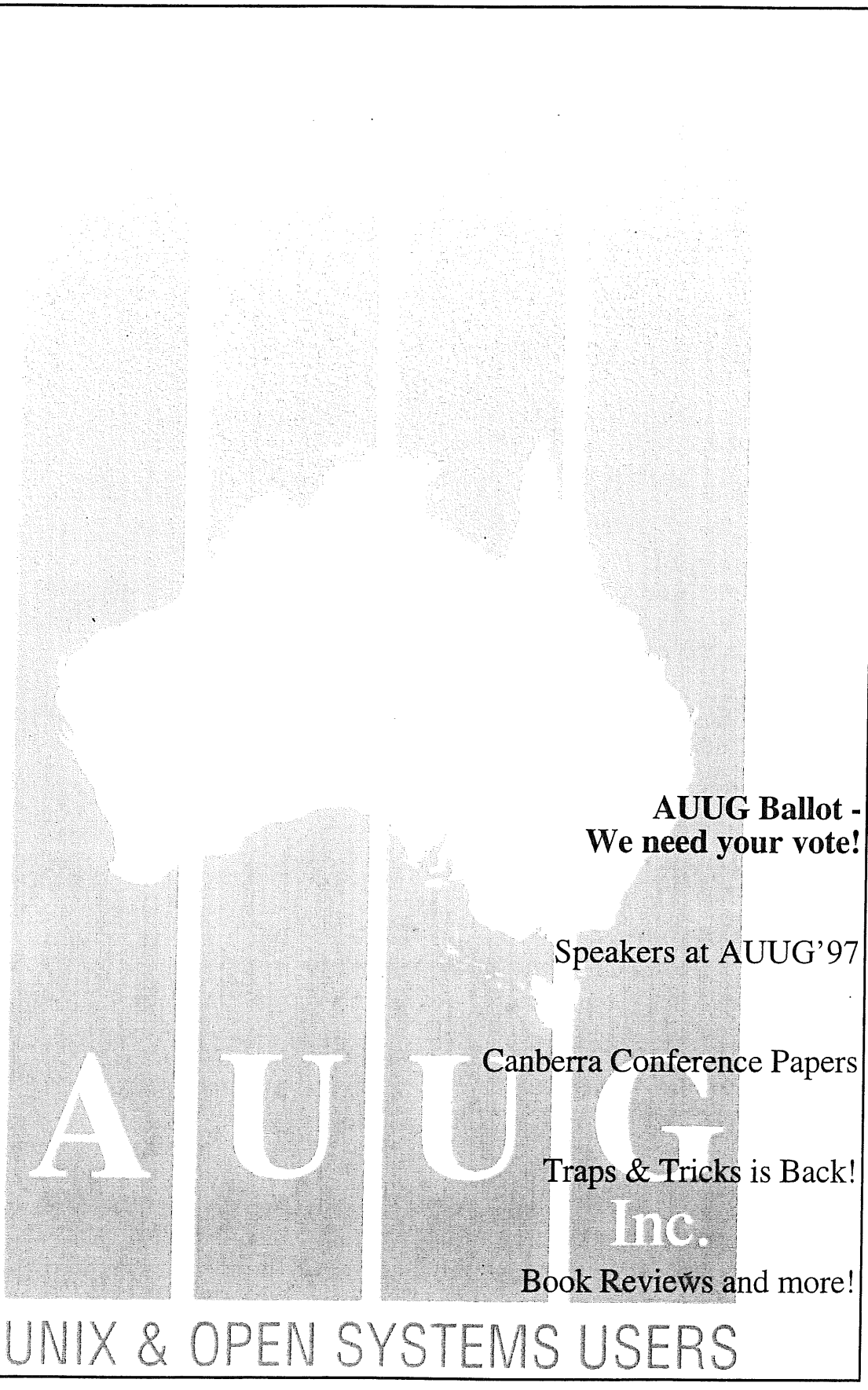
AUUGN

The Journal of AUUG Inc.
Volume 18 • Number 3
August 1997



AUUG Ballot - We need your vote!

Speakers at AUUG'97

Canberra Conference Papers

Traps & Tricks is Back!

Book Reviews and more!

ISSN 1035-7521

Print post approved by Australia Post - PP2391500002

AUUGN

The Journal of AUUG Inc.

Volume 18 • Number 3
August 1997

Table of Contents

AUUG Membership and General Correspondence

The AUUG Secretary
PO Box 366
Kensington NSW 2033

Tel: 02 9361 5994
Fax: 02 9332 4066
Toll Free: 1800 625 655
Internet: *auug@auug.org.au*

AUUGN Correspondence

Please send all correspondence regarding AUUGN to:

AUUGN Editor
PO Box 366
Kensington NSW 2033

Internet: *auugn@auug.org.au*

Submission Guidelines

Submission guidelines for AUUGN contributions are regularly posted on the aus.org.auug news group.

They are also available from the AUUG World Wide Web site at:

http://www.auug.org.au

Alternately, send email to the above correspondence address, requesting a copy.

AUUGN Back Issues

A variety of back issues of AUUGN are still available; for price and availability please contact the AUUG Secretariat, or write to:

AUUG Inc.
Back Issues Department
PO Box 366
Kensington NSW 2033
Australia

Conference Proceedings

A limited number of copies of the Conference Proceedings from previous AUUG Conferences are still available. Contact the AUUG Secretariat for details.

Mailing Lists

Enquiries regarding the purchase of the AUUGN mailing list should be directed to the AUUG Secretariat.

Tel: (02) 9361 5994
Fax: (02) 9332 4066

During normal business hours.

Disclaimer

Opinions expressed by the authors and reviewers are not necessarily those of AUUG Inc., its Journal, or its editorial committee.

Contribution Deadlines for AUUGN in 1997/98

Volume 18 • Number 4 • November 1997 : **October 7th 1997**

Volume 19 • Number 1 • February 1998 : **January 7th 1998**

Volume 19 • Number 2 • May 1998 : **April 7th 1998**

Volume 19 • Number 3 • August 1998 : **July 7th 1998**

Editorial

Günther Feuereisen <gunther@ibm.net>

Time flies .. doesn't it? It seems like it was only yesterday, everyone was planning for AUUG'97. Now, it's just under a month away .. where did all the time go?

Come September, the Brisbane Convention Centre won't know what hit it when the AUUG Circus comes rolling into town.

This year looks like being another great conference. If you haven't made plans to attend, it's not too late! Check out the web page <http://www.auug.org.au> for more details, or contact the AUUG Office.

On to the main event ..

Traps & Tricks is back because YOU, the reader, want us to be back. Solaris Musings has taken the issue off .. David has been snowed under with a ton of other work.

Once again we have some book reviews, as well as some articles from the Canberra Conference earlier in the year. We'll be presenting the remainder of these in upcoming issues.

