ISSN 1035-7521

Australian UNIX systems User Group Newsletter

AUUGN

Volume 13, Number 2

April 1992

Registered by Australia Post, Publication Number NBG6524

The AUUG Incorporated Newsletter

Volume 13 Number 2

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

AUUG Business Manager

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

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

Welcome to AUUGN Volume 13 Number 2.

As has been said elsewhere the summer conferences were a huge success. In this issue we have two more reports, namely the Sydney and Canberra conferences. Papers from the conference include, Remote Terminal Emulators and Other Tools for Cost-Effective Software Quality Assurance by Ken McDonell presented at a number of the conferences, Computer Crime by Ken Day presented at the Melbourne conference, UNIX System 5 Release 4, Use and Administration by Frank Crawford and A Filesystem for a Multi Gigabyte Jukebox by Rex di Bona and Ray Loyzaga, presented at the Sydney Conference and TCP/IP/ISDN by Hugh Irvine at the Adelaide conference. More papers from the summer conferences will be published in upcoming issues.

Despite the large number of papers available from the conference I am still interested in others, so don't hesitate to contact me.

Don't forget that the nominations and elections for the committee are coming soon. If you want to be heard, participate. I've heard that a few of the current committee members will be standing down.

Finally, the Australian Computer Society (ACS) has asked us to inform our members that they are running courses in Sydney on UNIX System V Release 4 Migration and introduction to TCP/IP and the Internet on July 3rd and 1-2nd, 1992 respectively. For pricing and further information contact ACS on (02) 283 5544.

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 2 717 3885
Fax: +61 2 717 9273

AUUGN Book Reviews

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

A number of books are currently being reviewed. These reviews will be published in future issues.

Contributions

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

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:

AUUG 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 27/04/1992

A.J. Mills & Sons Pty Ltd
A.N.U.
AAII
Adept Business Systems Pty Ltd
Adept Software
AIDC Ltd.
Alcatel Australia
Amdahl Pacific Services
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 Defence Industries Ltd
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 (A.C.T.)
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 Sciences of Australia Pty Ltd
Computer Software Packages
Corinthian Engineering Pty Ltd
CSIRO
Curtin University of Technology
Cyberscience Corporation Pty Ltd
Data General Australia
Deakin University
Defence Housing Authority
Defence Service Homes
Dept. of Agricultural & Rural Affairs
Dept. of Conservation & Environment
Dept. of Defence
Dept. of Foreign Affairs & Trade
Dept. of I.T.R.
Dept. of Minerals & Energy (NSW)
Dept. of the Premier and Cabinet - VIC
Dept. of the Premier and Cabinet - SA
Dept. of the Treasury
Dept. of Transport
Dept. of Treasury & Finance
Digital Equipment Corp (Australia) Pty Ltd
DMP Software Pty Ltd
Duesburys Information Technology 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
Gemco
Genasys II Pty Ltd
General Automation Pty Ltd
George Moss Ltd
GIO Australia
Golden Circle Australia
Grand United Friendly Society
Hamersley Iron
Harris & Sutherland Pty Ltd
Hermes Precisa Australia Pty. Ltd.
Highland Logic Pty Ltd
Honeywell Ltd
Honeywell Ltd
I.B.A.
IBM Australia Ltd
Iconix Pty Ltd
Infonetics
Information Technology Consultants
Insession Pty Ltd
Internode Systems Pty Ltd
Ipec Management Services
James Cook University of North Queensland
Labtam Australia Pty Ltd
Lancorp Pty. Ltd.
Land Information Centre
Leeds & Northrup Australia Pty. Limited
Macquarie University
Mayne Nickless Courier Systems

AUUG Institutional Members as at 27/04/1992

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
NSW Agriculture
Nucleus Business Systems
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
Prentice Hall Australia
Prime Computer
Prospect Electricity
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
Royal Melbourne Institute of Technology
SBC Dominguez Barry
Sculptor 4GL+SQL
SEQEB Control Centre
Shire of Eltham
Silicon Graphics Computer Systems
Snowy Mountains 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 Services
Telecom Payphone Services
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
Vibro Acoustic Sciences Ltd.
Vicomp
VME Systems Pty Ltd
Wacher Pty Ltd
Walter & Eliza Hall Institute
Wang Australia Pty. Ltd.
Water Board
Westfield Limited
Wyse Technology Pty. Ltd.

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.

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.

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.

--

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, '91 and '92 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:

Tuesday, 26 May 1992
Wednesday, 8 July 1992
Thursday, 20 August 1992
Tuesday, 29 September 1992
Wednesday, 11 November 1992
Thursday, 17 December 1992
Tuesday, 2 February 1993
Wednesday, 17 March 1993
Thursday, 29 April 1993
Tuesday, 8 June 1993
Wednesday, 21 July 1993

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@clcs.com.au*) or **John Carey** (ph. (03) 587-1444, e-mail: *john@labtam.oz.au*), or look for announcements in the newsgroup **aus.auug**.

The WAUG Column

Hi again. WAUG (the Western Australian UNIX systems Group) has just held its AGM. There have been a lot of changes on the committee this time.

Glenn Huxtable was re-elected to the Chair. Adrian Booth is the new Vice-Chairman. Sue Hoddinott is still the Secretary. Patrick Ko is now the Treasurer. All the officers were elected unopposed.

The ordinary committee members are Mark Baker, Luigi Cantoni, Don Griffiths, Janet Jackson, Major, and David Taylor. Major is still the Membership Secretary and I am the Newsletter Editor, so one of the other four suckers will get the job of Meeting Organiser, commonly acknowledged to be the worst, and most important, job in WAUG. The committee meets to decide on this on Tuesday 28 April.

After the formal meeting we had a rather rapid-fire talk from Indulis Bernsteins of IBM. Indulis described the Distributed Management Environment (DME) being built by the Open Software Foundation (OSF). DME builds on OSF's Distributed Computing Environment (DCE). As is the way with OSF, DME and DCE incorporate software from various organisations. Some that come to mind are IBM's NetView/6000, Tivoli WizDom, and Project Athena's Palladium. It was interesting to hear how they're being used by OSF, and I would like to hear more about them. Any of them could easily be the topic of a whole talk at a WAUG or AUUG meeting or conference.

The AGM is probably the worst meeting of the year for the speaker, because the audience have already sat through the formal proceedings. Indulis did a good job despite these adverse conditions.

The April issue of our newsletter YAUN (Yet Another UNIX Newsletter) has just come out. Adrian Booth has reviewed both the January and March meetings, as well as AUUG '92 Perth (this review also appeared in the previous issue of AUUGN). He also reviewed the O'Reilly handbook *Managing NFS and NIS*, by Hal Stern.

I worry about what we will do if Adrian falls ill, or gets tired of writing. Adrian usually does a *UNIX Tricks and Traps* column too, but since he was so busy, I filled in for April and wrote about the hassles of `ln(1)` and the usefulness of `cron(8)`.

I was pleased to be able to run three advertisements in April YAUN - one in conjunction with Adrian's book review. Advertisements help pay for the newsletter, and are seen by an interested audience.

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>

Third Annual Canberra Conference and Workshops

Ross Hand

The Summer 92 AUUG Canberra Conference was held on the 17th and 18th of February at the National Convention Center and the Australian National University.

Day One

The first day of the conference consists of low cost workshops intended to give novice and expert computer users an introduction to specific areas of computing. The titles and presenters of this years half day workshops where:

X11
David Baldwin, ANU
Liz Keith, Verity

Improving and checking the security of your UNIX system
John Barlow, ANU

Networking LANS and WANS
Geoff Collins, ADFA
Peter Elford, AARN
Geoff Huston, AARN

OSI demonstrated
Mike & YinLeng Husband, NEC Information Systems Australia

Low cost UNIX software
Gustav Meglicki, ANU

Exploiting UNIX with AWK, SED & Shell
Bede W. P. Seymour, ANU

As in previous years the workshops where held at the ANU campus and where given by experts in the subjects. The most popular workshop this year was Networking LANS and WANS given by Geoff Collin, Peter Elford and Geoff Huston. All the other workshops where well subscribed and from feedback obtained from attendees where well presented and appropriately priced. The half day format is very popular, being neither too long or too short. We have resisted the temptation to extend the workshops to a full day as it would either lengthen the conference to 3 days or restrict attendees to one workshop. We also believe that it would be too demanding on our volunteer presenters.

The profit we make from the workshops funds the monthly meetings we hold during the year and also provides us with a float for following years conference. Through cautious planning and spending we have been able to increase the profit from the workshops each year, which then enables us to improve the quality and scope of the conference itself.

Day Two

The second day is the presentation of papers which for the first time was held off campus at the prestigious National Convention Center. This venue was slightly more expensive than a lecture theatre at ANU but the organizing committee felt that we where not attracting enough of the government computing people and that a commercial venue would be an advantage. The National Convention Center also provided the catered lunch and relieved the organizing committee of many of the minor, but time consuming tasks associated with conference organizing. The papers presented on day two where::

Are Vendors Committed to an Open Systems Approach?
John Sarapak, Manager, ICL Australia
John is a member of the OSIcom Australia management committee. He will outline what is happening with Open Systems and how organisations are responding to the GOSIP policy. He will give some examples of where Departments are embracing the OSI standard.

What is the Australian User Alliance for Open Systems?
John Goddard, Chairman, Australian User Alliance For Open Systems
John will outline the aims of the Australia User Alliance for Open Systems (AUAOS), how it will function in Australia and the links already established with similar organizations. Through a series of working groups, AUAOS will develop education programs, position papers on Open System Environments and influence the user and vendor community adoption of open system strategies.

The Real Impact of Open Systems
Richard Cousins, Director, Cousins & Associates
Richard will examine the impact of open systems in terms of the current turmoil and forces shaping the computer industry. He suggests that the Australian UNIX market is larger than commonly perceived, with a marked difference from the US market. He questions the breadth of acceptance for both OS/2 and OSI.

Open Systems, An IBM Perspective
Bill Sadve, Open Systems Manager, IBM Austin Texas
Bill is the Open Systems Manager, reporting to Donna VanFleet, the head of AIX software development. He is an engineer by education and has experience in the development of hardware and software products at IBM. He will discuss the topics of Open Systems Solutions in relation to International Standards. The importance of enterprise wide networking and the evolution of Open systems will also be covered.

UNIX Applications within the Australian Electoral Commission
Rod Medew, Director Computer Services, Australian Electoral Commission
Rod will present some of the experiences with UNIX applications at the Australian Electoral Commission (AEC). He will discuss the Election Night Information Systems and the use of spatial information, using the UNIX operating System. He argues that UNIX is now a commercial operating system that can accommodate a wide range of applications, as experienced at the AEC.

Appropriate Use of Computing Technology
Chris Johnson, Dept of Computer Science, The Faculties, ANU
Computing technology can be both a benefit and a danger to the computer aware and to the general community alike. It is a tool with potential in peace, war, profits, political activity, economic efficiency, saving labour, entertainment, development, communication, law enforcement, daily convenience, daily temptation, and addiction. This is a review of some of the trends and dangers, and a reminder that the creators and providers of computing cannot treat it as a value-free technology.

Secure File Transfer with TCP/IP
Dr Lawrie Brown, Lecturer, Computer Science, Australian Defence Force Academy
Lawrie will discuss the design and implementation of security enhancements to support added authentication and secrecy options in the FTP file transfer utility. This work is based on earlier work enhancing security in the Telnet remote terminal utility.

TCSEC B Level Secure UNIX
Peter James, Principal Consultant, Easams Australia
Peter will give an overview of the functional requirements defined by the US Trusted Computer Security Evaluation Criteria (TCSEC) Orange Book. He will discuss how the TCSEC requirements can be achieved under UNIX.

The ADFA Campus Mail System
Geoff Collins, Senior Programmer, Computer Centre, Australian Defence Force Academy
The PH mail nameserver on one central machine acts as an electronic post-office for email to users at ADFA. It resolves a range of human-friendly mail addresses to a specific user code and machine for delivery. Geoff will describe the advantages of such a system, as well as some of its shortcomings.

Wide Area Information Services
David Baldwin, Senior Programmer, ANU
David will discuss the delivery of information services in a wide area network arena. He will look in particular at WAIS, a joint project between Thinking Machines, Apple Computer and Dow Jones.

Engineering a Connection to AARNet
Peter Elford, Australian Academic and Research Network
AARNet is a TCP/IP wide area network connecting the Australian Universities, CSIRO and the global Internet. Peter will provide an overview of the AARNet affiliate membership program which allows eligible organisations to connect to AARNet.

The Use of ISDN in the ACT Government Computing Service.
Greg Mills, Greg Mills and Associates.
Greg described the rationalisation of computing networks within the ACT with particular reference to the use of ISDN services.

Sponsorship

Financial sponsors this year where Hewlett-Packard Australia, ICL Australia, IBM Australia, Sun Microsystems and Australian Technology Resources.

Publicity

Two large advertisements, the first approximately 11cm by 16cm, the second 11cm by 8cm in the Canberra Times and a mailout of a professional produced conference brochure and registration form to our 250 plus mailing list comprised the main publicity for the conference. Sponsors where also given multiple copies of the conference brochure to distribute to their customers. The conference was of course announced in the Electronic News, but many of our attendees do not have news access and so obtain information about the conference through the more conventional channels.

Summary

We gave over 150 half day workshops at $40 for AUUG members and $50 for non members. The workshop fee included a morning or afternoon tea but not lunch.

The second day was attended by approximately 110 people (including speakers) at a cost of $45 for AUUG members and $55 for non members. A three course lunch as well as morning and afternoon teas was provided. The notable Canberra Times Computer journalist David Ives was our lunch speaker as well as chairing the Open Systems session of the conference.

There was no financial cost to the national AUUG as we used the profits of last years conference for deposits for the venue and publicity and mailing costs.

The conference and its organizing is of the same format as that in 1990 and 1991 with the exception of increasing the number of workshops offered and using a commercial venue for the second day.

Many people worked on the organizing committee and gave many hours of their time for what was our most successful conference to date. Special mention should go to John Barlow, Liz Keith and Peter Wishart.

NSW Summer Conference — Report

This year's Summer Conference NSW was a great success. Fifty-four people turned up to hear many technical presentations, and to natter over a cup of tea.

The lineup of speakers included Ray Loyzaga with his paper on *Bruced* — a program to do what Bruce (a system administrator at Basser Dept. of Computer Science) used to do — Andrew McRae with his poor-man's logic state analyser, Greg Rose talking about some of what he did at IBM (no, it wasn't MVS hacking!) and other people talking about things they've done with UNIX.

I'm not going to give abstracts for all the papers, but would like to make a few general comments.

The unofficial theme of the day seemed to be Administration and Measurement. Six of the twelve papers had something to do with measuring things or administering things: Ray Loyzaga and Peter Grey's papers on system administration, the paper Max Mattini presented on the OpenEyes performance monitoring tool, Andrew McRae's paper on profiling the kernel, Rex Di Bona's talk on the optical filesystem, and Greg Rose's talk on the MIPS farm all had system performance and administration as an underlying theme. The rest of the papers ranged from pure speculation (John Lion's paper about UNIX in the 21st Century) through description of the latest commercial UNIX (SVr4/ES: enhanced security) by Chris Schoettle to my paper on culture clash between kernel hackers and software engineers.

The full program was:

Peter Elford	*Engineering a Connection to AARNet*
Andrew McRae	*Hardware Profiling of Kernel Network Code*
John Lions	*UNIX in the 21st Century*
Greg Rose	*The Farm: Harvesting MIPS*
Chris Schoettle	*Overview of SVR4.1ES*
Peter Chubb	*Softway Engineering, or, Structure UNIX Kernel Hacking*
Frank Crawford	*UNIX System 5 release 4, Use and Administration*
Ray Loyzaga	*Bruced — Remote, Reliable System Administration*
Peter Gray	*Access Control Made Easy*
Punya Palit	*UNIX on a Fault Tolerant Platform*
Rex Di Bona	*Building a File System on a Multi Gigabyte Jukebox*
Max Mattini	*OpenEyes: A Performance Monitoring Tool for UNIX-Based Systems*

After the day, quite a few of us went to a SWiGS meeting. This was a fitting conclusion to the day, with interesting conversation about the next conference (and the one just completed) over some wine.

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) 717 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	MHSnet	UUCP
		TCP/IP	SLIP	
		Other (Please specify): ____________________		
A6.	Type of connection:	Direct	Modem/Dialin	Modem/Dialout
		X.25/Dialin	X.25/Dialout	
		Other (Please specify): ____________________		
A7.	If **modem**, connection type:	V21 (300 baud)	V23 (1200/75)	V22 (1200)
		V22bis (2400)	V32 (9600)	Trailblazer
		Other (Please specify): ____________________		
A8.	Estimated traffic volume (in KB/day): (not counting netnews)	< 1	1-10	10-100
		> 100: estimated volume: ____________________		
A9.	Do you require a news feed?	Yes	No	
		Limited (Please specify): ____________________		
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 TCP/IP Other (Please specify): ____________	MHSnet SLIP	UUCP
B2.	Type of connection:	Direct X.25/Dialin Other (Please specify): ____________	Modem/Dialin X.25/Dialout	Modem/Dialout
B3.	If **modem**, connection type:	V21 (300 baud) V22bis (2400) Other (Please specify): ____________	V23 (1200/75) V32 (9600)	V22 (1200) Trailblazer
B4.	Maximum traffic volume (in KB/day): (not counting netnews)	< 1 > 100: acceptable volume: ____________	1-10	10-100
B5.	Will you supply a news feed?	Yes Limited (Please specify): ____________	No	
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:			

Remote Terminal Emulators and Other Tools for Cost-Effective Software Quality Assurance

Ken J. McDonell
System Technology Laboratory
Pyramid Technology Corporation

Abstract

Software Quality Assurance (SQA) is a critical stage in the certification of serious software products. The process is inherently time-consuming and labour-intensive. Consequently the use of software tools is quite appealing, and both special-purpose and general-purpose software tools have been used, principally to ensure functional correctness.

