ISSN 1035-7521

Australian UNIX systems User Group Newsletter

AUUGN

Volume 13, Number 4

August 1992

Registered by Australia Post, Publication Number NBG6524

AUUGN Volume 13 Number 4 August 1992

The AUUG Incorporated Newsletter

Volume 13 Number 4

August 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
Phil McCrea
phil@softway.oz.au
Softway Pty. Ltd.
79 Myrtle Street
Chippendale NSW 2008

Vice-President
Glenn Huxtable
glenn@cs.uwa.oz.au
University of Western Australia
Computer Science Department
Nedlands WA 6009

Secretary
Peter Wishart
pjw@lobo.canberra.edu.au
EASAMS Australia
Level 6
60 Marcus Clark St.
Canberra ACT 2600

Treasurer
Frank Crawford
frank@atom.ansto.gov.au
Australian Supercomputing Technology
Private Mail Bag 1
Menai NSW 2234

Committee Members

Rolf Jester
rolf.jester@sno.mts.dec.com
Digital Equipment Corporation
P O Box 384
Concord West NSW 2138

Chris Maltby
chris@softway.sw.oz.au
Softway Pty. Ltd.
79 Myrtle Street
Chippendale NSW 2008

John O'Brien
john@wsa.oz.au
Whitesmiths Australia
#5 Woods Centre
Business & Tech. Park
Lucas Heights NSW 2234

Michael Paddon
mwp@iconix.oz.au
Iconix Pty Ltd
851 Dandenong Rd
East Malvern VIC 3145

Greg Rose
ggr@acci.com.au
ACCI
723 Swanston St
Carlton VIC 3053

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 an update.

The Annual General Meeting of AUUG Inc. will be held at 5:30 pm on 10th September, 1992, at the World Congress Centre, Melbourne.

AUUG Newsletter

Editorial

Welcome to AUUGN Volume 13 Number 4.

This issue should be received by members just before AUUG'92. Thus an update on the conference and the AGM notice have been included as a final remainder.

In this issue in addition to our standard sections, we are introducing a new section, **!AUUGN**, which will contain re-prints of items published in early issues of AUUGN. We are starting off with one that is often mentioned (*e.g.* in the recent election statements by nominees).

The response to the AUUGN new cover design competition has been poor. I am sure there are people out there with ideas, please send them in.

I am fast running out of papers from the summer conferences so anyone with some articles don't hesitate to contact me.

A letter to the editor has been included. It is towards the back of this issue, due to the fact that it arrived after most of the preparation had been completed. Such letters are always welcomed.

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. He has just changed jobs, so please contact me 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 5	Friday 25th September
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.

AUUG Institutional Members as at 03/08/1992

A.J. Mills & Sons Pty Ltd
A.N.U.
AAII
Adept Business Systems Pty Ltd
Adept Software
Alcatel Australia
Allaw Technologies
Amdahl Pacific Services
Andersen Consulting
ANI Manufacturing Group
ANSTO
Anti-Cancer Council of Victoria
ANZ Banking Group/I.T. Development
Apscore International Pty Ltd
Ausonics Pty Ltd
Australian Airlines Limited
Australian Bureau of Agricultural and
Resource Economics
Australian Bureau of Statistics
Australian Computing & Communications Institute
Australian Defence Industries Ltd
Australian Electoral Commission
Australian Information Processing Centre Pty Ltd
Australian Museum
Australian National Parks & Wildlife Service
Australian Taxation Office
Australian Technology Resources (A.C.T.)
Australian Wool Corporation
Automold Plastics Pty Ltd
Bain & Company
Ballarat Base Hospital
BHP Information Technology
BHP Minerals
BHP Petroleum
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
Classified Computers Pty Ltd
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
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 I.T.R.
Dept. of the Premier and Cabinet
Dept. of the Premier and Cabinet - SA
Dept. of the Treasury
Dept. of Transport
Dept. of Treasury & Finance
DEVETIR
Digital Equipment Corp (Australia) Pty Ltd
Easams (Australia) Ltd
EDS (Australia) Pty Ltd
Electronics Research Labs
Equinet Pty Ltd
FGH Decision Support Systems Pty Ltd
Financial Network Services
First State Computing
Fremantle Port Authority
Fujitsu Australia Ltd
GCS Pty Ltd
GEC Alsthom Australia
Geelong and District Water Board
GEMCO
Gemco
Genasys II Pty Ltd
General Automation Pty Ltd
GeoVision Australia
GIO Australia
Golden Circle Australia
Grand United Friendly Society
Great Barrier Reef Marine Park Authority
Gunnedah Abattoir
Haltek Pty Ltd
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
Information Technology Consultants
Insession Pty Ltd
Insurance & Superannuation Commission
Internode Systems Pty Ltd
Ipec Management Services
IPS Radio & Space Services
James Cook University of North Queensland
KPMG Solutions
Labtam Australia Pty Ltd
Lancorp Pty. Ltd.
Land Information Centre
Leeds & Northrup Australia Pty. Limited
Liquor Administration Board (NSW Govt.)

AUUG Institutional Members as at 03/08/1992

Logica Pty Ltd
Macquarie University
Mayne Nickless Courier Systems
Medical Benefits Funds of Australia Ltd.
Mentor Technologies Pty Ltd
Metal Trades Industry Association
Mincom Pty Ltd
Minenco Pty Ltd
Ministry of Consumer Affairs
Motorola Computer Systems
NEC Australia Pty Ltd
NEC Information Systems Australia Pty Ltd
NSW Agriculture
Nucleus Business Systems
Office of the Director of Public Prosecutions
Olivetti Australia Pty Ltd
OPSM
Ozware Developments Pty Ltd
Parliament House
Paxus
Philips PTS
Port of Melbourne Authority
Powerhouse Museum
Prentice Hall Australia
Prime Computer
Prospect Electricity
pTizan Computer Services Pty Ltd
Public Works Department
Pulse Club Computers Pty Ltd
Pyramid Technology
Queensland Department of Mines
Queensland University of Technology
Redland Shire Council
RMIT
Royal Melbourne Institute of Technology
SBC Dominguez Barry
Scitec Communication Systems
Sculptor 4GL+SQL
SEQEB Control Centre
Shire of Eltham
Software Developments
Softway Pty Ltd
South Australian Lands Dept.
Stallion Technologies Pty Ltd
Standards Australia
State Bank of NSW
State Revenue Office
Steelmark Eagle & Globe
Sugar Research Institute
Swinburne Institute of Technology
Sydney Ports Authority
Systems Union Pty Ltd
Tasmania Bank
Technical Software Services
Telecom Australia
Telecom Australia Corporate Customer
Telecom Network Engineering Computer Support Services
Telecom Payphone Services
Telectronics Pty Ltd
The Far North Qld Electricity Board
The Fulcrum Consulting Group
The Preston Group
The Roads and Traffic Authority
The Southport School
The University of Western Australia
TNT Australia Information Technology
Tower Computing Services
Tower Technology Pty Ltd
Tradelink Plumbing Supplies Centres
Triad Software Pty Ltd
Turbosoft Pty Ltd
TUSC Computer Systems
UCCQ
Unidata Australia
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
Vibro Acoustic Sciences Ltd.
Vicomp
VME Systems Pty Ltd
Walter & Eliza Hall Institute
Wang Australia Pty. Ltd.
Water Board
Westfield Limited
Wyse Technology Pty. Ltd.

AUSTRALIAN OPEN SYSTEMS USERS GROUP

AUUG Inc.

Annual General Meeting, 10th September 1992

Date:	Thursday, 10th September 1992
Time:	5.30 pm - 6.30 pm
Place:	World Congress Centre, Melbourne

Agenda

1. Accept the Minutes of 1991 AGM
2. Returning Officers Report: 1992 Election Results
3. Presidents Report
4. Secretarys Report
5. Treasurers Report
6. Conduct of 1992 Elections
7. Chapter Development
8. Other Business

Peter Wishart
AUUG Inc. Secretary

AUUG President´s Report

This is my first report as President of AUUG, so I should let you know why I decided to stand as President. This report paraphrases some points that were in my election policy statement.

1. Technical/Commercial polarization

I am concerned that the current polarization amongst some members re the technical v commercial orientation of AUUG should not cause a split in the AUUG ranks. AUUG should represent the interests of both the technical and the commercial community.

It's nice to romanticize about "the way AUUG used to be" when UNIX was a topic of some academic interest in Universities. But it is also important to cater for the needs of people, both technical and management, who have been introduced to UNIX fairly recently under the Open Systems push.

As a result, the focus of AUUG should be two-fold:

a. to provide a forum for the sharing of ideas and concepts about the Technical world of UNIX as an evolving operating system;

b. to provide a forum for people involved in the Commercial world of Open Systems, which of course are based on UNIX. This group is more interested in issues such as standards, GOSIP, etc.

Australia is too small a country to support 2 different UNIX User groups, and AUUG should clearly recognize that it should provide services to both communities. The current Management Committee certainly is in agreement on this, and we are have started to put appropriate programmes in place.

2. Neutrality of AUUG

It is important that AUUG be independent of UNIX political factions (UI, OSF etc), as well as hardware vendor independent.

Consequently it is desirable (although not essential) that the AUUG spokesperson, ie the President, should be as independent as possible, so that when AUUG's opinion is sought on various issues, which I hope it will be on many occasions, then the comments are perceived as not being biased in favour of any particular organization or company.

3. Liaison with other Professional Organizations

There are other professional organizations within Australia with overlap in the software area - ACS, IEEE, IREE, and IE Aust. Each of these organizations is starting to address the issue of Open Systems, and as a result AUUG will be liaising with these bodies and cooperate where appropriate in seminars, conferences, etc.

It is interesting to note that the IE(Aust) and the IREE have, for all intents and purposes, merged, with the IREE now representing the interests of electronics fraternity within the IE(Aust). Whilst I am not suggesting a merger of AUUG with anyone else, there are obvious advantages in cooperating with organizations where there is some common ground.

For the record, as well as being an AUUG member, I am a member of both the ACS and the IEEE, and have strong connections with the IREE and IE(Aust).

4. Distributed AUUG

Australia's distances make it essential to devolve more AUUG activities along State lines, with the formalization of local Chapters. This is being addressed by the Management Committee at present, and we should have formalized our Chapter structure very soon.

For those of you who do not know me, I am Managing Director of Softway Pty Ltd in Sydney. I am not a kernel hacker, but have been associated with UNIX since 1975 when it arrived at The University of NSW. I am in fact a lapsed academic ...

I am looking forward to steering AUUG through the current interesting times. If you have any comments or suggestions you would like to make, please feel free to mail me on phil@softway.sw.oz.au.

Phil McCrea

AUUGN New Cover Design Competition

We are looking for a new cover design for AUUGN to start with the first issue in 1993. To obtain member input we have decided to run a competition open to all AUUG members, with the entries to be judged by the AUUGN editor.

What is required to appear on the cover is:

1. The Name of the publication: AUUGN
2. The Name of the organisation: AUUG Inc.
3. The Volume and Issue numbers
4. Date, *i.e.* the Month and Year of the publication.
5. The registration number (ISSN 1035-7521).
6. Australia Post registration, currently, Registered by Australia Post, Publication Number NBG6524

The design should be easily reproducible and modifiable, (*i.e.* changes in date, *etc.*, also keep in mind the information on the spine). The current cover is black on coloured background (the colour changes monthly: white, red, green, yellow, orange and blue). Such a colour scheme is preferred, however there is limited opportunity for use of basic colours, keeping in mind that ease of reproduction is required and cost has to be kept down.

As with papers, I prefer documents to be e-mailed to me in postscript or troff. Hardcopy submissions should be on A4, addressed to:

Jagoda Crawford
AUUGN Editor
PO Box 366
Kensington, NSW, 2033
AUSTRALIA

E-mail: *jc@atom.ansto.gov.au*

Phone: +61 2 717 3885
Fax: +61 2 717 9273

The competition closes on Friday September 11th 1992. The entries will be displayed at the AUUG'92 for comments. The result will be announced sortly afterwards with the designer of the selected entry receiving a free year's subscription to AUUG.

AUUG '92 Update

Just a remainder that AUUG'92 is in the week of 8 - 11 September 1992. This year, Australia's premiere conference and exhibition will be held in Melbourne at the World Congress Centre.

8 September is a full day dedicated to tutorials:

We are offering 2 full day sessions:

- SVR4 Internals
- BSD Internals

AM 1/2 day sessions:

- Insights into Guru-level C Programming
- Cruising the Network
- Intro to the UNIX Operating System

PM 1/2 day sessions:

- PERL
- The Easy Route to X-Windows
- Business Requirements and Open Systems

9 - 11 September offers 3 morning Keynote and Plenary sessions and 3 streams, Technical, Management, and Combined for the afternoon.

- Technical sessions are designed for the computer scientist or programmer wanting in depth knowledge. The how and why of programming details will be presented.
- Management sessions are designed to present the business case perspective on topics of general concern to managers in the computer industry.
- Combined stream, new to AUUG, is designed to provide enough technical and management information to assist MIS directors and systems administrators in making educated decisions.

Over 53 local and international speakers will provide their global and expert insights with respect to such topics as Rightsizing, Free Software, Fault Tolerance and the Future, Virtual Reality, Hardware Profiling of a UNIX Kernel, Log Structured File Systems, the New Security Paradigm, CICS for Open Systems, Open Systems, The Search for the Holy Grail, Xcelsis, A new Approach to Spreadsheets, Porting C++ to an Embedded Environment.... and many more!

Fees for AUUG '92 are as follows:

Tutorials:	*On or before 28 Aug*	*After 28 Aug*
Full Day	**$280.00**	**$330.00**
Half Day	**$160.00**	**$210.00**
Conference	*On or before 28 Aug*	*After 28 Aug*
AUUG Members	**$300.00**	**$350.00**
Non Members	**$500.00**	**$550.00**
Day Fee Members	**$150.00**	**$200.00**
Day Fee Non Members	**$200.00**	**$250.00**
Full-time Student	**$100.00**	**$150.00**

Please note in an effort to reduce costs, luncheon is NOT included. If you desire to partake in a two course light luncheon, it is an additional $25.00 per day. The cocktail party on Wednesday, 9 September and the Gala Dinner on Thursday, 10 September is included in the conference fee.

You should also note in the Conference Program/Registration brochure, that in the General Information section under travel, special arrangements with Ansett via Performax Travel have been negotiated to secure discount rates. In addition to the special accommodation rates other local hotels are sited for your convenience. A map on page 16 of the brochure will assist in locating the best site for you.

We look forward to seeing you in Melbourne for a very successful conference and exhibition!

The Program Committee:
Peter Karr - Chair
Liz Fraumann
Ian Hoyle
Robert Elz

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, UniForum, 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

Following is a list of prices† provided by UniForum.

PUBLICATION ORDERS	Price		Postage/Handling		
	Member	Non-Member	Domestic	Canada	Overseas
CommUNIXations back issues*	$3.95	$5.00	$3	$5	$5
UniForum Monthly back issues*	3.95	5.00	3	5	5
UniNews Newsletter subscription	30.00	60.00	8	11	30
1992 UniForum Products Directory	45.00	95.00	7	15	55
1992 UniForum Proceedings	20.00	25.00	4	5	11
Your Guide to POSIX	5.00	10.00	3	4	9
POSIX Explored: System Interface	5.00	10.00	3	4	9
Network Substrata	5.00	10.00	2	3	6
Network Applications	5.00	10.00	2	3	6
The UniForum Guide To Graphical User Interfaces	4.95	9.95	2	3	6
Electronic Mail De-Mystified	5.00	10.00	3	4	9
The UniForum Guide To Distributed Computing(*)	4.95	9.95	2	3	6

† Prices in US dollars
(*) please specify issues

Canberra Chapter of AUUG Inc.

General Meeting

8:00pm Tuesday 13th Oct 1992
(venue to be advised)
''Unix Multiprocessing - Status and Trends''

The talk will be given by Tony Hampel of Advanced Systems Division, Wyse Technology Inc, USA. The talk is organised by Trilogy Business Systems Australia PL.

4th Annual Canberra Conference and Workshops

Call For Presentations and Workshops

AUUG in Canberra is holding its 4th annual conference and workshops on Tuesday and Wednesday the 16/17th February 1993. As well as a selection of international and national speakers, we are looking for presentations from local individuals and organisations in any area of UNIX or Open Systems. Presenters for half or full day workshops on any subject are also welcome (workshop presenters receive a modest stipend).

The 1992 conference and workshops were attended by over 100 people from throughout the Canberra region. We look forward to seeing you at the 1993 conference and workshops.

For further details contact:

Presentations:	Peter Wishart ph (06) 2612894 fax (06) 2613806 email: pjw@lobo.canberra.edu.au
Workshops:	David Baldwin Ph (06) 2495026 fax (06) 2493992 email: David.Baldwin@anu.edu.au
Sponsorships/Advertising:	Elizabeth Keith Ph (06) 2434818 fax (06) 2434848

Canberra Chapter of AUUG Inc.

Annual General Meeting

8:00pm Tuesday 10th November 1992
(venue to be advised)

The Canberra Chapter of AUUG Inc. will have its Annual General Meeting on Tuesday the 10th of October 1992. We are looking for more enthusiastic volunteers to join our committee. We are also looking at adopting a constitution and formalising our relationship with AUUG (national body). Nominations are called for President, Secretary, Treasurer, and general committee positions. The returning officer for the nominations is John Barlow. You can nominate at any time by written submission, signed by the nominated candidate, and all nominees are encouraged to write a few words about themselves for publication on the day of the ballot (the AGM). Nominations will be accepted on the night. The ballot will be a simple show-of-hands. More details will be mailed out in October.

If you have any enquiries on the positions, nomination, or proxy-votes please contact John Barlow ph: (06) 2492930 (BH), (06) 2821925 (AH), (06) 2490747 (fax).

Review of Canberra Chapter Events: Dialup service.

The Canberra Chapter of AUUG Inc is currently setting up a dialup UNIX box to provide email and news services to Members. Currently the system is undergoing change (ironing out the bugs, identifying problems). The idea is to provide email and news via a dialup service, with some restriction on hours of connect time per week, to AUUG members in the Canberra region. Conditions of use are still being finalised, as is the choice of software that we will attempt to support.

Current status of the system is:

Hardware:	386DX - 25MHz, 4MB RAM, 300 MB ESDI disk, 150 MB cartridge tape, 6 serial ports, 1 9600/2400 baud modem (on loan, and one on order).
Software:	SCO Unix, ELM mailer, MH mailer, NN news handler.

The hardware and software have been arranged via Allaw Plus, Genitech, ComTech and Adept Software.

We are connected by UUCP to the Adept Software UNIX system, which is connected via MHSnet to the Australian Defence Force Academy Computer Centre dialup service. The machine has an MX record for email delivery via CC ADFA, and is called canb.auug.org.au.

Current status includes mostly working news and email (just a few concerns on the mail return-address). We have been experimenting with downloading news/email to a local users PC via different programs and different news/mail readers (FSUUCP, UUPC, PC ELM, Easymail, SNews). The system looks very promising, and we are encouraging members to try it out.

Important Dates:

15th September	Committee Meeting at I-Block
13th October	General Meeting at ANU on ''multi-processing UNIX''
10th November	Annual General Meeting
16/17th February '93	Summer Conference and Workshops

Mr John Barlow,
Secretary, Canberra Chapter of AUUG Inc.
ANU, Parallel Computing Research Facility, (06) 2492930 (work) (06) 2490747 (fax)
John.Barlow@anu.edu.au (email).

The WAUG Column

The speaker at WAUG's June meeting was John Karabin of Randata, talking about Data Encryption. Unfortunately I missed the meeting, but I'm told the talk was informative and the attendance above average.

At the July meeting Glenn Huxtable, UNIX systems manager at the Department of Computer Science at The University of Western Australia, spoke on UNIX password security.

Glenn is also Vice-president of AUUG and Chairman of WAUG. After dealing with some administrative matters and introducing himself :-), he performed a nifty metaphorical hat-switch to become the speaker.

Among other things, Glenn is my boss. Luckily for me, my reaction to his talk was positive! He explained the UNIX password mechanism and the concepts and issues of password security from the viewpoints of both the user and the systems manager. A system's passwords are its first line of defence against unauthorised access, so their security is important to most installations. Especially if you are on a network, have sensitive information to protect, or have users - such as students - who may be inclined to hack (in the nasty sense of the word), you need to make your passwords hard to crack and hard to get at.

Every user, whether a first-year student or a managing director, needs to know why they should choose a good password - where "good" means difficult to guess or crack but easy to remember and type - and why they should keep it secure. Glenn illustrated his talk with anecdotes about how passwords have been obtained or guessed. He showed us some statistics on the time taken to crack various kinds of passwords by brute force. He also explained what makes a bad password (names, dictionary words - that sort of thing) and gave some ideas for making up good ones.

For the systems manager, Glenn discussed various issues, such as the root password and use of root access, the security problems of Network Information Service (formerly called Yellow Pages), and password shadowing - whereby the encrypted passwords are kept in a separate "shadow" or "adjunct" file that is accessible only to root.

We had a good attendance at the meeting and everyone seemed to find Glenn's talk interesting. It generated quite a bit of discussion. It was a little longer than planned, but I don't think anyone but Glenn noticed.

If you'd like to speak at WAUG or have an idea for a speaker, please contact our Meeting Organiser, Mark Baker, at baker@telecomwa.oz.au or on (09) 420 6813.

If you're interested in joining WAUG (the Western Australian UNIX systems Group) or contributing to our newsletter YAUN (Yet Another UNIX Newsletter), our address is PO Box 877, WEST PERTH WA 6005.

With any luck I'll see some of you at AUUG '92.

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

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:

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**.

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, 1990, 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	*FAX:*	(02) 332 4066
	AUUG Inc	*E-Mail:*	auug@atom.lhrl.oz
	P.O. Box 366		
	Kensington N.S.W. 2033		

Technical enquiries to:

Michael Paddon	(mwp@iconix.oz.au)	
or		
Frank Crawford	(frank@atom.lhrl.oz)	(02) 717 9404

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?	Yes	No	
	If **yes**, approximately how much?	____________________		
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)?	Yes	No	
	If **yes**, approximately how much (please also specify units, *e.g.* *$X/MB* or flat fee)?	____________________		
A13.	Once connected, are you willing to provide additional connections?	Yes	No	
A14.	Additional Comments:			

Existing Sites

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

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

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

AUUG Book Club

Book Reviews

AUUG Inc and Prentice Hall Australia have formed the AUUG Book Club to give AUUG members a chance to obtain Prentice Hall books at a significant discount.

To obtain copies of the books reviewed here, fill in the order form that appears at the end of the book reviews. Don't forget to deduct 20% from the listed retail prices.

Review copies of these books were kindly provided by Prentice Hall.

If you would like to review books for further offers from the AUUG Book Club, please contact the AUUGN Book Review Editor (see page 5).