We've been chatting with our friends at USENIX about getting selected issues from ;login: reprinted in AUUGN once more. That should be back by V18/#4.

Finally, a special note of thanks to all the AUUG Editorial team who keep chasing up content and articles from people. Remember people, we are here to share information with you. To share information, we need someone to provide it.

Time to get back to the juggling act. Enjoy AUUG'97 .. I'll see you all in November!

❖

President's Column

Michael Paddon <Michael.Paddon@auug.org.au>

Why do I like Unix? Stepping back and asking fundamental, or even stupid, questions can sometimes lead to some useful insight.

When I played this game recently, I found I didn't actually care a toss for Unix per se. After all it is just a trademark, and indeed the really useful and interesting systems are allowed to use it. So maybe a better question is: why do I spend so much time and effort evangelising Unix and its friends like BSD and Linux? Why pit oneself against the interests of large multi-nationals who promote lowest common denominator systems?

I guess a large part of this motivation comes from the quality and economic benefits of the "free" software industry that grew up around Unix. A strong argument can be made that this never would have occurred without the maverick influence of a vendor neutral platform. I wrote at length on the new dynamics of the software industry in my last column, so I'll let that speak for itself.

Unix viewed as a virtual machine is an interesting proposition, especially in the light of the runaway success of that newest of such things: Java. There is something fundamentally attractive to every programmer about a properly constructed VM, and whilst Unix spoke to a relative few, Java has acted as a wake up call to legions. Not that these are the only two good VM's around, but something in the mix has set these apart, and that something probably bears closer examination. I'll get back to this point, soon.

But, for me, the thing that sets Unix apart is a philosophy. People have tried to describe facets of this in abstract terms... "a toolbox model", "an open system" (whatever the hell that really means), "orthogonal" (if you believe that I have a bridge for sale), and "portable" to name a few. But this is all overintellectualising the issue.

Let me be succinct. The words "It can't be done", in one form or another, are becoming commonplace in our business. This tends to trigger the old red mist because what is usually meant by that is "I opened up the preferences dialogue, and there's no checkbox for that".

There's a great divide in the computing world today, and it is not a rift that can be healed. The multi-

nationals want cost effective product. So long as it sells, everything's OK. If the product isn't adaptable to your needs, learn to fit in the box. And remember, they're on your side.

I want to solve problems, to invent and create, to build, to push the envelope. That's why software is interesting. Humans drive the software, not the other way around.

Perhaps when you distill my argument down, it all reduces to an issue of basic intellectual freedom. This is why Unix lives on, beyond all the expectations of the pie charts of marketing consultants whose judgements don't factor in such intangibles. It's probably why the aforementioned legions of programmers are deserting the Microsoft locking for the Java virtual machine. Its certainly the motivator for the free software out there that is slaughtering the commercial competition in the marketplace.

Live free or die.

Remember those words when you next ask yourself why you are an AUUG member. And also remember that, whilst the word Unix traditionally precedes them, it isn't the only system with a rightful claim to that space.

A President's column at this point in the year that didn't mention the Winter Conference, would be a lot like a marguerita without salt. You'd hardly expect that to happen, and so I won't disappoint.

By the time you read these words, the conference will be close or underway. You'll have probably made your decision to attend (or not) by then, so I won't exhort you any further here. Similarly, enough words have been written on our programme that if you don't know how bloody good it is by now, nothing I say will help.

Suffice to say that I hope to see you there. Don't forget that the AUUG Annual General Meeting will be held during the event. Check the conference timetable for the exact time and location.

Finally, you'll notice that there is a ballot included with this AUUGN which proposes a constitutional change. More specifically, it proposes a change to one of the wind up clauses and is necessary for AUUG to secure tax exempt status with the Australian Taxation Office.

AUUG is currently incurring considerable expense on taxation matters and will continue to do so unless we secure this status. Personally, I think that there are much better things to do with our members' money.

I ask all members to please ensure you complete the ballot and return it promptly.

❖

AUUG '97 Keynote Speakers

It's just around the corner .. here's a brief preview of some of the keynote speakers at this years conference:

ERIC ALLMAN

Eric was the original author of sendmail. He was the Chief Programmer on the INGRES database management project and an early contributor to the Unix effort at Berkeley, authoring syslog, tset, the -me troff macro's, and trek. He received his M.S. in Computer Science from U. C. Berkeley in 1980. He designed database user and application interfaces at Britton Lee (later Sharebase), and contributed to the Ring Array Processor project for neural-network-based speech recognition at the International Computer Science Institute. He was also Chief Technical Officer at InReference, Inc. He co-authored the ``C Advisor'' column for Unix Review magazine for several years and is a member of the Board of Directors of USENIX Association.

Eric has been accused of working incessantly, enjoys writing with fountain pens, and collects wines which he stashes in his extensive cellar.

JON "MADDOG" HALL

In addition to Jon's work with Digital UNIX, Jon is also Executive Director of Linux International, a non-profit organisation dedicated to promoting the use of Linux, a freely distributable re-implementation of the UNIX operating system. Digital is the first system vendor to join Linux International, and is a Corporate Sponsoring Member. Jon is directly responsible for the port of Linux to the Alpha processor.

Jon started his career programming on large IBM mainframes in Basic Assembly Language, but his career improved dramatically when he was introduced to Digital's PDP-11 line of computers as chairman of the Computer Science Department at Hartford State Technical College. There he spent four glorious years teaching students the value of designing good algorithms, writing good code, and living an honourable life. He has also been known to enjoy discussing aspects of computer science over pizza and beer with said students.

Maddog (as his students named him, and as he likes to be called) has his MS in Computer Science from RPI, his BS in Commerce and Engineering from Drexel University, and in his spare time is writing the business plan for his retirement business.

PHIL KARN

is a Staff Engineer for QUALCOMM Inc in San Diego, California. He is well known in the security and ham radio communities, and is the author of widely used public domain packages such as S/KEY and KA9Q. Phil is also pursuing free availability of cryptography by suing the US government for preventing an export license for the diskettes associated with "Applied Cryptography", and has testified to the US Congress on the subject.

DR. MARSHALL KIRK MCKUSICK

writes books and articles, consults, and teaches classes on UNIX- and BSD-related subjects. While at the University of California at Berkeley, he implemented the 4.2BSD fast file system, and was the Research Computer Scientist at the Berkeley Computer Systems Research Group (CSRG) overseeing the development and release of 4.3BSD and 4.4BSD. His particular areas of interest are the virtual-memory system and the filesystem. One day, he hopes to see them merged seamlessly. He earned his undergraduate degree in Electrical Engineering from Cornell University, and did his graduate work at the University of California at Berkeley, where he received Masters degrees in Computer Science and Business Administration, and a doctoral degree in Computer Science. He is a past president of the Usenix Association, and is a member of ACM and IEEE.