The shortcomings of the traditional approaches arise from the growing complexity of application code, the omnipresence of full-screen and graphical user interfaces and the requirement for performance auditing to be included in SQA.

This paper concentrates upon the adoption of a Remote Terminal Emulator (RTE) (from benchmarking and performance analysis) to perform a constructive role in SQA – of particular relevance is the extent to which a sophisticated RTE can both reduce SQA costs and improve the quality of the certified software. The proposed solution features an RTE operating in concert with an array of standard UNIX* text manipulation filters and tools.

1. Introduction

This paper identifies several constructive roles that generic software tools and general-purpose Remote Terminal Emulators (RTEs) may play in the process of Software Quality Assurance (SQA). Of particular relevance is the extent to which a sophisticated RTE can both reduce SQA costs and improve the quality of the certified software.

Pyramid's *sscript* product is an RTE with a rich set of support tools that are particularly helpful in developing and executing tests for SQA. The examples in this document will be based upon *sscript*, but the concepts are readily translated to any RTE with comparable flexibility and ancillary support tools.

SQA covers both the development and maintenance phases of the software life-cycle. SQA is a multi-dimensional activity, with different classes of tests designed to establish different aspects of software "quality". For the present purposes, software quality is assumed to span functional correctness, robustness and performance[1].

* UNIX is a trademark of Unix System Laboratory.

1. Note particularly that this definition of "quality" excludes techniques based upon software "science" metrics, code coverage analysis and β-release programs – these methods may provide increased confidence about software quality, or be used in conjunction with SQA techniques, but the more important criteria is the deployment of production software whose "quality" is inversely proportional to the incidence of bug reports, application failure, end-user surprises and unacceptable performance.

The following taxonomy of testing modes is suggestive of the procedures involved in the SQA process, although in practice individual tests, or parts thereof may be re-used to establish quality in more than one area.

1. Unit tests of functional conformance. For a particular procedure, form or screen, does the application produce the desired results for all expected user inputs (both valid and invalid)?
2. Module tests of functional conformance. Does a particular transaction or application execute correctly?

   Unit and module tests are sometimes collectively referred to as **regression** or **validation** test suites.
3. Single-threaded performance. Is response time and/or throughput acceptable for one user on the designated platform?
4. Multi-user performance. Quantification of response time and/or throughput as a function of increasing concurrent load on a designated platform.
5. Capacity planning. For a given workload profile, what size platform is required to support the expected community of users?

Each mode of testing implies the application(s) have successfully completed all the preceding modes of testing, e.g. single-threaded performance presumes functional conformance at both the unit and module levels.

In the following sections we will expand upon the classes of tests and techniques appropriate to each of the above modes of testing.

Potentially both the creation **and** execution of the required tests for SQA is labour-intensive, time-consuming and demands the co-operation of developers and end-users.

The software tools approach to SQA can reduce **both** the labour and elapsed time components, at the same time, **increasing** the effectiveness of the process and the resultant software quality.

In general terms, SQA testing benefits from the RTE approach when one or more of the following conditions holds.

1. The application supports a form or full-screen user interface. Typically these applications will not function correctly if their input comes from a file or simple input stream such as a UNIX pipe, rather the test driver must provide the full services and functionality of a pseudo terminal – this is exactly what an RTE provides in conjunction with a remote login connection, e.g. back-to-back serial lines or *rlogin* or *telnet* or a "pad" to an X.25 connection.
2. The application produces output which legitimately varies from run to run with the same input. The best examples are screens that include date and time fields. We require the tests to be insensitive to these variations, but retain sensitivity to variations in other aspects of screen appearance and/or application behaviour. Because an RTE provides synchronization at defined points of the user-application interaction, the RTE can skip over parts of the application's output whilst checking for the presence of the expected cursor motion, prompt or other output that is deterministic (i.e. does not vary from one

run to the next).

3. User input should not be sent until the application is ready for it because some applications discard "typeahead" characters before a keyboard read. For a particular test, this may disturb the sequence of input characters in a unpredictable fashion, typically leading to unexpected output from the application. An RTE provides strict "send and wait for the expected response" synchronization, so typeahead may be avoided as required.
4. Realistic multi-user performance measurements require the simulation of end-user think times and inter-keystroke delays. These facilities are generic for serious RTEs.
5. Low-level tests are to be combined and re-used to implement the higher-level testing modes. An RTE provides a single scripting language and support tools that allow parts of the script composition process to be an automated "cut and paste" operation.

2. Unit Tests for Functional Conformance

This mode of testing is the most common for an application developer/maintainer, and one can assume that some testing has been performed before the application as passed from the developer/maintainer to the SQA staff.

For SQA, it is imperative that

a. Selection of input values provides complete **coverage** of all possible **field combinations** as per the specifications of the software unit. For example, a screen with 1 mandatory input field and 2 optional input fields requires 4 tests to cover all of the possible combinations of user-supplied values for the 3 fields.

b. Selection of input values provides reasonable **coverage** of input **values** that have different semantics. For example, positive and negative dollar amounts, maximum, minimum and illegal values for "quantity", boundary conditions (especially time periods that cross year or month boundaries), local and remote warehouse locations, etc.

c. Each "bug fix" generates at least one additional test for the error condition that had been previously undetected.

d. Periodically, **all** unit tests be run, irrespective of the modules that have been recently changed – this permits unexpected side-effects of software changes to be located, e.g. changes in a data entry module may expose programming errors or data dependencies in other software modules that have not been changed for a long time.

Increasing the "coverage" from the unit tests can often be automated, given the correct tools. The program *datagen* in the *sscript* distribution produces all possible permutations of data values as described in a simple declarative language.

For example, an application may require 3 input values, separated by tab characters. The first value is "1", "2" or "3". The second value is a description that is optional if the first value is "1". The third value may be "Y" or "N".

The corresponding *datagen* specification is

```
{ { 1 "\t" { "" | "descr" } } |
  { { 2 | 3 } "\t" "descr" } } "\t" { Y | N }
```

From which *datagen* produces the following 8 possible input sequences.

```
1            Y
1            N
1     descr  Y
1     descr  N
2     descr  Y
2     descr  N
3     descr  Y
3     descr  N
```

Once the user input has been generated, the next requirement is to generate the reference copies of validated output that will be used in all subsequent testing.

For batch-oriented applications, tools like a powerful command language (i.e. a "shell" in UNIX parlance), flexible re-assignment of input and output streams, pattern-based stream editors and a differential file comparator provide the building blocks from which effective SQA procedures may be constructed. It is exactly this approach that is used for the *sscript* product!

For interactive applications with more complex user interfaces, the same ideas may be used, but the inclusion of some more sophisticated building blocks is required; and RTE is an example, but its usefulness and productivity is critically dependent on the support tools for the process of RTE script development, execution and performance analysis.

For the process of RTE script development, for example, in the *sscript* environment, tools are provided to perform the following steps.

1. Capture user keystrokes from a sample session, e.g. a test case which validates the correctness of recent software change.

   Alternatively the keystroke file may be generated by *datagen* else manually created by someone with a thorough understanding of the application.

2. Execute a keystroke file and save all of the response from the application; the output is an interleaved "dialogue" of user input, followed by system response. The program *mkdialogue* does this.

   During this step, the dialogue is displayed on the user's terminal, so it is possible to visually verify correct application behaviour.

3. The program *mkscript* generates an *sscript* script from the dialogue. This script incorporates the user's input and the trailing edge of the system's response as a string to wait for before sending the next fragment of user input.

   Once generated, the script may be executed an arbitrary number of times and each successful execution verifies that the application is functioning correctly in a general sense (more on this a little later).

Assorted other *sscript* and UNIX tools may be used to add complexity to this simple scenario – the important point is that changes to the application's input or output specifications may necessitate re-doing the process, and given the large number of tests that may be involved the objective is to **automate** the process to the maximum possible extent.

The requirement that the complete suite of tests be run periodically (certainly before each release of a new revision) benefits most from a testing environment in which a large number of tests can be run automatically, the results verified and only exceptions need be reported.

This is exactly the methodology that is used in the testing of *sscript* itself, and the relevant scripts are distributed as an integral part of the *sscript* distribution. Each test consists of the following parts;

- a test script (if executed correctly, i.e. with no synchronization errors when a particular application response is expected but not received, this script verifies that the application is functioning correctly in a general sense, however additional tests of application output may be required to increase confidence in application correctness),
- optional input file(s), e.g. a list of valid part numbers from which pseudo-random selections will be made,
- an optional file containing command line options to be used when the application is invoked,
- an optional shell[2] script to filter the output, e.g. to translate non-deterministic text, or to strip login welcome messages, or to post-process the output file(s),
- the expected (correct) output.

In addition two shell scripts are provided to create (or recreate) the expected output file, and to execute one or more tests, compare the observed output with the expected output and report any variations.

3. Module Tests for Functional Conformance

Module Tests are typically composed of sequences of Unit Tests that constitute a complete transaction or unit of work or a complete user session.

Hence the techniques used to develop and run these tests are very similar to those employed for the Unit Tests, with the additional advantage that some preplanning of the Unit Tests will allow Module Tests to be automatically created by ''cut and paste'' techniques applied to the Unit Tests.

Additional checks applied at the completion of a Module Test may include

- file/table/record **presence**, if it should have been created,
- file/table/record **absence**, if it should have been deleted,
- aggregation over external files or database tables, e.g. count records, sum the balance fields,
- check data values in files or records against expected values, and
- check the committed transaction count.

Again the shell and the associated text processing tools may be used to build generic tools. For example the script below will **create** SQL queries, run them, capture the output, filter out the noise and report the relevant data.

2. Here ''shell'' refers to the UNIX command interpreter, e.g. */bin/sh*, rather than the *sscript* RTE interpreter.

```
count()
{
    echo "select count(*) from $1 \
    | sql \
    | sed -n '/^[0-9 ][0-9 ]*$/s/ //gp'
}

#
# count the tuples in the critical tables
#
for table in part supplier emp
do
    echo "$table: `count($table)`"
done
```

4. Single-threaded Performance

Testing for single-threaded performance is very directed – we are interested in locating application areas (functions, screens, modules) whose performance is so bad to suggest,

a. poor algorithm selection in the application,

b. poor algorithm implementation in the application,

c. inappropriate data structure or data base access paths (most commonly a missing index over some critical attribute of a table),

d. ''brain-dead'' DBMS or operating system performance, requiring an application re-design to workaround.

Selected Unit or Module Tests may be employed, but the coverage does not need to be as complete.

Performance will usually be measured in terms of throughput and/or response-time, and any sensible RTE will provide these statistics, but other batch-oriented testing techniques are unlikely to provide the response-time measures.

Note that the emphasis is not ''is the performance acceptable'' (that comes next, because single user performance is rarely of interest in determining ''acceptable''), but is something pathologically broken? If the answer is ''yes'', and re-implementation is undertaken then **both** the functional conformance and single-threaded performance tests must be re-run with the modified application.

5. Multi-user Performance

The hardest task at this stage is to devise **workload profiles** that constitute a representative and accurate characterization of how a community of users will use the system or application.

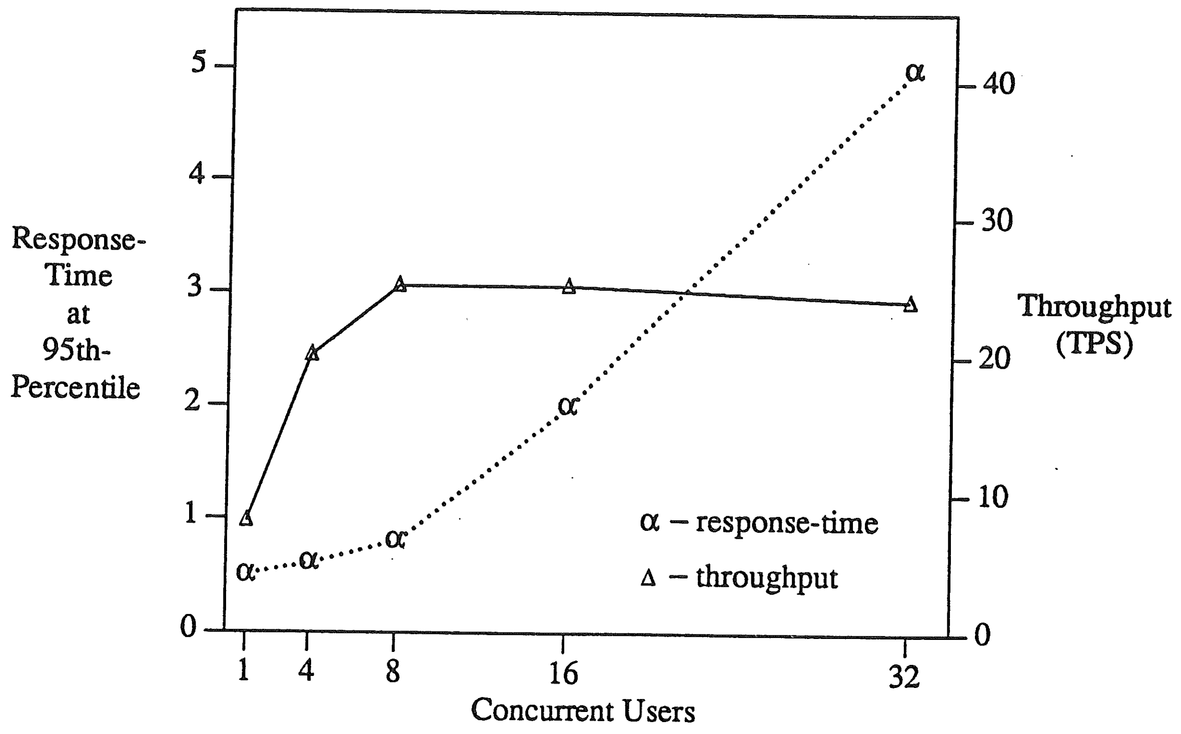
This task is a common requirement for benchmarking and performance analysis, but it is not generally well-understood and it is difficult to do well; some special assistance may be warranted.

The script parameterization facilities within *sscript* are most useful here in building generic scripts that generate a constant amount of work without artificial duplication of activity, e.g. different scripts retrieve different account records from the database.

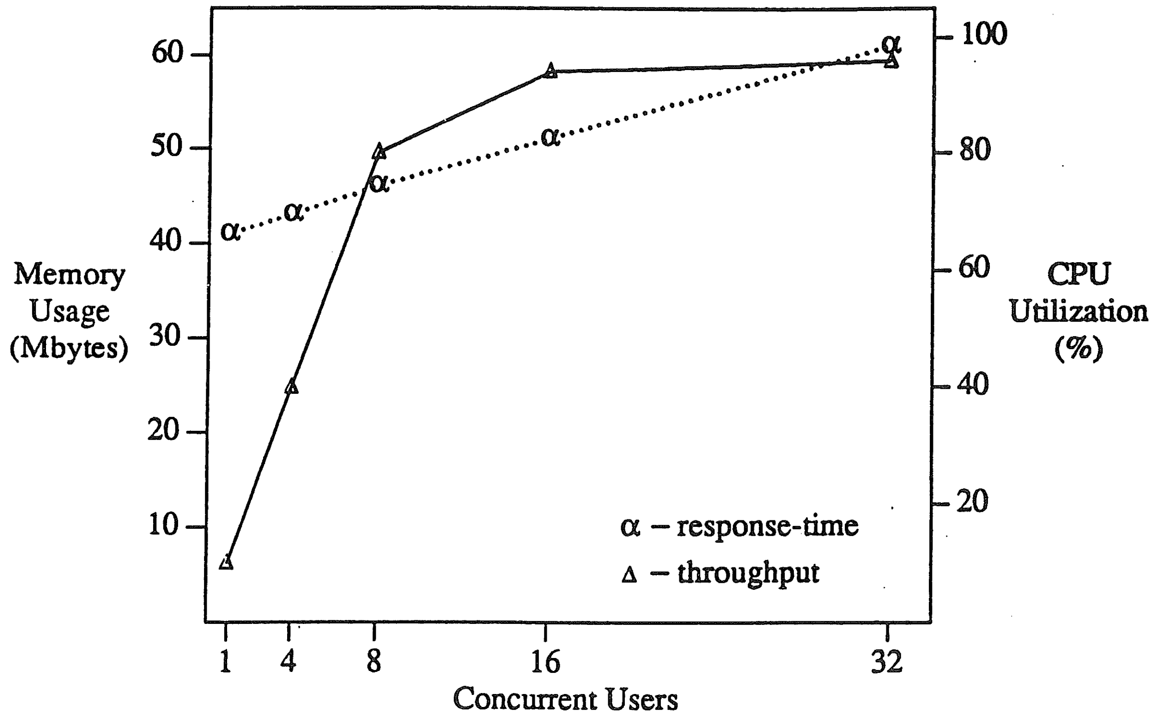
In many cases, these parameterized scripts can be automatically created from the specific scripts used for the Unit or Module Tests, by using the UNIX file editing tools.