COMPREHENSIVE C

by David SPULER
Prentice Hall, RRP $ 44.95
ISBN: 0-13-1565141-1

Reviewed by
Mark White
National Centre for Studies in Travel and Tourism
<markw@cltr.uq.oz.au>

The appearance of yet another introductory-level guide to the C programming language may be treated with an understandable indifference these days. A recent browse through a local up-market (but decidedly non-technical) bookshop produced literally dozens of different volumes all offering up-to-date knowledge bases of one or more of the more popular C language implementations. Further searches produced an even greater number of books discussing C++, development within various graphical environments, and so on. All of which may prove daunting to someone who not only wishes to purchase a single useful and complete introduction to C, but also a reference that defies obsoleteness. Spuler's work, fortunately, falls into the latter category.

The book is equally divided into two parts - an introduction to the C language, and a more advanced section, covering issues such as efficiency, development techniques, and an excellent discussion of the benefits of good programming style. The reader is assumed to have at least some working knowledge of the basic ideas of computer programming, and therefore we are spared any lengthy introductions. The logical presentation sequence of the language operators, types, constructs and libraries permits the reader an unhindered learning path throughout the book, and the inclusion of a useful and practical series of exercises at the end of each chapter further enhances my opinion of this book as an excellent self-education tool.

It is in the second section, titled "Advanced Issues", that Spuler's book continues to impress. Any of the chapters in this section could be studied independently, although again the presentation sequence is quite logical and appropriate. It is this section that sets this volume apart from other introductory works on C, in that its usefulness does not deteriorate once the reader has progressed beyond a certain level of skill. Of special note were the final chapters, devoted to a discussion of the art of programming on a UNIX system, and program efficiency. Each section was dealt with in a concise and logical manner - comments that also apply thoughout the book in general.

With a few appropriate exceptions, the language is presented independently of any operating system peculiarities, without necessarily limiting the scope of the various discussions. In instances where system-specific functions or attributes apply, the reader is presented with an example from one or more platforms (usually UNIX or DOS based) and then urged to consult their particular implementation's documentation. In doing this, the author is able to target and reach a much wider audience than may have been possible than if he had simply focussed on, say, one of the more popular PC implementations of C. He also uses the system variations as practical examples of the importance of portability in the software development process.

I can highly recommend "Comprehensive C" not only as a resource for beginners in the C language, but also as an interesting and valuable reference for more experienced programmers.

OPEN SYSTEMS INTERCONNECTION

by Gary Dickson and Alan Lloyd
Prentice Hall, RRP $49.95
ISBN: 0-13-640111-2

Reviewed by
Warren Simon
NEC Information Systems Australia
<warren@pdnsw.pd.necisa.oz.au>

Many of those working in the commercial Unix arena will be aware that OSI is becoming an increasingly important topic, and interest is likely to rise significantly with the release of 4.4 BSD later this year, which is reported to include an OSI stack and utilities (based on ISODE 7.0).

OSI is a huge and complex ballgame, the learning curve can be long and slow, and reading the standards is laborious and often boring. The alternative is to seek books such as this which provide a cohesive overview of OSI and a very good introduction to all the buzzwords. The book aims to provide a broad coverage of the OSI standards at a moderately technical level. It is partly based on a training course run by the authors, and has an interesting style, in that many paragraphs have bold keyword headings in the left margin, making it very easy to pick up the book for a casual browse.

The stated objective of the book is to help communications managers, programmers and engineers to:

- clarify any misunderstandings about OSI
- comprehend the fundamental concepts and broad picture of open networking
- understand the major issues in buying conformant OSI systems
- develop skills necessary for further study of the standards themselves
- gain some insight into future industry trends

The book is divided into six parts with Part 1 being an introduction to the OSI reference model, layering concept, and terminology. It covers some history of OSI, why we need standards, the various standards bodies and the rationale behind OSI.

Part 2 covers the lower 4 layers of the 7 layer model. These include the physical, data link, network and transport layers. There is discussion of connection oriented and connectionless networks, network addressing, ISDN, the various LAN standards (ethernet, token ring, token bus), link protocols (HDLC, LAPB, LAPD, LLC), and routing protocols. The data link and network layers are given comprehensive coverage and a chapter is devoted to internetworking of subnetworks.

Part 3 deals with standards for information exchange, covering the upper 3 layers of the model (session, presentation and application). A chapter is devoted to an explanation of the OSI encoding scheme known as Abstract Syntax Notation One (ASN.1).

Part 4 covers standards for distributed applications. There is a chapter devoted to each of X.400 message handling, X.500 directory services, file transfer and access methods (FTAM), virtual terminals (VT), systems management and distributed transaction processing.

Part 5 covers document handling standards such as Open Document Architecture (ODA), document transfer and manipulation, and electronic data interchange.

Part 6 deals with system integration, functional standards, conformance testing and procurement matters. It discusses how standards are developed, implemented, tested and finally turned into a product. There is brief mention of the UK, Australian and US GOSIP's. Also a chapter on buying OSI discusses the importance of convincing top level management that the short term penalties of migrating to OSI will be overcome by long term benefits.

An appendix contains a list of OSI and related standards. It includes both CCITT and ISO standards. There are also answers to the questions posed at the end of each chapter.

The book is very readable, supplemented with many diagrams and tables. On the whole it provides an excellent introduction to the concepts and terminology surrounding OSI.

UNIX Curses Explained
by Berny Goodheart
Prentice Hall, RRP $58.95
ISBN: 0-13-931957-3

Reviewed by
Mark White
National Centre for Studies in Travel and Tourism
<markw@cltr.uq.oz.au>

With the proliferation of books discussing each of the various windowing systems currently available, the appearance of a reference book devoted to programming for devices as seemingly archaic as ASCII video terminals is somewhat paradoxical. However the author explains, in the book's preface, that despite the onset of windowing systems into the UNIX scene, the ASCII terminal still "dominates the market". This book aims to serve two purposes: to provide a series of tutorials enabling a competent C programmer to create programs using the curses package, and also to provide a reference for the curses function library.

The book's initial chapters overview the curses package, and describe it's essential components - the window structure, the terminfo database, the <curses.h> include file, and the curses environment. Chapters 4 to 6 explain the structure of a curses program, gradually introducing many of the available functions such as window and attribute manipulation, the use of colours, and the use of windows larger than the terminal screen, called pads. The author displays a thorough knowledge of his subject, and also the ability to present the material in an interesting and logical manner. I particularly found many of the program examples both useful and pertinent.

Of further interest are two chapters describing, in quite some detail, the Terminfo terminal database, and the procedures involved in building terminfo descriptions for any given terminal. Although not immediately useful to an applications builder, these sections do provide an insight into the complexity surrounding the curses package.

The final (and largest) chapter provides (in standard UNIX format) a description of each function in the /usr/lib/libcurses.a library. I found these manual pages to be a worthwhile substitute for the limited curses documentation supplied with my system (UNIX V.3).

Of course, any recommendation (or otherwise) of the benefits to be gained from the purchase of this book will greatly depend on your need to produce or maintain software specifically for an ASCII terminal user base. If so, this book will serve as both a practical introduction and a useful reference.

AUUG BOOK CLUB & PRENTICE HALL AUSTRALIA

20% DISCOUNT TO AUUG MEMBERS

Please send me a copy/copies of the following books —

Spuler/ **Comprehensive C**

___ ISBN: 0131565141 Paper 1992 RRP $44.95*

Dickson/ **Open Systems Interconnection**

___ ISBN: 0136401112 Cloth 1992 RRP $49.95*

Goodheart/ **Unix Curses Explained**

___ ISBN: 0139319573 Paper 1990 RRP $58.95*

*Deduct 20% from listed retail price

Name: ______________________ Organisation: ______________

Address: ____
(Street address only)

______________________________ Telephone: ______________

☐ Please send my book/s on 30-day approval *(tick box)*

Enclosed cheque for $ ________ (Payable to 'Prentice Hall Australia')

Please charge my: ☐ Bankcard ☐ Visa ☐ MasterCard

Credit Card No: ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

Expiry Date: __________ Signature: ______________

Mail or fax completed order form to Prentice Hall Australia, PO Box 151, Brookvale NSW 2100

OR Use our FAST PHONE SERVICE by calling Sandra Bendall.
SYDNEY (02) 939 1333

A.C.N. 000 383 406

Prentice Hall Pty. Ltd.
7 Grosvenor Place, Brookvale NSW 2100.
Tel: (02) 939 1333 Fax: (02) 905 7934
A Paramount Communications Company

Enterprise Transaction Processing

Chris Schoettle
UNIX System Laboratories
Australia and New Zealand
Tel: +61-2-906-8953
Fax: +61-2-436-4673
Email: cts%uslp@usl.com

Terence Dwyer
UNIX System Laboratories

1. INTRODUCTION

Traditionally, Transaction Processing (TP) has been performed on centralized mainframe computers running proprietary operating systems, and proprietary TP system software. In the last several years, we have seen the emergence of TP software, including high performance Relational Database Management Systems (RDBMS), e.g. [INFORMIX] and [ORACLE], and TP Managers such as TUXEDO® System /T [USL], for computers running the UNIX® Operating System. However, the real promise of transaction processing on UNIX-based computers lies not in the ability to provide the "stand-alone" TP model common to proprietary systems, but in the ability to provide the hub of an integrated, distributed Enterprise Transaction Processing (ETP) System. This paper describes the hardware and software architecture of an ETP System. Such a system will enable proprietary TP users and vendors to capitalize on the trends toward decentralized computing, to migrate to UNIX-based TP systems, and to protect their investment in proprietary TP systems.

Figure 1 shows a typical ETP system configuration. This configuration is composed of the following components:

- Tier-1: Personal Workstations (WS)
- Tier-2: UNIX TP Servers (UTPS)
- Tier-3: Proprietary TP Servers (PTPS)

The Tier-1 WS machines, running a variety of proprietary operating systems (e.g. MS-DOS™, OS/2™, MACOS™, etc.) as well as the UNIX Operating System, are connected to a network, perhaps a Local Area Network (LAN), as shown in Figure 1. These machines are used to provide user interface processing. They allow the possibility of the attachment of hundreds of users to each UNIX TP Server, and offer the possibility of a wide variety of new interfaces for TP applications, including Graphical User Interfaces (GUI), in addition to the traditional forms-oriented TP-input paradigm.

Tier-2 consists of a networked set of powerful mid-sized computers running the UNIX Operating System and TP system software, such as TP monitors and RDBMS systems. The network connecting Tier-2 machines could be a LAN (perhaps the same one to which the Tier-1 machines are connected, as shown in Figure 1), or a Wide Area Network (WAN). The Tier-2 machines provide distributed TP services in the UNIX environment, including access to a variety of TP applications using the high performance RDBMS systems now available on UNIX platforms. In addition they link the workstations to the proprietary machines.

Tier-3 machines are mainframe class computers running proprietary operating systems and proprietary TP monitors such as IBM's CICS [IBM-1]. Today, such machines do the bulk of TP processing for most corporations, and contain a large investment in programs and stored data. Access to these programs and data will be required as

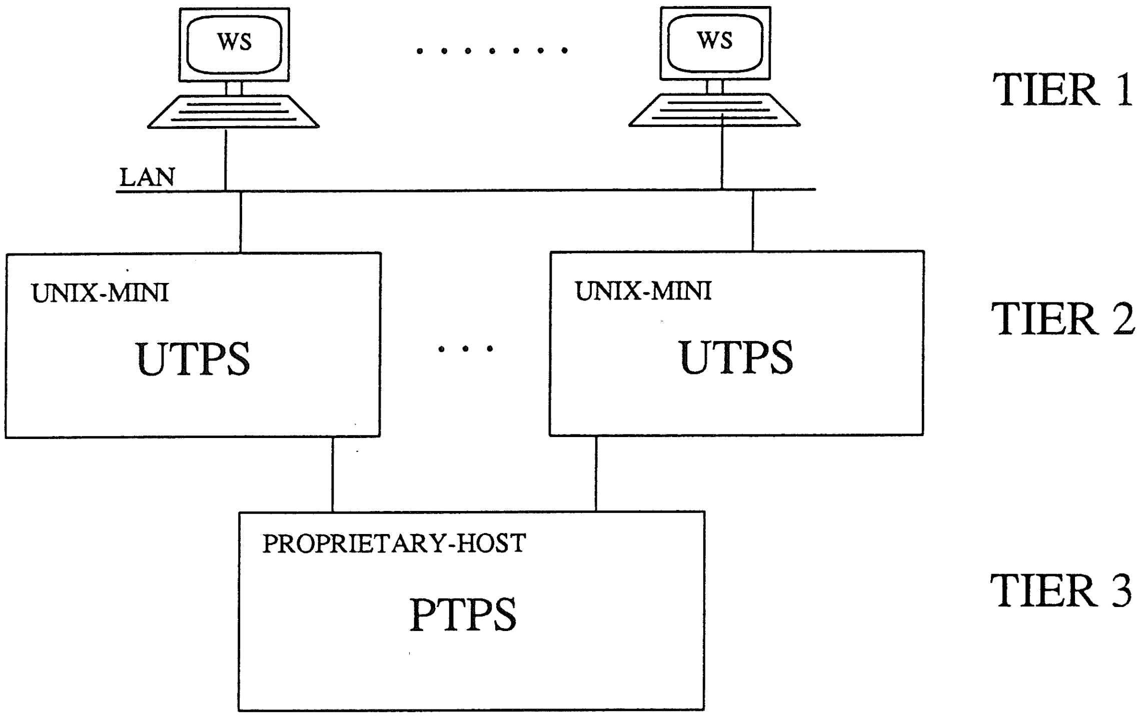


Figure 1. ETP Architecture

corporations move to TP Systems based on computers running the UNIX Operating System. Increasingly, Tier-3 machines will take on the role of proprietary TP servers. Figure 1 depicts a single Tier-3 machine with point-to-point connections from two Tier-2 machines. The exchange of data between Tier-2 and Tier-3 machines is likely to be carried over special networks supporting the proprietary protocols required to interface to Tier-3 machines.

Thus, an ETP system is composed of a collection of heterogeneous machines (and attendant operating systems), ranging from the personal computer to large proprietary mainframes. The Tier-2 (UNIX-based) machines play the central role in this system, providing local TP services to the workstation community, and connecting them to proprietary environments.

The TP platform software of the ETP system is the "glue" which binds together the hardware tiers into a unified TP System. As such, it provides communications, transaction, and administrative services to applications programmers and administrators, and may exist on all tiers. Key to the integration of an application across the tiers is the existence of a common TP Application Programming Interface (TP-API) for intermodule communication and transaction control. At the workstation level TP-API provides for the communication of input requests to the Tier-2 machines. At the Tier-2 level, TP-API provides for the reception of workstation input and the invocation of Tier-2 application services, or the forwarding of the requests on to Tier-3 machines. At the Tier-3 level, TP-API provides for the execution of TP service requests received from the Tier-2-machines. Goals of TP-API include consistency of syntax and semantics, location transparency of invoked modules, and transaction semantics on executed actions throughout the levels.

Subsequent sections of this paper expand on the architecture of ETP. Because of its central importance, this paper begins in Section 2 with a more complete description of Tier-2. Section 3 shows how Tier-1 is incorporated in ETP. Section 4 provides considerations for the inclusion of Tier-3 (i.e. proprietary TP) systems into ETP. Section 5 reports on the status of the construction of a commercial grade ETP system.

2. TIER 2: THE UNIX TP HUB

We start with Tier-2, the "middle tier" of the ETP System. [Landis] provides good insight why UNIX-based computers provide excellent functionality to play the middle role in ETP. This level consists of a networked set of powerful minicomputers running the UNIX Operating System. The use of a set of minicomputers offers several advantages including:

- the growing price advantage of machines smaller than mainframes
- the ability to mix heterogeneous machines, each suitable for particular tasks
- the ability to integrate several department size TP applications, each running on dedicated hardware, into a single application domain.

It should be noted that Tier-2 is more than a switcher of workstation requests to Tier-3 machines. Tier-2 machines themselves contain executable application code and shared databases. As TP applications are made available on Tier-2 machines (either new applications, or migration of Tier-3 applications to Tier-2 machines), many of the requests initiated from Tier-1 machines, or originating from within Tier-2 itself, may be completely satisfied at Tier-2. Tier-2 machines thus contain important, potentially "mission critical", resources for the corporation. As such, they are accorded the security and administration (e.g. backup) due traditional mainframe TP resources.

The TP environment for Tier-2 can be provided by an extension of TUXEDO® System/T. As described in [Andrade], TUXEDO System /T provides a powerful client/server model suitable for building high performance TP systems on Symmetric Multiprocessor (SMP) computers running the UNIX Operating System. Features of the System /T architecture include a high performance, shared-memory based name server, called the "Bulletin Board" (BB), and interprocess communication via System V messages. Requirements for the extension of such an architecture to a distributed Tier-2 architecture include:

- Communications Support for inter-machine client/server interactions
- Distributed Transaction Support (DTP)
- A TP oriented API.
- Centralized Administration

Figure 2 depicts a two node, Tier-2 system built upon this extended architecture.

2.1 Communications

Extension of System /T to the distributed case can be implemented by:

1. the distribution of the "Bulletin Board"
2. the extension of the messaging system

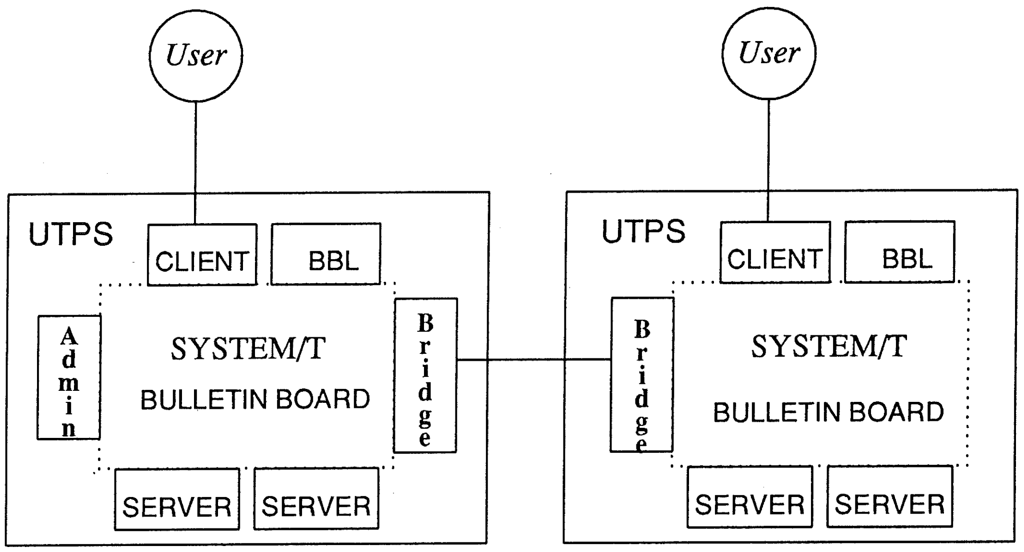


Figure 2. Tier-2, Two Node System /T Configuration

2.1.1 Name Server Distribution The performance constraints of TP systems require a fast method for determining client/server rendezvous. The shared-memory implementation of the BB in the SMP implementation of TUXEDO System /T fulfills this requirement. In the distributed case, it is desirable for each node to retain this fast access. One way to do this is to replicate the BB on all of the nodes. However, the BB contains two types of information: name-to-address mapping, and statistics. The former is used to provide location independence for client-server requests, and the latter is used for both administrative purposes and for load balancing. The name-address mapping information represents (relatively) stable information in a TP system, while the statistics are much more volatile.

The propagation of the BB's stable data can be accomplished through a special set of distributed administrative server processes called "Bulletin Board Liaison" (BBL) processes. Statistics, on the other hand, are kept locally at each site, and are made available administratively. These statistics are too volatile to be propagated throughout the system, and are not used as the basis of load balancing. Instead, a round robin method is used at each site to balance service requests originating from that site.

2.1.2 Inter-Machine Messaging As described in [Andrade], System V messages have very good properties for TP systems. In particular, they provide for a priority-based, reliable datagram service upon which efficient client/server interactions may be built. Using System V's networking facilities and providing a generalization of the name space for message queues, it is possible to provide a robust inter-machine messaging facility. The key implementation vehicle is a set of cooperating bridge processes which act as message forwarders. Since the bridges utilize reliable transport mechanisms, such as Systems V's TLI, the effect is to provide a "reliable datagram" service between client and server processes on different machines.

As in the SMP case, services are requested by name. To application programmers, the network is invisible (as are message queues in the SMP case). When a client requests a service, System /T selects a server by using the local copy of the BB, and then sends its request message to the selected server, using the bridges when the server is not co-located with the client. Likewise, System /T routes the reply to a service request to the originating client, whether it be local or remote.

2.2 DTP

The distribution of client/server interactions across the Tier-2 machines heightens the need for (distributed) transaction control. Transactions [Bernstein] provide a method to encapsulate a set of actions into a single atomic unit of work. This unit of work either wholly succeeds or has no effect. The results can be used to advance a set of distributed, logically related resources, e.g. data base systems, from one consistent state to another. One way to provide transactions for the Tier-2 machines is to implement the model of transaction control described in [X/OPEN-1]. In this model, a Transaction Manager (TM) coordinates transactions throughout a set of computers by providing communications paths with transaction semantics, and by interfacing to Resource Managers, e.g. DBMS systems, for the purpose of transaction control. In order to do this, the TM:

1. Generates Global Transaction Identifiers
2. Tracks sites participating in the transaction
3. Executes a two phase commit protocol when the application signals that all of the work is done
4. Executes a recovery protocol when a site is restored to operation after an outage

System /T has been extended to provide transaction facilities according to the X/Open model. In addition to the library routines which provide the application transaction management functions, the implementation of transaction control is as a special set of administrative server processes (not shown in Figure 2) which coordinate commit and recovery. These processes utilize data structures both in volatile and persistent memory.

2.3 TP-API

The use of a set of computers at Tier-2 places additional requirements on the application. For example, requests which effect permanent resources on multiple sites need to be grouped into transactions. If the computers have heterogeneous cpu architectures, it will be necessary to convert data types as data is exchanged. The following facilities are required of an API suitable for the high performance distributed TP interactions which occur within Tier-2:

- Transaction control
- Client/Server Communications
- Presentation Services (data conversion)

System /T provides these functions to applications running on Tier-2 through a TP-API called the Application Transaction Manager Interface (ATMI).

2.3.1 ATMI Transactional API The transaction control functions of ATMI allow an application to delimit a series of requests as comprising a single unit of work, called a transaction [Bernstein]. Generically, they consist of the procedures to begin work, signal completion of work, and undo work. In ATMI the functions which provide these services are called tpbegin(), tpcommit() and tpabort(), respectively.

2.3.2 ATMI Client/Server Communications API The client/server communication functions of ATMI allow the invocation of distributed services by name. Providing requests by name provides location independence of the requester from the server, thus allowing the server to be relocated without compromising the requester's ability to direct requests to it. Useful request/response paradigms include:

- synchronous calls
- asynchronous calls

• one-way calls

Synchronous calls block until the results are returned and are used when the requested results are required immediately, Asynchronous calls may be used to improve throughput when an application has several operations which may be performed in parallel. One-way calls are used when the results of the operations need not be known by the requester. In ATMI these services are provided by the functions tpcall() and tpacall().