In his spare time, he enjoys swimming, scuba diving, and wine collecting. The wine is stored in a specially constructed wine cellar (accessible from the net using the command ``telnet winecellar.McKusick.COM 451").

PAUL MOCKAPETRIS

is an internet advocate and Chief Technology Officer at Software.Com, where he oversees the development of internet infrastructure technology for messaging, directory and DNS. In the past, Paul has been IETF Chair, Director of Engineering for @Home, a member of the Internet Architecture Board, Director of the HPCC Division at USC/Information Sciences Institute, and Program Manager for Networking at ARPA.

Paul is known as the creator of the DNS, and the first implementor of the SMTP mail protocol. He has learner's permits in Physics and Electrical Engineering from MIT, and a PhD in Information and Computer Science from the University of California, Irvine.

DUANE WESSELS

is a Principal Investigator from the University of California, San Diego and is currently a visiting researcher the National Center for Atmospheric Research in Boulder, Colorado. He is currently working on a grant funded by the National Science Foundation to research and develop a national Web caching infrastructure, including the Squid Cache software. Other interests include Internet measurement, multicast, and missions to Mars.

❖

A letter from the Secretary

Dear AUUG Member,

NOTICE OF ANNUAL GENERAL MEETING

The 1997 AUUG Incorporated AGM will be held in conjunction with the AUUG97 Conference and Exhibition.

The AGM will be held at:

The Main Hall
AUUG97
Brisbane Convention and Exhibition Centre
Corner of Merivale Street and Glenelg Street
South Brisbane, NSW, 4101

The meeting will start at 5:10pm on Thursday, 4th September 1997. All AUUG members are invited to attend. Although not required to gain entry, it would be appreciated if you could bring your membership card or membership receipt.

AGENDA

(1) Apologies

(2) Minutes of the previous meeting.
- draft minutes were published in AUUGN v18n1, Feb. 1997.
- draft minutes are also available on-line at <http://www.auug.org.au/agm96.html>
- matters arising.

(3) Returning Officer's report.

(4) President's report.

(5) Secretary's report.

(6) Treasurer's report.

(7) Ratify appointments of Management Committee members.

(8) Ratify appointment of Assistant Returning Officer.

(9) 1998 Chapter Technical Conferences.

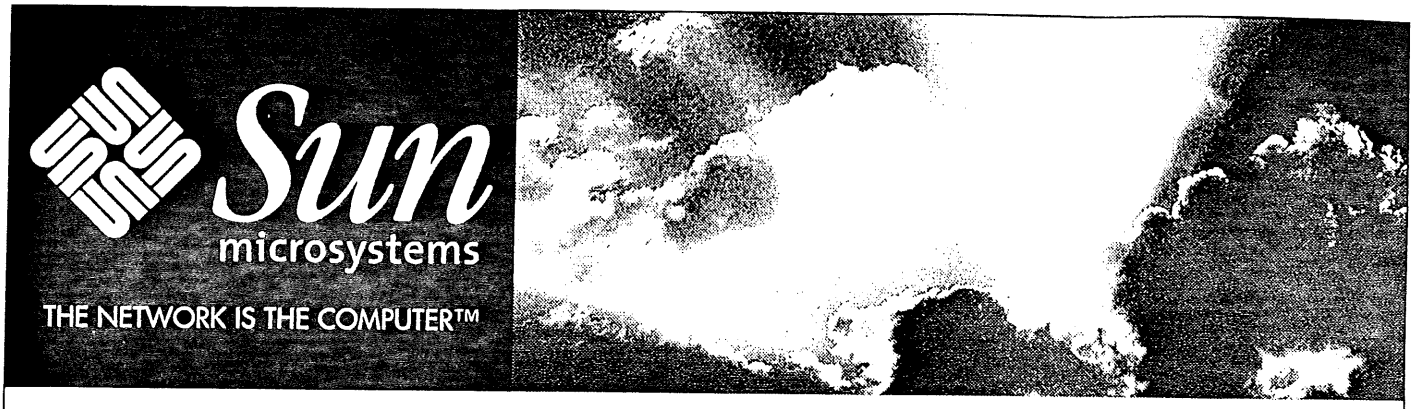
(10) AUUG98.

(11) Any other business.

Sincerely,

David Purdue
Secretary
AUUG Incorporated

❖



Sun Microsystems Inc is a powerful force in the world of enterprise computing. We provide leading edge hardware, software and services. We establish progressive intranets and revolutionise the internet. Sun has annual revenues of $10 billion and a strong presence in over 150 countries.

Systems Engineer

Regional Responsibility

This newly created position reports to the SunSoft Systems Engineering Manager and whilst it is based in Artarmon, you will have regional responsibility for Australia, New Zealand and certain Pacific Rim countries.

This challenging role will be directly responsible for providing pre-sales technical support on the SunSoft product range to both the SunSoft and channel sales teams. Duties will include supporting SunSoft's channel with presentations, demonstrations and preparation of tender responses.

Critical to your success will be your ability to keep abreast of new product releases, their market positioning, competitive advantage and technical superiority, but of equal importance will be your effectiveness in communicating this information to the field force. Naturally, travel to the USA can be anticipated for this purpose. You should have up to five years hands on experience with UNIX and networked systems either in presales or supporting roles, or an equivalent combination of experience and skills. Tertiary qualification in Computer Science and fluency in an Asian language an advantage.

This outstanding technical support role will appeal to experienced candidates seeking to further develop their career with a leader in advanced technology.

For more information call the Head-Start Job Info line : 02 9770 3510 or alternatively, mail/fax your resume, quoting ref no 0779.

HEAD-START RECRUITMENT PTY LTD
Locked Bag 808,
Milsons Point NSW 2061
Fax : 02 9966 1377

h t t p : / / w w w . s u n . c o m

System Administrators - Be Security Aware

Paul Ashley <ashley@fit.qut.edu.au>
Researcher and Consultant
Information Security Research Centre
Queensland University of Technology

System Administrators have it tough. Their day to day work is often stressful and their day rarely ends at the 5:00pm bell. I must agree with the list of skills necessary for a good system administrator (Craig Bishop, "The Network is Down", May '97 AUUGN (quoting directly)):

complex problem solving skills, often under pressure;
performance of installations, upgrades and maintenance;
capacity planning and capital expenditure budgets;
project management skills, so that planned activities are completed - on time and on budget with minimal impact on the day to day business;
can understand complex technical manuals;
can bridge the technology gap for your general staff.

However there is one emerging skill not on the list that is becoming essential for many system administrators: knowledge of security and the impact of the Internet. The story goes something like this. Senior management continually read about the business boom on the Internet. Profits will rise, market share will increase, if they are not on the Internet then they are not in the market. Even organisations who are not profit orientated (for example Government Departments), will see the service benefits of a Web presence. In their infinite wisdom they come to the obvious conclusion: we must be on the Internet and we must be there now!