The goal of these tests should be to produce graphs like the one below, i.e. to characterize performance and resource utilization as a function of increasing concurrent load, for a particular platform configuration and workload profile.

Once a single script has been developed for the "typical" user, the *sscript* tools[3] make producing these graphs an automated and straight-forward process.



3. In conjunction with the UNIX text processing tools, e.g. *grap*, and the Pyramid performance monitoring tools like *psnap*.



The tests will help produce the data for Capacity Planning, but also serve to highlight situations in which performance bottlenecks are encountered, either in the application, the DBMS, the operating system or the underlying hardware.

6. Capacity Planning

Once the apparatus for Multi-User Performance Testing is in place, this may be used to produce capacity planning guidelines, either in a general sense (e.g. we can support *N* ''typical'' users per Brand X CPU, each requiring *K* Mbytes of real memory) or for a specific customer and platform.

Capacity planning for a specific customer is most important, in that the application developer is able to generate a realistic (i.e. operational) load using the RTE, measure the resource consumption and performance and then be confident of the quality of service the real users will see in their production environment. The value of this sort of ''certification in advance'' cannot be under-estimated, and should be used as part of the on-going SQA process. Specifically, we would like to be able to avoid the scenario where installing a new version of an application brings attendant performance problems that are not evident until the load increases in the production environment (by which time, rolling back the software version may be a very costly option).

7. External Environment

No test runs in isolation, and the environment will influence the results obtained.

Particular attention should be paid to,

- Login name, passwords, login shell configuration and home directory contents.

  The typical *sscript* approach to this problem is to parameterize the script (for login name and password in particular), and to use the first few send-receive interactions to establish a particular shell configuration (e.g. start a shell, set the prompt and set the critical environment variables). In this way, the tests are largely self-contained, with minimal (error-prone) procedures required on the remote system.

- Initial database state.

  Scripts have to be developed to either re-establish an known database state, else undo the residual effects of previous tests.

- Initial data files – same as database state.

8. Desirable RTE Features

Since the RTE is emulating the behaviour of an end-user, the accuracy of that emulation and the ease with which RTE scripts can be developed have a critical impact on the cost-effectiveness of whole process.

The development of *sscript* has been driven by benchmarking and SQA requirements as they have been identified by users both within Pyramid and external customers. This process has produced an environment with many support tools and a scripting language that is extremely flexible. Of particular note, the following attributes increase the productivity of an *sscript* user.

- Tools to automate the script development from keystroke capture (or generation) through to a working script.
- For multi-user applications, the program *drive* has been specifically designed handle logging in large numbers of users, rendezvous after login, graceful cleanup after normal termination of all scripts, enforced cleanup following a user initiated termination, and control over optional shutdown when one script fails.
- All files created by *sscript* and the associated tools are ASCII text files, allowing full use of the UNIX file manipulation and editing tools for data extraction, ''cut and paste'' and differential file comparisons.
- Since UNIX files may be read from *sscript* scripts, it is a simple matter to parameterize the scripts to send data values drawn randomly or serially from sets of valid values.
- Since *sscript* scripts can create UNIX files, test results, error messages, audit trails, etc. can be saved from a test for subsequent processing.
- The statistics collection, post-processing and report tools allow analysis to be confined to a particular real-time window, make selection of desired interactions a simple matter, and produce reports of either transaction throughput and/or response-time distributions.

9. Conclusions

Effective SQA is a technically achievable, and economically critical, goal that can be attained by adopting the proven techniques and philosophy from the world of software tools, namely keep it simple, select the right tool and build new tools.

SQA effectiveness (and hence software quality) are enhanced when the SQA process embraces the notions of generalized solutions to a classes of related problems, incremental development and tool re-use, and evolution through flexibility.

Within this framework, the UNIX shell and file manipulation tools are of fundamental importance, and special-purpose tools such as RTEs offer significant added value to SQA productivity and effectiveness.

COMPUTER CRIME

Det. Sgt. Ken Day
Australian Federal Police
Computer Crime Section

INTRODUCTION

The true extent of computer crimes being committed in Australia is unknown, however we are aware of the types of criminal activities that can be and are being perpetrated by computer criminals. The actual and potential threat from this class of criminal should be recognised by every person or organisation that uses or operates computer systems. The resulting effect from these illegal activities can vary from the unsuccessful attempt to access a computer system to the permanent disabling of an entire computer network.

In this presentation I will give you a general overview of the commonwealth computer crime legislation and what your options are when such an offence is committed on your computer system. I hope at the end of this discussion you will have an understanding of the commonwealth computer crime legislation and how you can assist the Australian Federal Police in the investigation of these offences. I would welcome any questions at the conclusion of this presentation.

OFFENCE SECTIONS

Commonwealth computer offences are found in the Commonwealth Crimes Act 1914, Part VIA, which includes sections 76A to 76F inclusive. These offence sections list the circumstances that constitute an offence and the degree of potential penalty associated with those actions.

The commonwealth has the jurisdiction to enforce these laws in 2 types of situations and they are:

(a) when the criminal acts are directed against a Commonwealth computer, which includes a computer containing data on behalf of the Commonwealth,

(b) when the criminal acts are directed against any computer via a commonwealth facility (example - Commonwealth owned and controlled communication mediums such as Telecom, Austpac and OTC.).

There are basically four categories of offences in this legislation.

The first category of computer offences relates to unlawful access to a computer. The maximum punishment of this act is six (6) months imprisonment.

The second category of the computer offences relates to unlawful access to a computer with the intent to defraud any person. The maximum punishment for this act is two(2) years imprisonment.

The third category of offences relates to unlawful access to computers and the viewing of certain types of data namely data that relates to:

- The Security/defence/international relations of Australia;
- The existence or identity of a confidential source of information relating to the enforcement of a criminal law of the Commonwealth or of a State or Territory;

--

- The enforcement of a law of the Commonwealth or of a State or Territory;
- The protection of public safety;
- The personal affairs of any person;
- Trade Secrets;
- Records of a financial institution; or
- Commercial Information the disclosure of which could cause advantage or disadvantage to any person.

The person must have some knowledge of the type of data they are viewing to satisfy the conditions for this type of offence. The maximum punishment for this offence is two (2) years imprisonment.

The fourth category of offences relates to 3 certain types of effects to a computer system that have been caused intentionally by a person.

The first part of this category makes it illegal to destroy, erase, alter data stored in, or insert data into a computer whilst the second part makes it illegal to interfere with or interrupt or obstruct the lawful use of a computer.

The maximum penalty for this offence is ten (10) years imprisonment.

RESPONSE TO AN INTRUSION INTO A COMPUTER SYSTEM

The Commonwealth computer legislation was enacted in July 1989. Prior to this there was no national legislation that adequately covered the occurrences of computer crimes. The advent of this legislation has given the Australian Federal Police the jurisdiction to investigate and prosecute computer offenders in Australia regardless of where the victim computer systems are physically located.

The Australian Federal Police are now actively involved in the investigation of computer offences in this country. We have been actively investigating computer offences for more than two years and during this time we have gained experience in dealing with these types of offences. One reasons why I am here is to let you know that the Australian Federal Police have a Computer Crime Section. The other reason is to inform you that if someone unlawfully accesses your computer system you can report that matter to the Australian Federal Police for investigation.

The details, relating to these instances of unlawful accesses, that we are interested in are:

1 Date and time of any illegal intrusions into the victim computer system;

2 The technique by which the suspect made connection to the victim computer system, ie via Internet, direct modem connection, Austpac etc;

3 Details of the computer system that has been accessed;

   (i) Its physical location;

   (ii) The name and IP address (if applicable) of the computer system;

   (iii) The administrator(s) of the computer system;

   (iv) Details of what the intruder has done whilst on the victim computer system. This may include copies of history logs, any captured data of their login sessions or details of other computer systems they have connected with.

   (v) The name of the account (if applicable) that was used to access the victim computer system and details of the legitimate owner of that account.

4 Details of any information left on the victim computer system by the person who unlawfully accessed it. More often than not 'hackers' hide their information in hidden directories.

I am aware that there are many instances of minor 'hacking' that occurs in this country. In isolation these incidents may appear to be insignificant therefore it may be tempting to merely ignore those occurrences and not report them. Those isolated events may well be just that but they may be also part of a larger conspiracy between experienced 'hackers' to unlawfully access computer systems in Australia and overseas. My proposition is that I would prefer it if you would report these minor incidents to us, then this would let us collate all the information and identify any pattern of major 'hacking' incidents.

If your computer system is unlawfully accessed please do not hesitate to contact me or any member of the Australian Federal Police computer crime section on (03) 607 7777.

UNIX System V Release 4, Use and Administration.
A User's Perspective.

Frank Crawford

Aust. Supercomputing Tech., Private Mail Bag 1, Menai 2234
(frank@atom.ansto.gov.au)

ABSTRACT

UNIX System V Release 4 has been around for some time, and versions from various vendors are now reaching the market. It is poised to become the standard version of UNIX, but what is it really like? Is it really standard? How is it different to what is currently available? What problems will you face the first time you use it?

This paper will attempt to answer those and many more questions. It will describe what it is like to use and to administer, what new features have been added, what has been removed and more importantly what has subtly changed. It will concentrate more on things that affect the ordinary user or the person trying to administer such a system rather than those affecting programmers.

1. *History*

UNIX has been through many versions over the years from Edition 6 (and even earlier versions) through to the latest version, System V Release 4. However, it has not been a simple progression from one to the other, rather at times there have been many different versions available at the same time (see Figure 1.), some tailored for specific markets, others competing virtually head-to-head. For example, just a few years ago there were three main version, System III, the official AT&T product, BSD4.2, the most popular version at the time, and Xenix, the version aimed at the low end of the market.

Today, there is one standard version System V Release 4 (SVr4) (this doesn't preclude other specialised versions, such as BSD4.4 or even OSF/1). This standardisation was achieved by merging many of the best features from the different versions, including Xenix, BSD4.3, SunOS and the various research version of UNIX from AT&T Bell Laboratories.

Even more importantly, with the formation of UNIX System Laboratories (USL), and other related moves by AT&T, most vendors are making SVr4 the basis of their own UNIX versions, *e.g.* Sun's Solaris 2, Fujitsu's UXP/M and Pyramid's DC/OSx.

2. *Overview of SVr4*

Although SVr4 was announced a number of years ago, only in the last 12 months have commercial versions started to appear, and many more are planned for release in the near future.

In all descriptions of SVr4 much is made of the new and enhanced features, but the differences are much more than these. Users of any of the current versions of UNIX will find something that is new or modified, whether it is a System V or BSD based system. Because of this it is important not to make incorrect assumptions about the system. Even more there are many new features that have to be accounted for.

Although the changes cover all areas, some of the highlights include:

- Implementation of both the Internet and Berkeley networking utilities (*i.e.* ftp, telnet, rsh, rlogin, *etc.*) using STREAMS,
- Implementation of a virtual file system interface and a number of different file systems, including the standard System V file system (*s5*), the Berkeley fast file system (*ufs*) and a two networked file systems (*rfs* and *nfs*).

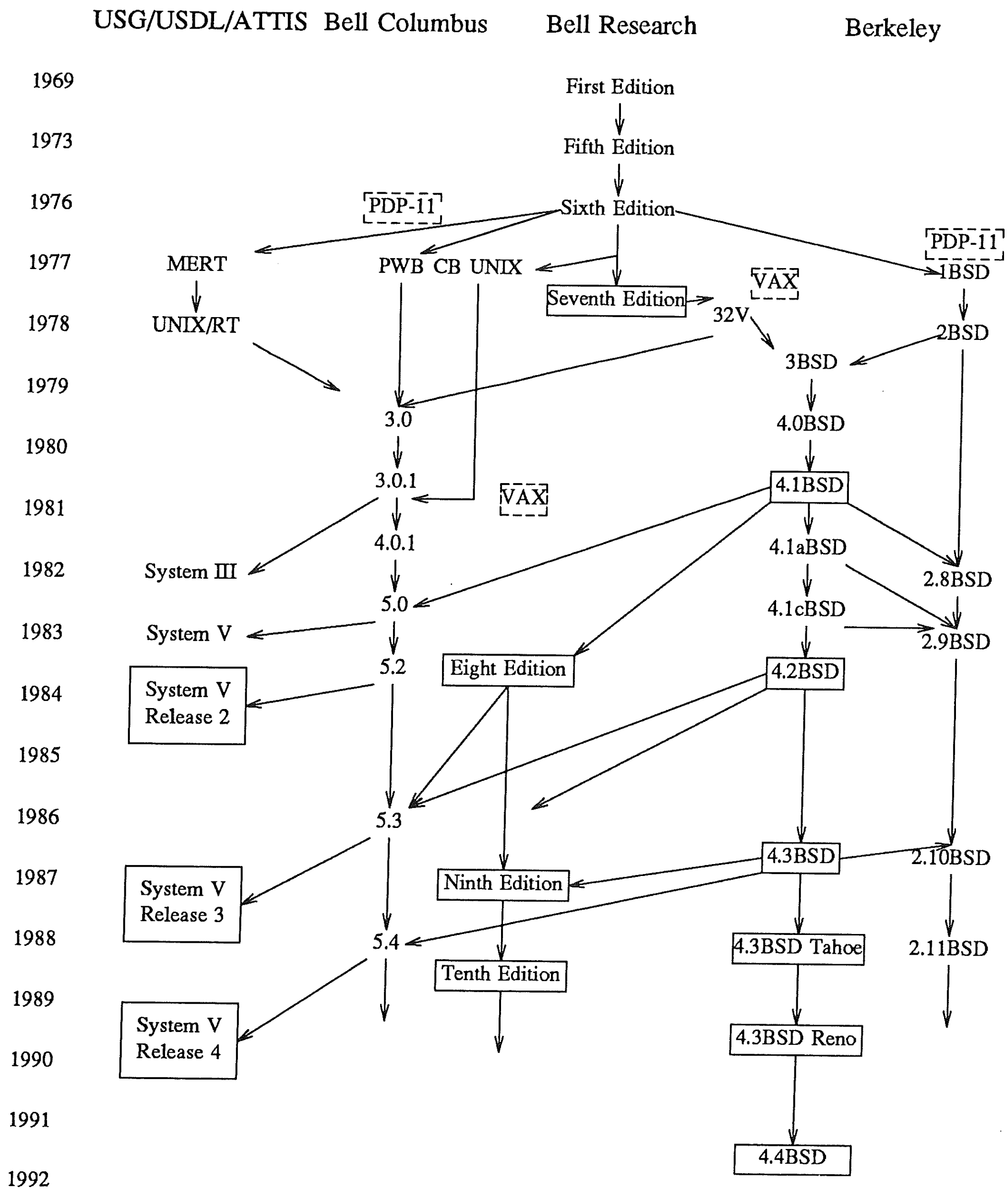


Figure 1. The UNIX System Family Tree.

- Inclusion of symbolic links on *s5* and *ufs* file systems.
- Implementation of */proc* and *fdfs*, *i.e.* the first is an interface to running procedures, especially useful for debugging, and the second an interface to currently open file descriptors. Both are implemented as virtual file systems.
- Restructuring of the system directory tree, including the addition of */var* and */stand* file systems.
- Dynamic linking and *ELF* linkage format.

3. *User Differences*

Depending on what system you are used to, you will either experience a major difference or else wonder what all the fuss is about. SVr4 has a number of major changes from standard SVr3.2, but many of them were common extensions, such as networking utilities and BSD compatibility.

3.1 Symbolic Links

Symbolic links are basically a file which contains the name of another file to be used in its place. Unlike normal (or hard) links they can be used to refer to normal files or directories, they can also refer to objects on other file systems, or even to non-existent objects.

These links are not without their problems. These include the getting "lost" in the file system, due to *cd <dir>* not being reversed by *cd ..* (although some shells such as */usr/bin/ksh* internally track the directories and attempt to backtrack for you). Also the links are interpreted at the point they are found, so loops are possible (although they fail after a fixed number of links), e.g. *link* $\rightarrow$ *link.*

3.2 Common Locations

Over the years the number of different locations for programs has multiplied. With SVr4 most of the common programs are located in */usr/bin*, there is, however, a symbolic link from */bin*.

This is the theory, unfortunately in practice everyone adds their own additional. For example, Fujitsu's UXP/M adds a directory */usr/uxp*, most versions of SVr4 have a */usr/lbin* and most installations will add a directory */usr/local/bin*. There is also a */usr/ucb* which is covered in the following section. Even SVr4 includes */usr/ccs/bin* which includes the C Compilation System (*i.e.* compilers, linker, *etc.*), although these programs are often linked to */usr/bin* (or if not, then the administrator should do so).

3.3 BSD Compatibility

One of the most widely publicised features is the BSD compatibility. If you are moving to SVr4 from a Berkeley compatible system then this is invaluable, on the other hand, if you are used to System V then this will have very few uses. In addition in the long run (*i.e.* over the few releases) this will be removed.