Figure 3a depicts the standard stacking paradigm of request/response interactions. Here, SERVER 1 receives a request, does some processing, and calls SERVER 2 to do some more processing. While SERVER 2 is processing, SERVER 1 is blocked, waiting for its reply. When this is received, SERVER 1 replies to the requester. Efficiencies may be gained in TP applications by providing facilities for a "bucket brigade" style of processing, as depicted in Figure 3b. In particular, if SERVER 1 has no more processing to do after calling SERVER 2, it can drop out of the request processing, and pass responsibility for responding to the requester to SERVER 2. SERVER 1 then becomes free to handle other requests.

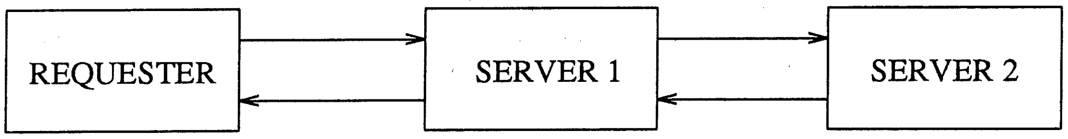


Figure 3a: Stacked Requests/Replies

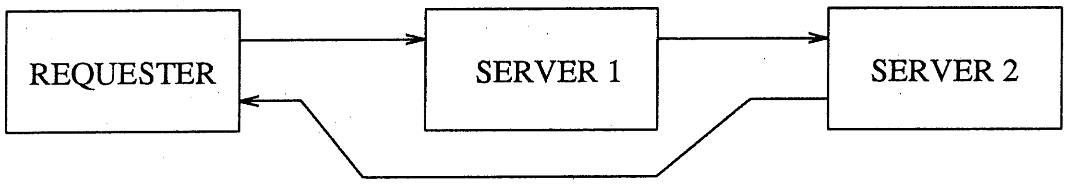


Figure 3b: Forwarded Request/Reply

ATMI provides this paradigm via the function tpforward().

2.3.3 Communicating Transactions A key concept in the ATMI model is that transactions accompany communications. For example, if a client begins a transaction, and then communicates with a server (e.g. by issuing a tpcall() function), the work done by the server becomes part of the transaction started by the client. Likewise, if the server makes requests of other servers, their work is also encapsulated by the transaction. In effect, transactions are propagated to all called services, whose work is then either committed or rolled-back when the originator calls tpcommit() or tpabort(), respectively.

2.3.4 ATMI Presentation Services While it is desirable to allow for Tier-2 to be comprised of computers of different cpu architectures, it is also desirable:

1. to minimize those differences for application programmers
2. perform conversions only as necessary

The messages sent between requesters and servers by ATMI calls are images of "typed buffers" [X/Open-1]. A typed buffer is a buffer with an associated string-named handle, called its "type". A typed buffer is created via a call to the function tpalloc(), and is destroyed by a call to the function tpfree(). When a message is sent, the type of the associated buffer is used to select a conversion function. Likewise, the type is used to invoke an "unconversion" procedure when the message is dequeued in the server. Typically, the supplier of a type, e.g. a system's programmer, provides its conversion routines, so that its user, e.g. an application programmer, can use it in client/server calls without regard to the architectures of the communicating machines.

System /T provides several built-in types, including character arrays, null terminated ASCII strings, C structures, and an attribute-value abstract data type called a Field Manipulation Language (FML) buffer. This latter type is a kind of heap data structure, in which elements are referenced by name. Built-in types, except character arrays, are automatically converted when passed between machines of dissimilar architecture. Character arrays are passed through without any conversion. Applications are free to add their own buffer types, but in so doing must supply the associated conversion functions.

System /T calls conversion functions only when source and destination machines are of different architecture, as indicated in a configuration file. Thus, if all of the machines at Tier-2 are of the same architecture, no conversion will be done for exchanged data.

2.4 Administration

Although Tier-2 consists of a network of machines, it is often required that they be administered as a unit, allowing an administrator to tend to the entire system from a single terminal. Such administration would typically consist of booting or shutting down the system, monitoring its performance, adjusting parameters, making backups, etc.

3. TIER 1: INCORPORATING WORKSTATIONS

There are several reasons to incorporate workstations in a TP environment. One of the most important is to offload CPU processing for a human interface. The asynchronous terminal, the traditional UNIX input device, imposes a significant burden on the UTPS for TP applications. The reason for this is that each input character requires the servicing of an interrupt. Additionally, most TP input is forms-oriented, and since forms packages most often read the screen in "raw mode", the forms handler, an application program, must be scheduled by the operating system on a per-character basis. The CPU overhead to accommodate such processing is enormous, perhaps consuming as many cycles as the rest of the software combined, including application, dbms, networking protocol and operating system logic.

Workstations, on the other hand, can be used as block mode devices, a type well suited for TP system input. In this mode, an entire message is received with one interrupt. In addition to relieving the UTPS of per-character processing, the cpu and memory provided by the workstation can be used to provide alternate forms of interface, including GUIs.

The model of workstations assumed in ETP is that workstations are intelligent devices (i.e. ones with cpu and memory) that are requesters of TP services. They need not be machines running the UNIX Operating System, but must be capable of generating the protocol required to talk to a Tier-2 machine. In the ETP model, workstations are assumed to contain no shared or persistent resources, such as databases, do not themselves offer any services, are personally administered, and have no security features.

3.1 Gateway to Tier-1

An important goal for ETP is to allow the connection of the large numbers of users typical of proprietary TP systems to Tier-2 machines. The offloading of the cpu cycles for the forms interface is not sufficient to provide this connectivity. It is usually the case that each user logged on to a UNIX system has one or more processes attached to his or her terminal. The context associated with these processes, including memory, file descriptors, process table slots, etc., is unacceptably large. What is required is a method of connecting many TP terminals with much less context. One way to provide the needed functionality is by providing a special gateway process, called the Workstation Gateway (WSG) to provide communications with the workstation community.

Figure 4 shows the architecture of WSG. WSG is a multistated process, which provides connectivity for many workstations with a minimum amount of context per workstation. Its primary job is to act as a surrogate client for the "real" client software modules, which are executing on the workstations. As a surrogate for many workstations, WSG cannot afford to block while waiting for replies to service requests, and must be specially constructed to handle the blocking calls of its connected workstations.

3.2 Workstation TP-API

One way to integrate the tiers of ETP is to provide the same TP-API on them, when practical. Unlike Tier-3 machines, workstations have not traditionally been used for TP applications, and do not have existing TP-APIs. A natural choice then is to provide the Tier-2 TP-API on Tier-1 machines. However, as workstations serve only a requester role in ETP, the API provided on them need only be the "client side" of TP-API. For Tier-2 configurations running TUXEDO System /T, this means providing the client calls of ATMI, called WS-ATMI, for the workstation machines. WS-ATMI includes transaction demarcation and control functions (tpbegin, tpabort, and tpcommit), typed buffer manipulation functions (tpalloc and tpfree), and service request functions (tpcall and tpacall). The client/server paradigm of Tier-2 is thus extended to Tier-1 machines via a uniform API for service requests. As a set of library routines, WS-ATMI may be used with a variety of forms and graphics packages executing on the workstations to inject inputs into, and receive outputs from Tier-2 machines.

3.3 WS Transactions

The role of workstations in transaction control needs particular attention. Since workstations are considered to

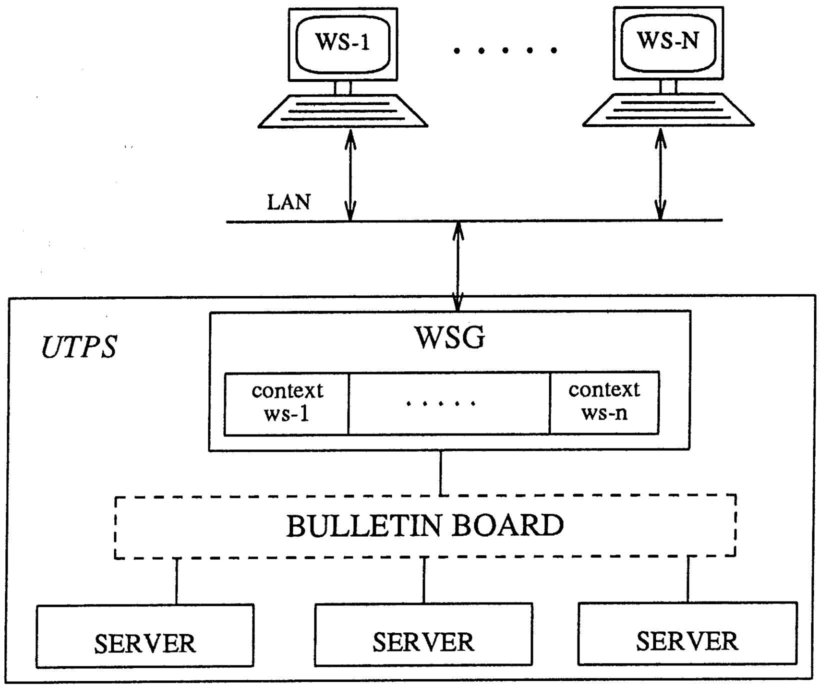


Figure 4. Workstation Gateway (WSG) - Multi-stated Client Surrogate

be personally administered, and thus may be turned off for extended periods, they should not be counted upon to provide transaction coordination for two-phase commit [Bernstein]. Instead, when an application on a workstation calls commit, transaction coordination needs to be delegated to a Tier-2 machine.

3.4 Tier-1 Administration

Although workstations themselves are considered to be personally administered, their connection to an ETP system should be subject to the administration of that system. In particular, an ETP administrator should be able to determine activity to/from the workstation, enable/disable its connection to the system, and advise it of abnormal conditions (e.g. imminent ETP system shut-down). A natural way to do this is to have WSG provide surrogate administrative services.

4. TIER 3: INCORPORATING PROPRIETARY TP SYSTEMS

As mentioned in the Section 1, the bulk of commercial TP processing is currently handled by proprietary TP systems. As TP users and TP vendors incorporate UNIX based solutions, there will continue to be a need to access the programs and data on proprietary TP systems. Overall, the approach taken to accommodate Tier-3 machines into an ETP system is to provide gateways from Tier-2 machines to Tier-3 machines. Within a given ETP system it is entirely likely that multiple heterogeneous proprietary systems may need to be incorporated. Such a scenario is depicted in Figure 5.

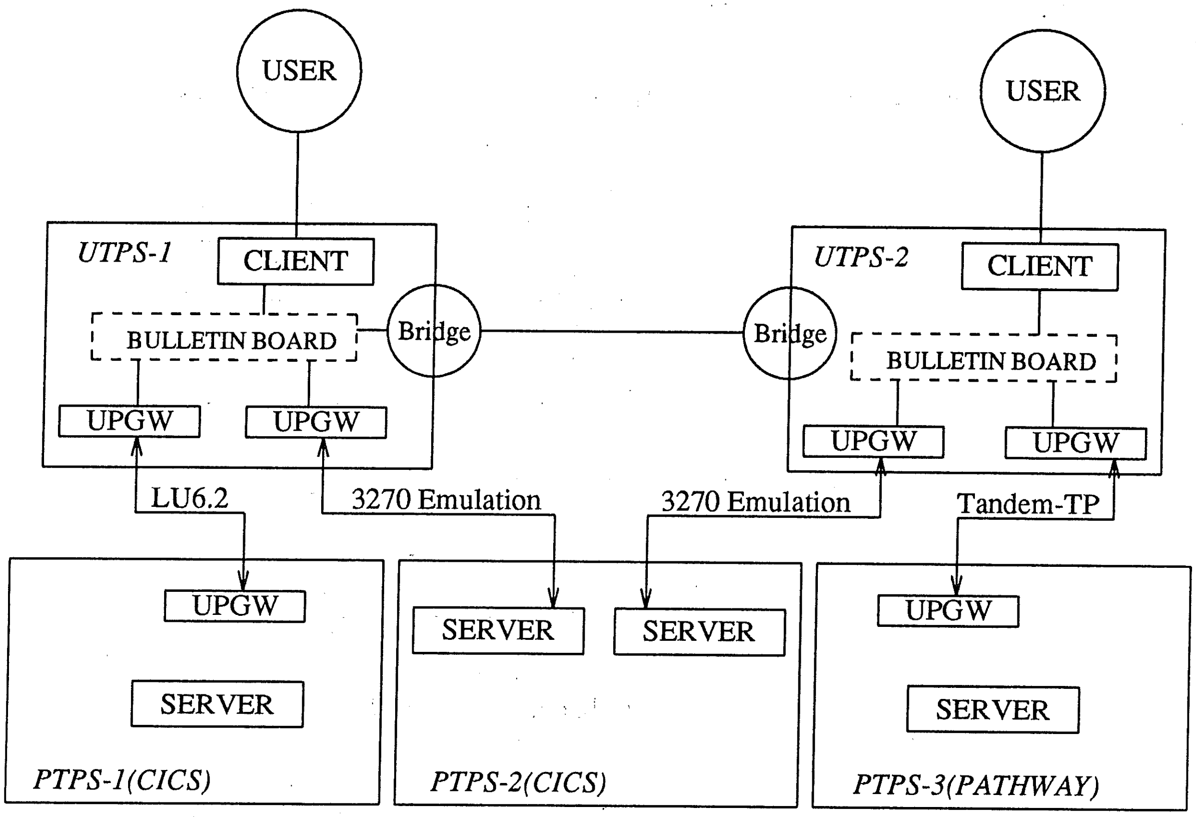


Figure 5. Multiple Heterogeneous Tier 3 Systems

In this figure, two Tier-2 machines (UTPS-1 and UTPS-2) are connected to three Tier-3 machines (no Tier-1 machines are shown). UTPS-1 is connected to two 370-compatible mainframes (PTPS-1 and PTPS2), each running an instance of the CICS Transaction Monitor [IBM-1]. UTPS-2 is connected to PTPS-2, and in addition is connected to PTPS-3, a Tandem computer running the PATHWAY Transaction Monitor [Tandem].

4.1 Gateway to Tier-3

In the ETP model, Tier-3 machines are considered to be servers. The basic paradigm of request/response is extended from Tier-2 to Tier-3 via UTPS-PTPS Gateway processes (UPGWs), depicted in Figure 5. By advertising proprietary services on Tier-2 machines, UPGWs can act as surrogate servers for the "real" servers, which reside on the proprietary system. Requests for proprietary application services, which appear to the system to be processed on Tier-2, are really forwarded by the UPGWs to servers on the proprietary system. To do this, the UPGWs need to have access to mappings of local service request name to proprietary server name, and a method of transforming data to the format of the proprietary systems.

The actual method of interface to the proprietary system is encapsulated within each UPGW. For gateways which interface to the proprietary system via terminal emulation, only a Tier-2 side gateway need be written. In this case, servers on the proprietary side are really terminal-bound processes, and the gateway needs to convert inputs and outputs to terminal format via a mapping language. Such a case is shown between the Tier-2 machines (UTPS-1, UTPS-2) and the PTPS-2 machine in Figure 5, where the protocol is 3270 emulation. For gateways which interface to the proprietary system via a program-to-program interface, for example IBM's LU6.2 [IBM-2], it is likely that a peer gateway on the proprietary side needs to be provided to yield the request/response paradigm available on Tier-2. This case is depicted in Figure 5 for the PTPS-1 and PTPS-3 machines, each of which has a gateway partner for the gateways on the connected Tier-2 machines. Of course, it should be possible to mix both program-to-program and terminal emulation encapsulations, even to the same machine, within one application.

4.2 Tier-3 API

The location transparency of requests originating on Tier-1 and Tier-2 machines means that Tier-2 processes should not be aware that their requests are processed on a proprietary mainframe. The service could migrate from Tier-3 to Tier-2, and the requesting module should see no difference.

Several choices for an API on the Tier-3 machines themselves are present:

- Provide the same API as in server modules of Tier-2.
- Accommodate the semantics required of Tier-2 interactions, but in a syntax more natural to the proprietary system, e.g in its native TP API.

Unlike the workstation case, where there are not existing APIs for TP, the proprietary TP systems have APIs for TP. So, the choice here is not an obvious one (i.e. provide the same API as on Tier-2), and may depend on many factors, including the ease of implementation and the acceptance of a new API on the proprietary machine. Whatever the choice, automatic conversion of data formats is highly desirable.

4.3 Tier-3 Transactions

It is also highly desirable to have transaction semantics available across the Tier-2/Tier-3 boundary. For example, this allows Tier-2 and Tier-3 database updates to be bound into an atomic unit of work. The protocol for transaction control with a Tier-3 System will depend on that system, and, in general, will be proprietary. For those proprietary systems whose protocols are compatible with the two-phased commit with presumed abort protocol [Mohan], the interface to transaction control can be provided by the XA interface described in [X/Open-2]. To do this, all or part of the proprietary system could be regarded as a Resource Manager, and XA calls are made on the Tier-2 system by the UPGW (or other administrative processes) to drive the transaction protocol. The implementation of XA for the proprietary system is split between the invoking Tier-2 machine and the associated Tier-3 machine. Note that it is likely the case that transaction semantics would only be supported for gateways whose protocol is program-to-program.

4.4 Administrative Integration

It is also highly desirable to provide the administrator of an ETP system with tools to determine the status of the Tier-3 machines which are incorporated into the ETP system. A natural way to provide this is to have the gateway processes also serve as administrative surrogates for the Tier-3 machines. In this case, each UPGW is responsible for booting any software needed on the Tier-3 system for interactions with the Tier-2 system. Likewise, UPGW is the vehicle by which Tier-2 informs Tier-3 that Tier-2 is shutting down. Finally, UPGW also responds to administrative requests as to the status of the Tier-3 System. In this latter regard, it will generally be impossible to keep transparency, and the Tier-2 System will need a method to allow commands specific to the proprietary system to be passed through via UPGW to the proprietary system.

4.5 UPGW Instantiations

In addition to pairwise instantiations for particular proprietary TP systems, two instantiations of UPGW are of particular interest:

1. ISOTPGW. As ISO/TP [ISO/TP] moves towards reality, vendors will begin to make it available in their proprietary environments. Such a protocol then becomes the preferred method for interacting with proprietary systems. Basically, a generic gateway type, ISOTPGW, will be able to accommodate interactions with all ISO/TP conforming proprietary systems, although some customization may be needed for administrative purposes.
2. UUGW. A particular instance of a Tier-3 System might not be a proprietary TP system at all, but rather another Tier-2 System. Since Tier-2 systems have a well defined input port, the WSG described in section 3.1, an instance of UPGW, called UUGW, can be constructed by using the implementation of WS-ATMI for a UNIX workstation. Its partner is the WSG. A set of such gateways working in the opposite directions can provide complete connectivity between the two ETP "domains". The result is that it is possible to create a very large TP system composed of domains of administratively autonomous ETP systems. The domains are joined at the Tier-2 level. The construction of such large systems begins to require the application of more advanced techniques, including the use of a standards-based naming service and the incorporation of standards-based administrative services.

5. ETP Status

The development of a full three-tiered ETP system is now complete and availabe in TUXEDO Enterprise Transaction Processing System Release 4.2. The TUXEDO System is currently available from a variety of hardware and software vendors.

6. SUMMARY

An architecture which integrates multiple levels of computer processing into a complete TP system has been presented in this paper. The architecture provides for the migration of proprietary TP solutions to those centered around a network of computers running the UNIX Operating System, and allows users to take advantage of the trend towards decentralized operations, while protecting their investment in proprietary TP systems. All elements of this architecture are currently available.

7. ACKNOWLEDGEMENTS

We would like to thank the members of the TUXEDO project for their ongoing efforts in the realization of the ETP architecture described in this paper. Mark Carges, Jane Dwyer, Howard Elder and Glenn Rose provided valuable comments on drafts of this paper.

8. REFERENCES

[Andrade]	J. Andrade, M. Carges, and K. Kovach, "Building a Transaction Processing System on UNIX Systems", 1989 UniForum Conference Proceedings, pp. 167-176.
[USL]	USL, "TUXEDO® Enterprise Transaction Processing System Release 4.2 Product Overview", 1991.
[Bernstein]	P.A. Bernstein, V. Hadzilacos, and N. Goodman, "Concurrency Control and Recovery in database Systems", Addison-Wesley, 1987.
[IBM-1]	IBM, "Customer Information Control System CICS/DOS/VS, Application Programmer's Reference Manual (Command Level)", SC33-0077-5, Sixth Edition, July, 19876.
[IBM-2]	IBM, "Systems Network Architecture, Format and Protocol Reference Manual, Architecture Logic for LU Type 6.2", SC30-3269-3, Fourth Edition, December, 1985.
[INFORMIX]	INFORMIX, Inc., "Informix-SQL Users Guide", October, 1986.
[ISO-TP]	ISO/TC 97/SC 21C N 2274, "Information Processing Systems-- Open Systems Interconnection -- Distributed Transaction processing -- Part 3: Protocol Specification," March 1988.
[Landis]	Ken Landis, "UNIX Calms Wall Street Chaos", CommUNIXations, August, 1990, p.22.
[Mohan]	C. Mohan, B.G. Lindsay, R. Obermarck, "Transaction Management in the R* Distributed database Management System," ACM Transactions on Database Systems, December 1986, vol. 11, no. 4, pp. 378-397.
[ORACLE]	ORACLE Corporation, "SQL Language Reference Manual Version 6.0", February, 1990.
[Tandem]	Tandem Computers, "Introduction to NonStop SQL™", March, 1987.
[X/Open-1]	X/Open Company Limited, Transaction Processing Working Group, "Interim Reference Model for Distributed Transaction Processing", July 7, 1989.
[X/Open-2]	X/Open Company Limited, Preliminary Specification, "Distributed Transaction Processing: The XA Specification", April, 1990.

OpenEyes A Performances Monitoring Tool for UNIX-based Systems

Perry Fisch, Masayuki Hatanaka, Andy Law, Max Mattini
Unix Commercialisation Group (UniComm)
Fujitsu Australia
tel: 61-2-410 4556

Abstract

The scalability of Unix from the PC level to the mainframe level is certainly one of the reasons which makes UNIX popular in Open Systems installations. However, the scalability of the operating system should be accompanied with a similar level of scalability in functionality. For example, the system administration/management functions of a UNIX mainframe should be a superset of the management functions of a UNIX workstation.

This paper addresses the performance monitoring aspect of systems management. It is based on the experience gained by the UniComm group during the development of an advanced prototype of OpenEyes, a performance monitoring tool.

1. Introduction

Performance reflects the amount of work completed per unit of time. Improving the performance of a system involves getting more work accomplished in the same time through the more effective utilisation of the available resources. In order to improve the overall performance (e.g tunning) of a system, first we should understand the systems themselves. By monitoring performance, OpenEyes helps a user gain in the understanding of the system. Monitoring is the first step toward automatic tunning, and expert system based, capacity planning in the future.

In Open Systems, computers of different sizes, vendors and missions co-exist 'glued' by the network. Preferably, the commercial users want to see 'The Computer System' supporting their business in a transparent manner. The users do not want to be aware of all the multitudes of hardware and software components which comprise their system. This transparent vision of the system/network should also be reflected in all aspects of system management.