This is where the system administrator comes into play. A memo will work its way down until it reaches the person in charge of the network. It will say something like this "we need to connect to the Internet, we need to have a Web presence, we need to be secure, and by the way we need to there by the end of this quarter".

This is where many good system administrator's come unstuck. They are generally very much in control of their job, but connecting to the Internet in terms of security is a new game. Most system administrator's are comfortable with the closed environment that is typical to most organisations, but allowing traffic in and out from the Internet, protecting against hostile attackers, is difficult. All of a sudden system administrators need to be experts on firewalls, cryptology, attack strategies, and a range of other topics not normally part of their everyday workload. Unfortunately to date there is much carnage out there. I have experienced the following disasters:

- connections to the Internet without firewalls;
- firewalls badly configured;
- firewall technology used that is far too weak for the nature of the data it is trying to protect;
- organisations relying on a third party to configure their firewalls and not really understanding anything about what they have done or how to maintain it.

Often the system administrator's first knowledge of their poor configuration is when the original senior manager calls the person into their office to please explain why their networks have been attacked, and what the person did to stop it.

So what can system administrator's do to prepare themselves for their forthcoming Internet connection. I suggest the following strategy:

Become security aware. Start reading books and articles on information security so that you understand terms like 40-bit encryption, RSA and proxy firewalls.

Investigate joining AUSCERT, the organisation who's sole purpose is understanding and reacting to Internet Security problems.

Look much closer at the security of your system as a whole. Even if you are not connected to the Internet, could your systems be more secure? Do you really need sendmail on every machine, are your patches up to date?

This should at least get you started. But suppose next week the memo does arrive, what are you going to do? Here is a simple checklist:

Your organisation must have a Security Policy. This document may be just a few pages and should outline what you are trying to protect (just how valuable is the data behind the firewall) and what services you should be allowing through your firewall.

You must have a firewall. Firewalls come in all sorts of shapes and flavours so you must understand which firewall is suitable for you and why.

You must have a Management Procedure for your firewall system. It should outline how configuration changes are to be made and tested, how you are

keeping patches up to date and how you are keeping up to date with newly discovered security bugs in your software.

You should have a periodic security audit. There are many organisations now that can give you an independent assessment of the security of your firewall system, and in most cases will find vulnerabilities.

All of the above unfortunately will result in an extra commitment by system administrator's, but can also be very rewarding. It is much easier to justify expenditure, and you become much more valuable to the organisation if you understand how to manage and control your Internet connections. So don't let your organisation and yourself become another Internet statistic, do as the Scout's do "Always Be Prepared!".

❖

[Over the next few issues, we will be presenting selected articles from the Canberra Conference earlier this year. Thanks to all for making these available to us. - Ed.]

Simplified LP configuration

Peter Gray <pdg@uow.edu.au>
Information Technology Services
University of Wollongong

Introduction

The move from BSD based operating systems (such as Solaris 1 (SUNOS)) to SVR4 based systems (such as Solaris 2) can cause significant problems to the system administrator, not to mention the users. One of the major areas of change is in the printing subsystem.

The BSD lpr/lpd system uses a single daemon to handle all printing requests, both local and remote. The lpd daemon is controlled by a single configuration file. The system is simple to administer but does suffer from some lack of flexibility.

The SVR4 print system uses one daemon to schedule print requests (lpsched) and others to handle connection to remote printers (listen, lpNet). The system is much more complex, cannot be controlled easily by configuration files but does significantly increase flexibility.

The LP system uses a program (usually a shell script) called the model interface program, to handle the actual connection to the printer. The administrator can determine which models are assigned to which printers.

The LP system is complex enough that many administrators choose to simply run the old lpr/lpd system on their SVR4 boxes. This can lead to confusion amongst the user base who may be using standard documentation.

This paper describes a system developed to have the best features of the BSD print system, namely ease of configuration and central control, while still running the SVR4 print subsystem. The system also allows for easier control of printer access and print quotas.

Configuration

One aim of the system was to have all the information about the site's printers reside in one place. A directory hierarchy is used. The top level name of the directory is configurable. At our site it is ``/packages/lp" which will be used in all examples. Subdirectories and their contents are:

bin	Executable programs and scripts.
lib	Other files, such as postscript header files.
servers	Text files describing print servers.
printers	Text files describing available printers.
access	Text files describing any access restrictions.

This area can be shared between machines via NFS or replicated via rdist.

There are 2 main scripts which reside on the ``bin" directory. The startup is run to set up a print server or a print client. It first removes all the old information about existing printers or servers. It then reads the files in the ``servers" directory. If it finds an entry for the current machine, it sets up the machine as a print server, using the information present in the file. It next reads the files in the ``printers" directory and installs a print queue for each nominated printer. After the setup script completes, the machine is ready to function as a print server or client and all printers are configured.

The other script is a model interface script, a replacement for the ``standard" model shipped with the system. It understands how to handle directly connected printers, either serial or parallel as well as Appletalk printers (via the Columbia Appletalk Package (CAP)) and printers on serial connections to Annex terminal servers. As well as being able to handle the above types of printers, it also allows the system administrator to override most of the programs used in the script. This provides almost

unlimited flexibility to handle printers which do not fall into the above categories.

File formats

All the configuration files are sourced directly into the scripts described above. This means they must be valid Bourne shell code. The purpose of the configuration files is to set certain shell variables used by the scripts. The names of the variables and their meaning are documented below.

Print Server Configuration File

Variables which may be set in a print server configuration file are:

SERVER_NAME	The fully qualified name of the print server.
SERVER_PROTOCOL	The print protocol used by the server, either BSD or S5. This is unused in the current version. All servers are set up to listen on both protocols.

Variables which may be set in a printer configuration file are:

PR_NAME	Name of printer (print queue) (required).
PR_TERM	Printer type (required).
PR_DESCRIPTION	Printer description.
PR_CONTENT	Content, usually postscript or simple.
PR_REMOTE	Remote print server if printer is not local.
PR_REMOTE_QUEUE	Remote print queue if PR_REMOTE is set.
PR_REMOTE_PROTOCOL	Print protocol to use to remote server.
PR_CAP	True if printer is on Appletalk.
PR_ANNEX	Name of Annex terminal server.
PR_ANNEX_PORT	Port number on Annex server.
PR_QUOTA	True if quotas apply to this printer.
PR_QUOTA_HOST	Host running the quota daemon for this printer.
PR_DEVICE	Device for local printer (required, but can be /dev/null).
PR_BANNER	True if banner page wanted.
PR_FORCE_BANNER	True if user should not be allowed to suppress banner page.
PR_TTY	True if printer device is a tty style device.
PR_STTY_MODES	Arguments to stty(1) to set up printer device.
PR_PRINT_PROGRAM	Program to do printing (last stage in pipeline).
PR_BANNER_PROGRAM	Program to generate banner.
PR_TRAILER_PROGRAM	Program to generate trailer.
PR_FILTER_PROGRAM	Filter program.
PR_LPCAT_PROGRAM	Program to print file.
PR_QUOTA_PROGRAM	Program to handle print quotas.