To enable most of these function you need to include */usr/ucb* in your **PATH**. To choose Berkeley functions over SVr4 you have to specify */usr/ucb* before */usr/bin* in your **PATH**. More importantly, some utilities, such as */usr/bin/sh*, change their actions depending on the order specified in the **PATH** variable.

3.4 Job Control

One other important addition for BSD is job control. This is the ability to pause jobs, and to move them from the foreground (*i.e.* the job accepting input from the terminal) to the background (*i.e.* a job running unattached to the terminal). This is a very useful feature which has long been available in BSD systems.

4. Administration Differences

The largest differences are found in the area of system administration. For some time AT&T have been enhancing the administration, both with procedures from other commercial versions of UNIX, and from the internal research versions.

New features in SVr4 include a unified approach access to the system, facilities for virtual file systems, the */stand* partition for bootable images, reorganised system partitions, new backup procedures and new networking procedures.

4.1 File System Reorganisation

With the advent of workstations, diskless systems and other networked systems a reorganisation of the system layout is long overdue. The reorganisation has been designed with the aim of sharing files across the network. This has involved splitting files into the following directories:

root	which contains files necessary for booting the system.
/usr	which contains shareable files that are static over the life of the system. This file system can be mounted *read-only* and generally contains architecture dependent files.

/usr/share	which contains architecture independent files, *e.g. termcap* and *man* pages.
/var	which contains files and directories whose contents change over the life of the local system, such as system log files.
/home	which contains the home directories and files of the system's users.

Any of these can be mounted as a separate file system depending on the system size and requirements (and in fact some have to be *e.g. /stand*).

Along with this restructuring, various system utilities have been reorganised. Many of the utilities that were previously located in */etc* or */bin* have now been moved to */sbin* or, if they are considered non-essential, in */usr/sbin.*

Finally, the */dev* directory has also been reorganised, which related devices located in common subdirectories, *e.g.* all terminals are in */dev/term*, pseudo-terminals in */dev/pts.*

4.2 Virtual File System and vfstab

In many ways a new feature is the *virtual file system* interface and the ability to have many different file system types. Basically this is an extension of *vnodes*, introduced for *nfs*, which allows many different file system structures to share common interfaces.

Some file system types supported include:

s5	The traditional UNIX file system.
ufs	An implementation of the BSD *fast file system*, which optimises disk usage for larger files.
rfs	Remote File Sharing, *i.e.* the standard distributed file system type for System V UNIX.
nfs	An implementation of the Network File System, originally developed by Sun Microsystems.
/proc	The process file system type, which is a mechanism for accessing the address space of running processes.
bfs	A very simple file system that provides support for file-system-independent booting.

Along with the introduction of these file types there comes the problem of administration. In the case of the *virtual file systems* this is handled by specifying in the file */etc/vfstab* various information including the file system type. To handle various administrative procedures, a generalised wrapper program invokes the appropriate specific instance for a particular type. For example, */sbin/fsck* checks the file system type in */etc/vfstab* and then calls the appropriate version of *fsck* from */usr/lib/fs/<type>* directory.

4.3 Booting Procedures

Another area of major change has been in system booting. With the advent of the various file system types it would be too difficult to write a bootstrap program that handled all types. To cope with this SVr4 has introduce a simple file system (*bfs*) specifically for handling standalone programs, such as the UNIX kernel. To handle this SVr4 has included specific types in the disk VTOC indicating which partition is of type *bfs* and thus usable by the bootstrap program.

Once the system boots you have the choice of which run-level to select (there is a way to set a default level). This has been standardised much more than previously, with the following levels.

TABLE 1. UNIX Run Level Description

Level	Description
s	Single user mode
0	Halt/Power off
1	One user mode
2	Multiuser mode
3	Distributed mode
4	Unassigned
5	System diagnostics
6	Reboot

Obviously some modes are dependent on what is possible, for example level 0 will only power off where possible while level 5 will often only halt the system, leaving it up to the user to run diagnostics. Aside from the standardisation, SVr4 also introduces a new definition for level 3, *distributed mode.* In this mode various distributed file systems are imported and exported. File systems are imported by the use of the *mount* procedure (similar to BSD systems), while the export procedures are more complicated (see below).

>From the above table you will see that there are effectively two similar level, *s* and 1. The major differences are that level *s* invokes */sbin/sulogin* to login as *root* with nothing generally mounted, whereas level 1 is effectively a fully running system, but only accessible from the system console.

One final difference is that when the system changes to level 2 or 3 it also invokes the *service access facility* which controls general access to the system.

As with previous versions of System V, all these levels are controlled by the file */etc/inittab* and so these levels can be change, but it is not recommended.

4.4 Service Access Facility

With the varying accesses methods now available to the system, the previous methods such as *getty* were found to be wanting. To address the problems of no central control SVr4 introduces the *service access facility* and the related *port monitors.* The programs to control this are *sacadm* and *pmadm.*

A widely known example of a *port monitor* is *inetd,* which, although it hasn't been fully integrated, is run as one type of *port monitor.* Another, more familiar to System V users is the *listen* service, which is another *port monitor,* which has been fully integrated with the *service access facility.*

One related idea is the ability to disallow all logins if the file */etc/nologin* is present. This has been available under BSD systems but has only now been included in System V systems.

4.5 New Networking Facilities

As is widely known, the fully Internet and Berkeley networking facilities are now available, along with an implementation of the *socket* library. However, in general, these are not implemented under SVr4's *transport layer interface (TLI).*

For procedures using the TLI (which includes *remote procedure calls (RPC)*), access to various networks is control by the file */etc/netconfig.* This lists such details as the networking family (*e.g. inet*), the device to use, the access methods (*e.g.* connectionless, virtual circuit, *etc.*), and which shared libraries to use.

4.5.1 File System Export The control for exporting file systems is now handled by a number of new procedures. These include such procedures as *share,* all of which are driven by the script */etc/dfs/dfstab.* This is invoked by many programs including *mountall* and by *init* on change to run level *3.*

4.6 Network Printing

One feature in BSD systems that has long been requested has been remote printing. Along with many other changes to the SVr4 printing system the facility to do network prints is now available. This includes an emulation of BSD facilities and the ability to connect to BSD systems.

This is very simple to implement, and involves defining a remote systems using *lpsystem* and specifying those systems as the destination, rather than a device. To accept remote printer requests the SVr4 *listen* facility is used, with a request to also listen on the BSD printer port allowing connections from BSD systems. One point with this, for systems with multiple network connections, each need a separate entry.

One other problem at present is that there still appear to be many implementation problems with most implementations.

4.7 Backup Procedures

Another area that has undergone considerable reworking is the backup facilities. SVr4 implements a "unified" approach to many different backup procedures, from a file by file basis, to a full partition image, to a disk image. These are controlled by a set of routines that allow backups to be scheduled and run automatically or run under operator control. They also optionally keep detailed logs of the files backed up and have commands to move these to other locations. Further, there are now commands for users to request restores of their own files and directories, as well as the administrator to restore partitions or disks.

The command to run a registered backup is *backup*, which may then invoke *bkoper* to communicate with an operator and also invokes various other utilities to perform the backup. The programs to perform the physical backup are all located in */etc/bkup/method* and have specific procedures for communicating with *backup*. Unfortunately while the new programs are good (although they still have some bugs as would be expected in a new system), the backup formats are still the traditional ones, *cpio*, *volcopy* and *dd*. There is still nothing that can handle file deletion or recreate individual files that are missing blocks (as is found with many databases).

On a positive note, however, BSD's *dump* and *restore* are also supported (called *ufsdump* and *ufsrestore*), but only on *ufs* file systems.

4.8 Security

Although there has been an increased level of security awareness over the years, some of which is reflected in SVr4, there are still problems to be overcome.

SVr4 implements a shadow password file, password aging, account expiry and detailed password checking. However, many of the system changes increase the need to override these security procedures. For example the printer scheduler *lpsched* now runs as *root*, similarly because of the use of file system permission in the */proc* file system *ps* and related programs now require to be setuid *root*.

5. Conclusion

The advertising people claimed some years ago *System V - Consider it Standard*, this is now coming true. Most vendors are implementing versions of it and it is by far the dominant version in the market at present. It is even so standard that many of the bugs are common across all platforms, *e.g.* the **TZ** variable does not handle Australian daylight savings under UXP/M or Interactive.

Despite these initial problems, SVr4 is an exciting development, it provides most of the functionality that users have been asking for for some time and yet provides many new features and "goodies" to keep administrators busy for some time.

6. References

[1] UNIX System Laboratories (1990): *UNIX System V Release 4 Migration Guide*, Prentice Hall, ISBN 0-13-933821-7.

[2] Leffler, S.J., McKusick, M.K., Karels, M.J. and Quarterman, J.S. (1989): *The Design and Implementation of the 4.3BSD UNIX Operating System*, Addison-Wesley, ISBN 0-201-06196-1.

TCP / IP / ISDN

Hugh Irvine

Irvine & Associates Pty Ltd
29 Fitzgibbon Crescent
Caulfield 3161 Victoria

ABSTRACT

Telecom Australia's recent launch of their Macrolink and Microlink ISDN services offers greatly increased capacity and flexibility to users of digital communications links. These increases are accompanied by extremely attractive pricing policies, making possible new and innovative network configurations. Coupled with significantly enhanced functionality in both internetworking devices and sophisticated graphics terminals, ISDN enables network designers to provide much higher performance at much lower cost.

This paper will describe the Port of Melbourne Authority's state-of-the-art TCP / IP wide area network implemented using ISDN. The presentation will necessarily be a snapshot of a rapidly evolving technology. The initial network design will be shown, along with some of the "futures" that were taken into account in deciding how best to lay a suitable foundation for future growth. Comparisons will also be made in order to characterise the performance available to various types of network traffic over single and multiple ISDN "B" channels.

Hugh Irvine is an independent consultant specialising in Unix systems, large-scale TCP / IP networks, internetworks and X11 window environments. His assignment with the Port of Melbourne Authority was to design and install both a local and a wide area network of several hundred X11 window terminals.

1. Introduction

The Port of Melbourne Authority is a Victorian Statutory Authority. It is the landlord of the Port of Melbourne and Western Port, and it is responsible for the entire Victorian coastal area including all shipping and navigation. Early in 1990 the PMA began to implement an entirely new IT strategy. This strategy was based on three fundamental technologies: the Ingres relational database system, X11 windowing terminals and the UNIX™ operating system. This IT strategy also covered all of the PMA's operational centres, based at the World Trade Centre in central Melbourne, together with the docklands area, and also including regional offices in Bairnsdale, Hastings, Lakes Entrance, Western Port and Point Lonsdale (twenty-six locations altogether).

2. History

The Port of Melbourne Authority's IT resources evolution has been fairly typical of many large organisations. An ICL mainframe was used internally together with remote access to an IBM bureau, and of course, PC's and Macintosh's had proliferated.
Docklands locations and remote offices typically used ICL dumb terminals or PC's (or both), connected via 2400 baud dialup modems. These circuits very often were established in the morning and left nailed up all day! Needless to say, it was not unusual to find both an ICL terminal and a PC on the same desk, with two different modems and telephone lines!

3. Requirements

The PMA's requirements were two-fold. Firstly, to deploy the new applications developed under the IT strategic plan (X11 windows based terminals running Ingres Windows4GL applications). Secondly, to integrate all of the existing terminal and PC based connectivity into a seamless networking platform. Given that X11 windowing terminals were to be installed in remote locations, it was apparent that a TCP / IP network was required. And further, that significant bandwidth would be required to support the Ingres Windows4GL graphical applications. Just how much bandwidth would be required was ascertained by conducting subjective experiments on "typical" users in a test-rig environment. The test-rig consisted of three X11 window terminals connected to a small ethernet LAN that was itself connected to the computer room LAN via IP routers and back-to-back high speed modems. The modems were then clocked at 64kbs, 128kbs, 1024kbs and 2048kbs, and a series of tests were conducted at each link speed.
The tests consisted of typical operations such as downloading server code, logging into the system, starting a window manager and starting selected user applications.
The users in the experiment were also asked to give their subjective opinions as to the usability of the various configurations. Somewhat surprisingly, the "knee" of the subjective response curve occurred at a link speed of 128kbs, with link speeds lower than 128kbs being deemed unusable (even for a single terminal), and link speeds higher than 128kbs not making a significant subjective difference (even compared to a local ethernet).

*see Diagram 1

4. Cost Analysis

ISDN services are extremely attractively priced, and since the minimum bandwidth requirements for the wide area network were 128kbs, single ISDN Microlink services (2 x 64kbs "B" + 1 x 16kbs "D" channels) were selected for all remote locations. These Microlinks all connect to a single Macrolink service (30 x 64kbs "B" + 1 x 64kbs "D" channels) in the PMA computer room.
As will be seen from the following figures, ISDN 64kbs links are comparable in cost to 9.6kbs DDS links, and offer almost seven times the bandwidth.

*see Diagram 2.

5. Design

The design goals for the wide area network were to provide high-bandwidth TCP / IP connectivity for X11 window terminals, remote printers connected to terminal servers and local LAN connectivity for existing PC's and workstations. Further, a growth path had to be provided for incremental increases in bandwidth capacity to accommodate expected increases in network traffic as more and more terminal devices are installed. To limit the scope of the initial project, it was decided to install the wide area network in stages, with six sites (out of a total of twenty-six) being chosen for a "pilot" implementation. The six sites were divided equally into remote and metropolitan areas, with Bairnsdale, Hastings and Point Lonsdale as remote locations, and the Harbour Control Tower, Port Emergency Services and Central Workshops as metropolitan locations.

6. Implementation

There are essentially two options available for the implementation of an ISDN based wide area network. The options depend on how the ISDN "B" channels are aggregated into a single (at least logically) high speed channel. The aggregation can be accomplished in two fundamentally different fashions. Firstly, each ISDN "B" channel can be treated as a separate IP point-to-point link, with the IP routers at each end configured with the requisite number of serial ports, and some form of load-balancing being implemented in the router's gateway software. This approach will work with a limited number of connections, but quickly becomes prohibitively expensive and unwieldy especially at the central hub (imagine fifty or a hundred high speed serial ports in a rack of IP routers!).
The alternative is to handle channel aggregation as a separate function, independent of the IP routing function, and utilise purpose built hardware for the task. This is how the PMA's pilot network is implemented. A central IP router is configured with six high speed serial ports, each of which is connected to channel aggregation equipment in the ISDN primary rate interface device. In the current configuration, each remote site is connected via ISDN Microlink, hence two "B" channels are aggregated to form single 128kbs links between the central and remote IP routers. The remote sites are basically mirror images of the central site, except that individual ISDN terminal adapters (the only ones available at the time) are required for connection to the Microlink service.
Obviously, the reason for implementing the network in this way, is that additional bandwidth can be provided merely by adding Microlink connections to the channel aggregation equipment, nothing has to be changed at the IP level.

*see Diagram 3

7. Future Directions

There are two future issues that are of immediate concern to the Port of Melbourne Authority. The first, of course, is the cost of an implementation such as the one just described. It is evident that not all locations have a requirement for more than a single terminal device (plus a printer), and supplying terminal adapters, channel aggregation and IP routers is not justified in this configuration. To address this requirement, the PMA has entered into an R&D agreement with Labtam Australia to develop an ISDN Microlink direct-connect X11 window terminal. This project is well advanced, and a preliminary prototype device is currently being tested by the PMA. This terminal will include support for both ISDN "B" channels as point-to-point links together with IP load balancing for maximum performance. It should be noted that most X11 window terminals (including those from Labtam) feature serial ports that are accessible via TCP, for use as terminal or print servers. This is how the PMA will support all remote printers.

The second issue that the PMA is actively pursuing is a cleaner, more elegant interface between the central IP router and the ISDN channel aggregation and Primary Rate Interface (Macrolink or PRI). Once again, adding individual high speed serial ports to both devices for each additional remote location is not cost effective. Therefore, the PMA is currently supporting a development effort to provide time division multiplexed (TDM) interfaces on both the IP router side and on the PRI side, so that only one interface on each device will be required to accommodate up to thirty individual separate "B" channels for point-to-point IP links (this development is currently in progress between cisco Systems and Summit Communications).

8. Conclusions

The most interesting conclusion to be drawn from this project is that a new age of high speed, cost effective digital connectivity is truly upon us. Availing oneself of this level of data service is no more difficult than ringing up one's local Telecom Business Office and placing an order (this author's domestic ISDN Microlink service was installed within a fortnight of placing the order!). Significantly, this is an area in which commercial availability is preceding R&D by a long way. IP routing itself must be extended to cope with "virtual links" which can be set up and torn down, almost on a per-packet basis. Fortunately work is well advanced in these areas and products like domestic X11 window terminals are very close to commercial availability.