In figure 1 (below), the operator controls the hosts on the network from a X-windows workstation. The performance data from the monitored systems is stored in a file server for a later analysis. Functionally, the multi-host administration is centralised, while physically, the software is distributed over the network.

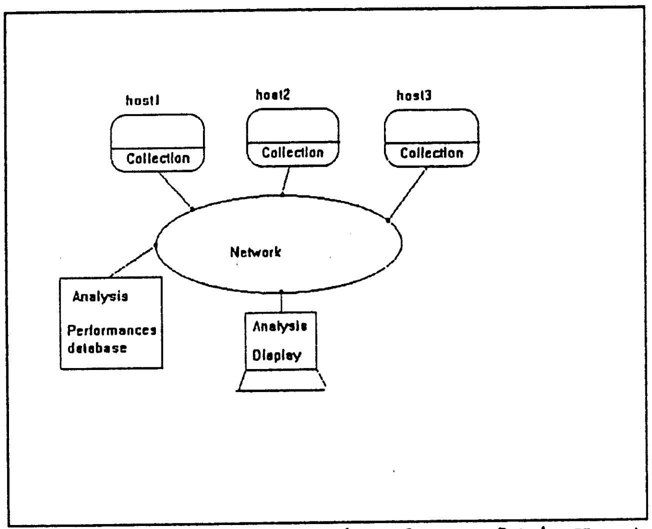


Figure 1 Centralised Multi-Host System Management

2. OpenEyes - Performance Monitor Tool

The OpenEyes prototype was used to monitor the performance and resource usage of Fujitsu's UTS/M or UXP/M hosts (UTS/M is based on UNIX V v2, while UXP is a mainframe implementation of UNIX System V v4.0). Up to 12 hosts can be monitored from one physical workstation. OpenEyes collects, analyses and records the relevant host performance information. It then displays this information to the user in a user friendly and graphical format.

Collector processes on the hosts (see figure 2) gather host information in an optimised fashion, while analyser processes on the workstation record, normalise and analyse the information. Finally, display processes on the workstation present the information using the OpenLook graphical user interface. The hosts and the OpenEyes workstation communicate via the TCP/IP protocols.

Two classes of monitoring capabilities are provided in OpenEyes: snapshots and trends. Snapshots provide a non-critical real-time 'picture' of the system. Trends show the evolution of performance across the time.

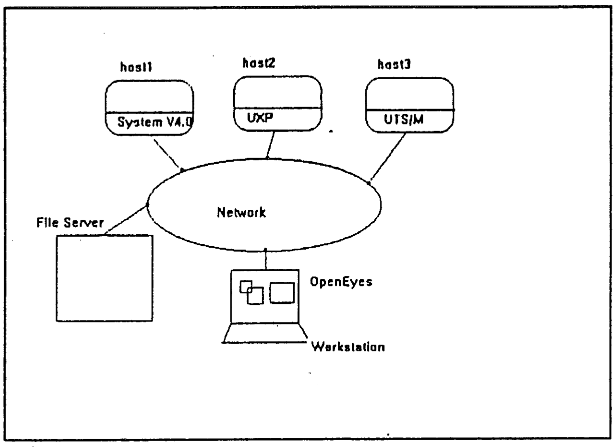


Figure 2 Open Eyes - A Centralised Performance Monitoring Tool

3. OpenEyes Framework

In order to support the concept of centralised multi-host management, OpenEyes is based on a modular framework (see figure 3 below). The framework isolates the hardware/system dependent aspects of performance and caters for emerging new hardware and operating systems. Tunning and capacity planning are not supported currently by OpenEyes but will be in the near future.

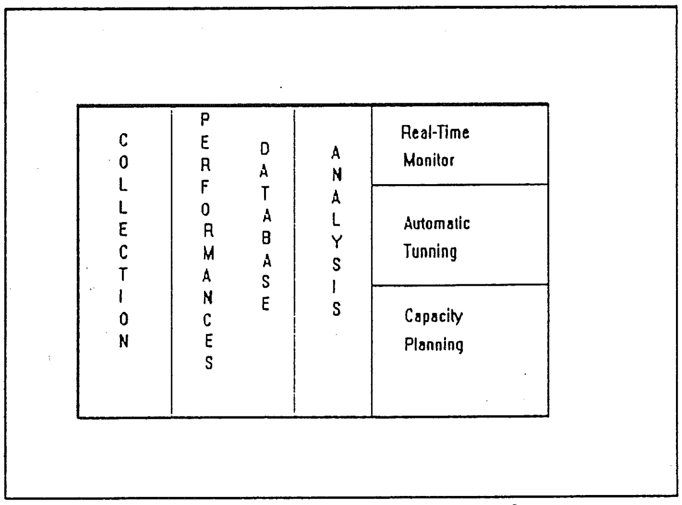


Figure 3 OpenEyes Framework

3.1 The Performance Database

"If it cannot be measured, it cannot be managed". The Performance Database is a collection of all performance and resource usage measurements, collected on all hosts during operation time. 'Operation time' means the time of operation of every monitored host in the system. The integrity and completeness of the Performance Database must be maintained in the event of a network failure.

The Performance Database is a logical database structured utilising the hierarchical concepts of subsystems, entities, and fields. The collection of related fields forms an entity, in turn, the collection of related entities forms a subsystem.

This logical database can be implemented using relational databases techniques or, later, using object oriented database techniques.

The structure of the Performance Database mirrors the major host subsystems:

- CPU subsystem
- Memory subsystem
- I/O subsystem
- Application/workload subsystem

3.2 The Collection Module

The task of the Collection module is the gathering of the raw data from the System V kernel or the various UNIX subsystems (such as a transaction monitor). Special care is taken in writing the collection module software so that the performance of the monitored host and the network are not affected.

3.3 The Analysis Module

The task of the Analysis module is to check the data received from the Collection module, to normalise the data, to insert the data in the database and finally to pass the data to the Display module.

Normalisation of data means the conversion of hardware/system dependent performance information into a uniform and logical format. Hopefully, in the future, international standards could define a common set of performance entities global to all hosts/operating systems.

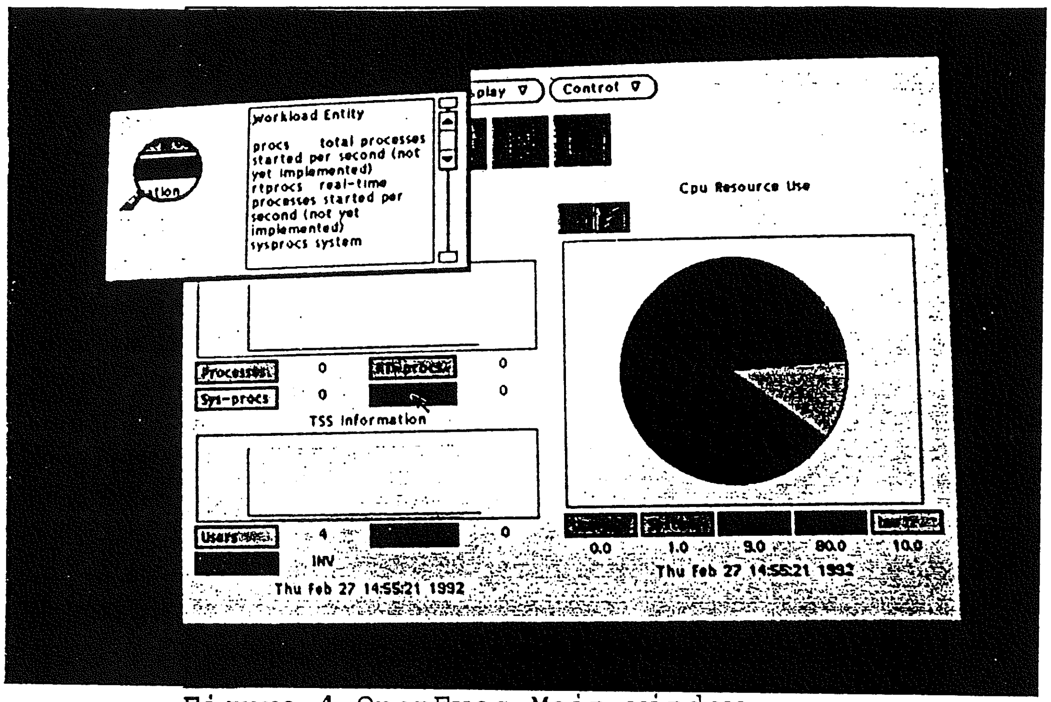
3.4 The Display Module

The Display module is the user interface module of OpenEyes. It displays the information gathered on the host and transmitted across the network. Internally, the Display module processes the information received from the host, in order to present it to the user in a clear and easily readable format.

4. OpenEyes Functionality

OpenEyes Provides:

- Centralised multi-host monitoring.
- A hierarchical view of performance.
- Snapshots and trends graphics.
- Threshold control.
- Customisation.



Workload Entity

procs total processes started per second (not yet implemented)
rtprocs real-time processes started per second (not yet implemented)
sysprocs system

Thu Feb 27 14:55:21 1992

Figure 4 OpenEyes Main window

4.1 Hierarchical View of Performance

Centralised multi-host monitoring involves the management of a huge amount of information. In practice, one operator will monitor the performance of up to 12 hosts (minis, mainframes and workstations) on the network. For this reason the information should be presented to the operator in a structured fashion, separated by levels of abstraction, as needed, and in a user friendly X-Windows environment. For example, instead of having 6 control consoles to control 6 hosts, only one workstation is used. Instead of having the operator to poll the consoles looking for abnormal behaviour, the operator can define some threshold conditions and be notified when abnormalities happen. Instead of being lost in a 'flood' of numbers and data, a hierarchical and iconic interface is provided allowing the user to navigate throughout the system.

Figure 5 (below) shows the file systems available on a given UXP mainframe. By selecting the file system icon, the operator can obtain the specific file system information.

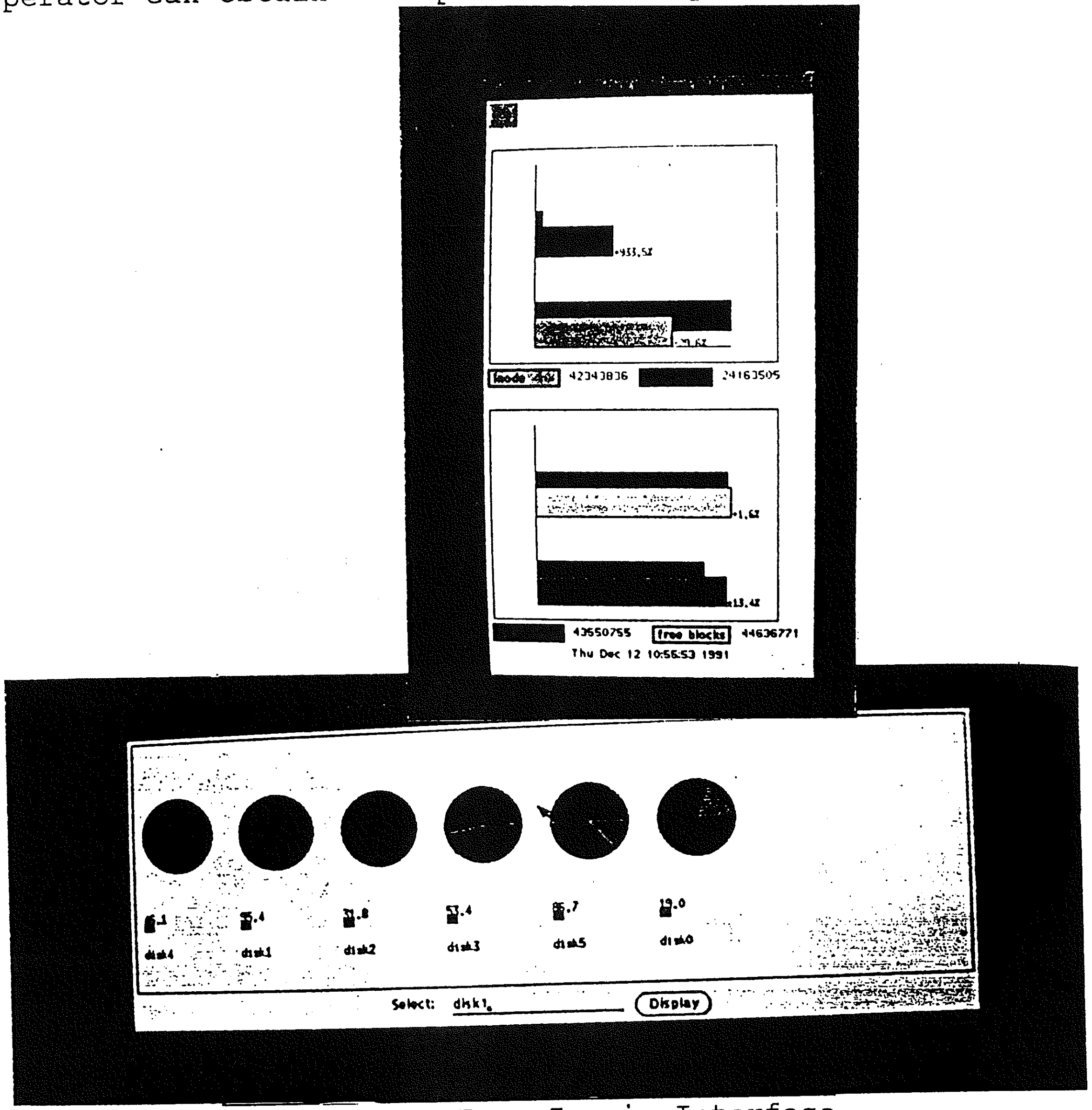


Figure 5 OpenEyes Iconic Interface for UXP File Systems

4.2 Snapshots and Trends

Figure 6 shows two classes of monitoring capabilities provided to the OpenEyes user: snapshots and trends. Snapshots provide a non-critical real-time view of the system. Trends show the evolution of performance and resources usage entities across the time.

Snapshots are of two types:

- Short period [5 second to 50 minutes]
  This type of snapshot determines the performance values during the last collection period (e.g the percent of CPU idle the last 10 seconds).

- Long period [1 hour to 24 hours]
  This type of snapshot gives the performance average during the long period (e.g the average percent of CPU idle for the last 12 hours).

By comparing the snapshots of the short period to the long period the user may detect abnormalities in the behaviour of the host. Both periods are user adjustable.

4.3 Threshold Control

The user can control the collection period of all of the entities. This is done simply by adjusting the sliders in the control window associated with each entity (see figure 7 below). In addition, the user can define the threshold values associated with every measured performance field.

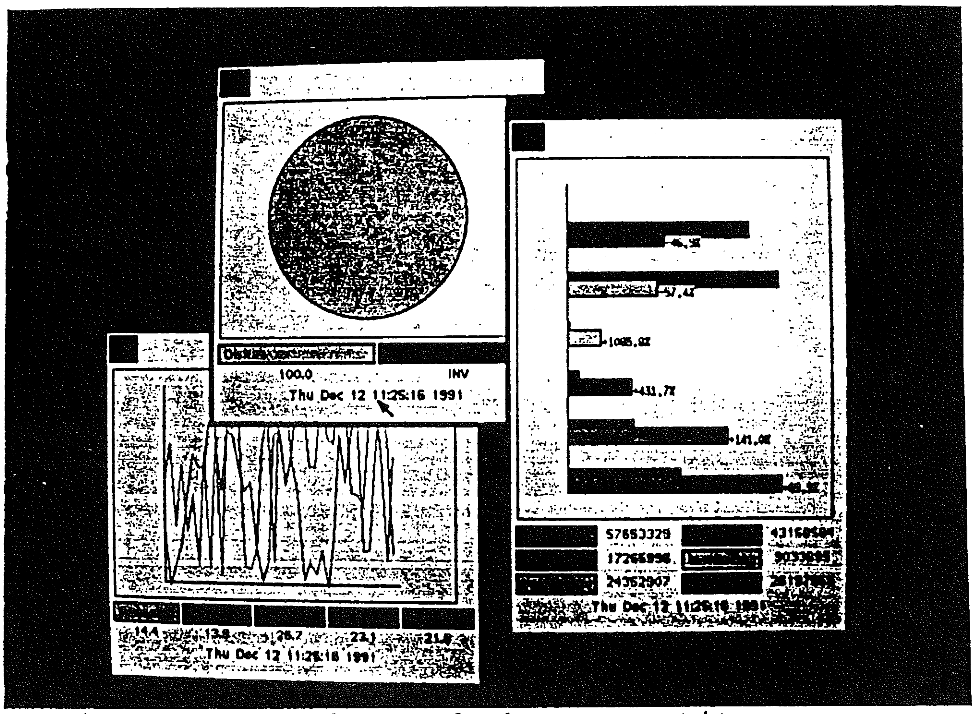


Figure 6 Snapshots of the CPU Entity

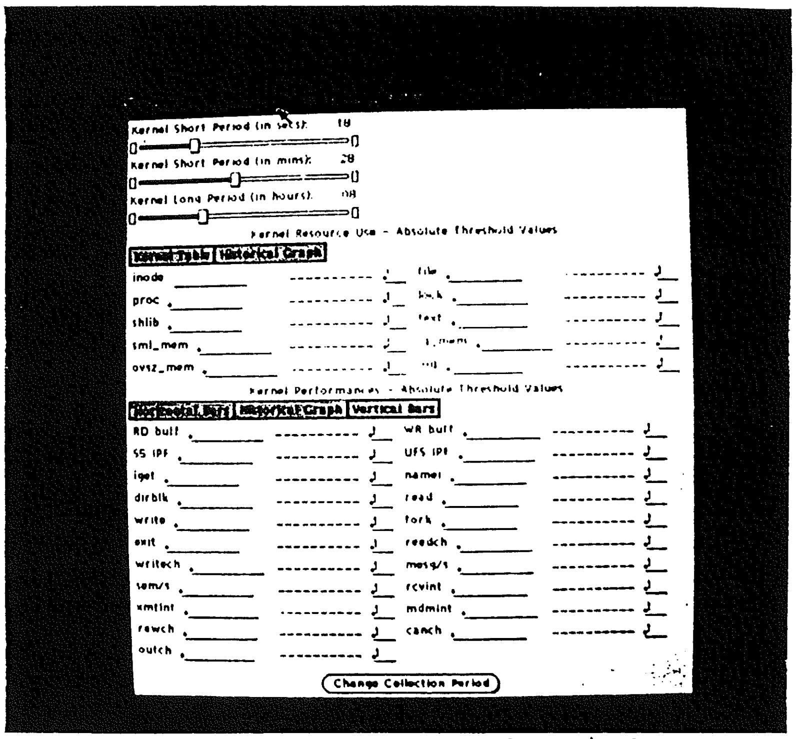
When the threshold value is violated the operator is notified of the event. An alarm/notification is triggered and is propagated. The alarm/notification mechanisms mirrors the logical database structure: the operator can detect/identify the origin of the alarm at the subsystem level, entity level or the field level.

5. Customisation and User Preferences

Customisation is of primary importance in monitoring a multi-host environment. This helps the operator associate the different graphical icons and windows with the different monitored hosts.

User customisation includes:

- The choice of entities to be monitored. For example, in a "panic" situation the operator may disable all monitored entities except the memory entity relieving the network from the added overhead.
- The style of graphical display (pie chart, histograms, bars) associated with each entity. Each kind of graphic presents the performance data in different way giving an added value to the raw data.
- The collection period associated with each subsystem.
- The threshold values associated with each field.
- Colour selections.



Kernel Short Period (in secs): 18

Kernel Short Period (in mins): 28

Kernel Long Period (in hours): 08

Kernel Resource Use - Absolute Threshold Values

inode		file	
proc		lock	
shlib		text	
sml_mem		l_mem	
ovsz_mem		[illegible]	

Kernel Performances - Absolute Threshold Values

RD buff	WR buff
SS IPF	UFS IPF
iget	namei
dirblk	read
write	fork
exit	readch
writech	mesg/s
sem/s	rcvint
xmtint	mdmint
rawch	canch
outch	

Figure 7 The Control Window: Collection Period and Threshold Values Definition

User preferences can be stored/retrieved across different monitoring sessions, relieving the operators from setting or resetting the environment every session and allowing them to tailor a default (standard) monitoring environment.

6. Conclusion

OpenEyes combines UNIX scalability, networking and the X-windows graphical user interface (GUI) to provide an elegant solution to centralised multi-host performance monitoring.

This iconic/graphical environment, presents the information to the operator in a 'easily absorbed' format making the understanding of system performance easier and user friendly. Finally, the measuring/understanding of system performance prepares the way for a more challenging task: automatic tunning and capacity planning.

Acknowledgments

OpenEyes is the first project of the Unicomm group in Fujitsu Australia. OpenEyes project's team members: Perry Fisch (Unicomm's Manager), Masayuki Hatanaka, Isao Hosokawa, Andy Law, Angus MacDonald and Max Mattini.

References

UNIX System V Release 4, System Administration's Guide
UNIX System V Release 4, System Administration's Reference Manual

Experiments in Network Distributed Image Synthesis

Steven Hayes
Postgraduate Student
steveh@godzilla.cgl.citri.edu.au

Michael A. Gigante
Director, ACGC
mg@godzilla.cgl.citri.edu.au

February 26, 1992

Abstract

The need for a network distributed image synthesis platform is discussed. An analysis of the features required for such a rendering platform is presented, and a series of experiments are performed with a number of software packages that provide protocols for distributed network computation. The packages invesitgated include SR, ISIS, PVM, C/Linda, Paragon and the Argonne Macros.

Introduction

Computer generated photo-realistic images are expensive to produce. The expense can be measured in terms of processing time required to render an image, and the memory required to contain the data base which describes the image being rendered. By distributing the data base across a network of workstations and mini-computers, it will be possible to quickly render very large, complex scenes.

A direct consequence of dividing the global scene description information over a number of networked computers is the need for frequent communication between participating nodes. This communication can lead to network saturation if not carefully managed. Using too many nodes in a networked computation can have adverse affects on the computation time. In our talk we shall discuss the affects of network overhead on computations , providing some quantitative results.

Evaluation Criteria

A number of criteria have been identified which the distributed image synthesis platform should meet:

- The platform should be able to function on a heterogenous network of UNIX workstations and mini-computers. Implicit in this requirement is the need for unambiguous data sharing between different processor architectures. For example, a network of workstations may include some machines which use a big-endian processor architecture, while others use a little-endian architecture. Ideally source code compiled for one machine architecture should compile on a different machine achitecture with no source code modifications necessary.

- The distributed processing package should be sufficiently high level so that existing image synthesis software may be converted to run as network distributed software without requiring a total re-write. The programming model espoused by the distributed network processing package largely determines the ease with which such modifications to existing software are made.

- A controlling process must be able to preside over all of the nodes involved in a distributed image computation. Work must be carefully allocated to nodes, ensuring that no node is left idle when other nodes are over-burdened with work. Additionally, the controlling process must be able to single out particular nodes for special tasks. For example, there may be some computations involved during image synthesis which are particularly well suited for execution on a 8 CPU shared memory machine. It would be silly and inefficent to allocate this task to a single CPU work station. Where a task may be performed on any available node, dynamic process allocation should occur.