Access control

The SVR4 print system does allow the system administrator to restrict access to printers to particular users. However, the system is impractical for a large site where the administrator may want to allow or deny many hundreds or perhaps thousands or users.

Our system allows access to printers to be allowed or denied based on user name or group names. Each printer can have an access control file associated with it. If no access control file exists, access is unrestricted.

Access control files consist of a number of lines, each line containing 3 fields. The first field is the access field and can be ``allow'' or ``deny''.

The second field is the selector field, and selects whether the line refers to a group or a user name. Valid values for this field are ``user'' and ``group''.

The third field is the user/group field. It contains either a valid UNIX group name (from /etc/group) or a valid user name (from /etc/passwd). It may contain the special value ``all'' to match all groups or users.

The algorithm used to control access is simple. The access control flag is initially set to TRUE (access allowed). Each line of the file is read in turn and the access control flag is set to the value of the access field if the current user matches the user/group field. The user's supplementary groups are included in any group checking.

This form of checking will not work where a print server does not share a group file and passwd file with the print clients.

Print quotas

Print quotas are not an integral part of the system. The method used is simple and depends on a separate filter to count pages and decrement a user's print quota.

The local print quota code is not currently included in the distribution because it is still under development.

Example configuration files

The following is a sample configuration file from the ``servers" directory. It describes a single print server.

```
#
#
#
SERVER_NAME=wumpus.its.uow.edu.au
SERVER_PROTOCOL=bsd
```

The following is a file from the `printers" directory. It describes a printer connected via a serial port on an Annex terminal server.

```
PR_NAME=48a
PR_DESCRIPTION="Dot-matrix printer in 2nd
year lab"
PR_CONTENT=simple
PR_TERM=seikosha
PR_ANNEX=csci-ts2.cs.uow.edu.au
PR_ANNEX_PORT=7017
PR_QUOTA=OFF
PR_QUOTA_HOST=wraith.cs.uow.edu.au
PR_DEVICE=/dev/null
PR_BANNER=ON
PR_BANNER_FORCE=OFF
```

The following printer definition file was artificially created to show some of the flexibility of the system.

```
PR_NAME=dummy
PR_DESCRIPTION="This printer does nothing
useful"
PR_CONTENT=simple
PR_TERM=vt100
PR_REMOTE=draci.its.uow.edu.au
PR_REMOTE_QUEUE=draci
PR_REMOTE_PROTOCOL=bsd
PR_QUOTA=OFF
PR_DEVICE=/dev/term/a
PR_BANNER=ON
PR_FORCE_BANNER=OFF
PR_TTY=true
PR_STTY_MODES="ospeed 1200 ispeed 1200"
PR_PRINT_PROGRAM='cat - /usr/pub/ascii'
PR_BANNER_PROGRAM='echo A very small
banner'
PR_TRAILER_PROGRAM='echo End of print job.'
```

This is an example access control file.

```
deny   user   all
allow  group  cs-uow
allow  user   mick
```

Conclusion

The system has evolved out of continued efforts to streamline administration of our printers. It has been in production for almost 12 months with minimal problems. The code is considered beta, mainly because it has not been ported to any other systems and there is little documentation. It can be found at ftp://ftp.uow.edu.au/pub/software/lp.tar.Z

❖

The Making of an Internet Access Point

Karl Auer
PCUG (ACT) Inc.

Background

In late early 1994, the PCUG was considering ways in which it could obtain cheap Internet access for its members. For the average person, the Internet was still pretty specialised, but interest was growing. Also, the limitations of Fidonet were becoming very obvious, and our Bulletin Board was not providing the services the Committee needed in terms of what we would now call an Intranet.

At the same time, AUUG was reaching the limits of its rather crappy Internet access point. They had a battered old Sun and a couple of 9600bps modems sitting in the ANU machine room, with no interactive access to the Internet.

To cut a long story short, we decided to get together. AUUG would supply the initial server, we would join forces on the additional hardware, and we'd share stuff like the bandwidth costs in proportion to subscriptions. AUUG would be able to supply the initial expertise, while PCUG would supply (we hoped) critical mass and thus enough cash to be able to afford the very expensive 64Kbps link to AARNet.

The negotiations took quite a while because it took quite a while to beg, cajole, bully and berate some parts of the PCUG Committee, which was looking at a substantial outlay and a rather intimate relationship with a group it had only just been introduced to. However, agreement was reached eventually.

We kicked off with a Sun 4/370 server which we kitted out with about 11Gb of hard disk and a backup tape drive. We installed a fairly standard set of software on it - WWW, ftp, mail and news - and custom built an accounting system. Cisco and Sun were very good to us - Sun provided the server and Cisco gave us a very nice price on the two 2511 terminal servers we used. Maestro came to the party with a very nice price on some modems too, so with a hub and some cabling we were away.

The official start date we decided upon was 1 February 1995. We estimated that we'd need 250 subscribers to make it at all, and about 300 to break even. We offered all sorts of deals (some of which came back to haunt us) and a free modem as a prize to be drawn from those who subscribed before the opening day. By the time of the official opening, we had over 600 subscribers - to a system which did not exist when most of them had sent their money!

We now have over 2700 subscribers out of a combined membership of about 3700.

Technical Description

We started with one Sun 4/370, 3 AUUG modems and 13 PCUG modems on two Cisco 2511's, all running over a 64Kbps link to AARNet. That soon expanded to 48 PCUG modems and 7 AUUG modems (and the AUUG user to modem ration was still better that the PCUG ratio!).

Now we have two Sun servers, supreme and cheese. Cheese hosts the AUUG web and runs the news server, supreme runs everything else.

Dialups are provided for the PCUG by a Cisco AS5200 - 60 channels coming in on two ISDN Macrolinks. Each channel can accept either a normal call from an ordinary telephone service and a modem, or an ISDN call from people that have the appropriate equipment. This AS5200 is about to handle our soon-to-be-installed DDS Fastway link to the Internet, which will be replacing our current 128Kbps ISDN link.

Dialups for AUUG are provided by the seven modems attached to a Cisco 2511. The 2511 also handles the "private network".

Routing to the Internet is currently handled by a Cisco 2503I, which aggregates the two channels of a Microlink to provide a single 128Kbps link to Access One.

We run extended TACACS on the 2511 and the AS5200; this talks to an extended TACACS server on supreme and drives a custom-built accounting system. When people log on, we check if they have credit and refuse access if they don't. When people log off, we update our accounting database, reducing their credit appropriately.