9. Acknowledgments

The author would like to thank the following organisations for their participation in this project:

Bull Information Systems
cisco Systems
Datacraft
Labtam Australia
Summit Communications
Telecom Australia

Diagram 1
Wide Area Network Tests Chart

	64kbs	128kbs	1024kbs	2048kbs	10mbs	2 x 64kbs ISDN
Download Server (1mb)	3:00	1:45	0:40	0:39	0:30	2:22
Download Server (0.6mb)	1:40	1:15	0:27	0:25	0:24	1:30
Start Windows 4GL	0:40	0:20	0:06	0:06	0:05	0:25

Diagram 2
Cost Comparison

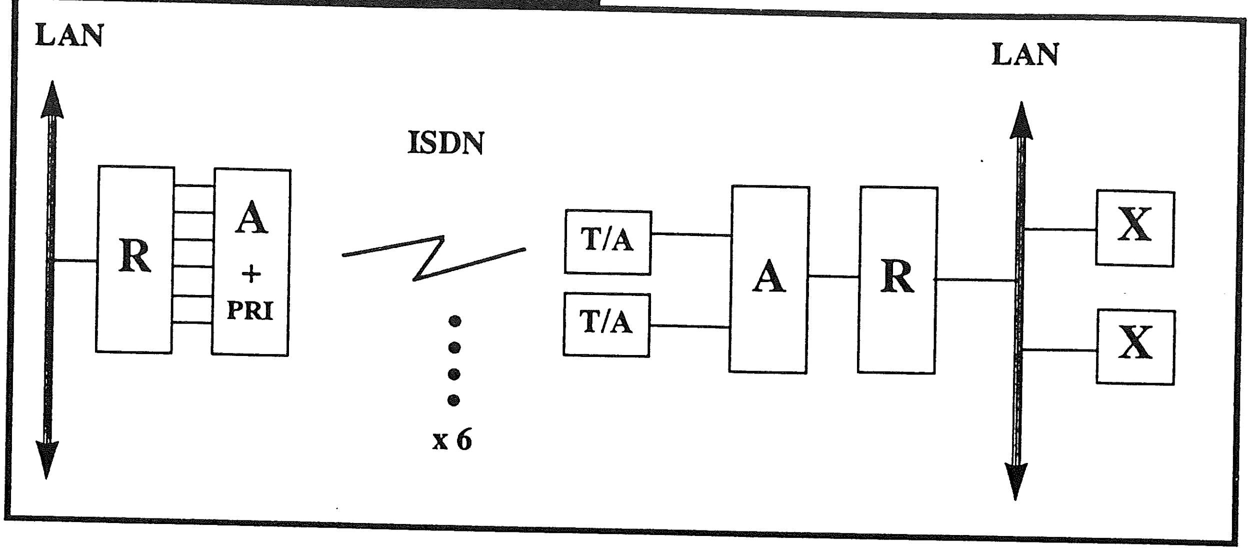
Metropolitan (< 12 km)

	ISDN 64kbs	DMS 9.6kbs
Installation	$360	$720
"	$360	$720
Access	$912	$2376
"	$912	$2376
Semi-permanent	$2028	
First Year Total	$4572	$6192
Recurring	$3852	$4752

Wide Area (Sydney - Melbourne)

	ISDN	DDS 9.6kbs	DDS 48kbs
Installation	$360	$765	$1510
"	$360	$765	$1510
Access	$912	$3540	$6612
"	$912	$3540	$6612
Semi-permanent	$16836	$9264	$22056
First Year Total	$19380	$17874	$38300
Recurring	$18660	$16344	$35280

Diagram 3
Functional Overview



R	=	IP Router
A	=	Channel Aggregation
X	=	X11 window terminal
T/A	=	Terminal Adapter

A Filesystem for a Multi Gigabyte Jukebox

Rex di Bona
Ray Loyzaga
The University of Sydney

ABSTRACT

Data Storage Devices are rapidly increasing in capacity. This has created problems in both backup and archival of data stored on these devices. With the advent of cheap, removable, optical storage devices, and jukebox control units it is now possible to have many gigabytes of archived or backed up data online.

1. The Disk Storage Problem

Each system administrator has to cope with the problem of long term disk storage. It is a well known problem that users accumulate, in fact almost aggregate, files. Each file so important that it could not be trusted to such a flimsy mechanism as tape.

Instant access to each of these files is also required. A user will desire instant access to any datum that they have collected, and to search through tape is a long and arduous process. This, while being perhaps slightly facetious is an accurate outline of a user who is not charged for their storage use. A further advantage is the reduction in human interaction to obtain tapes, and less wear and tear on tapes.

The Basser Department of Computer Science uses, at last count, 11 Machines with 15 Gigabytes of online storage for the academic users. Some of these machines are dedicated to specific projects, whilst others are available to the general academic populace. The overwhelming bulk of the data stored on the disks is archival, but *required for immediate use*. It must be stated that a lot of the disk, actually several gigabytes, is sources to various systems, including X11R4, which we currently use, and X11R5 which we are moving to.

We are trying to archive most, if not all, of the archivable material, but onto a medium more accessible than magnetic tape. To achieve this purpose the department used a Hewlett-Packard HP6300 Jukebox kindly donated by Hewlett-Packard for this project. The jukebox is a SCSI device that can hold up to 20Gb of data online as 32 disks each holding 600Mb, and two drives, each capable of reading and writing one side of a disk. Onto these disks the archival material will be stored, and older disks will be removed from the jukebox as the device fills up.

As a point of terminology each disk is thought of as containing two platters, an up platter, and a down platter. This is because to read the down platter the disk has to be physically flipped by the autochanger.

2. The File System

The jukebox presents an interface similar to that of two normal disks. Each disk drive is an independent unit, capable of reading and writing to a single platter. The platters are moved by an autochanger mechanism which is a separate SCSI device. The controlling machine is responsible for co-ordination between the drives and the autochanger.

Because the jukebox was to be a system wide resource, and because the Department has considerable experience with building user mode NFS filesystems it was decided that the jukebox would be presented to the kernels of the machines as a single NFS filesystem. The filesystem would appear to be the sum of the sizes of each of the platters. It was originally envisaged that there would be one jukebox wide filesystem, but for testing purposes the current implementation has a limitation that files cannot cross platter boundaries.

A more severe problem with attempting to have one filesystem span the entire jukebox is the problem of addressing. A thirty two bit pointer is insufficient to address individually each byte of storage on the jukebox. Either files would be limited to four gigabytes, or to the size of a platter, and the easiest solution was to limit files to the size of a platter.

2.1 An NFS Protocol Filesystem

As each of the machines in the department could mount NFS file systems, and the jukebox was to be a department wide resource, it was decide to make the jukebox controlling software implement the NFS protocol. The software to run the jukebox was also designed to be as system independent as possible, and to require minimal kernel changes to function. This led, along with the experience in the department with user mode NFS systems, to the jukebox controlling software being a user mode NFS server. The software is run as an ordinary user, and requires no kernel support for the NFS protocol interface.

This led to the software being easily debuggable by an ordinary debugger, and changes to the software did not require a kernel rebuild and machine reboot. To improve the ability to debug the initial software each platter has a self contained filesystem on it, a filesystem that the standard utilities, *fsck*, etc would work on. This meant that a large body of code didn't have to be written especially for the jukebox file system and improved the development cycle. As there was a real system supported filesystem on each platter the routines provided by the kernel for manipulating filesystems could be used to manipulate files on the individual platters.

2.1.1 The File Handle One draw back in using a standard filesystem and in using the kernel to manipulate the files is the problem of a file handle. In the NFS protocol files are identified by file handles, opaque 32 byte structures that can be passed to a file server to access a file, or directory. Because NFS is stateless a file handle must be unique across reboots, so some sort of internal data structure cannot be used as it would be lost across reboots. For kernel bound NFS servers this is not a problem as an inode can be used as the file handle. The kernel can manipulate files given just their inodes, but a user level process cannot, it can only manipulate open files or files specified by pathnames.

To solve this a filehandle was devised that could be used to map onto a filename in a one to one mapping. Each file handle would map onto a single file and would be constant across reboots. The file handle thus devised contained the platter identifier, the inode of the file (or directory), and the inode of the parent directory. Given these pieces of information the only system dependent routine was created, *getp()* took the file handle and returned the path to the file specified by that handle. It did this by reading the blocks associated with the parent directory inode, and storing the inode of its parent and the name of either the file or the child directory. Doing this recursively the algorithm would produce the full pathname for the file working from the file to the root inode of the platter. When the routine reached the root it would terminate. This 180 line routine needed to know the internal structure of directories on the platter, the only routine in the server that needed to know any internal structure of the file system.

2.1.2 NFS Problems Chosing NFS was not without its problems, the filehandle was a problem, and other, more interesting problems were found as development progressed. The NFS specification defines several structures as being opaque, their values were not to be interpreted by the client machine. It was found that the NFS reference implementation implemented by manufacturers did indeed look *inside* these structures and interpret the values therein. This caused some problems as our implementation didn't use these structures for the same purpose as the reference implementation.

One of the more intriguing problems was that the length of a directory was supposedly held inside an opaque structure, and was used on an end of directory message to represent the size of the directory. We left this value as zero on this message which resulted in zero length directories, and even more worrisome, each file in the directory would appear twice. This was because the client kernel would receive the first return message which contained all the entries and had the end of directory flag set. The client kernel would ignore the end of directory flag, request more information with the start pointer as the first entry, which resulted in a duplicate set of entries being sent.

For large directories we had the opaque pointer being a pointer to an internal data structure, so we ended up having directories being several megabytes in length. This problem was solved, but required more work to implement. As the supposedly opaque data structure was not so opaque a mapping has to be kept in the server with *offset in the directory* to *position in the data structure* information.

A more serious problem was the failure of the NFS clients to include user or mode information in create requests. A create request supposedly had two fields that contained the uid and mode to create the file or directory with, but these were sent as zero and 07777 respectively, resulting in publically writeable, set-uid root files, a grave security risk. The information could be obtained for the user from the credentials with the request, and it was determined that a mode of 07777 was the *don't set* mode, instead of -1 as the standard stated.

2.2 The Root Directory

The root directory is an imaginary directory. It holds an entry for each visible platter, and two special files, `ctl`, and `status`. An example is shown in Figure 1. This directory is created when the server is either started or reinitialised.

```
joyce # ls -lisa /jukebox
total 49
     2    1 drwxr-xr-x   2 root      root           512 Mar 11 20:08 .
     2    3 drwxr-xr-x  27 bin       bin           1536 Mar 11 03:00 ..
   100    1 drwxr-xr-x   2 root      root           512 Mar 11 20:08 0000000
   101    1 drwxr-xr-x   2 root      root           512 Mar 11 20:08 0000001
   102    1 drwxr-xr-x   2 root      root           512 Mar 11 20:08 0000002
   103    1 drwxr-xr-x   2 root      root           512 Mar 11 20:08 0000003
   104    1 drwxr-xr-x   2 root      root           512 Mar 11 20:08 0000004
   105    1 drwxr-xr-x   2 root      root           512 Mar 11 20:08 0000005
   106    1 drwxr-xr-x   2 root      root           512 Mar 11 20:08 0000006
   107    1 drwxr-xr-x   2 root      root           512 Mar 11 20:08 0000007
   108    1 drwxr-xr-x   2 root      root           512 Mar 11 20:08 0000008
   109    1 drwxr-xr-x   2 root      root           512 Mar 11 20:08 0000009
   110    1 drwxr-xr-x   2 root      root           512 Mar 11 20:08 0000010
   111    1 drwxr-xr-x   2 root      root           512 Mar 11 20:08 0000011
     3    0 --w-------   2 root      root             0 Mar 11 20:08 ctl
     4   33 -r--r--r--   2 root      root         16384 Mar 11 20:17 status
joyce #
```

Figure 1. A Root Directory

Each platter is identified by a unique platter number, given to it at platter creation time. When the server initialises this directory it firstly creates the `ctl` and `status` files, then it scans all the resident platters and obtains their platter numbers. This directory is unique in the jukebox server in as much as it is the only directory that cannot be modified, even by root. Any attempt to modify this directory, to chmod a file, to create a file, or remove a file, will result in a permission denied message. The date of the entries, apart from the status entry, do not change, and are the time when the server was started. The status entry changes continuously so that an attempt to read it results in an actual read, not the cached copy from the local NFS client.

2.2.1 Platter Directories Each platter has its own file system, mapped into the jukebox hierarchy starting at the platter number. To the NFS client it appears that a seamless transition is made between the root directory and the root of each platter, so doing a listing of the platter root directory doesn't result in any unusual inode numbers for the `.`, or `..` entries.

```
joyce # ls -lisa /jukebox/0000001
total 20
     101     1 drwxr-xr-x   2 root      root             512 Mar 11 20:08 .
       2     1 drwxr-xr-x   2 root      root             512 Mar 11 20:08 ..
2097156     0 -rw-------    1 root      root               0 Jan 29 15:01 0000001
2097155    17 drwxr-xr-x   2 root      root            8192 Jan 20 18:16 lost+found
2097280     1 drwxr-xr-x   3 root      root             512 Jan 28 15:05 pgrad
joyce #
```

Figure 2. A Root Directory on a Platter

The inode numbers returned by NFS are unique across a filesystem, so the real disk inode number could not be used as the returned inode number (properly called fileid in NFS terminology). Instead the inode number on each disk is prepended with a platter number set sufficiently high that duplicates will not occur. This results in extraordinarily large inode numbers as can be seen in Figure 2.

2.2.2 Control Files The two control files `ctl` and `status` are ASCII files similar to those implemented in the Plan 9 operating system. `Ctl` is a write only file that accepts control messages, and `status` is a read only file that describes, in a fixed format, the current status of the jukebox and platters. We shall present the actions of these files later.

The server will respond to a file system stat request by returning the sum of available and used data, which interestingly enough overflows the `df` command's calculations, resulting in interesting statistics on the jukebox as shown in Figure 3.

```
joyce # df /jukebox
Filesystem                Type  kbytes      use    avail %use  Mounted on
joyce:/jukebox             nfs -948358        0 -962068   0%  /jukebox
joyce #
```

Figure 3. Df output for the Jukebox

2.3 The Platters

Each platter is formatted and set out as a single filesystem. This was originally to allow for the use of the standard filesystem tools to interact with the platters, tools that would have to be rewritten if a different filesystem format was picked. It also allowed for the kernel routines to be used to access the data stored on a platter. This causes a problem however, it is now impossible using the current implementation to create a file that is greater than platter in size, and a file cannot span more than one platter. This results in the situation where even though there is free space on the jukebox a file cannot grow as its platter is full. We will discuss ways that this problem can be circumvented, but we still have a hard limit on file size of four gigabytes imposed by NFS.

2.3.1 Changes for a Single Filesystem There are two ways we can allow for files to grow larger than a single platter, we can either redefine the platter filesystem format to a format that allows a file to be spread over multiple platters or we can arrange for an additional layer of software, above the current jukebox software, to handle the problem, and arrange for a file to be split as necessary.

The idea of redefining the format of the filesystem was played with for a while, and may be implemented in future work, but would require the creation of the equivalent of newfs/mkfs, fsck, dump, restore, and a lot of the kernel routines. This is a worthwhile research project, but not necessary for our use.

Since the jukebox was to be used for an archive server it was noted that most of the data on the system would be static. Once on the jukebox it would neither grow nor shrink, but would just be read. If a data file was desired to grow the file would be moved from the jukebox back to fast disk, and manipulated there. When the file was unreferenced for a suitable amount of time it would be moved back to the jukebox, but not necessarily in its original position. Since we know the size of a file when it is being moved to the jukebox we can move it to the platter that has sufficient free room, or if necessary we can have the last file on a platter overlap onto a new platter. This would require only one file per platter extending across to a new platter, a simple enough special case.

This idea of only allowing one file to grow across a platter boundary has not been implemented currently, but will be incorporated in the version of software under development. This will allow files to grow to the maximum size allowed by the NFS protocol.

2.4 Conclusion

The filesystem presented above has been implemented and is currently running on a MIPS Magnum RC3330 computer. It uses 350Kb of virtual memory, including the caches that will be described shortly. It achieves a throughput of 300K reading, and about 100K writing, due to the speed of the optical disks. The source is just under 4500 lines of C code with that being 1600 lines of NFS interface code, 1200 lines of file handling code, 500 lines for both the filehandle and file caches 190 lines for the filesystem dependent code, and 1000 lines for both the jukebox autochanger control and the pseudo root directory with ctl and status files.

The system performs quickly, and can recover from most soft errors well. Its worse fall back is to do an automatic rescan of all the platters and reinitialise all its internal tables. This action is lengthy, taking about 15 seconds per platter, so for a fully populated jukebox it takes about 8 minutes to reinitialise.

The main problem encountered has been that the server code is single threaded. This means that operations that require a platter change hold up operations that do not need to be delayed. This is one area that needs more work, but for an archive device a delay of about 16 seconds for any file was deemed satisfactory performance.

3. The Caches

Since it takes a considerable amount of time to change platters it was imperative that caching of various things occur to improve performance. The three things that the jukebox server uses are, the mapping from filehandle to pathname, the *stat*'ing of a file, and the reading and writing of a file. All three of these were cached to improve performance. The main objective of each cache was to reduce, or eliminate the number of platter moves that occurred.

3.1 The Filehandle Cache

The filehandle cache keeps a cache of filehandles to pathnames. Each time a filehandle is created or looked up using the *getp()* routine the pathname that corresponds to that filehandle is stored along with the filehandle in a thousand element LRU cache. This cache is indexed by filehandle only.