- Error handling and robustness of the distributed computation platform are mandatory. It is particularly important that the controlling node be notified quickly if a process node is unable to perform an allocated task. A rejected task should be rapidly re-allocated to another node, as the task to be performed will often be a 'stepping stone' for some more complex task.

Overview of Packages

Six packages that allow network distributed parallel computations to be performed are discussed below. Some of the packages will be excluded from further consideration for the image synthesis platform because we believe that they are inherently ill-suited to the task.

SR

The Synchronised Resources [1] programming language is designed for writing distributed applications of any scale. SR supports only homogeneous networks of machines. This restraint, along with the requirement of recoding existing raytracing code in the SR language, makes SR unsuitable for our application.

Our experiences with SR have shown deficiencies in the way SR handles processes running on separate machines. Often processes would remain running after a distributed computation had been completed. Interfacing SR code to standard C library routines also proved to lack the required robustness. Additionally, SR modifies the run-time stack on each machine executing SR programs, requiring the use of assembly language. Porting SR to new platforms if difficult!

ISIS

The ISIS system, from Cornell University, is a series of low level C routines which place emphasis on fault-tolerant distributed computing. The ISIS pacakge, due to its low-level approach to distributed computation, suffers a gradual learning curve and is more demanding of the programmer than some of the systems reviewed here. Because of this reason, we have decided against using ISIS for our distributed platform.

Argonne Macros

The Argonne Macros [2], from Argonne National Laboratories, implement a variety of parallel programming models, including message passing and shared memory programming. The macros were designed to make parallel programs very portable, which results in the macros being quite low-level in nature.

We decided against using the Argonne Macros for the image synthesis platform because of their lack of abstraction of the distributed computation paradigm. We also encountered difficulties installing the macros that supported network distributed computation.

PVM

The Parallel Virtual Machine (PVM) system [5] is designed specifically as an environment for distributed programming on a network of heterogeneous machines. PVM is implemented as a library for C or Fortran programs running on UNIX machines. Data transfer between machines of desparate architecture is provided through use of the XDR libraries for machine independant data representation.

PVM provides ample control of processes running on network nodes through a set of library routines, and can allocate specific tasks to specific nodes with ease. The PVM system allows SIMD, MIMD and single CPU machines to all co-orperate in completed a single distributed task.

PVM nodes communicate with each other using messages, which ride on ethernet packets. As the volume of data to be transmitted acrosss the network incleases, the network overhead also increases, due to the fixed size ethernet packets.

Paragon

The Paragon system [3] is a set of C++ classes which cater for data parallel programming on shared memory machines or on a network of machines. The PVM system provides the underlying support required for paragon to operate over a network.

Data parallel programming essentially deals with arrays where each element may be processed in parallel. Parallel unary and binary operators are supported in Paragon using C++ operator overloading. Index one parallel structure by another is also allowed. Other programming systems which utilize data parallel structures include Fortran-90, and the C* programming language for the connection machine.

We consider the data parallel model of parallel programming ill suited to many image synthesis applications, particularly ray tracing and radiosity methods. The data parallel programming paradigm demands that old programs be completely re-written, which is a major violation of our criteria. Paragon is copyrighted public domain software.

Linda

The C/Linda system [4] implements an entire distributed programming system with a library of six function calls. Linda introduces the idea of a *Tuple Space* in which processes and data reside. A process to be executed is placed into the tuple space, where it will stay until a network node can remove and execute it. The allocation of processes to nodes under linda is completely dynamic. If a node has nothing to do, it simply looks in the tuple space for some work.

Programming C/Linda is very simple. Only four basic library calls are required. A small number of extensions to the standard C language syntax are made, allowing pattern-based query of data in the tuple space. Linda's abstraction of network distributed computation has removed a lot of low level control, and it is not possible to force a process to run on a particular machine.

Linda implements network communications using the UDP protocol, which offers low network overhead regardless of the amount of data being transmitted. The current versions of Linda only operate on homogeneous networks of machines, though this limitation is currently being addressed.

Linda can run on shared memory multiprocessors as well as over a homogeneous network of computers. Although the functions used for tuple space manipulation are identical for either Linda system, the programmer must be aware that Linda assumes a 'fork' programming model for shared memory machines, and a 'message' model for a network of machines.

Presentation

In this talk we will present out experiences with the SR, PVM and Linda packages. We will give quantitative results of a series of experiments designed to identify the relationship between distributed computations of varying complexity and the optimal number of network nodes which should be involved in these computations.

Acknowledgements

We wish to acknowledge Kim Libreri and Angus Montgomery for the work they did with the SR package. We would also like to thank Ken Musgrave for his discussions on the virtues of ISIS and C-Linda.

References

[1] Gregory R. Andrews and Ronald A. Olsson. Report on the SR programming language version 1.1. Technical report, Department of Computer Science, University of Arizona, Tucson, Arizona 85721, 1989.

[2] J. Boyle, R. Butler, T. Disz, B. Glickfeld, E. Lusk, R. Overbeek, J. Petterson, and R. Stevens. *Portable Programs for Parallel Processors.* Holt, Rinehart and Winston, inc., 111 5th Avenue, New Yourk, NY 10003, 1987.

[3] Craig M. Chase, Alex L. Cheung, Anthony P. Reeves, and Mark R. Smith. Paragon: A parallel programming environment for scientific applications using communication structures. Technical report, School of Electrical Engineering, Cornell University, Ithaca, NY 14853, 1991.

[4] Scientific Computing Associates inc. *C-Linda Reference Manual.* 246 Church Street, Suite 307, NewHaven, CT 06510, 1990.

[5] V.S. Sunderman. PVM: A framework for parallel distributed computing. Technical report, Department of Math and Computer Science, Emory University, Atlanta, GA 30322, 1990.

Terminal Interfaces †

Robert Elz
Computer Science
University of Melbourne

This short opinion on the properties that a terminal interface could have, or ought have, is intended to provoke a discussion along similar lines at the forthcoming UNIX Users' Group meeting in September, with a possible outcome being the standardization of

a) a basic terminal driver interface, to both the user and his programs.
b) a uniform and accepted method for various installations to make their local modifications (eg: to suit a particular local terminal type.)
c) a standard set of system calls, since the current stty, and gtty are clearly inadequate.

Before continuing, I must say that the following views are personal, and do not necessarily reflect those of other Melbourne University users. Also, I have not yet had the opportunity to examine UNIX Version 7 nor UNIX V32, so the comments contained herein should be understood to apply to Version 6 UNIX only.

Some of the 'features' discussed below are currently incorporated in the Melbourne tty driver, others are under consideration for a forthcoming version, the remainder are either not really useful, or too difficult to implement in a UNIX setting to be really serious contenders.

1 Motivation

Before getting down to the nitty gritty, I am inclined to spend a paragraph or two justifying the amount of time I invest in constant meddling with the terminal interface.

Apart from such pragmatic considerations as the gross inefficiency of the original Bell design for Interdata hardware, and the apalling jumble of code in 'canon()', my principal motivating force is my view that the terminal driver is the single most important component of a timesharing system to the user of that system.

† Re-printed from AUUGN Voume 1 Number 5

Others regard the editor, or the command interpreter as being more signifigant, however I feel that however marvellous such utilities may be, the user will never be truly comfortable unless he is able to communicate with them in the manner with which he feels most at home. After all, while a user may spend 50% of his time in the editor, and most of the remaining 50% using the command interpreter, <u>all</u> of his logon time is spent using the terminal.

2 What the terminal interface should offer the user

a) There must be a facility to delete the previous character on the current line, and the user ought to be able to obtain visible confirmation that the character has indeed been deleted. On a CRT this should be accomplished by actually erasing the character (with proper allowance made for erasing tabs). On hard copy terminals (or storage screen types, like the Tektronix) the user ought to have the option of seeing the character deleted if he desires (not always - it takes both time and paper), or of simply getting the erase character echoed in some visible form. An attractive idea for erasing on hard copy terminals would be to write a cross ('\' ← '/') over the character deleted, if only it weren't for the fact that most hard copy terminals run at 300 baud (or less). In all cases, if there is nothing left to erase, then nothing at all should happen on the terminal.

b) There should be the ability to delete the whole line and start it again. On a CRT it would be ideal if the line could be made to vanish, on other terminals the user ought to have the option of whether a new line should actually be taken or not. Because of the drastic consequences of an accidental use of this command, it ought to be reversible (most usefully using the erase mechanism above), though possibly with restrictions on when this may be done. If the line delete character is erased, the terminal should be restored as nearly as is possible to the state it was in before it was entered (ie: the 'un-killed' line ought to re-appear).

c) There should be a mechanism to have the contents of the current line redisplayed. If there is a deleted line which could still be recovered, an indication of this ought to be given (perhaps its contents too).

d) There should be mechanisms for signalling a process from the terminal and for indicating that there is no more input to be sent.

e) It should be possible for a user to temporarily suspend output from being sent to his terminal, and (of course) to restart it again. This is particularly important for terminals which decide for themselves when they have had enough, and send a command to the processor to stop sending.

f) There should be a means to divert output from the terminal to another destination (especially to the bit bucket) without the consent of the issuing program.

g) There should be an escape mechanism to permit any of the above commands to be entered as data.

h) All of the command characters should be re-definable by the user at will, especially those mentioned in (e) (which vary from terminal to terminal) and (a) which is the most often used.

i) It should be possible to have all characters typed on the terminal echoed in some visible form, that is, there should be an option to prevent there from being any character that could be treated as input to a program, and yet leave no sign on the terminal that they have been typed.

j) The terminal should be able to operate in any of three modes.

(i) continuous output.
(ii) fixed paging.
(iii) variable paging.

(i) is the same as the way most terminals are used on UNIX now.

(ii) causes output to the terminal to be suspended every so-many lines, to give the user a chance to read it. Output should not recommence until the user requests it. Input from the user at this time should be treated as commands indicating what he wants done, and as a minimum he should be able to move to the next page, move some part of a page, and exit page mode (temporarily). The only time that this halt should not occur is when the line that fills the page was typed by the user. Whenever a new page is started the screen should (optionally) be cleared.

(iii) is similar to (ii) except that any input from the user indicates the next line is the beginning of a new page, rather than just an input line at the very end of a page.

k) Lines being typed should be able to be given priority over output from some program, if the user wants this. That is, program output should be suspended until the user finishes typing his line.

l) Terminals with limited line widths should be catered for by optionally folding long lines if this is not performed by the terminal hardware. In any case where folding takes place, the terminal handler should be aware of this and recognise that an extra line has been printed.

m) Terminals with limited character sets should be supported as fully as possible, in a manner that places the minimum of strain on the user.

3 Terminal / Program Interfaces

a) A program ought to be able to determine the modes in which a terminal is currently being used, and alter any it chooses to.

b) It should be possible to obtain a unique identification of the type of terminal in use, or inform the system of the type if the program deems itself to be a better judge.

c) There should be a standard set of terminal control characters to perform specific terminal functions (such as clearing the screen, or moving the cursor to the left). The interfacing software ought to take care of any mapping required (possibly into a character sequence.)

d) A process should be able to determine if there is any input from the user waiting to be read and determine if this amounts to a complete line or simply some stray characters. It should be possible to discard typeahead, but this should not be a by-product of other alterations.

e) It should be possible to define a set of characters to be line terminators, which delimit 'lines' in the users input.

f) The terminal's width and depth should be able to be ascertained, as should the current characters being used by the user for control purposes.

g) Output delays for the various usual functions requiring them should be individually settable over a generous continuous range. A program ought to be able to specifically request a delay at any point in the output stream.

h) Options should be independantly switchable - selecting a particular format should not imply that some other attribute either must or cannot be selected.

4 Local Modifications

When designing a terminal interface to be used in a number on enviroments, it is important to recognise the fact that individual installations will have to customize it for their own particular requirements.

This may amount to no more than the deletion of unnecessary code, eg: if there are no terminals using any character other than the standard for some particular terminal function, then the mapping ought to be discarded.

In other cases, the pecularities of a particular terminal may warrant the addition of some code, eg: to output the escape sequence to return a particular type of terminal to full duplex after an especially foolish user has sought out and maliciously depressed the 'break' key, which is carefully hidden adjacent to 'return', or to prevent transmitting the relevant control codes to those terminals which will happily reply with the complete contents of the screen, unless the program to receive it all has indicated its willingness to accept this data.

Since such modifications will take place, it is best to allow for them by providing installation dependant interface mechanisms, and specifying only that some value is the default to allow portable software. This might take the form of 'empty space' in a system call data block, where zero is to represent the normal case, and any other value will invoke some installation dependant option.

5 Relationship with UNIX

The terminal interface offered by UNIX is not too far removed from the ideal situation for there to be any great difficulty in making some useful improvements.

This has indeed been done at many UNIX sites, but almost without exception the changes have been more in the nature of repairs than renovation. The best indicator of this is the 'stty' system call. In a heroic attempt to remain as close as possible to the released version of UNIX, most modifiers of the terminal handler have sought to remain with the standard 'sgtty' structure, and have been content with alterations like:

> "We don't really need backspace delays, so let's use that bit for ....."

I propose a total redesign of 'stty' to permit a much larger parameter area, since there is just so much that can be packed into 48 bits, and it isn't enough. Interdata UNIX allows 96 bits (3 integers, 32 bits each) and the current Melbourne terminal handler makes use of all but about 4 of them, and is still missing a lot that I regard as essential (even without the frills).

It may be appropriate to provide extra system calls, and assign each a particular role, dividing the control information among them upon a basis of likelihood of use, so that information rarely needed would not be being continually moved around. (Eg: very few programs have the slightest interest in the speed that the terminal runs at, and those that currently do usually only want to calculate how many fill characters are necessary to simulate a delay. If there was a means to request a delay of a specific length, then effectively no program at all would want to examine the terminal speed. This information would be better confined to the programs that really require it.)

At the present time, I have no fixed ideas as to what ought to replace, or at least augment the 'stty' and 'gtty' sys calls, but there should be at least 64 bits of on/off flags; 8 bit delay values for new lines, tabs, carriage returns, form feeds, vertical tabs (and probably backspaces); about a dozen user level control characters (erase, kill, interrupt, quit, eof, suspend, continue, discard, redisplay, escape); character sequences for clear the screen, simple cursor movements, and bell ringing; an array of 128 bits to specify line terminating characters; 8 (or 4?) bit fields for transmitter and receiver speeds; and lots of gaps for things we haven't thought of yet. In all a minimum of the order of 400 bits.

6 The Current Melbourne Interface

As mentioned above, 'stty' and 'gtty' are still used at Melbourne, in much the same form as in the standard Version 6 UNIX, except that there is twice as much space. This excess has been used to allow 32 flag (mode) bits (which still include the delay choice), 6 settable user interaction control characters, and two speeds (though Interdata hardware utilizes only one of them, there is no 'split speed').

Following an outline similar to sections 2, 3, and 4 above, there follows a brief summary:

6.1 User Interface

a) Erase is much like the UNIX standard, except that the actual erasing is performed when the erase character is read, rather than when a program requests the line. This means that on a CRT the erased character can be made to vanish (and it is, except for tabs, which are not handled correctly at all). On other terminals, the user can choose either to see the characters that are being erased, enclosed in '#' delimiters, or to have his 'erase' character echoed. If that character is 'rubout' (ASCII 0177) he may choose to have it displayed as a '#' instead. If there are no characters in the line to erase, nothing is echoed.

b) The current line may be deleted (as in any other UNIX). It is possible to request that a 'newline' sequence follow the echo of the 'kill' character if you want that. The 'kill' may be erased, but only if the character immediately following the 'kill' is 'erase' (there are a couple of exceptions actually). If this is done, then, if a new line was taken the 'un-killed' line will be echoed, and the cursor left at the end of it, otherwise the standard erase sequence is performed, illustrating the 'kill' character being removed.

c) It is possible to redisplay the current line. If there is a killed line that could be ressurected, then that line is displayed (it appears in the same manner as if the user had just typed it).

d) Interrupt and quit are as in standard UNIX. It is possible to use the terminal 'break' key to signal interrupts (and it can be left enabled in 'raw' mode) as an alternate to the ordinary character. If 'break' is to be the only interrupt key, then both 'interrupt' and 'quit' can be set to the same value, the driver guarantees to signal 'quit' if the character is typed. End of file is as in standard UNIX.

e) It is possible to suspend output, and resume again later. Both 'interrupt' and 'quit' also end a suspension. Input typed while output is suspended is echoed when the suspension is lifted.

f) Output cannot be discarded nor redirected.

g) Any of the characters above can be escaped, except 'interrupt', 'quit', 'pause', and 'suspend'. Missing the latter two is a definite bug, I am unsure whether is should be possible to escape 'interrupt' and 'quit' characters.

h) The six characters that can be altered are 'erase', 'kill', 'interrupt', 'quit', 'end of file', and 'escape'. There are two choices for each of 'suspend' and 'pause' (to correspond to the terminals that we use) - either character may be used. 'Redisplay' is control-A, and cannot be altered.

i) The only control characters that are ever graphically displayed (other than in performing actions such as erasing a character) are ESC (ASCII 033), which can be displayed as a '\' if desired (and it always is) and RUBOUT (ASCII 0177), which can be displayed as a '#'. These conventions arose from the common use of ESC as the terminal 'escape' character (replacing the much overworked '\') and RUBOUT as 'erase' (replacing '#' which is too hard to type).

j) There is no form of paging.

k) There is no line folding.

l) Output from a program cannot be restrained from interrupting the users input line.

m) Limited character set terminals are handled much as in standard UNIX, but the user has the choice of whether he wants the pre-translated character echoed, or the post-translated character (to make life easier on terminals that can display lower case but not transmit it).

6.2 Program Interface

a) 'Stty' and 'gtty' operate as in any other UNIX.

b) Terminal identification is not available.

c) On the one terminal type that does not use backspace (ASCII 010) for cursor left, a translation is made. No other common sequences are handled.

d) There is no way to determine if the user has typed anything without attempting a 'read'. It is possible to discard typeahead ('stty' as is usual), but it is also possible to cause 'stty' to retain typeahead, if the terminal speed is not altered, and there is no switch between 'raw' and 'cooked' modes.

e) Lines are terminated by 'newline' or 'end of file' as is usual.

f) Page width and depth are not known to anyone, except the poor user who must explicitly specify them wherever a program needs the information.

g) Delays are close to the UNIX standard, except that there are no backspace delays, but there are three different vertical motion delays. Programs cannot request delays (other than by emitting a relevant character).

h) Flags are comparatively independant. In particular, 'raw' mode does not disable 'crmod' or 'lcase' or just about anything else.

6.3 Local modifications

There are even a few 'features' that are considered peculiar to our installation, with both the possibilities mentioned in section 4 being included, as well as a peculiar 'fast' mode akin to 'raw', but supposedly more efficient for reading large amounts of data at high speed, and a mechanism to put the line into a 'space' condition for about 250 milliseconds, both intended for some form of intermachine communication over RS232 lines.

7 Conclusion

I have attempted to keep my ideas within the bounds of reasonable possibility for a system like UNIX, and have resisted the temptation to include extras like the ability to retrieve previous lines, or the 'the way it looks on the terminal is the way I want it read' philosophy, which has much merit, but is not easy to implement.

There are doubtless countless other small facilities that would be useful if included, but which I haven't ever encountered and thus remain ignorant of. If any reader has any suggestions I would be grateful to learn of them, either at the Users' Group meeting, ot at some other time, whether they be minor improvements, implementation methods, or straight out laughter.

The Melbourne terminal driver is available to anyone who would like a copy (send me a tape), but you should be warned that there is much in it that cannot be made to work on a PDP-11, (though nothing at the level of this discussion - mostly at the hardware interface stages) and that there is a major revision in the offing. I intend to include paging, and line folding, and probably variable line termination if I can make them all fit. This will certainly require modifications to the 'stty' format so I will not embark on this until after the Users' Group meeting, in case there is a decision made as to what things really ought to be like.

An Update of UNIX-Related Standards Activities †

Stephen Walli <*stephe@mks.com*>
Report Editor
USENIX Standards Watchdog Committee

Report on the IEEE Standards Board

An Anonymous Friend of USENIX reports on the December 3-5, 1991 meeting in New York, NY :

[Ed. - The report writer asked to remain anonymous. Anyone wishing to send comments to the writer may do so through me. - SW]

The IEEE Standards Board is the committee responsible for overseeing all standards related activities within the IEEE. The IEEE produces standards for the entire electrical engineering spectrum, not just the Computer Society. The Technical Committee on Operating Systems — Standards Subcommittee (TCOS-SS), is the IEEE Computer Society committee responsible for the POSIX family of standards.

As usual, the December 1991 meetings of the IEEE Standards Board produced a plethora of new Project Approval Requests (PARs) and approved projects, some new rules to apply to the standards process, and one more new Organizational Representative that can ballot POSIX standards.

Acknowledgments

Perhaps the new rule that most impacts the IEEE community is one concerning the use of acknowledgment sections in standards. You've probably seen one of these sections before; they're the ones that thank your company/university/organization/mother for providing the means for you to contribute your thoughts and ideas to that lovely thing known as the standards process. It's usually found in the front or back of the standard in what we jargon-savvy folks know are informative sections of the document, so it's not part of the official standard. (Don't confuse it with the foreword or introduction, which discuss the technical and historical development of the standard and list the working group and balloting group.)

The IEEE Standards Board Procedures Committee (whew! that's ProCom for short) felt that the IEEE could be legally liable if the standards mentioned a company without first asking their permission. A policy was proposed that said a working group could include one of these sections if each member obtained written permission from the companies/etc. involved, to be kept on file with the IEEE Standards Department. There are form letters for you to follow, but nonetheless it's an extra step you'll have to take.

Of course, the question comes up as to whether you should be doing this work at all. What if one company says yes and 20 say no? Do you have an acknowledgments section that only lists a few companies? For a family of standards like POSIX, should some standards have this section and some not? As always, things rapidly get complicated. Because of this, the POSIX technical editors had a lengthy discussion on whether to have these sections at all in their documents. Opinion was wide-ranging and varied; the interim decision was to go to our individual working groups and ask for their opinions. Based on those discussions, the technical editors will decide whether to keep these sections in the future.

The Curse of Acronyms

As we all know, standards-writing groups have a seemingly inexhaustible ability to create acronyms. Indeed, after a while our conversations seem to consist entirely of abbreviations, and woe betide the person who tries to understand our arcane internal code.

Of course, the Standards Board has to do just that when they look at our PARs (oops! that's Project Authorization Requests). They understandably get frustrated. Because of that, the New Standards Committee (NesCom) has said that they don't want to see incomprehensible acronyms on PAR submissions in the future. The NesCom members come from all societies of the IEEE, not just Computer, and many power-engineering standards developers can't begin to guess at what an acronym means that you've used since the first time you touched a keyboard.