Our bandwidth usage is a perennial problem. We started with 64K and increased that after less than one year to 128Kbps using MPP to aggregate the two B-channels of our Microlink. We are about to increase it to 256Kbps. No matter what though, we need to minimise bandwidth usage - if not because it is overloaded, then because it now costs money in the form of volume charges. We run a squid proxy server and cache and we use DNEWS to control our news needs somewhat, but over the next year we will have to be more imaginative.

Business Description

Pricing and cash flow

The single biggest item in business terms is cash flow. Capital costs are interesting of course, but at least for us the cost of bandwidth dwarfed any other single cost.

We made a big mistake in the first year - we thought people would use a lot of time online. We sold 400 hours for $120, without any time limit on using it. One of the special introductory offers we made actually doubled that amount. We discovered at the end of the first year that somewhat less than a third of that time had actually been used, and we were in the uncomfortable position of looking at having to service about four hundred people for three years or more before they ever paid us more money!

We have since rectified that mistake, but it would not have been possible for a commercial entity to do what we did, which was simply say to our members "we made a boo-boo and we need to fix it, here is what's going to happen".

Our pricing is now set up so that each member will pay $120 per year, more if they spend a lot of time online. It is important to remember that the service doesn't stop when the customer hangs up; any pricing scheme has to cover your ongoing cash flow requirements.

Because we did not see ourselves as competing, the issue of competitiveness didn't enter our pricing equations and it still hasn't. If we find our members leaving in droves to join a better cheaper ISP, we will fold the Project with a smile.

Service levels

A lot of the commercial ISPs have no service level agreements at all - we felt that this was just too dangerous - you should have an agreement that covers what you will provide for the money people are paying you.

Our service level statement is simple - we guarantee nothing and reserve the right to change anything at any time. Some have called this the Microsoft Method. The policy has served us well, especially

since in practice our service has been stable and effective.

Where we have lost out is in supply. We have hit a modem crunch twice now - one in October 1995 when we were running about 16 modems for 1500 people, and again while we waited desperately for the Cisco AS5200 E1 card to arrive to take us from 44 lines to 60. Our motto has always be "we do what we can with what we have" - this is unlikely to be adequate for a commercial provider, so careful capacity planning is essential.

Bandwidth is another issue, though strangely not nearly as important as modems. As long as data is getting through, most people are happy for it to take as long as it takes. They get annoyed when their online time is expensive or when transfers fail, but they get much more annoyed if they can't get on at all.

That said, insufficient bandwidth is a very frustrating thing, so your bandwidth planning needs to be done carefully. For the small ISP, bandwidth is likely to be the single most expensive item, with the possible exception of wages. There are many rules of thumb; I don't know if any of them work.

"Enough" is defined as much by who your users are as by how many you have. We have found that our bandwidth has been used steadily more - even though the number of simultaneous users has not increased that much, because as users get more sophisticated, their bandwidth needs increase.

Policy

From day one, we have had an Acceptable Use Policy in place, and have required people to sign it when they picked up their login details. In the current legislative climate, we feel that we need written protection from our subscribers, and we make sure that no minor gets an account without a countersignature from their parent or guardian.

The AUP can be summed up as "we guarantee nothing; we can change anything we like whenever we like; anything you find on the Internet is your problem; you may not share your password; you will behave like a reasonable person".

We take this extremely seriously, and it puzzles us why commercial ISPs do not.

Support

Being a PC Users Group, support was patchy but in general pretty effective. We ran courses, we sent people email, we ran clinics once a month, we had newsgroups where we announced stuff and explained things. For the most part though, people worked it out themselves. This was certainly our intention, and part of the reason for giving people online time at such low rates - we wanted people to feel they could play and learn without worrying about the ringing cash register in the background.

However, we had the luxury of volunteer labour and we could tell people to figure it out themselves. A commercial operation has to do better. A word of advice - whatever money and work you put into a foolproof software kit and plain English instructions will be repaid a hundred times in support calls not taken.

We set up a help alias, too. This is just a mailing list, with about thirty people on it. People with problems email the alias, and whoever can answer does so, copying the reply to the list. The result is that the questioner is very likely to get an answer, plus all the helpers learn a bit. We have found that the list has been extremely effective. The help newsgroup has also been popular and effective - the simple fact is that people enjoy helping others, and you can take advantage of this to minimise your support problems.

Redundancy

We run very close to the edge. We take tape backups of everything, but in technical terms we have little or no redundancy. In theory we could move all server functions onto a single box, but that's about it. We have been squirrelling away spare parts for the Sun boxes as we came across them.

This ties into service levels. We have a support contract with Cisco and are going to enter one with Sun. If we were commercial the risk of even one day's outage would be unacceptable, and we would have to purchase either a gold-leafed support contract or have a very good disaster plan in place.

Political Description

The Internet Project is run by a group called the Internet Project Management Committee, IPMC for short. This has three reps from each of the two Groups, and has control of the Internet Project according to the agreement between the Groups.

While this has been very effective in technical terms, it has been something of a problem in political terms and in some practical ways. The IPMC has no authority to spend money, so every cent has to be extracted from the PCUG and AUUG Committees. Sometimes this has been like getting blood from a stone.

In a game that moves so fast, it is vital that you have a technical management structure that can deal with problems fast and take advantage of changes rapidly.

The future?

In the immediate future, I see smaller providers getting together and forming peering arrangements. This reduces their costs in terms of paid-for volume, takes much of their traffic off their Internet links, and offers benefits in areas such as hot standby bandwidth to the Internet, offsite backup of critical data, account portability, emergency dialup overflow capacity and so on.

In the longer term, say the next two years plus, I think the access market is likely to change radically. With Telstra's Big Pond entering the act, with competition between access providers increasing and with mooted changes in all sorts of areas such as local call costs, ISDN costs and so on, most access seems likely to end up the hands of a few big providers.

This process will be hastened because people will start to want services that only the big providers can offer - roaming accounts, 13 access numbers and so on.

However, the opposite may happen! If telecommunications costs don't go down, the small provider may find a niche in a kind of "Internet club" market, providing boutique services or even just caching services to a small, local clientele.

❖

Hosting Virtual Domains

Jeremy Laidman
Canberra Institute of Technology
JPLaidman@ACSLink.net.au

Whether we like it or not, the Internet is moving into a commercial phase. Businesses are keen to have a presence on the Internet to tap into the global marketplace it provides. Creating a complete Internet-presence package for business can be financially rewarding for a service provider. In addition, many non-profit or subsidised organisations with Internet connections are being financially squeezed, and are looking for new ways of generating revenue.

Virtualising Internet domains allows Internet services for different domain names to be physically located on a single computer, saving computing and communications hardware. A single computer is capable of hosting hundreds of virtual domains, a potential money-spinner for an organisation that already has the hardware in place for their own services. This paper investigates some of the ways of providing virtual services, and looks specifically at web-page, email and finger services.