3.2 The Stat Cache

The most common NFS operation is a file lookup. This takes a file handle for a directory, a file name for a file in that directory, and returns the filehandle for the specified file and a stat structure for the file. To allow for this the filehandle cache was extended to have the stat structure held along with each filehandle, and with each directory a list of the files in the directory was optionally held.

The list of files stores the name of each file and the inode number for that file. This allowed a lookup operation to operate entirely within the cache, as the first filehandle would be used to obtain the structure for the directory, the list of files would be traversed looking for the specified file and if not found an error returned. If found the inode number for the file was taken with the inode number for the directory and the platter id for the directory, both obtainable from the filehandle for the directory itself, and combined to form the filehandle for the file. This was then used to search the cache and if found the information returned. If the cache was hit both times the platter containing the files wouldn't be accessed at all.

3.3 The File Cache

To speed up read accesses to the jukebox a write through cache of files was created. This cache will cache up to the first three megabytes of a file on a fast disk. The cache is accessed on full jukebox pathname. The cache is write through in case of unexpected system shutdown or platter removal. The jukebox server software doesn't keep a list of files cached in the file cache, when a request comes in for a read or write the server will create the file if necessary.

There is a problem with sparse files, and files containing blocks of zeros in the cache. The NFS protocol allows out of order writes, so a file in the file cache may be sparse even though data exists in the holes in the file. To account for this problem the server checks each block read from a cached file. If a block reads as all zeros then the actual block from the real disk is read. This is because it is not possible to tell whether the block was all zeros or was a hole in the cached file.

The server only creates files in the file cache. A separate process is used to trim the size of the cache. It is run infrequently, and removes all files not accessed in a certain period. If the cache fills up then the performance of the server is impaired, but the correct data is still returned.

3.4 Rebooting

When rebooting the jukebox server machine it is preferred that both a shutdown and an unmount message be sent to the jukebox server. This ensures that all filesystems are consistent and that the current layout of the jukebox platters in their storage slots is saved into a configuration file.

When the jukebox server starts up it checks for a configuration file, and if present initialises itself from this file instead of from the jukebox itself. If on any access after that a discrepancy is found between the information in the configuration file and the actual jukebox layout the server will automatically reinitialise itself. For fully populated jukeboxes the configuration file reduces start up time drastically. If the server crashes unexpectedly, and the old configuration file still exists the old configuration file can be used for the jukebox server startup. This is possible as the jukebox server tries to keep platters in the same slots whenever possible which means that the configuration file stays accurate. This slightly increases the work that autochanger has to perform, but the increase is a marginal one, being a disk flip on occasions.

4. Kernel Modifications

The jukebox server is implemented on a RISC/os 4.52 kernel with additional SCSI commands for controlling the autochanger, and some additional IOCTLs for manipulating the platter header information. These changes were added to the SCSI driver and the kernel rebuilt. The additional commands for the autochanger were implemented as IOCTLs on the raw disk device. Most of the changes to the kernel were because the kernel assumed that all devices were either tapes or disks, and that all disks were fixed. As removable media become more prevalent the need for these types of changes to the kernel will be reduced.

One kernel change discarded as too costly was to modify the file type of archived files to a special archive type. This was discarded as it would require modifications to every kernel that would want to use the jukebox system. Whilst this method would give the nicest user interface the amount of work for system administrators would be prohibitive, and would delay the adoption of new operating systems at sites until the appropriate modifications were done. As the jukebox system currently stands only the server machine requires kernel modifications, all other machines access the jukebox through NFS, and the kernel changes are only due to deficiencies in the kernel which should be corrected soon (hopefully)!

4.1 Changes to the SCSI Driver

The changes to the SCSI driver included adding the SCSI commands to make the autochanger move, to obtain information about the number of disks, the layout of the disks, the layout of the disk drives in the jukebox. It is expected that these commands will be standard in kernels soon, as they are all part of the SCSI standard.

4.2 Additional IOCTLs

The IOCTLs added to the system were those necessary to implement the additional SCSI commands, and one to make the system reinitialise its volume information about a fixed disk. This IOCTL was required to correct the assumption that all disks were fixed disks and as such couldn't be removed, this resulted in the partition information not being read after a platter exchange.

5. The Control Interface

5.1 The Status File

Figure 4 shows the output from the status file. It is a fixed format ASCII file, and is available on any host that the jukebox is mounted on, which allows for control of the jukebox to be done from a remote host.

```
joyce # cat /jukebox/status
Jukebox Status
Transport Elements     : 1 (0 to 0)
Storage Elements       : 32 (11 to 42)
Input/Output Elements : 1 (10 to 10)
Data Transfer Elements: 2 (1 to 2)
Disk Drive 0 (SCSI 4L0) holds: Active: 0000001 (295086Kb, 281470Kb avail)
                              Inactive: 0000000
Disk Drive 1 (SCSI 5L0) empty
Storage Element   0 holds: Up: 0000001 (295086Kb, 281470Kb avail)
                         Down: 0000000
Storage Element   1 holds: Up: 0000002 (295086Kb, 295077Kb avail)
                         Down: 0000003 (295086Kb, 295077Kb avail)
Storage Element   2 holds: Up: 0000007 (295086Kb, 295077Kb avail)
                         Down: 0000006 (295086Kb, 295077Kb avail)
Storage Element   4 holds: Up: 0000008 (295086Kb, 295077Kb avail)
                         Down: 0000009 (295086Kb, 295077Kb avail)
Storage Element   5 holds: Up: 0000011 (295086Kb, 295075Kb avail)
                         Down: 0000010 (295086Kb, 295075Kb avail)
Storage Element   6 holds: Up: 0000004 (295086Kb, 295077Kb avail)
                         Down: 0000005 (295086Kb, 295077Kb avail)
joyce #
```

Figure 4. Example Output from the Status File

The platter `0000000` doesn't have associated with it the actual disk values for size of platter, and free space available. This is because that filesystem was not unmounted correctly due to a system crash and is still dirty. The jukebox software recognises that a filesystem is on that platter, but will not clean it until necessary. Manual intervention could be done by locking drive zero and manually `fsck`'ing the platter. The status file holds information about the entire system, number of drives, mailslots, storage slots, and autochanger arms. It also contains dynamic information, the platters that are mounted in each drive, the home storage spaces for each platter, and the space available on each platter.

5.2 The CTL File

The commands that the control file accepts are presented in Table 1. These commands are sent as ASCII strings, so the echo command can be used to control the jukebox's functions.

Command	Action
dump	Dump out the internal caches and structures (Used for debugging).
lock	Lock a drive and remove it from jukebox access.
noremove	Disallow front panel control of the jukebox. This is a security measure to ensure the software's idea of the state of the physical jukebox corresponds to reality.
remove	Allow front panel control of the jukebox.
rescan	Reinitialise the internal idea of where disks are.
shutdown	Write out the current configuration of the jukebox.
unlock	Allow a drive previously locked to be used by the jukebox again.
unmount	Remove the platters from the drives and return them to their storage spaces. This is used in preparation for an orderly shutdown of the system.

TABLE 1. Commands Accepted by the Jukebox Server

6. Other Uses for the Jukebox

The jukebox as shown is used as an archive server. It could however be used for other purposes. One possible use for the jukebox is as a backup device. Each night a script would run that made copies of changed files to the jukebox, and then during the day these changed files could be backed up from the jukebox to tape. The files also remain on the jukebox to provide a first level user accessible backup store. When Basser had spare disk space a copy of files were kept in an online backup directory called /backup for each device, and copies made in there each night. This proved remarkably effective with most backup requests able to be satisfied from the /backup directory.

7. Conclusions

The removable media jukebox provides an alternative to tape for archival of infrequently used data. It provides the random access patterns of regular disk, with a slight loss in speed, but provides the ability to have much greater quantities of data available. This paper has presented one method where this storage medium was used to provide archival of this infrequent data at low cost to the department. Overall the performance of the jukebox has been more than satisfactory, with data availability being much higher, and faster, than with tape storage.

An Update on UNIX–Related Standards Activities

Stephen R. Walli

Report Editor, USENIX Standards Watchdog Committee

What is POSIX.0 and why isn't it in ballot?

There is a lot of confusion surrounding the POSIX.0 working group. There are many different co-ordination problems in a set of related projects the size of POSIX.

- How will all the new system interface extensions like real-time, security and transparent file access fit into and onto POSIX1, the base interface standard?
- How will profiles layer on top of the base interfaces? Especially considering the current dynamic state of some of these documents and that no one knows how they completely integrate?
- How do test methods layer onto the base interfaces?
- How will language independent specifications map the C-based documents, and how are test methods affected?

One might presume that the POSIX.0 working group was addressing all of these thorny problems. They are not. Over time several ad hoc committees and steering committees have formed to address various symptoms of these problems.

POSIX.0 is building an overall guide book to open systems environments, the "Guide to Open Systems Environments". It is supposed to act as an informative guide to all of the issues involved in open systems, developing a model of how all these standards and technologies inter-relate. It will enable the design of a profile by an application systems analyst. The profile will be used for systems procurement, systems design and development. This would be a very useful document to have in existence. ISO is considering using POSIX.0's work as an ISO technical report.

Problems arise with the resources of the working group. POSIX.0 has long been labelled the "room with the suits." It is where management and strategic planners go when they attend POSIX meetings. These people feed back to their organisations how open systems technologies will be used. They help build and maintain the business case for the rest of our participation. Their goal in building the guide is a necessary one, and an essential part of the overall industry move to open systems.

They have recognised that their strengths are strategic and not technical. They have requested that other working groups provide the detailed technical resource that is needed to develop and review their respective parts of the guide. This is where the problems begin.

The other working groups are already stretched very thinly. They are busy working on the interfaces and profiles they desire to be standardized. They are also dealing with the added quality assurance requirements like language independent specifications and test methods. They have their own resource problems.

The working groups really are a group of fiefdoms, each addressing their area of technical experience and expertise, serving their part of the industry. While they generally appreciate the needs of the POSIX.0 working group, they haven't the bandwidth to service those needs. They're waiting for the "Guide" to go to ballot, knowing that they have to comment then or forever remain silent.

POSIX.0 has existed for almost four years. They are up to draft 14 of their document. They are only now trying to go to mock ballot. In any other IEEE standards body outside of the TCOS-SS sponsored POSIX family, people might begin questioning their existence.

POSIX.0 requested and received the support of the TCOS-SS Sponsor Executive Committee to force other working groups to attend the "architecture" co-ordination meetings held once a week during POSIX working groups. I was POSIX.4's stuckee.

I witnessed two generally prevailing attitudes in the October meeting: compliance by avoidance or compliance with complaint. Some working group members used the meeting to sit quietly getting caught up on other things, either reading they had brought with them, or meeting quietly with other people they had to co-ordinate with that week. Others were just upset that they were spending valuable time in a working group other than their own, commenting on a document that was not their concern.

Time is a priceless commodity at IEEE POSIX meetings. Key people within individual working groups, and the people with more global views of POSIX, are already stretched too thinly to participate in a development effort in another room.

As long as the balloting cycle remains for POSIX.0, I don't believe there will be active participation from outside of the POSIX.0 working group. Mock ballot is not real. The people that should comment on the document will still wait for real balloting to start. They are just too busy.

It is very difficult to submit one's hard work for comment and criticism by the anonymous ignominy of a balloting group. Often, balloting group members were not a part of the working group. They were not part of the hard discussions which shaped initial drafts of the document. They are not always polite and constructive in their criticisms.

Balloting, however, is not a "checkpoint" or milestone. Documents are not presented as completed technical or business reports, as they might be in our normal working environment. Balloting is an entire phase in the standards development process. The POSIX.0 guide will be shaped by the balloting process. This is a fact. One only has to look at the existing examples of POSIX.1, and POSIX.3 (Test Methodologies). The chairs of the POSIX.2 (Shell), POSIX.4 (Real-time), and the language bindings (Fortran and Ada) working groups have all seen their documents re-shaped in ballot. This has happened over a balloting period of up to a couple of years.

Attempting to perfect the document before ballot only delays the process of developing this necessary guide. The Guide is not something to be presented to a balloting group for approval. It is presented to have it critiqued, corrected, and shaped into the best possible, consensus built document it can be.

The POSIX.0 guide should be sent to ballot. Only then will it receive the genuine comment and constructive criticism it requires to become the educational guide the commercial world is looking for.

Report on POSIX.0: The Guide to Open Systems Environments

Kevin Lewis <klewis@gucci.enet.dec.com> reports on the October 21–25 meeting in Parsippany, NJ:

The October meeting for POSIX.0 was quite focused: get ready for the mock ballot. This required the group to pass through the document four times, each time with a deeper level of scrutiny. The level of consensus was quite high given that the group has worked diligently over the last year towards this goal. (Ya know, it's funny how a "line in the sand" can make a group of people come together . . .).

Two major changes were made to the ballot. The Software Development Environments section was moved to an appendix. The Fault Management section was merged into the section on Information System Management. The remaining changes were editorial.

The mock ballot will start in mid-November and end on January 7. Our overall objective is to begin the ballot resolution at the January meeting. We will also begin drawing up plans for the formal ballot, which right now is scheduled to begin in July.

There is still an issue dogging POSIX.0. People still presume that POSIX.0 is where one goes to find out how all of the POSIX.n standards fit together. We have determined and stated several times that it is *not* the place. The guide's purpose goes beyond the efforts of the current POSIX groups. It maps existing and emerging standards that are well outside the POSIX effort in such as way as to reflect the general requirements of a complete open system, with the POSIX standards playing a key role.

We do understand the need for this type of overview information and see why some people assume that POSIX.0 is the place to get it. The group intends to discuss this at the January meeting with the objective of pursuing some kind of resolution. Off the cuff, we felt that UniForum's "POSIX Explored" series may already be addressing this need.

Aside from this, the group is now holding its breath to see what kind of response it gets via the mock ballot. Stay tuned . . . .

Report on POSIX.2: Shell and Utilities

David Rowley <david@mks.com> reports on the October 21–25 meeting in Parsippany, NJ:

Summary

POSIX.2 (Shell and Utilities) Draft 11.2 closed its "changes-only" recirculation ballot on October 21. The draft received 80% approval, therefore POSIX.2 will not be recirculated until ISO comments have been received (sometime in May 1992). An IEEE standard is expected by late summer 1992. Draft 8 of POSIX.2a (UPE) is due out early in December.

POSIX.2b (see below) is underway, with the first draft distributed to the working group. The group continues to wrestle with 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. Work on POSIX.2a test methods is scheduled to start in January at the Irvine meeting.

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, including the new PAX archive format and the command changes required to provide an interface to the symbolic link work from POSIX1a.

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 11.2 ballot objections was completed with 80% approval. The final step to becoming an IEEE standard is to resolve comments from ISO. Draft 11.2 is also registered as ISO/IEC Committee Document (CD) 9945–2.2. International balloting continues, and ISO comments are expected at the May 1992 WG15 meeting. The Chair of POSIX.2, Hal Jespersen, will only produce a Draft 11.3 for recirculation at the IEEE level if significant ISO problems are encountered. Draft 12 would then be produced in June of 1992, with the standard approved at the next IEEE Standards Board meeting.

The IEEE is experimenting with electronic access to standards documents. The current drafts of POSIX.2, POSIX.0, and related documents are now available via ftp, courtesy of the IEEE and Andrew Hume at AT&T Bell Labs (andrew@research.att.com). Documents are available in both postscript and plain ASCII formats. From a reviewer's standpoint, it is extremely useful to be able to `grep` the drafts. This will also increase the number of people with access to these materials. I hope the IEEE will expand this trial.

POSIX.2a Status

Draft 7 was produced and recirculated, and the ballot closed August 19. Ballot resolution is complete, and Draft 8 should be available early in December.

Draft 8 will also be submitted to ISO as a Proposed Draft Amendment (PDAM) for eventual ballot as ISO 9945–2, Amendment 1. Expect the approval of POSIX.2a as a full-use standard anywhere from three to six months after POSIX.2.

POSIX.2b Status

Command interfaces to the POSIX.1a work will be a large component of the POSIX.2b work.

Dawn Burnett (USL) has volunteered to write up a proposal on modifications to the existing POSIX.2 command set to accommodate symbolic links. The proposal will be based on existing System V support.

David Korn (AT&T) suggested creation of a new **find** command, as an interface to the file-tree-walking services provided in POSIX.1a. The proposal was turned down due to lack of existing practice.

Eric Horner (Unisys) is preparing a proposal on adding `getsubopts`, since many commands have the need to process sub-options. The work will be based on the System V Release 4 implementation.

Karen Schafer, the Chair of POSIX.10 (Supercomputing Profile) and POSIX.15 (Supercomputing Batch Interfaces), has requested that a command interface to the new POSIX.1a checkpointing service be added. The POSIX.15 working group will submit a proposal.

The Chair mentioned that ISO may possibly request that the macro processor `m4` be added to POSIX.2b. Apparently there is a sizable demand from the ISO community. `m4` is specified in the X/Open Portability Guide.

Hal Jespersen has volunteered to Chair the POSIX.2b group, but has explicitly requested that he *not* chair the corresponding test methods work, due to the amount of personal time and effort required. Resources to perform this function will have to be found before test methods work can start. At the meeting, Hal Jespersen was reconfirmed as Chair of POSIX.2 and POSIX.2a. The term is for a period of three years. Many groups were asked to reconfirm their officers.