This means we'll have to get used to standards titles that are even longer than they are now! When filling out a PAR, you'll have to remember to expand acronyms appropriately. You wouldn't want to have the PAR rejected on these grounds. This subject will be discussed in more detail at the next NesCom meeting.

One Man, One Vote

Questions have arisen as to whether or not members such as Institutional Representatives and similar reps in the power engineering realm vote twice on a document, once as an individual and possibly again representing their organization. The Board agreed to appoint an ad hoc committee to look at the issue of one man, one vote. More information should be available from forthcoming meetings.

† Re-printed from login: Volume 17, Number 3

In other IR related news, SPARC is now officially approved by the IEEE Standards Board as an IR and has the right to vote on all POSIX documents.

[Ed. — The following lists are provided to allow the reader to appreciate the full breadth of control the IEEE Standards Board has as its mandate. These are still just the Computer Society related standards. The reader should note P1279, P1281, and P1282. Andrew Hume regularily warns in his ANSI WORM standards reports that the WORM standards may have a far broader impact than people think. Here, in P1282, we even see them "worming" their way into POSIX.1 (ISO 9945-1).]

And here's the information on Review Committee (RevCom) and NesCom Computer Society activity:

Approved New Computer Society PARs

P1278 (C/SCC) Standard for Information Technology-Distributed Simulation Applications-Process and Data Entity Interchange Formats

P1279 (C/SCC) Standard for Information Technology-CD-ROM Architectural Profile

P1281 (C/SCC) Standard for Information Technology-Use of ISO 9660: 1988 System Use Fields

P1282 (C/SCC) Standard for Information Technology-Interchange of ISO 9945-1:1990 Filesystems via the ISO 9660: 1988 File Structure

P802.1j (C/TCCC) Standard for Managed Objects for MAC Bridges (Supplement of 802.1D)

P802.1k (C/TCCC) LAN/MAN Management Information for Monitoring and Event Reporting

P802.2C (C/TCCC) PICS Proforma for LLC Type 1 Operation and LLC Type 2 Operation

P802.1D (C/TCCC) Technical and Editorial Corrections to Std. 802.1D

P802.2f (C/TCCC) Standard for LLC Sublayer Management

P802.6k (C/CC) Distributed Queue Dual Bus Subnet work of a Metropolitan Area Network, Supplement for MAC Bridging

Revised Approved Computer Society PARs

P1209 (C/SE) Recommended Practice for the Evaluation and Selection of CASE Tools

P802.1F (C/CC) Common Definitions and Procedures for 802 Management Information

P1155 (C/MM) Standard for VMEbus Extensions for Instrumentation: VXIbus

P1175 (C/SCC) Trial Use Standard Reference Model for Computing System Tool Interconnections

P1396 (C/MM) Standard for a Communication Bus (TELECOM Bus): Reference Models

Withdrawn Computer Society PARs

P1101.5 (C/MM) Standard for Mechanical Core Specification for Microcomputers-Desktop Form Factor

Approved New Computer Society Standards

610.6 (C/SCC) Standard Glossary of Computer Graphics Terminology

1029.1 (SCC20 & C/DA) Standard for Waveform and Vector Exchange (WAVES)

1175 (C/SCC) Trial Use Standard Reference Model for Computing System Tool Interconnections

P1212 (C/MM) Standard Control and Status Register (CSR) Architecture for Microcomputer Buses

Withdrawn Computer Society Standards

IEEE Std 662-1980, IEEE Standards Terminology for Semiconductor Memory (ANSI)

Report on POSIX.0: The Guide to Open Systems

Kevin Lewis <klewis@gucci.enet.dec.com> reports on the January 13-17, 1992 meeting in Irvine, CA:

The POSIX.0 working group adjourned the October meeting wondering what the mock ballot would yield. This uncertainty was focused not only on the size of the return, but also on whether there were any hidden or significant issues lurking in the darkness.

Twenty six mock ballot responses were received: 13 users, 9 producers, and 4 general interest participants. This reflects a healthy balance. In total, there were approximately 760 objections/comments. Some ballots covered specific sections, while others addressed the entire guide.

It appears that the issue of "public specifications" that has been lurking around in other venues has arisen here. For those of you not familiar with this, I cannot do it justice here. Suffice it to say that it involves the use within public procurements of specifications that are not currently in the formal standards process but which have widespread industry use and acceptance.

POSIX.0 feels that such specifications are acceptable under specific conditions which include consensus, availability, lack of encumbrances, and proper documentation. (There is much, much more to this, so get a copy of the guide or call someone in POSIX.0 if you are interested or concerned.)

The decision was made in January to move forward on the formal ballot. POSIX.0 has notified the IEEE and a letter forming the formal ballot group will go out in the March-April time frame. The goal is to begin the formal ballot in July. In parallel, POSIX.0 will be submitting the guide to the international standards community in order to obtain review and comment and to prepare the way for it as an ISO Technical Report.

Report on POSIX.2: Shell and Utilities

David Rowley <david@mks.com> reports on the January 13-17 meeting in Irvine, CA :

Summary

The end is in sight. POSIX.2 (Shell and Utilities) Draft 11.2 closed its recirculation ballot last October 21. Draft 11.3 is due out any day now. A full draft (Draft 12) will be recirculated to the IEEE working group before the final standard is adopted. POSIX.2a (UPE) Draft 8 closed its recirculation ballot on January 24. Both standards are expected to be approved as full-use IEEE standards at the September meeting of the IEEE Standards Board.

Work on POSIX.2b continues, including the contentious new file format for PAX and extensions to the POSIX.2 utilities to handle symbolic links.

The first cut at test assertions for POSIX.2 has been wrapped up, and assertions for POSIX.2a have begun.

Background

A brief POSIX.2 project description:

POSIX.2 is the base standard dealing with the basic shell programming language and a set of utilities required for the portability of shell scripts. It excludes most features that might be considered interactive. POSIX.2 also standardizes command-line and function interfaces related to certain POSIX.2 utilities (e.g., popen(), regular expressions, etc.). This part of POSIX.2 , which was developed first, is sometimes known as "Dot 2 Classic."

POSIX.2a , the User Portability Extension or UPE , is a supplement to the base standard. It standardizes commands, such as vi, that might not appear in shell scripts, but are important enough that users must learn them on any real system. It is essentially an interactive standard, and will eventually be an optional chapter to a future draft of the base document. This approach allows the adoption of the UPE to trail Dot 2 Classic without delaying it.

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

POSIX.2b is a newly approved project which will cover extensions and new requests from other groups, such as a new file format for PAX and extensions for symbolic links.

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

Hal Jespersen, Chair of POSIX.2 , is about to send out Draft 11.3. This is likely the last "changes-only" draft of POSIX.2 .

The POSIX.2/D11.2 recirculation ballot closed October 21, and resolution of ballot objections has completed.

Balloting of Draft 11.2 has been held open pending the arrival of ISO comments. All changes for the next draft (11.3) will be forwarded to ISO through the US TAG.

It is expected that a final draft 12 of POSIX.2 will be made ready in time for the May WG15 meeting in New Zealand, and should be approved as a Draft International Standard.

The technical content of the standard has more or less stabilized. Most recent changes relate to clarifications in wording.

POSIX.2a Status

POSIX.2a is also coming down the home stretch, as the technical content has stabilized. Ballot resolution for POSIX.2a (UPE) Draft 8 was completed. The ballot closed on January 24. The next draft will likely be a quick "changes-only" recirculation, labelled draft 8.1. It should appear any day now.

The ISO ballot ends in April. All comments will be rolled into a Draft 9 which will be produced in time to be carried to ISO in May for approval as a Draft International Standard (DIS).

Hal Jespersen expects that both standards should be given final full-use IEEE approval at the September meeting of the IEEE Standards Board.

Internationalization Inadequacies

Randall Howard, President of MKS, put forward a proposal to the POSIX.2b working group to define a system API to the internationalization information embodied in a POSIX.2 locale. Routines to access collation elements, detect membership within a character class and extensions to the strftime() call were presented. The group felt that since it was a system API, not a utility, it rightfully belongs in POSIX.1 . When the same presentation was given to POSIX.1 , they expressed the opinion that parts of the proposal were better suited to the ANSI or ISO C Standard efforts. Unfortunately, they don't necessarily want it since they haven't (yet) adopted the POSIX.2 definition of a locale. This all demonstrates that the POSIX process cannot effectively deal with issues that cut across a number of working groups and/or standards. Perhaps the Systems Interface Coordination Committee (SICC) that has recently been formed within POSIX can help address some of these issues.

Comments on ISO 10646

The ISO working group that is responsible for the ISO 10646 character set standard (which now includes the Unicode work,) has asked the POSIX.2 working group for their opinion on their current proposal.

The working group expressed much concern over the use of null octets within the valid character codes. Since computer languages such as "C" make use of nulls as a string termination marker, a lot of existing code would have to be heavily modified in order to support the new standard. The working group was against the proposal for this reason. Apparently the ISO/ANSI C working group has expressed similar concerns.

Symbolic Links

Dawn Burnett from USL submitted a proposal on extending the POSIX.2 and POSIX.2a utilities to support symbolic links, based on the System V implementation. The problems that arise from symbolic linked directories were discussed at length. There is nothing more irritating than changing to a directory, printing the current working directory only to find that you have been magically warped to a completely different spot in the file system. A proposal to maintain both physical ("Where am I") and virtual ("How did I get here") paths was offered. The text will find its way into the next draft of POSIX.2b for further discussion.

Test Methods

Real progress was made completing the remaining test assertions for POSIX.2 , and beginning the POSIX.2a work. A style guide for writing consistent assertions has yet to appear, but the group seems to have found its stride and is working well.

Test assertions for the interactive utilities have yet to be tackled, but it is expected that it will not be as difficult as first anticipated. The assertions for `vi`, `talk`, etc. will describe (in precise English) what action must take place upon the stated input. The process whereby the results are verified will be left up to the test suite implementor.

New PAX Archive Format

Work continues on the new PAX archive format. A consensus is (slowly) starting to brew. The issue of supported filename code sets is a thorny one, especially since POSIX has not addressed any code set issues in a general way (such as adopting the X/Open `iconv` utility and API).

The problems stem from wanting to use the format to address both universal archive transportability as well as local file system backup and restore, one concentrating on a standard common ground, the other wanting the flexibility of representing the full local filename character set. This is the most contentious area of the format, and there will likely be much wailing and gnashing of teeth before the dust settles.

If you have any interest in this area, the group would be pleased to hear from you.

Report on POSIX.3: Test Methods and Conformance

Andrew Twigger <att@root.co.uk> reports on the January 13-17, 1992 meeting in Irvine, CA:

SCCT Matters

The Steering Committee on Conformance Testing (SCCT) met three times during the week and discussed a broad range of testing related issues. The major issues centered around fitting the test methods into the document structure, dealing with options in "base" standards, and test methods for profiles.

The higher level of document structure seems to have been resolved by introducing a parallel set of documents (and therefore project requests, or PARs) to the base standards. The test methods documents will be numbered by adding 1000 to the base standard number, i.e., POSIX.6 Security Extensions (P1003.6) will have test methods in a document numbered P2003.6. The IEEE would then resolve any accompanying publication issues.

The more granular issue of how to write assertions which can be easily merged along with the base standard was also briefly discussed, but not yet concluded. The integration of base standards (POSIX.1, POSIX.4, POSIX.6, etc.) is one of the major problems facing TCOS at the moment, but the solution seems as far away as ever. (The Technical Committee on Operating Systems – Standards Subcommittee, TCOS-SS, is the IEEE Computer Society TC responsible for the POSIX standards.)

From the test methods perspective, integrating assertions for a pervasive interface like open() introduces a considerable problem in defining which assertions relate to which base standard options. While solutions can be produced easily, these are generally inelegant.

The options issue, which was left over from the Parsippany meeting was readdressed with some further input from POSIX.1. The problem may not be as serious as previously thought and many of the issues can be resolved with some minor changes to POSIX.3.1 (POSIX.1 Test Methods). The remaining ones can be resolved by introducing an announcement mechanism, which most test suites have to provide, allowing the test suite to determine the implementation's option setting.

The SCCT reviewed the meaning of profile conformance and the use of conformance statements in profiles. They agreed that profile conformance statements should refer back to those in the base standards and should be validated by reference to the test methods for the base standards, where available, plus the specific test methods for the "mortar" defined in the profile. (The Profiles Steering Committee is reaching agreement on the rules for subsetting base standards, and how additional behaviour may be thought of as the "mortar" binding the standards together.)

Software Testing Environment BOF

On Monday evening BSI (the British Standards Institution) and Mindcraft called a Birds-of-a-Feather gathering to explain Software Testing International and the Software Testing Environment (STE). Software Testing International would be a non-profit organisation set up to administer the development of test suites for POSIX and other standards. Most of the attendees seemed reticent in their approval of the scheme, particularly when it became evident that Mindcraft would be the sole test suite authoring organization with a seat on the Board. Comments from the presenters that "POSIX testing is just starting to become serious" were also not well received. It seemed clear that both structural and perceptual changes would be necessary before the proposed scheme could make an impact in the POSIX testing arena.

The actual STE introduces an additional API layer on top of the current Test Environment Toolkit (TET), a freely available testing harness created jointly by X/Open, Unix International, and the Open Software Foundation. Initial impressions were that the main purpose of this layer is to allow Mindcraft's CTS based test suites to execute in the TET environment. (NIST is currently supporting the TET as their testing methodology of choice.)

Mindcraft promised to make the specifications available shortly and to provide an implementation at the end of quarter two. The testing community review the value of these extensions, but with significant aspects like distributed testing omitted it may not capture many peoples' imagination.

POSIX.3

The POSIX.3 working group continued in their relentless task of writing and reviewing assertions for the POSIX.2 (Shell and Utilities) standard. The latest draft (POSIX.3.2/D7) has been circulated for review and comment, though no comments have yet been received. At the end of the Irvine meeting it was expected that there would be no significant parts of POSIX.2 that were unaddressed by test methods, except its internationalisation aspects. The working group commenced the specification of test methods for POSIX.2a (UPE) towards the end of the meeting.

Other working groups were also developing test methods for their standards with progress being made by (at least) POSIX.6, POSIX.8, POSIX.12, 1224 and POSIX.17, as well as some of the profile groups. In general, these groups were developing a reasonable understanding of the task facing them, and in some cases good quality test methods have already been produced.

The question of language independent test methods was discussed in the POSIX.1 forum, though other groups (for example 1224) have also made progress in this area. The outcome of the POSIX.1 discussion was an estimate by a prospective contractor to undertake 2,000 or more hours of work to produce LIS test methods for POSIX.1. This looks like an exceedingly high estimate, and I would be very surprised if TCOS followed it up!

Report on POSIX.6: Security Extensions

Charisse Castagnoli <charisse@sware.com> reports on the January 13-17, 1992 meeting in Irvine, CA:

This was the first meeting of the POSIX.6 group since the ballot closed on January 6, 1992. Of the 232 official ballot members, 181 members responded. The response equals 78% of the ballot pool. (A minimum of 75% response is required by IEEE for a ballot to be considered valid.)

The 181 returned ballots were divided as follows:

Affirmative	Negative	Abstain
69	61	51
53%	47%	<don't count>

In order for a ballot to pass, there must be a 75% affirmative ballot. One would think this means 75% of the responses must be affirmative, but this is not the case. Only 75% of the non-abstaining votes need to be affirmative. Taken to an extreme, this means that regardless of the ballot pool size, if three people vote affirmative, 1 votes negative and the rest abstain, the initiative passes. The moral of the story is: abstain only as a last resort, there may be deleterious side effects.

The POSIX.6 committee is now divided into 3 groups: test assertions, new projects, ballot technical reviewers.

The test assertions group, led by David Rogers, is developing the companion document of test assertions. This is required to actually complete the ballot process.

The new projects group is working on new Project Authorization Requests (PARs). Three PARs were presented: one PAR for Identification and Authentication, one for data interchange, and one for terminal I/O.

In addition, PARs were prepared for the existing POSIX.6 functions. The current PAR for the existing functionality will eventually be transformed into POSIX.6a (Security Extensions to System Interfaces) and POSIX.6b (Security Extensions to System Utilities and Shell). [ed. — PARs are essentially administrative project control documents, but are becoming administrative nightmares in the IEEE standards development process.]

The ballot resolution group began reviewing the ballot objections. A preliminary analysis indicated that one common objection was lack of consistency within the ballot. Requests for consistency in function naming, calling parameters, data types, and return codes were frequent. After careful reflection, the ballot resolution group agreed this was a reasonable request and began to work out a set of guidelines to ensure consistency throughout the draft.

Highlights of the ballot resolution group discussions include:

> Should "set" and "get" be used for function names instead of "read" and "write?"
>
> Should data types be contiguous in memory? (That is, can a data object be copied with a bcopy()?)
>
> Should functions manage data storage and allocation or should the programmer manage them?

After arduous negotiations, the group developed a set of guidelines that resolved many issues that have plagued the drafts for years. The ballot resolution group will now join the State Department to support peace negotiations in the Middle East.

The ballot resolution group tested the guidelines by applying them to each of the primary functions in the draft. These functional areas are privilege mechanism, mandatory access control (including information labeling), and access control lists. The auditing functions were granted an exemption from this exercise, because they were being reviewed in light of the new data type guidelines and substantial interface modifications were expected.

Chris Hughes presented some options for new auditing interfaces. The existing interfaces, in addition to being inconsistent, lack good support for application level auditing. Additional work is needed on the auditing functions, and will be presented at the next POSIX meeting.

At the end of the meeting, we all agreed to try and complete the interface changes necessary to bring each function in line with the new guidelines. We also agreed to resolve as many ballot objections as possible before the April meeting.

Report on POSIX.14: Multiprocessor Profile

Rick Greer <rick@ivy.isc.com> reports on the January 13 - 17, 1992 meeting in Irvine, CA:

The multiprocessor working group plans to submit their draft profile to a mock ballot after the April 1992 meeting. Much of the January meeting was spent dealing with various trivialities in the draft in anticipation of the mock ballot. We did, however, encounter one major issue that could prevent the draft profile from ever becoming a standard. It seems that a draft profile cannot become a "POSIX Standard Profile" if it references documents which are not themselves official standards endorsed by a "recognized accrediting body."The POSIX.14 draft references both the POSIX.4 (Real-time) and POSIX.4a (Threads) drafts, as well as some ongoing ANSI X3H5 work defining parallel language facilities. It cannot become a standard profile until all of these make their way through the appropriate standardization mill.

The POSIX.14 profile is fairly simple, and likely to be ready for balloting long before its antecedent documents. This forces the POSIX.14 working group into one of a number of possible holding actions:

1. Hold up balloting the POSIX.14 profile until all of the referenced documents become standards. This will leave the working group with very little to do, except perhaps to work with the other groups to try and speed up acceptance of their work. Since most of the POSIX.14 working group are refugees of the POSIX.4a threads wars, there is very little enthusiasm within POSIX.14 for this approach.

2. Go ahead and ballot the POSIX.14 profile, but don't submit it to the IEEE Standards Board for approval until the referenced documents become standards. This gives the working group something to do over the next few months (i.e, work on ballot resolutions). In the long run it will only delay the inevitable: We are likely to run out of ballot objections long before the other documents become standards.

3. There are a number of "missing interfaces" that POSIX.14 would like to see added to POSIX.1 and POSIX.2 but, being a profile group, is unable to specify. What we can do is to recommend to other groups that they incorporate these interfaces into subsequent versions of their documents to better support multiprocessor operation. The general feeling within POSIX.14 is that if we do a thorough job of presenting well defined, well rationalized, multi-processor interfaces, the other working groups should pick them up with little argument (ha!). While waiting for the draft documents referenced by the POSIX.14 profile to become standards, the POSIX.14 working group could devote some effort to defining these missing interfaces.

We pretty much decided to go with holding action number 3 (primarily because it's more fun than items 1 or 2), but this course of action presents problems of its own. If we wish to include the missing interfaces into the profile, we will have to wait for them to become officially adopted into POSIX.1 and POSIX.2. This would, of course, put us right back where we started: waiting for referenced documents to become standards before the profile itself can be finished.

One way out of this dilemma is to include the missing interfaces in an appendix to the profile itself. Once the interfaces have become recognized standards, we can include them in the normative text in a later revision of the profile.

Report on POSIX.17 - Directory Services API

Mark Hazzard <markh@rsvl.unisys.com> reports on the January 13-17, 1992 meeting in Irvine, CA:

Summary

Once again, the POSIX.17 group made solid progress between meetings, completing all major homework assignments. The week in Irvine was a busy one. The Project Managment Committee (PMC) audited POSIX.17 and gave the group a clean bill of health. We also met with POSIX.12 furthering our discussion on a simplified API to the directory. Our Mock ballot input on the networking section of the Guide, POSIX.0 Draft 14, were reviewed with POSIX.0, with the promise that they will be reflected in the next draft. We completed processing input from our Mock Ballot of POSIX.17 Draft 2.0 and began drafting responses to our reviewers. We also identified work items and continued planning for an official IEEE ballot which begins April 7, 1992.

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 the joint XAPIA- X/Open 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. Draft 4 of P1224 has already entered IEEE ballot.

POSIX.17 is one of five "networking" groups that currently make up POSIX Distributed Services and as such, POSIX.17 comes under the purview of the Distributed Services Steering Committee (DSSC).

Status

The group chair was unable to attend the meeting in Irvine, CA, so yours truly once again assumed the duties of chair. There has been a low grumble about the ever increasing overhead associated with TCOS/POSIX working groups, and now I know why. A Project Management Committee audit Sunday morning, 2 Sponsor Executive Committee (SEC) meetings, 2 Systems Interface Coordination Committee (SICC) meetings, 2 Distributed Systems Steering Committee (DSSC) meetings, 2 Distributed Systems (DS) Plenaries, a Logistics meeting, a Distributed Security study and (I almost forgot) POSIX.17 working group meetings made for a noticeable lack of "spare" time.

Commitment within the group remains strong, with all other core members attending, and completing their "homework" assignments.

The TCOS Project Management Committee held the first audit of POSIX.17 on Sunday morning. The PMC recommended continued sponsorship of the work, splitting the work into two projects, increasing the size of the working group for ballot resolution and bringing our Issues Log current.

During the week, the group completed processing the comments received from our Mock ballot. We began to draft written responses which will be sent to all who took time to review the draft and provide us with comments and/or objections. Several of the comments/objections resulted in improvements to the specification and will be incorporated into the next draft (3.0). This will be the draft that goes directly to IEEE ballot on April 7th.

The Technical Editor completed the Language Independent Specification (LIS) and a first cut at test assertions as well. (X/Open followed through with their promise to fund our technical editor to write the assertions for POSIX.17. This made sense in that X/Open needs to have assertions for XDS.) The test assertions were reviewed by a consultant from POSIX.3 who had some problems with the way things were done. A lively debate ensued, but in the end, we caved in, and will incorporate the "suggestions." It is estimated that 90% of our assertions will require change. Hopefully, this can be automated.