Introduction

While installing a web server at the Canberra Institute of Technology, I began investigating the prospect of generating revenue by hosting another organisation's web pages on CIT's server. Very soon it became clear that to a paying customer, an Internet presence included their own place in the DNS space. It should be obvious how much more professional an organisation looks when their home page URL is http://www.myorg.com compared to http://www.someone-else.com/~myorg.

Around this time I decided to take a look at the Apache web server - based on the NCSA server's publicly available source code, it extended the capabilities far above those of its ancestor. Today it's the most popular web server software on the Internet, estimated at more than 40% of all sites - more than three times its nearest competitor. (Netcraft, 1997)

One major advantage of Apache was that it could host these virtual domain thingies. This meant that the web server would give you a different set of pages depending on how you addressed it. All you needed to do was point two different DNS entries at the same computer and Apache would handle the rest. After implementing a virtual web server or two, I began to investigate other services that could perform the same magic. I will describe some of them here.

Methods of Virtualisation

There are two methods of virtualising, depending on the strategies the server can use to find out what the client is trying to see. One method uses multiple IP addresses on the one host, enabling the server software to detect the IP address of its own side of a socket. The other method requires the protocol of the service to mandate that the client indicate to the server the domain name of the service it wants to use.

IP-Based Virtualisation

When a socket is established between a client and a server, the socket has a number of attributes, including the IP addresses and ports of both ends of the socket. For most servers, the IP address of its

own end of a socket is always the same, but for an IP-based-virtualising host this isn't the case. The server software looks at its own IP address attribute for the socket and modifies its behaviour accordingly.

A host can be given multiple addresses simply by installing more than one network interface card (NIC). Each NIC is given a unique IP address, and each address has a different DNS entry. The DNS names need not be in the same domain. For example, eth0 (ethernet adapter 0) hosts the local domain of www.localdomain.com.au, eth1 hosts www.otherdomain1.com.au and eth2 hosts www.otherdomain2.edu.se.

When it comes time to host your fifteenth virtual domain, your computer will probably come close to being full of NICs. This is where IP aliasing comes into play. IP aliasing allows a single NIC to own multiple IP addresses. Many versions of UNIX allow IP aliasing, including Linux, Free-BSD and Solaris. Under Linux 2.x, the ifconfig program is used repeatedly on the ethernet device, once for each IP address.

```
# insmod ip_alias
# ifconfig eth0 161.50.48.5 netmask 255.255.255.0 broadcast 161.50.48.255
# ifconfig eth0:0 161.50.48.7 netmask 255.255.255.0 broadcast 161.50.48.255
# route add -host 161.50.48.7 -dev eth0:0
# netstat -i
Kernel Interface table
Iface   MTU Met  RX-OK RX-ERR RX-DRP RX-OVR  TX-OK TX-ERR TX-DRP TX-OVR Flags
lo     3584   0   2667      0      0      0   2667      0      0      0 BLRU
eth0   1500   0 491391      4      0      0  33964      2      0      0 BMRU
eth0:0 1500   0      0      0      0      0      0      0      0      0 BRU
```

Protocol-Based Virtualisation

Some TCP/IP services are designed in such a way that multiple IP addresses are unnecessary. In particular, the HTTP/1.1 (world wide web) protocol requires that a client identify to the server the DNS name to which it is connecting. A number of conversations between a web client and a virtualising web server are shown in appendix 1 to demonstrate this. The author's keystrokes are highlighted.

Although the HTTP/1.1 specification has only recently become an RFC, many of its innovations have been implemented in browsers for some time. For example, Netscape from 2.x has been able to access protocol-based virtual servers because it sends the Host header to the server.

Examples of Virtualised Servers

Virtual Web Server

As described above, the HTTP/1.1 protocol specification allows for virtual domains without IP aliasing techniques. Apache from version 1.1.1 is able to perform protocol-based virtual hosting. Apache simply requires a <VirtualHost> section in the httpd.conf file, such as is shown here.

```
<VirtualHost www.virtual.com.au>
    ServerName www.virtual.com.au
    DocumentRoot
/usr/local/etc/httpd/virtual
</VirtualHost>
```

Other settings can appear within the <VirtualHost> block, allowing, for example, different log files and CGI directories for each virtual host.

As there are some browsers that don't yet conform to the HTTP/1.1 specification regarding the Host: header, the only way to ensure that all browsers can access all of your virtual hosts is to use IP aliasing. The configuration of Apache is the same as with protocol-based virtualisation - if no Host: header is present, it will revert to IP-based virtualisation. The DNS points different hostnames to different IP addresses instead of to the same address.

Virtual Email

Hosting email for virtual domains can be a very tricky business. Although the SMTP protocol provides enough information for the server to establish protocol-based virtual hosting, the sendmail program (the standard UNIX program for SMTP mail delivery) is not designed for virtual hosting.

There are two general solutions for virtual email domains, one provides an email forwarding service, the other allows user accounts to be held locally. The forwarding solution involves using sendmail's ability to use a database for resolving email address rewrites.

Account Synonyms

The solution that is used at CIT involves only some minor configuration changes in the file sendmail.cf. The effect is that users who have accounts on the mail machine will receive mail regardless of the domain in which they reside. The email addresses

<fred@realdomain> and <fred@virtualdomain> are synonymous.

Step 1 - Add domains for which we want to receive mail to the w class.

```
Cw localhost cit.act.edu.au aihs.edu.au
```

Any mail ending in one of these domains will be accepted by the mailhost. Any other mail will be rejected.

Step 2 - Turn off masquerading by having a null M define.

```
DM
```

This stops headers in outgoing virtualised mail from being rewritten as if coming from the non-virtual domain.

Step 3 - Kill and restart the sendmail daemon.

Step 4 - Create an MX record for the virtual domain, pointing at the master mail server. Here's a section of the db.aihs file.

```
$ORIGIN aihs.edu.au.
...
mailhost IN CNAME mailhost.cit.act.edu.au.
@        MX 10    mailhost
...
```

Mail delivery systems should consult the DNS system to determine the correct target host for a particular domain. The above entries will direct all mail for the aihs.edu.au domain to the host mailhost.cit.act.edu.au.

Account Forwarding

The forwarding approach involves creating a file as shown below. The syntax of the file allows for forwarding single email addresses, remaining unmatched addresses in a domain, or all addresses in an entire domain to matching addresses in another domain.

```
fred@virtualdomain1   fred@realdomain1
bill@virtualdomain1   harry@realdomain1
virtualdomain1        postmaster@realdomain1

@virtualdomain2       realdomain2
```

The sendmail configuration file needs to have several rules added which parse addresses using the file. For more information, and a more extensive description, refer to the URL http://www.westnet.com/providers/. (Candrera)

Virtual Finger

Because all user accounts at CIT are behind a firewall, the finger service provided at cit.act.edu.au were virtually useless. It was decided that a finger server that displayed the standard CIT email format would be more helpful. To investigate how a virtual server makes use of IP aliasing, I implemented a virtual finger server that displays different messages when accessed as different hostnames. A section of the finger server is shown in Appendix 1. The entire finger server (written in perl) is available at ftp://ftp.cit.act.edu.au/pub/virtual/finger.pl.

