrew@research.att.com`, 908-582-6262) or the committee chair, Ed Beshore. I think the two most useful documents are the current draft of the working paper (about 60 pages) and a programmer's guide to the draft (about 12 pages written by me). I will send you copies of the latter document; requests for other documents or more general inquiries about X3B11.1's work would be best sent to Ed Beshore.

The next meeting is in Merrimack, NH on October 21 – 25, 1991. Anyone interested in attending should contact either me or Ed Beshore (`edb@hpgrla.hp.com`).

An Update on UNIX–Related Standards Activities

Stephen R. Walli

Report Editor. USENIX Standards Watchdog Committee

P1202.1: Windowing Toolkit API

Luisa Johnson. Harris Space Systems, reports on the July 8–12, 1991 meeting in Santa Clara, CA:

The P1201.1 Working Group's attendance was significantly lower than that of previous P1201.1 quarterly working group sessions. Most participants expected much controversy due to the recent selection of the base and reference documents at the Boulder meeting in May. Fortunately, the participants that did show up for the week long meetings were more eager to start drafting the standards document than arguing over document selection. After all, these documents are only informational sources to be used in the drafting of the standard and the layered API standards document itself will most likely not resemble either of them.

The working group was fairly well represented by both layered API developers and current or future users. With the exception of the first morning session, no representatives from any major toolkit or hardware vendor actively participated in the P1201.1 sessions the rest of the week.

The working group spent the first morning discussing the usual administrative items and identifying a strategy to be used for the rest of week in order to generate the first standard draft document. The strategy consisted of:

- reviewing the strawman document outline,
- identifying areas to be deferred or to be omitted from the standard,
- identifying and describing a basic list of objects including their attributes.

As a result of the document outline review process, a few minor modifications and additions were made to the General section, the Terminology and General Requirements section, and appendices were identified by the group. Topics covered by the group included:

- internationalization concerns,
- geometry management and anchoring,
- color,
- cursors.

Rationale will be added regarding the basic requirements list, language bindings, and the process that was used to select the base and reference documents.

The next area tackled by the group was that of identifying areas to defer for future meeting discussions or topics to omit from the standard. If the area had been or is currently being addressed by other working or standards groups, then it was considered out of our standard's scope. Areas such as drawing, resource formats, and resource languages were identified as possible areas to expand on once the initial first draft is completed.

WNDX Corporation made a presentation to the working group. Their product allows developers to specify a look and feel that may be different from the underlying GUI's "look-and-feel ." It is implemented to the native library and emulates the style guide, so a developer could select the MacIntosh "look-and-feel" on a UNIX Windows environment. WNDX Corporation representatives informed the group of their desire to participate in this standards effort and the working group agreed that the standards effort could only benefit with the inclusion of new approaches and their lessons learned.

From the second day through the end of the week, P1201.1 worked diligently on the identification of objects and attributes. This became an iterative process by which the first pass was a simple candidate list of objects which became further defined each day. Attributes were assigned and refined throughout the week. No effort was devoted to the specific syntax and semantics to be utilized. Instead, for each object, pointers to both the Base and Reference documents were annotated for further details. By the end of the week, a robust set of objects and attributes had been identified and the working group members felt a sense of accomplishment which none had anticipated. Working group members felt that this had been one of the most fruitful meetings in their turbulent history. The next mailing will include the approved first draft.

With the lack of participation by any major GUI vendor, one can only wonder if the accomplishments achieved during this week could have been obtained had they not been so busy fighting the GUI PAR wars.

POSIX.7: System Administration

Martin Kirk <m.kirk@xopen.co.uk> reports on the July 8–12, 1991 meeting in Santa Clara, CA:

The July meeting of the POSIX.7 (System Administration) working group continued the new direction established over the previous two meetings.

Small groups continued work on Printer Management, Software Management, and further refinement of the "Big Sticky Issues," i.e. the global context of these activities.

The most important results from the "Sticky Issues" small group were recommendations for the style and content of POSIX.7 standards. Their final recommendations will bring the group into full alignment with the rest of POSIX. The overall structure for each functional area standard will have sections for:

- POSIX.1 style programmatic interfaces based on existing practice
- POSIX.2 style command line interfaces based on existing practice
- Managed object definitions to provide a basis for the distributed system administration functionality [Ed. — It is appropriate to mention that these have no relationship to the communications object types to be managed with the object management API being defined by P1224 and POSIX.17.]

This approach represents a compromise between "traditional" systems administration and the object-oriented approach. Where there are existing interfaces available they will be used. They will be supplemented by managed object definitions needed to provide uniform interoperability between different implementations.

Adopting this approach, along with the earlier decision to build separate functional area standards instead of a monolithic tome, should enable the group to progress more swiftly.

The Print Management group has been pursuing an approach based on the MIT Palladium distributed printing system. They received a strong contribution from the UNIX System Lab (USL) championing the System V lp print system. This was a timely interjection, allowing us the opportunity to address the issues that would have undoubtedly arisen during the balloting process. By identifying both the common subset and the differences, it should be possible to provide the appropriate rationale for the contents of the eventual standard.

The Software Management group continued to make good progress. They are working with contributions from several sources, including AT&T, DEC, HP, Siemens-Nixdorf, and SCO. (My apologies to anyone I left out.) As one would expect, all these differing systems are remarkably similar in terms of the functionality they present to the user, and thus the group found it relatively painless to identify the large common subset between them.

The group's other activity was to identify a third functional area in which to commence work. The chosen candidate was User Management as it was felt that many other system resources were managed in terms of their relationship to users.

By the time the next POSIX meeting takes place in October, the OSF Distributed Management Environment selection will have been announced. It will be very interesting to see what effect this has on the system administration standards process.

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 UNIX System Laboratories, Incorporated

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, 1992

.... 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.3 binary
☐ System V.2 source	☐ System V.2 binary
☐ System V source	☐ System V binary
☐ System III source	☐ System III binary
☐ 4.2 or 4.3 BSD source	
☐ 4.1 BSD source	
☐ V7 source	

☐ Other *(Indicate which)* ....

AUUG Incorporated
Application for Ordinary, or Student, Membership
Australian UNIX* systems Users' Group.

*UNIX is a registered trademark of UNIX System Laboratories, Incorporated

To apply for membership of the AUUG, complete this form, and return it with payment in Australian Dollars, or credit card authorisation, to:

AUUG Membership Secretary
P O Box 366
Kensington NSW 2033
Australia

- **Please don't send purchase orders — perhaps your purchasing department will consider this form to be an invoice.**
- **Foreign applicants please send a bank draft drawn on an Australian bank, or credit card authorisation, and remember to select either surface or air mail.**

This form is valid only until 31st May, 1992

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 UNIX System Laboratories, Incorporated

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, 1992

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 UNIX System Laboratories, Incorporated

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#* ________