Once again, we met with POSIX.12 (Protocol Independent Interfaces) in joint session and discussed their requirements on directory services. The POSIX.12 group wants a simplified interface to directory services for the users of their Detailed Network Interface (read sockets/XTI). We also discussed what objects POSIX.12 will need to be stored by the directory and how those objects will get documented. Given our need to freeze our draft for ballot and the lack of definition for both new objects and interface functions, we explored possible avenues for proceeding with the work.

We met in small group to continue the discussion. POSIX.17 participants left the meeting with a greater understanding of the issues, but no closer to a solution. We had a debriefing session afterwards and decided to produce a white paper documenting agreements, assumptions, issues, options, and proposed actions. This will be used to focus discussion at the next small groups meeting in April.

POSIX.17 and P1224 met again in joint session to review/revise test assertions for P1224. Draft 4 of P1224 has already entered ballot and we agreed to assist them in ballot resolution as time permits. Test assertions will be balloted in a recirculation. Since P1224 is a normative reference for POSIX.17, a stable version is essential for our ballot.

We sent a representative to POSIX.0's Architecture Framework BOF where the the results of their recent Mock Ballot were discussed. POSIX.17 had submitted comments/objections to the POSIX.0 Mock Ballot (Draft 14), focusing on the "networking" section. We were told that all our comments and objections were accepted and will be included in the next draft. The POSIX.0 model defined in the Mock Ballot draft seemed to recognize the need for APIs aimed at systems integrators as well as end users.

POSIX.17 shares a problem with P1224 and P1224.1. It seems that the objects defined in the base documents (XDS, XOM, X.400 API) reserved object ids (OIDs) in a vendor's (DEC) registered ISO name space. This might be ok for vendor consortia, but it won't cut it for a de jure standard. Because this issue touches more than one group, the DSSC discussed it and agreed to produce a recommendation on how to proceed by next meeting.

In Closing

Again, there are quite a few homework assignments between meetings. (I think there's a trend here.) Given this is our last quarter before ballot, we need to complete formation of the ballot group, fix the test assertions, finalize Draft 3, and respond formally to our Mock Ballot reviewers. We've also been asked by the DSSC and PMC to split our current Project Authorization Request (PAR) into two new PARs, one which addresses only the API to directory services and the other which addresses the POSIX name space issue.

The group will reconvene April 6-10, 1992 at the IEEE POSIX meeting in Dallas.

Report on the POSIX Study Group on Distributed Security

Laura Micks <uunet!aixsm!micks> reports on the January 15, 1992 meeting in Irvine, CA:

A study group has formed to investigate the feasibility of a project request (PAR) for Distributed Security.

One of the major topics raised at the Distributed Services Steering Committee (DSSC) was the problem of Security in a Distributed environment. This issue is not addressed by the Security working group (POSIX.6), nor any of the working groups under the DSSC.

A meeting was scheduled for all interested parties to discuss future directions in this area. Approximately 20 people attended and the application was made to be approved as a Study Group. If approved, a Study Group can be funded (from a logistics point of view) to meet for several meetings without an official PAR in place. The group plans to meet for an entire week next meeting cycle.

Most of the attendees were from the Security and Systems Management groups. Several people attended for general interest. It took the group quite some time to get rolling. There seemed to be two camps: one that wanted to define a conceptual model, identify services required, etc., and the other that wanted to pin down the existing implementations, choose one, and tweak it where necessary.

A PAR was actually drafted in October 1991 by Data Logic on behalf of Petr Janecek of X/Open. The PAR was not officially submitted to the POSIX Sponsor Executive Committee, probably due to potential lack of support and sponsorship within the POSIX community. The draft of this PAR was copied and distributed to the study group.

Known existing projects and organizations working on similar efforts were identified. The known models identified were as follows:

Open Software Foundation's Distributed Computing Environment (DCE)

NIS (Sun)

ECMA TC46 Technical Committee on Security Framework

ISO 7498-2 Security Addendum covering Architectural Framework/Security Svcs

The Andrew File System (AFS)

Project Athena

GSSAPI - A generic security API from DEC

Project MAXSIX

DNSIX - (Mitre)

Netware

GASSP (Generally Accepted Security System Principles)

U.S. Government OSI Profile (GOSIP)

We decided to further the study by arranging as many presentations as feasible from the list above for the April meeting. The meeting agenda will be to hear the architectural presentations on security models, and to determine selection requirements for base documents. A thorough evaluation will be made at the July meeting.

It is premature to assess the viability of this study group becoming an actual POSIX committee. The initial meeting was somewhat disorganized but in all fairness, there was little or no advance notice of this group's meeting, hence the attendees were unprepared. Given the sensitivity of the subject and the obvious differences of opinions raised at the January meeting, I don't expect that the exercise of selecting a particular model to be used as a base document will be trivial.

Can UNIX Designers Learn Anything from PCs? †

Jeff Haemer
<jsh@canary.com>

Editor's Note: An Open Letter to Marc Rochkind

Dear Marc,

Seeing your Invited Talk abstract in the 1992 Winter USENIX brochure (Can UNIX designers learn anything from PCs?), makes me wonder if I should propose an analogous talk for Comdex: Can PC designers learn anything from UNIX? Here are the first ten reasons I came up with.

- UNIX does more than DOS.

  There's a lovely old Ken Olsen quote about howUNIX is simple but VMS is complete. Today, UNIX is also complete. Steve Zucker points out: "In 1980, a single person could read and understand the entire UNIX kernel. In 1990, a single person couldn't lift it." Remember, if simplicity worked, the world would be overrun with insects.

- UNIX makes writing large programs convenient.

  Contrast the size of LOTUS 1-2-3 with the size of xclock. Think of how much more Shakespearian literature we'd have if he hadn't been hobbled by the sonnet form and the attention span of the audience at the Globe.

- UNIX permits complex system administration.

  A properly written labyrinth of */etc/rc[0-9].d* directories eliminates most by-hand operations during reboots, so that you can go make lunch while your machine is powering back up and putting all your files in lost+found. Appliances belong in the kitchen, and sometimes so do you.

- UNIX provides good programmer support.

  200 PCs only create a job for a single system administrator. 200 Suns create jobs for at least eleven: ten administrators *and* a system administration manager.

  Contrast the average salary of device-driver writers in the UNIX and DOS worlds, and you'll see why UNIX vendors brag about the dozens of devices and displays they support, while your latest DAK catalogue begs you to take their surplus: "$1,548.95 worth of CD-ROM software for just $149." Sure, you have to buy a $200 CD ROM drive, too, but you can bet it comes up as soon as you plug it in.

- UNIX encourages portability across platforms.

  DOS programmers were stuck with function prototypes faster than X3J11 could decide what it thought it might maybe think about doing. Because of the close relationship between C and UNIX, UNIX takes a more conservative approach.

† Re-printed from login: Volume 17, Number 3

Converting programs to ANSI is error-prone and you often have to debug the result, which means either putting in lots of printf statements or getting out the sdb manual. My vendor-supplied UNIX "ANSI C" compiler still generates big, slow code, and sets *argc* to 0, however many arguments you pass it. Things like this encourage use of the standard, UNIX "Portable C Compiler," */bin/cc*, which allows you to continue to compile your code on PDP-11/70s.

As a result, UNIX provides pretty good source-code portability across such different INTEL UNIX binary formats as a.out, x.out, coff, gpoff, elf, whatever Sun did for 386s, BCS, BCS-2, and the new System-V-consider-them-a-standard-dot-4 ABI. Indeed, source code portability often transcends UNIX. X Windows and OSF/Motif, for example, will soon run on everything from VMS to the Macintosh.

• UNIX is a potentially more lucrative software market than DOS.

Prices of Microsoft Word, Lotus 1-2-3, and the Norton Utilities for UNIX, show that all can have much higher profit margins on UNIX. (Admittedly, some of that difference is inflation, since the proper price comparison is with older DOS versions of comparable functionality.)

• UNIX provides device-independent I/O.

This lets state-of-the-art UNIX word processors, like ex and nroff, run on a wide range of devices, from teletypes, to terminals connected to 300-baud dial-up lines, to */dev/null*. In contrast, many standard Mac applications, like HyperCard, require mice, and don't work at all on ANSI-standard terminals or VT100s.

The optimizing, screen drawing package, curses, lets screen-oriented, state-of-the-art UNIX word processors, like *vi*, run quickly on the same range of devices.

• UNIX comes bundled with a lot of useful applications software.

At my local Office Club store, Reader Rabbit for DOS costs $20, but the far-more-flexible quiz, from AT&T, is a standard part of many UNIX distributions. Moreover, quiz doesn't restrict me to machines with speakers for audio output.

• UNIX is serious about formal standards.

PCs tend to be drawn into immediate practical solutions and d class de-facto standards. By the time formal ANSI standards are finally established, PC users often have a huge installed base of software that has to be upgraded if it wants to follow the rules. This is why the U.S. Government likes UNIX.

UNIX is also serious about international standards, and standardizing internationalization, and funny character sets, which, as John Quarterman points out, will allow the ls command to have a lot more options. This makes UNIX popular with foreign governments, like Belgium, Japan, X/Open, and what's left of Croatia.

• UNIX is a trademark.

Folks are always using "DOS" and "PC" as nouns, threatening the trademark-protected revenue streams of Microsoft and IBM, and discouraging further investment in software development. In contrast, AT&T has lawyers to help us remember not to omit the "System" from "UNIX System to UNIX System CoPy."

This week, on L.A. Law:

See, Benny, this entire letter uses UNIX as a noun instead of an adjective, so it's ungrammatical and, therefore, illegal.

Gee Arnie, maybe that's why my mom always made me use a syntax-directed editor.

Regards,

Jeff Haemer

USLE Column †

Julian Lomberg
Principal Consultant
USLE
London
United Kingdom

Julian Lomberg is Principal Consultant at UNIX System Laboratories Europe. He has extensive UNIX experience, and was a key developer on the UNIX System V Release 4 SPARC port.

UNIX System Laboratories announced the standard multiprocessing release of UNIX System V Release 4 - SVR4 MP - in October 1991. Julian Lomberg joins us in this edition to talk about the product.

For further information on this column, please contact Gill Smith on gill@uel.co.uk Gill is Marketing Manager at USLE.

UNIX® System V Release 4 Multi-Processor

Introduction

One characteristic of the computer market in the 1990s is the increasing use of multi-processor technology to boost performance in computer systems. Multi-processing is not new, it has been used in supercomputers since the 1960s. The trend emerging in the 1990s has been for this technology to be used in smaller systems, indeed now in 1992 a number of personal computer manufacturers whose desktop machines sell for only a few thousand dollars have now added multi-processor systems to their range. Today, if you have about $10,000 to spend, you can buy a multi-processor system.

The more widespread use of multi-processor technology in an increasingly competitive and cost-conscious computer market gives rise to some issues that did not exist ten or twenty years ago. Now for products to be accepted into the mass computer market they must adhere to standards, and this is where the multi-processing release of UNIX System V Release 4 (SVR4 MP) comes in. SVR4 MP is designed to enable users of the new generation of symmetric multi-processor systems, a term that I shall explain shortly, to benefit from the increases in throughput available from such systems without loosing compatibility with their existing applications.

The Target Architecture

There are many different types of multi-processor architectures and the one currently being exploited in low and medium cost systems is symmetric multi-processing. In this architecture, as shown in Figure 1, each processor shares the same system memory. Each processor has complete access to the whole of system memory without regard for other processors running in the system. If two processors attempt to access the same memory location simultaneously, the hardware will delay one of the processor's memory access until the other has completed it's access. This phenomenon is known as bus contention and is the reason why most symmetric multi-processor systems provide some cache memory local to each processor. Local cache memory reduces bus contention because data being used frequently by a particular processor will remain in that processor's cache, thus reducing the number of accesses that are needed to the shared memory.

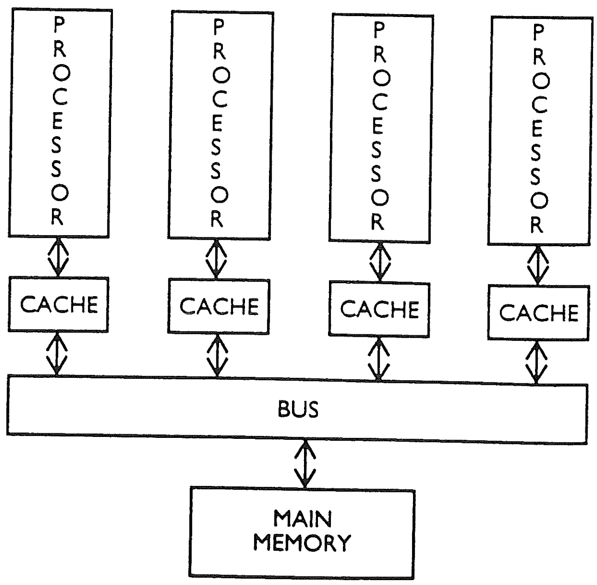


Figure 1: Symmetric Multi-processor Architecture

† Re-printed from EurOpen: Volume 12, Number 2

Local processor caches give rise to a problem referred to as cache consistency. If multiple processors are operating on the same piece of data from main memory they will each have a copy of that data in their local cache. If one processor modifies the data in its cache, then the corresponding copies of the data in the other processor's caches become invalid unless they are all updated to reflect the modification. As with bus contention, cache consistency is typically handled by the hardware; the SVR4 MP implementation assumes this to be the case.

Although we have discussed the symmetry of the target architecture with respect to memory access, the symmetry of I/O and interrupt delivery must also be considered. If all I/O devices are accessible from all processors then the architecture is said to have symmetric I/O. The alternative is asymmetric I/O where certain devices are only accessible from certain processors. An example would be a system where the main memory was accessed by the processor via a dedicated special purpose high-speed bus, but all the I/O devices are attached to an EISA bus that is only accessible from one of the processors. SVR4 MP can support either symmetric or asymmetric I/O. SVR4 MP can also support symmetric or asymmetric device-interrupt distribution. That is, interrupts can always go to one processor, different interrupts can go to different processors, or any interrupt can go to any processor.

Finally, the hardware must support a mechanism for one processor to raise an interrupt on another processor, and the processors used must support an atomic test-and-set operation for implementation of data structure locking (see later).

Design Goals

Before looking at some of the changes that were made to SVR4 in adapting it to a symmetric multi-processor architecture it is important to look at the design goals set for the system as these have influenced the design. Three major design goals have shaped the implementation of SVR4 MP:

Complete compatibility with uniprocessor SVR4. Not only does this mean that SVR4 MP complies to the same standards as SVR4, such as IEEE Posix, X/Open Portability Guide and USL's System V Interface Definition, but also that the SVR4 Application Binary Interface, and in the most recent SVR4 MP version for the 386 the Intel Binary Compatibility Specification extension 2 (iBCS2), are supported. This means that the user doesn't have to throw away his or her SVR4 applications when moving from a uniprocessor to a multi-processor implementation.

Minimal changes to uniprocessor SVR4 source code. This makes it easier to ensure the compatibility goals above are met, it also provides an easier implementation path for computer manufacturers migrating their operating system from a uniprocessor to a multi-processor base, and eases the movement of product enhancements between uniprocessor and multi-processor bases.

Ease of hardware portability. The code specific to a particular processor architecture is isolated, as is the code that is specific to a particular implementation of symmetric multi-processing such as starting and stopping individual processors and sending interrupts between them.

In addition to the above design goals, some performance criteria were also defined:

When running on a single processor, no more than 5% performance degradation compared to uniprocessor SVR4. There will always be some overhead introduced by the locking mechanism, explained later in this article, that need to be introduced for multi-processor systems. To run SVR4 MP on a uniprocessor system would not be sensible, given that the uniprocessor SVR4 implementation would be suitable, but it does provide a good way to measure the overhead introduced in making the operating system multi-processor capable.

An increase of 85% of available processor power for every processor added to the system. The ideal situation would be to achieve an increase of 100% of available processor power for every processor added, issues such as bus contention and locking overhead make this impossible to achieve in practice.

The Technology

This section examines some of the key changes that were made to the SVR4 kernel in order to adapt it for a multi-processor system. Most of these changes are completely transparent to the user and the application developer. The only changes that might effect these people are described in the final subsection, *New and Old Code*.

Scheduling

The UNIX System has always been a multi-processing (not to be confused with multi-processor) system. Multi-processing means that the system can have multiple processes active at any one time, for instance there may be two users on the system, one running an editor process and the other running a compiler. On a uniprocessor system the kernel uses a scheduling algorithm to share the one available processor between the multiple processes that are waiting to run. An available process is picked from the queue of available processes and allowed to run until the kernel decides to schedule another process from the queue. This scheduling algorithm is easily extended to a multi-processor environment. Instead of only having one processor available on which to run processes, there are now multiple processors. Each processor schedules its time independently, however they all select the processes that they are going to run next from the same queue. The scheduling algorithm on each processor will always pick the highest priority process that is on the queue, and since each processor is picking processes off the same queue the load will be evenly spread across the processors.

Data Locking

At any given time all processors will be running simultaneously. Each processor will have selected a process to run, either a user process, a system process, or an idle task. Each processor could either be executing the process itself, or else executing within the kernel. It is the possibility of multiple processors simultaneously executing the kernel code that causes problems. Changes must be made to the kernel that allow it to continue to function reliably when it is being executed simultaneously by multiple processors. The resulting kernel is said to be *multi-threaded.*

Imagine two processors attempting to increment the same variable in the kernel. They could both read the current value *n* simultaneously, add 1 to it, and then write the resulting value *n+1* back when the real value should have been *n*+2. The solution to this problem is to implement a locking scheme to prevent concurrent access to data structures where such concurrent access could cause corruption of kernel data structures. Before accessing a data structure a process must first acquire the lock associated with it, and, once it has finished its access, it releases the lock. Another process will be prevented from acquiring the lock until the first processes has released it, and hence will not be permitted concurrent access. These locks are called *mutex* (mutual exclusion) locks. The major task in multi-threading the SVR4 kernel was to protect kernel data structures using mutex locks.

Adding locking to a kernel must be done with care if optimal performance is to be achieved. A balance must be struck between coarse-grained and fine-grained locks. Coarse grain locks cover a large number of data structures. Creating locks that are too coarsely grained results in excessive contention on the locks. Processes may have to wait for a lock in order to access data that could be totally unrelated to the data currently being operated on by the process holding the lock. Setting the locks at too fine a grain, for instance on each individual variable, can mean that the kernel spends more time acquiring and releasing the locks than it does operating on the data. The granularity of the SVR4 MP locks were determined using debug and analysis tools that provide data on lock contention and other issues related to performance and possible deadlock conditions.

The mutex locks in SVR4 MP are of two types, spin locks and sleep locks. Spin locks are used for data items that are only locked briefly before being released. When a process tries to acquire a spin lock and fails it continually retries until it succeeds. Sleeps locks are used for data items that are going to remain locked for a longer period of time. When a process fails to acquire a sleep lock it is put to sleep and woken when the lock is released by the current holder. The behaviour of each lock in the system is set up at initialisation time, this means that a given lock can be switched between spin and sleep by changing the one place in the code where it is initialised.

Two other unusual features of SVR4 MP locks are also worth mentioning, both are motivated by the design goal of minimising the number of changes to the uniprocessor SVR4 source code.

It is possible, due to the recursive nature of the SVR4 kernel, for a process to try to acquire a lock that it already holds. In many multi-processor systems this would cause a deadlock, however this is permitted in SVR4 MP. The system keeps a count of how many times a process acquires the same lock so that when it is subsequently released the correct number of releases can be performed.

The second unusual aspect of SVR4 MP locks is their behaviour when the holding process goes to sleep. A process that goes to sleep whilst still holding a lock could have a disastrous effect on system performance. Many multi-processor systems solve this problem by ensuring that processes do not go to sleep whilst holding locks that other processes are likely to need. To implement this solution in SVR4 would require many changes to the code and hence an alternative solution was required. In SVR4 MP all locks held by a process are automatically released when that process is put to sleep and then automatically re-acquired when the process is woken up. The reason that this solution works in SVR4 is because, even for the uniprocessor SVR4 implementation, sleeps are significant events because they will cause a context switch and hence another process to take control of the processor. This means that the uniprocessor SVR4 code already ensures that kernel data structures are in a consistent state before a process is put to sleep.

Processor Private Data

There are certain variables in SVR4 that are processor specific such as the interrupt mask and the details of the process currently running on that processor. In SVR4 MP these variables are grouped into processor-private regions, one per processor in the system. Each processor's virtual memory map is set up so that it's processor-private area, at the same virtual address on each processor, is mapped to a different physical address where the processor variables for that processor are actually stored. The numerous references to processor specific variables scattered throughout the SVR4 code did therefore not have to be changed.

New and Old Code

Locking interfaces have been added to the new DDI/DKI specification to enable multi-threaded device drivers and streams modules to be written. Device drivers and streams modules are linked into the kernel address space and may be running simultaneously on more than one processor. This means that new code should use the new DDI/DKI locking interfaces to protect critical data structures.

It might not be possible to modify all device drivers to be multi-threaded. Perhaps you have an SVR4 driver for a peripheral that was only supplied in binary form and hence cannot be modified to include the necessary lock protection. SVR4 MP supplies support for single-threaded device drivers that conform to the original DDI/DKI specification. It is possible to bind a device driver to a single processor. This ensures that the specified device driver will only ever be allowed to run on the single specified processor. This means that the device driver can run secure in the knowledge that no other processor will be executing the same code and hence there will be no concurrent access to the data structures it is using.

Summary

SVR4 MP provides a standard operating system environment for multi-processor systems. Complete compatibility with uniprocessor SVR4 makes the move to a multi-processor environment transparent for the users and preserves their investment in application software. Due to some innovative new locking mechanisms, the internal changes to the SVR4 kernel have also been minimised allowing OEMs to adopt this new technology with a minimum of effort.

AUUG
MANAGEMENT COMMITTEE
MINUTES OF MEETING 6 APRIL 1992 †

Held at ACMS, Paddington

Present: Frank Crawford, Pat Duffy (from 1:30 - 2:00), Glenn Huxtable, Rolf Jester, Peter Karr (from 11:30), Chris Maltby, Scott Merrilees, Michael Tuke, and Liz Fraumann

Meeting commenced at 10:27am

Ross Hand and John Barlow of Canberra gave a presentation to the committee regarding the development of local chapters and their experiences with their summer conference. Ellen Gubbins of Symmetry and Wael Foda, AUUG Secretariat (ACMS) gave presentations to the committee and were present during relevant portions of the meeting.

1. APOLOGIES
   Apologies were received from Pat Duffy and Jagoda Crawford.

2. MINUTES OF LAST MEETING (20 JANUARY 1992)
   Moved (FC/SM) that the minutes were accepted. Carried.

   Items not specifically cited here are incorporate in the business of this meeting. The following items were carried over from the previous meeting:

2.1 AARNET Subscriptions ($250.) and connections for WF & PK
ACTION : CM

2.2 Tax exempt status
ACTION: MT

2.3 Registration of Assets
ACTION: WF

2.4 Feedback forms from summer conference
ACTION: GH

† Note that the minutes for the May meeting were published in Volume 13 Number 3 of AUUGN

3. PRESIDENT'S REPORT

Due to Pat's time constraints, the president's report was not given.