Project Management Committee Review

Both POSIX.2 and POSIX.2a were reviewed by the Project Management Committee (PMC) in October. Both projects were found to be meeting the stated objectives of their respective PARs.

New PAX Archive Format

The group continued to struggle with the new PAXarchive format. Mark Brown (IBM) completed revising the current proposal (now part of POSIX.2b), but the group is having difficulty reaching consensus on the scope of this effort.

ISO would like the new format to include some form of filename code set translation facility, and preferably a file content translation scheme. Some members want to go further, and add extensive pre- and post-processing services, including code set mapping, encryption, and compression.. Hal Jespersen will follow up with ISO to determine their exact requirements, but encourages the group to take on a leadership role in defining the specifications.

The general POSIX.2 philosophy of "divide and conquer", each utility performing one function simply and quickly, does not seem to be in effect here. The group is attempting to define code set mapping within an archive format, when there is no general-purpose mapping utility defined as part of the POSIX.2 effort.

X/Open includes the specifications for the **iconv** utility, as well as the **iconv**(), **iconvopen**() and **iconvclose**() system interfaces. Had POSIX.2 included these standard services for code set translation, creation of the new PAX format would be much less contentious. POSIX.2 is often referred to as a "command-line interface to system services". This new PAX format strays from that path (along with the internationalization utilities "locale" and "localedef"). Brand new functionality should not be created at the command level.

Test Methods

Tuesday to Thursday were spent writing test assertions in a joint meeting between POSIX.2 and POSIX.3.2. The situation has not improved since the previous meeting. Confusion continues to reign when writing assertions. The requested style guide did not materialize in Parsippany, but the group banded together to put the results of their guidelines discussion down on paper. This will allow the legacy of each meeting to be passed onto the next.

Work continued on reviewing test methods, including **bc**, **make**, Basic Regular Expressions, **awk** and the **localedef** grammar.

Either a "Mock Ballot" or a "Request for Comment" of the current POSIX.3.2 draft (Draft 7) will be distributed at the end of November. Comments are expected back by the end of February.

Writing test assertions for POSIX.2a will start in January 1992. Methodologies for writing interactive test methods (for testing **vi**, etc.) have yet to be developed. It is important some specific guidelines be set early on to provide the needed consistency. The challenges faced here will likely be the foundation for future test methods work, such as that required by the Graphical User Interface groups.

Report on 1003.3 POSIX Test Methods and Conformance

Andrew Twigger <att@root.co.uk> reports on the October 21–25 meeting in Parsippany, NJ:

SCCT Matters

Much of the Steering Committee on Conformance Testing (SCCT) discussion centered around the manner in which test methods for the various additive standards (to POSIX.1) could be combined into a single document. Many of the problems in combining test methods relate to the manner in which the additive standards use base functionality from POSIX.1 in a slightly modified manner. For example, the POSIX.4 (Real time extensions) draft standard makes copious reference to POSIX.1 functions and applies them to new file types. This means that many of the test assertions contained in POSIX.1 become applicable in the new context, while others remain applicable only to POSIX.1. Expressing this in a standard quickly results in the emergence of a complex matrix structure defining the applicability and testability of a particular assertion.

This problem is made worse by the failure of the various working groups within POSIX to clearly address the issues of interaction between the standards. The current philosophy seems to be that the first additive standard to be adopted won't have to worry about all of the others that are coming later. The problems of integration lie with those who are later in achieving adoption. This is an unsatisfactory state of affairs and there is need for a practical solution to the problem.

The SCCT is also faced with the issue of working groups wanting to quote subsets of the interfaces in POSIX.1. This has already started with POSIX.8 (Transparent file access) which is modifying previously mandated POSIX.1 behaviour. There were rumours circulating that one group wanted a subset consisting of only four POSIX.1 functions. This makes the task of the test methods writer much more difficult, since there are far fewer functions available to perform a test. If subsetting to this level becomes common practice, perhaps I will be able to get a conformance certificate for my watch!

A great deal of SCCT time was taken up in discussing what subsets were allowable and how many different options a supplier can choose from in providing a conforming system. Much of this problem relates to the manner in which POSIX.1 tried to deal with the differences in traditional behaviour and the variety of wording they used in the POSIX.1 standard to identify this. POSIX.3.1 (test methods for POSIX.1) has perpetually struggled with this terminology both in terms of the test assertions and the requirements that should be placed on a conformance document. I am sure that this discussion will continue for several meetings before any conclusion is drawn. I am unsure that many people are aware of the consequences of dividing standards into lots of optional pieces in a manner that becomes difficult for the users to understand.

The SCCT continues to struggle with ideas on testing profiles. The profile groups seem very eager to provide test methods for their profiles, but in many cases the manner in which they apply underlying standards is unclear. Indeed, in some cases the profile is being used as a tool to identify the gaps in the base standards rather than there being any real hope of the document being forwarded for ballot before these underlying standards are complete.

POSIX.3.1

The outstanding ballot objections to the POSIX.3.1 were reviewed with the balloters during the meeting. I believe that a concensus position was reached on practically all outstanding items. It is expected that a new, and hopefully final, balloting draft will be available in December with approval of the standard being targetted for early in 1992.

POSIX.3.2

POSIX.2 is the draft standard for shells and utilities, so it follows that POSIX.3.2 is the document containing related test methods. Most of the group concentrated on developing assertions for the remaining utilities with the aim of completing the majority of the document by the end of the meeting. Other parts of the group concentrated on reviewing work that had been done between meetings to align with recent drafts of POSIX.2.

Some discussion took place on the manner in which grammars could and should be tested. This discussion was not entirely fruitful, with a decision to leave the depth of grammar testing to the test suite author.

The group decided to formally conduct a review of the current test assertions, as it is believed that in many areas they are now relatively stable. While this was not considered to be a full mock ballot, it is expected to be a good preparatory stage towards that. The major items that remain outstanding are the application of Regular Expressions to the utilities and test methods to cover the Internationalisation effects on each command.

POSIX.4, POSIX.4a, POSIX.4b, POSIX.13: Real-time POSIX

Bill O. Gallmeister <bog@lynx.com> reports on the October, 1991 meeting in Parsippany, NJ:

Summary

The POSIX.4 working group met for four days this session. We approved our first draft of the extended POSIX.4b Real-time Proposal, and held a useful mock ballot of our current draft of POSIX.13 (Real-time Profiles). Work on a real-time networking interface, in conjunction with POSIX.12 (Transport Layer Interfaces) continued. (They're doing the work. We're answering the questions when we can). POSIX.4 was very close to going out for ballot recirculation, and has since been recirculated. POSIX.4a is still being readied for its recirculation.

Real-time Application Profiles

This week, we held a mini-mock ballot among the working group to get a feel for how close POSIX.13 is to being ready to ballot. POSIX.4 has produced four different profiles, matching different scales of real-time endeavor. [ed — These are described in earlier snitch reports.] One significant issue that must be addressed soon is the proposed subsetting of POSIX.1 that is required for the Embedded System application environment profile (AEP), the smallest real-time profile. This profile targets systems that may be unable to support the fork system call, due to the lack of a memory management unit (MMU). We have proposed a subsetting of POSIX.1 that would allow a system without fork to claim conformance to this Embedded profile, i.e. it would claim conformance to POSIX.13, not POSIX.1.

Is it POSIX, or is it UNIX?

The POSIX.1 working group has responded to this proposed subsetting with a resounding "no". In their opinion, POSIX.1 is the minimal set of functions that an application should always be able to take for granted when the system claims POSIX conformance in any way, shape, or form. Any subsetting, it seems, amounts to a diminution of the POSIX ideal. This may be fine in the case of a workstation, mainframe, or even PC-class machine. In the case of embedded computing systems, this position amounts to a statement that POSIX has nothing to offer these systems. POSIX.4 feels that this statement is wrong, and that we should take a pro-active stance in defining what POSIX conformance means when applied to such a small system, rather than letting this definition be done by the marketplace.

The next step in this procedure is for POSIX.4 to make a formal request for an extension to POSIX.1. Depending on POSIX.1's response, we will proceed from there. Stay tuned for our next exciting episode!

POSIX.4b First Draft Readied

POSIX.4b, the proposal for further real-time extensions to POSIX.1, POSIX.4, and POSIX.4a, had its first few chapters approved, and a first draft of this material should be available soon. The approved functions include a specification for *spawn* (a combined *fork* and *exec* which can be supported on MMU-less machines), and a chapter specifying timed-out interfaces for blocking POSIX calls.

Sockets

The POSIX.12 group continued its work on trying to accommodate real-time needs in the sockets facility. The issue du jour in Parsippany was whether *select* (or, alternatively, *poll*) provided sufficient functionality for asynchronous message passing, or whether *select* should be modified, or a new interface provided. The POSIX.4 working group was unable to come up with a clear direction at this meeting, and we will be revisiting the issue at our next meeting in beautiful Irvine.

Balloting Status

Many of the technical reviewers for POSIX.4 were present in Parsippany, and we were able to achieve the required 75% positive ballot response required for a recirculation. POSIX.4 has now been recirculated, so the technical reviewers will have some gripping reading material for the Holidays. POSIX.4a has been delayed a bit and has not yet gone to recirculation. As a matter of logistics, it is unlikely that POSIX.13 will go to ballot until the recirculations of POSIX.4 and POSIX.4a become more routine.

POSIX.12: Protocol Independent Interfaces

Tim Kirby <trk@cray.com> reports on the October 21–25 meeting in Parsippany, NJ:

Summary

The ongoing mission: two interfaces are proposed in language independent form —a Simple Network Interface (SNI) and a Detailed Network Interface (DNI). SNI is a proposal drawn from several sources with no (de-facto) standard. DNI is seen as a single language independent binding to which there are two valid C language bindings, sockets and XTI.

The POSIX.12 group took a hit this time in terms of attendance, with at least a couple of the members struggling to maintain funding/support for future attendance. One of these is the chair, Les Wibberley, who may not be around after the April meeting if he is unable to find funding to help with travel expenses to the tune of around US$3,000 per annum. If the funds are not found, we will be looking for a new chair to take over following the April meeting. Anyone who can either help or would be interested in the chairmanship of the group (corporate support is required), please contact Les at the following address:

Chemical Abstracts Service
2540 Olentangy River Rd.
Columbus, Ohio 43210
<lhw25@cas.org>, <lhw25@cas.bitnet>

Report

This report will sound very similar to the previous (July meeting) report as discussion of many of the same issues continued from the previous meeting cycle.

Once again, liaison with other groups absorbed a significant amount of the week. These groups were POSIX.4 (Real-time), POSIX.11 (Transaction Processing), and POSIX.17 (Directory Services). It has been proposed that POSIX.12 will encompass all the functionality required by POSIX.4 and POSIX.11. After careful study, the group's consensus was that one or two of the requirements would be well placed in the POSIX.12 standard, but those remaining (the majority) should be implemented above the POSIX.12 interface level. The scope of our project will be examined to ensure that any proposed standard is consistent with that scope.

The POSIX.17 (Directory Services) liaison meeting was productive and resulted in several action items to be worked on before the next cycle to synchronize the two interfaces.

Multicast support, as required for lightweight stacks such as XTP, is also receiving careful attention to ensure the proposed interface standards will support such functionality.

The long outstanding issue of the inadequacies of the `listio()` functions with regard to the networking interfaces was brought to some resolution. There is a requirement for some form of `select()` or `poll()` functionality that does not appear anywhere in the current POSIX interface arena. The Systems Interface Coordination Committee (SICC ... also known as the Steering Committee on Systems Interfaces, or SCSI) worked on this item, eventually passing it to the POSIX.1 working group. There's an action on POSIX.12 to provide a detailed specification of the requirement. SICC is a newly formed ad hoc committee of the base interface group chairs, whose function is to address some of the immediate coordination concerns between groups.

Other items worked on during the week were the glossary for the POSIX.12 standard, test assertions, and detailed discussions on the generation of Language Independent specifications for both DNI and SNI.

A significant number of actions were assigned for between-cycle homework in most of the areas mentioned. The current plan for POSIX.12 has a mock ballot sometime in the second quarter of 1992, with a real ballot following around the fourth quarter. Given the additional work items picked up from POSIX.4, POSIX.11, and the Multicast extensions, there is some project reviewing and possibly some reworking of proposed dates to come. Any additional support is most welcome from willing volunteers!

Report on POSIX.17 - Directory Services API

Mark Hazzard <markh@rsvl.unisys.com> reports on the October 21–25, 1991 meeting in Parsippany, New Jersey:

Summary

As promised, Draft 2.0 of POSIX.17 (Directory Services API) went to Mock Ballot before the Parsippany meeting. The group made solid progress processing input from the Mock Ballot. We also identified work items and began planning for an official IEEE ballot which is slated for 2nd quarter 1992. Significant work remains in the area of test assertions and general document formatting.

Introduction

The POSIX.17 group is generating a user to directory API (e.g., an API to an X.500 Directory User Agent). We are using XAPIA – X/Open's Directory Services specification (XDS) as a basis for work. XDS is an object oriented interface and requires a companion document, X/Open's Object Management specification (XOM) for object management.

XOM is a stand-alone specification with general applicability beyond the API to directory services. It will also be used by IEEE P1224.1 (X.400 API), and possibly other POSIX groups, and is being standardized by P1224.

Status

Only four of the six "core" working group members who attended the Santa Clara meeting showed up in Parsippany. Despite assurances from the missing members to be there next time, participation remains an issue. Due to the small number of committee members, there's a lot of work for everybody, especially in the next 3–4 months as we get ready for IEEE Ballot. Ours was one of several groups hit by the lingering recession. I guess standards need to be completed during prosperous years.

The Mock Ballot of POSIX.17 Draft 2.0 was completed between meeting cycles with comments received from less than 10% of the ballot group. We spent most of the meeting processing those comments.

Significant editorial changes will be required to resolve the comments, including separating and clearly labeling normative vs. non-normative text, adopting a stricter, more legalistic nomenclature and style for the normative sections and generally tightening up the draft. Several technical changes were suggested and these will be processed between meetings.

The Technical Editor accepted an action to complete the Language Independent Specification (LIS) and test assertions by the next meeting. Our plan is to complete outstanding work items and begin IEEE Ballot on April 1, 1992 with a complete draft.

Once again, we met with POSIX.12 (Protocol Independent Interfaces) in joint session and discussed their requirements for directory services. The POSIX.12 group produced an updated version of their directory requirements paper which was briefly discussed. We also addressed Mock Ballot comments submitted by the POSIX.12 chair, which by and large reflected the concerns of his group.

There was general agreement that the POSIX.17 API provided sufficient functionality and, in any case, POSIX.12 would be using only a small subset of the functions (primarily read and search). The main issues seem to be the complexity of the API, how the directory information tree (DIT) gets extended and the mechanics of how new (POSIX.12 required) objects get defined.

It was agreed that we would continue to refine the POSIX.12 requirements, using their white paper as a basis. We decided that a "small" group consisting of members of POSIX.12 and POSIX.17 should meet regularly to resolve outstanding issues.

POSIX.17 and P1224 met again in joint session to continue to review/revise the XOM specification. Many corrections were made, and a new draft will be released before the next meeting. P1224 is scheduled to go to IEEE Ballot in the first quarter of 1992. Since P1224 is a normative reference for POSIX.17, a stable version is essential for our ballot.

The group met with a consulting member of POSIX.3 (Test Methods) to review our test assertions to date. The feedback we received was positive from the perspective that, if anything, we had gone overboard on writing assertions, and that less effort than planned would be required to complete them. We were also informed that X/Open had agreed to fund our technical editor to write the assertions for POSIX.17.

The POSIX.0 group (POSIX Guide) held a review of the "networking" sections of their latest draft (D13). All the distributed services groups attended and expressed their concerns, some claiming that their input to POSIX.0 had been largely ignored. It was noted that the POSIX.0 model is seriously flawed in that it doesn't account for systems integrators and recognize their needs. This message was acknowledged and will hopefully be accommodated in subsequent versions of the POSIX.0 draft.

In Closing . . .

There are quite a few homework assignments between meetings, including forming the ballot group, rewriting/reformatting sections of the specification and responding formally to our Mock Ballot comments.

The group will reconvene January 13–17, 1992 at the IEEE POSIX meeting in Irvine, CA.

ANSI X3B11.1: WORM File Systems

Andrew Hume <andrew@research.att.com> reports on the October 21–25, 1991 meeting in Merrimack, New Hampshire and on the TC15 meeting on November 11–15, 1991 in Geneva, Switzerland.

Introduction

X3B11.1 is working on a standard for file interchange on random access optical media: a portable file system for WORMs or rewritable optical disks. TC15 is a committee within ECMA that works on file system standards. This report covers the last X3B11.1 meeting in Merrimack, New Hampshire and the last TC15 meeting in Geneva, Switzerland. The report describes the cumulative effects of both meetings, mainly as a set of changes. In brief, we are conducting a letter ballot on whether to forward the current draft on for processing as an ANSI standard, and simultaneously, the draft is undergoing editing for processing as an ECMA and eventually, as an ISO standard.

The Current Draft Working Paper

The draft is now essentially three distinct standards, more or less, embedded within a single document.