4. SECRETARY'S REPORT

Rolf Jester reported:

4.1 The organisation has 65 unfinancial members. A letter will be distributed to these individuals soliciting their membership one last time, otherwise they will be taken off of the membership roster.

ACTION: RJ/WF

4.2 A Business reply card will be developed to make it easy to respond to membership renewal.

4.3 Those not renewing will be surveyed to ascertain the reason(s) they have chosen not to renew.

4.4 Current membership statistics are attached.

4.5 It was moved that Liz should get information on "free post/bus. reply cards (PK/GH)

ACTION: LF

5. TREASURER'S REPORT

Frank Crawford reported:

5.1 Current bank account is at $27k. $15k was made over the past 2 months from membership and conferences.

5.2 We have spent $9k on conferences.

5.3 A budget was circulated and is attached, it was agreed by all that we must stay within this budget for this year conference and exhibition.

5.4 We can expect additional funds coming from AARNET connections.

5.5 Bottom line perspective: there was a small profit from AUUG '91 ($4-5K) but expenses have negated the profit.

5.6 Much discussion ensued regarding solutions to the flat budget for this year. Peter Karr suggested to review the format of AUUG '92 and subsequent conferences. Moved to review by RJ/SM. All agreed.

6. EDITOR'S REPORT

Frank Crawford reported:

6.1 Volume 13 #2 deadline is 17 April '92. We are awaiting the President's Report and it will be issued by the end of April.

6.2 Exec. Committee nominations are due 14 April.

6.3 It was suggested for future issues, that the January issue should contain the Call for Nominations for Executive Committee.

6.4 It was moved and so passed that due to time constraints the nomination forms would be mailed out in a separate mailing this year. RJ/PK

ACTION: RJ - items to WF
ACTION: WF - mailing

6.5 AUUGN now has 2 covers, the "old" and a new design done by Symmetry. A decision was reached that the "old" surplus stock would be used up prior to printing a new cover. The general consensus of the group was they would like to see a different design than submitted by Symmetry. Liz Fraumann suggested a contest by the general membership to submit designs. The selected design would award its creator with a free year's membership! Institutional members would be excluded. The idea was so moved and passed. RJ/FC. The contest rules will be announced in an upcoming issue of AUUGN.

6.6 Open Forum - is pending mailing to the membership. The Canberra and WA regions will receive ~200 copies for their distribution. ACMS will maintain 150 - 200 copies for distribution on inquiries and Peter Karr offered the OSR mailing list to distribute ~700 copies.

ACTION: PK - list to LF by Fri

6.7 Back issues of AUUGN are not being serviced. It was suggested that Liz Fraumann go to Softway, where they are currently being stored and handle the fulfillment. For future, a stock will be maintained at ACMS and handled by Wael.

ACTION: LF

6.8 Peter Karr informed the group of the "printed post" coming from the government in the next year or so. This will significantly increase the distribution cost of AUUGN. This was yet another motivation for the development and implementation of local chapters... To handle distribution.

6.9 Liz Fraumann suggested, in a previous suggestion to Jagoda, that "theme" ideas are built for the issues of AUUGN, making it easier for authors to submit papers, and easier for solicitation in advance of articles. This will be tried for the December and February issues of AUUGN and publicized in an upcoming issue.

ACTION: JC

6.10 It was moved and so passed that mail alises should be updated printed in AUUGN. RJ/GH

ACTION: SM

7. SUMMER CONFERENCES

Glenn Huxtable reported:

7.1 All accounts were the summer conferences were very successful. Perth, Adelaide, and Sydney, were reminders of "the old days." Format was a one day conference with lunch and am/pm teas. 3/4 Concurrent sessions were held. A final report will be submitted.

7.2 Perth - costs were ~ 1K and had over 70 attendees. $60 for members and $80 for non-members.

7.3 Adelaide - costs were ~ 500-600.

7.4 Canberra - sponsored, costs were -0-. Over 110 attendees and was held over 2 days. In addition, they held a security workshop. A full report/summary has been submitted with these minutes.

7.5 Sydney - costs were ~ 1k. and over 50 attendees.

7.6 Darwin - at the time of this meeting had not submitted information yet.

7.7 Papers presented at the summer conferences will be submitted to *AUUGN* editor for inclusion in the publication. It was also suggested the summer conferences work closely together and interconnect with their speakers.

7.8 It was suggested to hold conferences/work shops as an ongoing activity and have a "roadshow". Peter Karr offered use of the OSR mailing list which could be targeted to specific attendees.

ACTION: GH will produce full report on summer conf.

ACTION: LF will produce a conf. kit for organisers on "How to"...

7.9 It was moved and seconded that the formalisation of local chapters would commence. PK/RJ All agreed.

8. CHAPTERS

Ross Hand and John Barlow from Canberra proposed:

8.1 Local Chapters would offer better value to members with local meetings and conferences. Part of the membership fees would be directed back to each local chapter. A local bulletin board would be established and potentially reduce cost of administration. Costs for the bulletin board would be a 4.1k running cost.

8.2 Discussion ensued regarding the proposal. Options were discussed as to the handling of the dollar flow to the chapter. Two options were cited:

A. Guaranteed percentage of the overall membership gathered from the region.

B. Handled on a case by case bases. Necessitating a chapter request for funds.

8.3 It was suggested that Institutional Members could specify the desire for participating in multiple regions for an additional cost. However their primary place of business would dictate where the 2 representatives would participate for no additional cost.

8.4 It was moved and seconded that a sub-committee should be formed and draft a "Chapter" model to be submitted at the next Management Meeting. RJ/FC. This model will include, Funding, Structure (relationship to AUUG,Inc., and Mission of the local chapters. All agreed. The sub-committee is comprised of: GH, SM, RH, JB.

9. PUBLIC RELATIONS

Ellen Gubbins reported:

9.1 Article submissions have been focused on the posting of the Business Manager position and the Call for Papers. Coverage has been good and well received. RJ owes an article and Frank will submit an article on SVR4 use and administration this week to be used over the next month.

9.2 The media schedule for AUUG '92 was reviewed. It was felt by several committee members that the commencement date was too early, perhaps the targeted publications inappropriate, and advertising fees too high. The program committee will review these issues and amend appropriately.

9.3 It was restated that the projected media budget is a firm budget and can not be exceeded.

10. CONFERENCE

Liz Fraumann reported:

10.1 Currently tasks are proceeding on schedule. Review of the design, and deliverables took place. The design was suggested to be modified and the weight of paper stock selected to be changed to assist in bringing cost per item closer in line with the projected budget.

ACTION: LF will fax modified design to committee

10.2 Peter Karr proposed the following conference attendee price modifications: Members - $395 (SAME) Non-Members - 595. PK/SM Passed. Further discussion included the option for lunch w/$50 discount for those not partaking.

10.3 Discussion, at length, ensued chartering the program committee to commit to a high quality technical conference. The object of the conference is to educate not market.

10.4 Liz passed along Andrew McRae's suggestion for a terminal room and the potential of distribution of public domain software as a direct benefit of membership.

ACTION: LF call MHS to do mail service

10.5 Wael was asked about charges for last years' conferences. He explained the charges were for the use of the credit card acceptance for registration. It was suggested that we discourage their use in '92.

10.6 It was suggested that perhaps a PCP certificate may be available for participation in portions of the conference, and potentially free advertising.

11 BUDGET

11.1 It was suggested that the budget reflect more closely the pending amendments with the summer conferences and chapters.

11.2 It was proposed to consider a new class of membership, a Corporate Membership. A proper amendment will be drafted.

ACTION: FC/GH

11.3 Peter Karr offered a free ad in OSR for membership

11.4 Frank suggested there will be up to a 10K loss this year due to flat conference registrations, losses, and lean times.

12. OTHER ISSUES

12.1 AARNET article. CM promised to complete the article for publication OSR.

ACTION: CM

12.2 AUUG received a letter of congratulations from UniForum on the amount of members and their participation. There is a financial payment from UniForum to AUUG based on these numbers. How ever it could not be determined from the correspondence how these numbers were ascertained.

ACTION: LF call UniForum

12.3 It was reviewed that AUUG agreed to underwrite up to 5K per site for each summer conference.

12.4 Costs related to Symmetry were discussed at length. It was agreed a sub-committee would be formed to review/reestablish goals/objectives for Symmetry. Sub-committee consists of PD/RJ/CM/LF.

ACTION: PD/RJ/CM/LF

12.5 It was suggested to review statistics on the number of AUUG members reading news, OSR, etc. It was suggested that aus.auug and OSR run the same articles.

12.6 Pat reviewed a proposal from DataPro for the membership to participate in a "Study of UNIX Workstation Market." Respondents would receive executive summary. It was agreed a mailing would be done.

ACTION: LF/WF

12.7 It was also suggested that a place for those wishing to abstain in voting be placed on this years ballot.

12.8 An overtime procedure was established for LF's time.

12.9 The Technical Library proposal was reviewed. It was suggested that this is bettered handled with the event of local chapters and tabled.

12.10 It was suggested that a summary of auug.oz be placed in *AUUGN*. RJ suggested PD write the summary.

12.11 It was suggested to do a survey of the membership and benefits. Liz and Frank will draft a survey and fax to the committee.

ACTION: LF

12.12 Rolf will placed a call for agenda items for the next AUUG general meeting.

ACTION: RJ

12.13 It was suggested to review the current share of the conference and exhibition with ACMS.

ACTION: PD/CM

13. NEXT MEETING

Monday 18 May 1992 - 10:00am @ ACMS

Meeting adjourned at 5:40pm

Respectfully Submitted,

Elizabeth A. Fraumann

MHS RELEASE: AUUG MEMBERS NETWORK ACCESS

Under a special arrangement with Message Handling Systems - operators of TMX (The Message eXchange) - AUUG is providing members with a packaged offering for electronic mail and news access to the major networks.

The Message eXchange is a well established service that has been operating for over twelve months with a growing user community that includes individuals, large computer vendors and small software organisations.

AUUG members can dial-in to the TMX facilities to exchange messages with:

- other AUUG members
- the main Australian and overseas networks
- all or selected Australian and Usenet news groups
- the Clarinet professional news and information service.

The special price for registered AUUG members is a one-time $395 (the normal TMX fee is $495) which is inclusive of an initial 10 free connect hours and a special version of the MHSnet software. After the first ten hours, connection rates are $8.00 per hour. (For volume users there is an optional prepayment of $1200 for 300 hours at $4.00 per connect hour). There is a minimum fee of one hour per month.

Connection to TMX may be via V22, V22bis, V32 and PEP modems. MS.DOS and Apple MacIntosh connections are also available.

The Clarinet service, for which fees are applicable, includes UPI wireservice, computer industry news, financial information and up-to-date major world news stories.

The TMX service is currently available in Sydney and the Melbourne service is expected to be operational by July. While other centres would need to dial-in to either of these systems, it is intended to expand the service to other states based on traffic demand.

AUUG is pleased to be able to introduce this service to its members as a means to increase communication effectiveness. For more information please contact Elaine Pensabene at Message Handling Systems Pty Ltd.

Phone: (02) 550 4448
Fax: (02) 519 2551
Email: elaine@mhs.oz.au

Omegatrend Pty Ltd
A.C.N. 009 403 981
P.O. Box 475, Claremont, Western Australia 6010 Telephone (09) 347 6200 Facsimile (09) 347 6299

RECEIVED
-4 AUG 1992

27th July 1992

The Editor,
AUUGN,
P.O. Box 366,
KENSINGTON NSW 2033

Dear Jagoda,

I much appreciated Ross Parish's article in the June issue of AUUGN. I thought it was a well written overview of UNIX/PC networking.

I felt that I should however make a few comments about LanManager for UNIX. We have an NCR System 3000 running our main accounting/inventory/sales software on which is also installed AT&T/NCR Stargroup LanManager. We have a 386 PC clone also running SVR4 and Stargroup LanManager and some 50 PC's connected to this network. We are running entirely under OSI and have never had TCP/IP running.

The 386 clone runs as the primary server in our domain and the NCR as the backup server. The PC's use NetBIOS VT sessions to run our main application on the NCR and most of the network file space exists on the 386 PC.

Contrary to Ross's quote of various commentators re OSI, running DOS 5.0 and Windows 3.X we have no problems with RAM cram, particularly as all our machines on the network are 386's (SX or DX) and 99% of our applications are Windows applications.

I though this information may be of interest to other readers when considering the possibilities of UNIX networking.

Thanks for your tireless efforts in producing AUUGN.

Yours sincerely,

[illegible]

BOB SOUTHWELL - Executive Officer
Membership and Information Systems

AUUG Membership Categories

Once again a reminder for all ''members'' of AUUG to check that you are, in fact, a member, and that you still will be for the next two months.

There are 4 membership types, plus a newsletter subscription, any of which might be just right for you.

The membership categories are:

Institutional Member
Ordinary Member
Student Member
Honorary Life Member

Institutional memberships are primarily intended for university departments, companies, etc. This is a voting membership (one vote), which receives two copies of the newsletter. Institutional members can also delegate 2 representatives to attend AUUG meetings at members rates. AUUG is also keeping track of the licence status of institutional members. If, at some future date, we are able to offer a software tape distribution service, this would be available only to institutional members, whose relevant licences can be verified.

If your institution is not an institutional member, isn't it about time it became one?

Ordinary memberships are for individuals. This is also a voting membership (one vote), which receives a single copy of the newsletter. A primary difference from Institutional Membership is that the benefits of Ordinary Membership apply to the named member only. That is, only the member can obtain discounts an attendance at AUUG meetings, etc. Sending a representative isn't permitted.

Are **you** an AUUG member?

Student Memberships are for full time students at recognised academic institutions. This is a non voting membership which receives a single copy of the newsletter. Otherwise the benefits are as for Ordinary Members.

Honorary Life Membership is not a membership you can apply for, you must be elected to it. What's more, you must have been a member for at least 5 years before being elected.

It's also possible to subscribe to the newsletter without being an AUUG member. This saves you nothing financially, that is, the subscription price is greater than the membership dues. However, it might be appropriate for libraries, etc, which simply want copies of AUUGN to help fill their shelves, and have no actual interest in the contents, or the association.

Subscriptions are also available to members who have a need for more copies of AUUGN than their membership provides.

To find out if you are currently really an AUUG member, examine the mailing label of this AUUGN. In the lower right corner you will find information about your current membership status. The first letter is your membership type code, N for regular members, S for students, and I for institutions. Then follows your membership expiration date, in the format exp=MM/YY. The remaining information is for internal use.

Check that your membership isn't about to expire (or worse, hasn't expired already). Ask your colleagues if they received this issue of AUUGN, tell them that if not, it probably means that their membership has lapsed, or perhaps, they were never a member at all! Feel free to copy the membership forms, give one to everyone that you know.

If you want to join AUUG, or renew your membership, you will find forms in this issue of AUUGN. Send the appropriate form (with remittance) to the address indicated on it, and your membership will (re-)commence.

As a service to members, AUUG has arranged to accept payments via credit card. You can use your Bankcard (within Australia only), or your Visa or Mastercard by simply completing the authorisation on the application form.

AUUG Incorporated
Application for Institutional Membership
AUUG Inc.

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

AUUG Membership Secretary
P O Box 366
Kensington NSW 2033
Australia

• Foreign applicants please send a bank draft drawn on an Australian bank, or credit card authorisation, and remember to select either surface or air mail.

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
AUUG Inc.

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

AUUG Membership Secretary
P O Box 366
Kensington NSW 2033
Australia

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

This form is valid only until 31st May, 1992

I, ... do hereby apply for

☐ Renewal/New*	Membership of the AUUG	$78.00	
☐ Renewal/New*	Student Membership	$45.00	(note certification on other side)
☐ International Surface Mail		$20.00	
☐ International Air Mail		$60.00	(note local zone rate available)

Total remitted AUD$______
(cheque, money order, credit card)

* Delete one.

I agree that this membership will be subject to the rules and by-laws of the AUUG as in force from time to time, and that this membership will run for 12 consecutive months commencing on the first day of the month following that during which this application is processed.

Date: __/__/__ Signed: ____________________

☐ Tick this box if you wish your name & address withheld from mailing lists made available to vendors.

For our mailing database - please type or print clearly:

Name: .... Phone: .... (bh)

Address: .... (ah)

.... Net Address: ....

....

.... *Write "Unchanged" if details have not*

.... *altered and this is a renewal.*

Please charge $_____ to my ☐ Bankcard ☐ Visa ☐ Mastercard.

Account number: ______ ______ ______ ______. Expiry date: __/__.

Name on card: ______________________________ Signed: ____________________

Office use only:

Chq: bank ________ *bsb* ___-___ *a/c* ____________ # ____________

Date: __/__/__ *$* *CC type* ___ *V#* __________

Who: ____________ *Member#* ________

Student Member Certification *(to be completed by a member of the academic staff)*

I, ... certify that

.... *(name)*

is a full time student at .... *(institution)*

and is expected to graduate approximately ___/___/___.

Title: ______________________________ Signature: __________________

AUUG Incorporated
Application for Newsletter Subscription
AUUG Inc.

Non members who wish to apply for a subscription to the Australian UNIX systems User Group Newsletter, or members who desire additional subscriptions, should complete this form and return it to:

AUUG Membership Secretary
P O Box 366
Kensington NSW 2033
Australia

• Please don't send purchase orders — perhaps your purchasing department will consider this form to be an invoice.

• Foreign applicants please send a bank draft drawn on an Australian bank, or credit card authorisation, and remember to select either surface or air mail.

• Use multiple copies of this form if copies of AUUGN are to be dispatched to differing addresses.

This form is valid only until 31st May, 1992

Please *enter / renew* my subscription for the Australian UNIX systems User Group Newsletter, as follows:

Name: .... Phone: .... (bh)

Address: .... (ah)

....

.... Net Address: ....

.... *Write "Unchanged" if address has*

.... *not altered and this is a renewal.*

For each copy requested, I enclose:

☐ Subscription to AUUGN	$ 90.00
☐ International Surface Mail	$ 20.00
☐ International Air Mail	$ 60.00

Copies requested (to above address) ________

Total remitted AUD$______

(cheque, money order, credit card)

☐ Tick this box if you wish your name & address withheld from mailing lists made available to vendors.

Please charge $_____ to my ☐ Bankcard ☐ Visa ☐ Mastercard.

Account number: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _. Expiry date: __/__.

Name on card: ______________________ Signed: ______________

Office use only:

Chq: bank ________ *bsb* ___-___ *a/c* ____________ *#* ____________

Date: __/__/__ *$* *CC type* ___ *V#* __________

Who: ____________ *Subscr#* _________

AUUG

Notification of Change of Address

AUUG Inc.

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#* _________

	HCL-eXceed	HCL-eXceed Plus	HCL-eXceed HiRes	HCL-eXceed/W (MS-Windows)
TCP/IP transports supported:				
Beame & Whiteside Software BWNFS, rel. 2.15 or higher	yes	yes	yes	yes
FTP Software Inc. PC/TCP Network Software for DOS, rel. 2.02 or higher	yes	yes	yes	yes
Hewlett-Packard Company HP ARPA Services, rel. 2.0 or higher	yes	yes	yes	yes
IBM Corporation TCP/IP for DOS, rel. 2.0 or higher	yes	yes	yes	yes
Locus Computing Corporation TCP/IP for DOS, rel. 2.0.4 or higher	yes	yes	yes	yes
Microsoft Corporation Microsoft LAN Manager, rel. 2.0 or higher	yes	yes	yes	yes
NetManage, Inc. Chameleon TCP/IP for Windows 3.0, rel. 2.0 or higher	no	no	no	yes
Novell LAN Workplace TCP/IP, rel. 4.0 or higher	yes	yes	yes	yes
Novell/Excelan LAN Workplace TCP/IP, rel. 3.4 or higher	yes	yes	yes	no
Sun Microsystems, Inc. PC-NFS, releases 3.0.1 and 3.5	yes	yes	yes	yes
3COM Corporation 3+ Open TCP, rel. 1.1 or higher	yes	yes	yes	yes
Ungermann-Bass, Inc. Net/One TCP BNS/PC, rel. 16.2 or higher	yes	yes	yes	yes
Walker, Richer & Quinn, Inc. Reflection Network Series, rel. 1.1 or higher	yes	yes	yes	yes
Wollongong Group, Inc. Pathway Access for DOS, rel. 1.1	yes	yes	yes	yes
Wollongong Group, Inc. WIN/TCP for DOS, rel. 4.1 or higher	yes	yes	yes	yes
Digital Equipment Corporation	DECnet, rel. 4.0 or higher and DEC Pathworks TCP/IP, rel. V1.1A or higher			
X server highlights:				
X11 Release 5	yes	yes	yes	yes
Mode	real	protected	protected	stand.,386-enhanced
Maximum number of clients	16	32	32	32
3 button emulation on 2 button mouse	yes	yes	yes	yes
Menu driven configurator	yes	yes	yes	yes
Local fonts viewing in graphic rendition	yes	yes	yes	yes
Number of int'l keyboard mappings	16	16	16	16
Auto font builder	yes	yes	yes	yes
Stored image size	unlimited	unlimited	unlimited	unlimited
Motif, Open Look, uwm, twm compatible	yes	yes	yes	yes
Local Window Manager	no	yes, from Hummingbird (Motif look and feel)	yes, from Hummingbird (Motif look and feel)	yes, MS-Windows from Microsoft
Start-up methods:				
TELNET	yes	yes	yes	yes
PASSIVE	yes	yes	yes	yes
REXEC and RSH	yes	yes	yes	yes
X11 R.5 XDM Control Protocol				
indirect	no	yes	yes	yes
query	no	yes	yes	yes
broadcast	no	yes	yes	yes
Log file of host & server messages	yes	yes	yes	yes
Server extensions built-in	yes	yes	yes	yes
"Escape to DOS" & back to server	no	yes	yes	n.a.
Backing store & save unders	no	yes	yes	yes
Font size	64K	unlimited	unlimited	unlimited
Custom compose-key	no	yes	yes	yes
Non-Rectangular Extensions	no	yes	yes	yes
Xtrace (for application debugging)	no	yes	yes	yes
DOS - X concurrency	no	no	no	yes
X - DOS copy & paste	via use of HCL-eXtend (see below)			yes - text & image
UNIX-DOS file transfer Print UNIX/DOS files on local PC printer Access log file while server running	via use of HCL-eXtend (see below)			
HCL-eXtend	A suite of X Window clients which reside on a UNIX host. HCL-eXtend allows a user to access the DOS file system, printers and program execution facilities without leaving X.			

TURN PCs INTO X TERMINALS WITH

HCL - eXceed family of X servers for PCs

HCL-eXceed now supporting X11R

HCL - eXceed Plus	PC X server for DOS PC
HCL - eXceed/W	MS-Windows based PC X server
HCL - eXceed HiRes	High resolution (1280 x 1024 PC X server

INS
ACN: 003 338 476

Unit 3a, 3 Lanceley Place
Artarmon, NSW, 2064

INFORMATION NETWORK SOLUTIONS

CALL NOW FOR MORE INFORMATION ON:

PHONE: (02) 906 - 6335 FAX: (02) 906 - 6327