The first standard describes a generic scheme, upward compatible with ISO 9660, for recognising boot records, and which standard describes how the volume has been recorded. This allows media to be recorded with boot blocks for several different system architectures and different operatings systems. The goal is that a generic boot ROM could search for all the boot blocks for the current system, somehow ask the user for which one to use, and then load a boot block into memory and execute it.

The second standard is a volume structure standard. It provides mechanisms for labelling volumes and volume sets, managing unallocated space on a volume basis, describing partitions and a label of the partition's contents, and an optional scheme for bad sector management.

The third standard is a file system structure, corresponding fairly closely to a UNIX file system augmented by symbolic links and extended attributes.

Boot/Volume Recognition

This part was negotiated at the TC15 meeting and is identical with the corresponding part of the CDWO draft. In case you don't know what CDWO is, it is a WORM disk based on the CDROM technology. Under certain recording modes, a CDWO disk can be read on CDROM drives. On the whole, it seems a painful device to program for, but the dream is that the CDWO drives can be made quite cheaply and they can be used for reading ISO9660 disks and writing CDWO disks (say, for backup). The CDWO folks are also known as the Frankfurt group. (Why are standards groups known after the city where they first meet?)

The TC15 proposal says basically, at sector 16 there is an optional sequence of ISO9660 descriptors terminated by the ISO9660 terminator descriptor. Then follows an arbitrary (including zero) number of extended descriptor sequences, each begun and terminated with specific descriptors. The contents of these sequences are either generic boot descriptors (common amongst standards) or standard-specific descriptors. We defined one such descriptor; it simply points to our normal structures. The boot descriptors may well be the most useful part of this standard, particularly if vendors adopt it. The boot descriptors are quite flexible. They describe an extent of sectors on the media. They contain:

- a 32 byte architecture identifier,
- a 32 byte operating system identifier,
- a 64 bit load address for the extent and
- a 64 bit start address.

Volume/Partition Structure

This part is largely unchanged since my last report. We added a new kind of partition, a catenation of other partitions, in order to support hybrid media which is a mixture of different media types, say read only and rewritable. The other changes mainly have to do with simplifying things for smaller implementations, such as on memory-limited PCs. A new level of interchange has been added for low end implementations with some simplifying restrictions on recording volume descriptors (for example, a single extent at a fixed location).

File System

This part has changed in a lot of small details, mainly for harmonization with the CDWO draft. In a valiant attempt to reduce the number of registration authorities, we dropped a bunch of somewhat dubious extended attributes and systematised a general approach for registerable properties, such as file system type or extended attribute type. In general, these are 32 bit numbers and some authority, such as ECMA or ANSI, manages a registry mapping numbers to textual descriptions. Our registration type is a 32 bit number and a 32 byte field. (It happens that often you need some data to go with the registration number). Furthermore, a number of zero means that there is no official meaning but that the 32 byte field may specify the meaning in an implementation-defined way. This allows folks to implement and experiment before going through the bother of registering.

International Activity

The TC15 meeting was surprisingly effective and we made many useful changes. It was particularly rewarding (for me) to finally meet with the representative from Fujitsu, which has been quite active and vocal (and constructive) with its comments on our draft.

There is much emphasis on producing standards in a timely fashion within TC15. The current schedule, for both our draft and the CDWO draft, is that all technical details be fixed by the January meeting and that the January and March meetings be editorial in nature. We hope/expect that the drafts will be voted on and adopted as ECMA standards at the June 1992 ECMA General Assembly meeting. They then would start on the ISO "fast track" process and assuming no negative votes at the ISO level, would become an ISO standard around March or April 1993. (This is about 6 to 8 months before the draft could become an ANSI standard, assuming best case for the ANSI process.)

Because of TC15's preoccupation with its two drafts, work on the general storage architecture model has been postponed.

Electronic Distribution of Standards/ Drafts

Since I became technical editor of X3B11.1, my drafts have been available electronically by both *ftp* and email (*netlib*) from `research.att.com`. (For *ftp*, login as *netlib*.) For details, get `index` from `research/memo`.

This has been extended to various IEEE standards as an experiment with the permission and blessing of the IEEE Standards Board, resources provided by AT&T, and with the cooperation of the various editors involved, most notably Hal Jespersen. Currently, these include drafts, meeting notices and meetings for the POSIX committees P1003.2, P1003.2b, P1003.0 and IEEE 896 (Futurebus+). Drafts are available in PostScript or ASCII, compressed or not, and as page bundles (pages 101–120) or discrete sections (3.4.2). (For further details, get `index` from `posix` and/or `basc`.) This has proven to be very popular; without wide advertising there have been over 7000 requests for POSIX stuff in the first two months.

Other happenings

The IEEE is starting up a new committee on Hypermedia and related topics. One of the initial project requests (PARs) for this committee is (more or less) making the Rock Ridge proposal a standard. This strikes me as puzzling but I hope to find out why when they start having meetings.

The Rock Ridge proposal is an attempt to make the CDROM standard, ISO9660, which is by and large designed for use on VMS and MS-DOS, useful for recording UNIX file systems. It does so by recording the stuff ISO9660 left out in certain implementation use areas. I am yet unclear why this should be an IEEE standard when the CDWO draft provides a strict superset of the Rock Ridge functionality and the CDWO will probably be an ECMA standard. It maybe will even be a ISO standard before the Rock Ridge proposal could become an IEEE standard. (To say nothing of the apparent stupidity in having two standards, one of which is a subset of the other.)

Admittedly, there is a fine distinction between the two proposals. The Rock Ridge format is pure ISO9660; implementations just have to be changed to recognise and act on certain implementation use descriptors. The CDWO proposal adds new descriptors in an area just after where ISO9660 stops looking for descriptors — implementations have to be changed to look further for, and recognise, these new descriptors. I, and I suspect most users, think this distinction is pretty much meaningless; both proposals can be read (at least the ISO9660 information) by ISO9660 systems, and ISO9660 disks can be read.

The difference in implementation effort for either proposal seems small, particularly when you consider most of the work goes into the file system and kernel interface code. The two groups, Rock Ridge and Frankfurt, have agreed that their work occupies different niches and therefore justifies two separate standards, although I suspect this is a post-hoc rationalisation more to do with marketing and economic reasons rather than technical or user-related reasons. With any luck, the committee may decide that the Rock Ridge needs are sufficently met by the CDWO proposal.

At COMDEX, IBM made it known that it will be announcing the general release of an optical product in late 1991 or very early 1992 and this would include a volume and file system format. (This has been in field test for some time now.) I regret that although IBM was represented at the first few X3B11.1 meetings, they stopped coming and the IBM format is unrelated to, and uncoordinated with, the X3B11.1 draft.

Finale

It seems likely that something very close to the current draft will become an ISO standard for (essentially all) random access media. Even if you don't care about the file system aspect, you may (should?) be interested in the proposed booting and volume recognition scheme. It would be quite worthwhile for there to exist a generic way to boot off media, but it will only happen if it is useful and meets most vendor's needs. I would be very interested in feedback on this aspect of the draft.

If you wish to comment on the draft, get a copy electronically, or contact me or the X3B11.1 chair (Ed Beshore) for a copy. If you work for a company represented on the committee, comments ought to be funneled through that representative, but in any case, I will collect and present any comments sent to me. Comments would need to be received by me prior to December 27, 1991.

If you would like more details on X3B11's work, you should contact either me (andrew@research.att.com, 908-582-6262) or the committee chair, Ed Beshore (edb@hpgrla.hp.com, 303-350-4826). The critical document is the current draft of the working paper (about 80 pages). There is also a programmer's guide to the draft (about 18 pages written by me). I will send you copies of the latter document; requests for other documents or more general inquiries about X3B11's work would be best sent to Ed Beshore.

The next meeting is in Santa Clara, CA on January 6 – 10, 1992 and will address the ballots from the letter ballot. Anyone interested in attending should contact either me or Ed Beshore.

Report on X3J16: C++

Mike Vilot <mjv@objects.mv.com> reports on the November, 1991 meeting in Dallas, Texas:

Current Status

The ANSI X3J16 / ISO WG21 committee continued refining some of the important conceptual details of the C++ language. During this meeting, the committee made decisions on issues that had been discussed since the March meeting in Nashua, including name lookup semantics and translation limits.

November meeting

Texas Instruments hosted the meeting in Dallas. The week's major activities focused on the informal working group meetings. Many members had been unable to attend the June meeting in Sweden, so much of the effort went into regaining momentum lost since the March meeting.

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. Steve is the Convener of ISO WG21

During the summer, X3J16 conducted a letter ballot and voted in favor of converting to a Type I standards project. Steve reported that the necessary administrative matters to coordinate the work of X3J16 and WG21 had been completed.

He also relayed the requests of SC22 (WG21's parent organization) to consider certain topics important to the international community:

- support for international character handling
- support for language independent data types and arithmetic operations
- a portability annex in the eventual standard

Editorial

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

Much of the recent work on the document has been in clarifying or defining basic terms. The process of resolving the definitions of the two base documents continues. (These are the Annotated C++ Reference Manual (ARM) and the C standard.) For example, the C standard uses the term "compatible type", while the ARM uses the phrase "the same type". The editorial changes involve using one term or the other consistently throughout the document.

One minor change clarified that enumerations are distinct types, even though enumeration values still promote to int.

Formal Syntax

Jim Roskind, of Roskind Software, presented the work of the Formal Syntax group.

The group revisited issues it had raised at the March meeting, regarding changes to the delimiters used for template argument lists, and minor changes to the grammar involving *throw-expressions*.

The discussion on replacing the '<' and '>' delimiters in the template syntax (in favor of '(' and ')' or some other pair of "matching" tokens) uncovered no new issues. Document 91–0033 presents the full discussion of this issue. Since the parsing difficulties could be addressed with a slightly more complex grammar, and no alternative pair of tokens was clearly preferable, the final vote was to lay the issue to rest with no change in the syntax.

The discussion of the grammar involving *throw-expressions* resulted in a minor change to the language. Documents 91–0013, 91–0034, and 91–0048 discuss the details.

One other minor change involved the wording of the description of template type arguments. Document 91–0061 describes the issue.

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.

After much discussion (some of it heated), the group arrived at what they felt was an acceptable specification of name lookup semantics. A brief description of the proposed rule is that the scope of a name declared in a class consists not only of the text following the name's declarator (to the end of the class), but also all of the function bodies and constructor intializer in the class.

The Core group will have a precise description in writing for the next meeting.

Another issue the group addressed (but did not resolve) was the specification of the lifetime of implementation-generated temporary objects. At the March meeting, the group had considered the effects of early (end of expression) and late (end of enclosing block) destruction of such temporaries. Consensus seems to be forming on a middle ground, but requires a precise definition of its semantics.

Bjarne Stroustrup pointed out that the emerging position is roughly that "temporaries persist to the end of the enclosing basic block." The group still has to resolve some details, because the proposed semantics do not exactly match the usual definition of basic blocks.

Environment

Peter Chapin, of Vermont Technical College, presented the work of the Environment group.

The major topic of discussion was on what translation limits the eventual standard should define. Although there were some very strong arguments against specifying limits (the strongest being that such numbers turn out to be upper limits in too many "conforming" implementations), the committee decided in favor of specifying such limits. The environment group will propose specific limits at the next meeting.

The group also continued refining the semantics of static object initialization. Their current proposal is to introduce two types of translation units, to distinguish between objects in dynamic link libraries and objects that can be initialized before entering the `main()` function.

Libraries

I presented the Library group's work.

The most important result of the week was to scrap the existing proposal for string classes (document 91–0078) and start over. Uwe Steinmueller of Siemens-Nixdorf contributed his experience in implementing string classes to the effort. The group will have a new proposal for the next meeting.

Work progressed on standard exceptions, based on Jerry Schwarz's proposal (document 91–0116). The group proposed changing the default behavior of the `newhandler` to throw an `xalloc` exception.

We refined the iostreams proposal (document 91–0117). The group discussed several minor issues in the proposal:

- national character set streams
- file open modes
- wide character support
- interaction with other standards (e.g. ASN.1)
- names of header files
- specification of `streampos`, `streamoff` "types"
- exceptions thrown by `streambuf`
- specification of mode flag "types"

Notably, most of the issues involved accomodating International Concerns over character handling. The group will have a revised proposal at the next meeting.

The new item for the Library group involved proposals for standard "container" classes. Based on Chuck Allison's survey of existing libraries (document 91–0111), the group found a half-dozen abstractions recurring in each library (vector, list, queue, stack, bitset, hash table).

The basic design approach for these container classes will be template classes specifying concrete data types. That is, no elaborate Smalltalk-like Collection hierarchy, ultimately rooted at class Object. Members of the group with experience implementing the "easy" classes volunteered to write up proposals for the next meeting (specifically, bitset and vector).

The Library group will also investigate language independent data types, and arithmetic operations. For example, a standard class `complex` could be included in the library, rather than requiring a new built in data type.

Language Extensions

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

The group proposed an explicit procedure for considering and deliberating on submitted extensions. A recent series of postings to `comp.lang.c++` by Jim Adcock of Microsoft indicated his dissatisfaction with the "openness" of X3J16's work. Sensitive to the experience of the C committee (where a J. Hansberry invoked the formal procedures of ANSI to delay the publication of that standard by over a year), the Extensions group is going out of its way to give an unbiased hearing of every proposal submitted.

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

- support for run-time type information
- 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 the dot operator
- a name space control mechanism
- keyword parameters (like Ada's)
- programmer-defined operators new and delete for arrays
- new keyword `const`
- constrained genericity in template type arguments
- extensions to control the order of static object initialization

Every proposal will have at least one person (not the original author) examine the proposal and investigate its impacts on the existing language. Once the working group has reached a consensus on whether to recommend accepting or rejecting the proposal, it will bring the issue to the full X3J16 for a discussion and formal vote.

The group is working on a written guide to preparing language extension proposals, to encourage proposals which are complete enough to receive an adequate discussion.

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 Shopiro to decide which terms can be incorporated as C++ definitions.

Next events

The next three X3J16 meetings (and their hosts) will be:

- March 16–20 1992, London, UK (BSI and Zortech/Symantec)
- July 12–17 Toronto Canada (IBM)
- November 8–13, Boston MA (OSF)

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@cup.hp.com

Vice Chair: Stephen D. Clamage
TauMetric Corporation
8765 Fletcher Pkwy, Suite 301
La Mesa, CA 91942
(619)697–7607
FAX (619)697–1140
email steve@taumet.com

AUUG
MANAGEMENT COMMITTEE
MINUTES OF MEETING 20 JANUARY 1992

Held at ACMS, Paddington.

Present: Pat Duffy, Frank Crawford, Jagoda Crawford, Scott Merrilees, Chris Maltby, Glenn Huxtable,

Meeting commenced at 11:15am.

1 APOLOGIES

Andrew Gollan, Rolf Jester, Peter Karr.

2 MINUTES OF LAST MEETING (9 DECEMBER 1991)

Correction to item 12.3: "AUUG'91" should read "AUUG'92".
Moved (CM/FC) that the minutes be accepted as amended. Carried.

3 BUSINESS ARISING FROM PREVIOUS MINUTES

4.2 Completed.
4.3 Carried over.
4.4 Carried over.
4.5 Carried over.
5.3 Carried over.
5.4 Carried over - next year's budget
5.5 Carried over.
7.3 Completed.
7.4 In progress.
8.3 Completed.
8.4 In progress.
9.4 Awaiting clarification.
10.3 Carried over.
12.3 Completed, but Stephen prince is not responsible for AUUG'92 Programme.
13.1 Carried over.
14.1 Completed.

4 AUUG'92

Ian Hoyle has volunteered to join Robert Elz and Peter Karr on the AUUG'92 Programme Committee.

Wael Foda of ACMS joined the discussion regarding AUUG'92 and the Summer Conferences at 12:10 and left at 13:00.

5 CANBERRA CHAPTER

Moved (FC/PD) that we accept the Canberra Open Systems Users Group (COSUG) as an AUUG chapter and invite them to send appropriate members of COSUG to the next AUUG Executive meeting to discuss details. Carried.

6 PART-TIME CO-ORDINATOR/ADMINISTRATOR

It was agreed to hire Liz Trauman as a part-time co-ordinator/administrator to assist us with co-ordinating ACMS, Symmetry, AUUG'92 etc. The job is expected to be about two days per week at $30 per hour. Liz Trauman attended the meeting from 14:00. Frank Crawford will send letter of offer.

7 PR/SYMMETRY

There was discussion of Symmetry's duties.

Ellen Gubbins of Symmetry joined the meeting at 15:00. The proposed 1992 PR campaign was discussed. Ellen will send a revised quote by fax.

8 NEXT MEETING: FRIDAY MARCH 27, 10:00 AM AT ACMS.

The meeting concluded at 16:30

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 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 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

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
P O Box 366
Kensington NSW 2033
Australia

Please allow at least 4 weeks for the change of address to take effect.

Old address (or attach a mailing label)

Name:	....	Phone:	.... (bh)
Address:	....		.... (ah)
	....	Net Address:	....
	....		
	....		
	....		

New address (leave unaltered details blank)

Name:	....	Phone:	.... (bh)
Address:	....		.... (ah)
	....	Net Address:	....
	....		
	....		
	....		

Office use only:

Date: ___/___/___

Who: ____________ *Memb#* ________