After finding its own DNS name, the finger server displays a message file, named /etc/finger-message.domain where domain is the domain name of the connection being made. If a matching filename can't be found, /etc/finger-message is used.

Here's the file /etc/finger-message.aihs.edu.au.

```
$ finger @aihs.edu.au
[ aihs.edu.au]

Welcome to the Australian Institute of Hospitality Studies (AIHS)
standard finger response message.

To send email to a staff member at the AIHS, send to
    <firstname>.<surname>@aihs.edu.au
for example
    Peter.Clarke@aihs.edu.au

Any questions can be directed to postmaster@aihs.edu.au.
```

Conclusion

As shown in this paper, there is some scope for hosting virtual domains for a number of services. If a protocol itself can't handle virtual hosting, the use of IP-aliasing can allow for multiple domains. The requirement is that server software be capable of virtualising. A service provider should be able to use the techniques described here to host virtual domains for a range of services.

One generalised approach to this might be a virtualising super-server that performs a chroot before executing the real server, thus keeping each virtual domain in its own directory structure, holding its own documents and configuration files. This scheme is the subject of further investigation.

Appendix 1 - Example Web Transactions

Note that in the first two connections the GET command specified the absolute URI, including the DNS name of the server. Even though the connections in examples 1 and 2 were made to the same IP address, the server was able to provide a different page in the second case because the client identified where it was trying to connect. The clients conforming to the older HTTP/1.0 protocol don't use absolute URIs (in fact HTTP/1.0 forbids it for non-proxy connections), and so in the third case the server can only return from its default page set. The final example shows HTTP/1.1's Host: header in action. Even though the requested URI is not absolute, the client is still able to indicate the desired page set.

Example 1 - Absolute URI

```
$ telnet www.cit.act.edu.au 80
Trying 161.50.48.3...
Connected to spider.cit.act.edu.au.
Escape character is '^]'.
GET http://www.cit.act.edu.au/ http/1.0

HTTP/1.0 200 OK
Date: Fri, 17 Jan 1997 12:02:57 GMT
Server: Apache/1.1.1
Content-type: text/html
Content-length: 4201
Last-modified: Mon, 16 Dec 1996 04:36:14 GMT
...
```

Example 2 - Absolute URI

```
$ telnet www.cit.act.edu.au 80
Trying 161.50.48.3...
Connected to spider.cit.act.edu.au.
Escape character is '^]'.
HEAD http://www.aihs.edu.au/ http/1.0

HTTP/1.0 200 OK
Date: Fri, 17 Jan 1997 12:04:11 GMT
Server: Apache/1.1.1
Content-type: text/html
Content-length: 288
Last-modified: Fri, 09 Aug 1996 06:39:13 GMT
...
```

Example 3 - Relative URI

```
$ telnet www.cit.act.edu.au 80
Trying 161.50.48.3...
Connected to spider.cit.act.edu.au.
Escape character is '^]'.
GET / http/1.0

HTTP/1.0 200 OK
Date: Fri, 17 Jan 1997 12:08:29 GMT
Server: Apache/1.1.1
Content-type: text/html
Content-length: 4201
Last-modified: Mon, 16 Dec 1996 04:36:14 GMT
...
```

Example 4 - Relative URI with Host Header

```
$ telnet www.cit.act.edu.au 80
Trying 161.50.48.3...
Connected to spider.cit.act.edu.au.
Escape character is '^]'.
GET / http/1.0
Host: www.aihs.edu.au

HTTP/1.0 200 OK
Date: Fri, 17 Jan 1997 12:15:55 GMT
Server: Apache/1.1.1
Content-type: text/html
Content-length: 288
Last-modified: Fri, 09 Aug 1996 06:39:13 GMT
...
```

Appendix 2 - Section of Virtual Finger Server

The full server is available as ftp://ftp.cit.act.edu.au/pub/virtual/finger.pl.

```
use Socket;

    $sockaddr = 'S n a4 x8';

    # launched from inetd so the socket is STDIN

    $myaddr = getsockname(STDIN) || die "Invalid socket";
    ($af,$port,$inetaddr) = unpack($sockaddr,$myaddr);
    @myinetaddr = unpack('C4',$inetaddr);
    $myip = join(".",@myinetaddr);
    $myname = gethostbyaddr $inetaddr, AF_INET;

    # now we have $myip and $myname we can virtualise
...
```

References

Berners-Lee, T et al, *Hypertext Transfer Protocol -- HTTP/1.0 RFC 1945*, MIT/LCS May 1996

Candreva, Chris, *Scripts and Patches for Internet Service Providers* <http://www.westnet.com/providers>, Jan 1997

Fielding, R et al, *Hypertext Transfer Protocol -- HTTP/1.1 RFC 2068*, MIT/LCS, Jan 1997

Mirzadeh, Aram *Configuring Linux to run Apache virtual hosts* <http://www.qosina.com/~awm/apache/virtual.html>, Jul 1996

Netcraft, *The Netcraft Web Server Survey* <http://www.netcraft.co.uk/survey>, Jan 1997

Pancamo, Dan, *VirtualWeb-Mini-HOWTO* <http://sunsite.unc.edu/mdw/HOWTO/mini/Virtual-Web>, Nov 1995

Pillay, Harish, *Mini How-to on Setting Up IP Aliasing On A Linux Machine* <http://sunsite.unc.edu/mdw/HOWTO/mini/IP-Alias>, Nov 1996

Postel, J, *Simple Mail Transfer Protocol RFC 821*, USC/ISI, Jan 1982

Ring, Inc, *Virtual Domains* <http://www.linux-consulting.com/FAQ_virtual/>, 1994-6

Trümper, Winfried, *Virtual FTP-servers with wu-ftpd* <http://sunsite.unc.edu/mdw/HOWTO/mini/Virtual-wu-ftpd>, Oct 1996

Wall, Larry and Schwartz, Randal L, *Programming perl* , O'Reilly & Associates, Jan 1991

Zimmermann, D, *The Finger User Information Protocol RFC 1288*, Center for Discrete Mathematics and Theoretical Computer Science, Dec 1991.

❖

Performance Analysis of Software

John Wright
CSC Australia, Canberra
jsw@act.csa.com.au
February, 1997

Introduction

In order to determine whether software and/or hardware being considered for a particular application will adequately meet performance requirements, it is sometimes necessary to conduct full-scale tests on the proposed system. In this context, "system" encompasses the hardware, network and all software (operating system, commercial-off-the-shelf (COTS) and purpose built software) that, when used together, provides to the users with an application. I have been involved with commercial systems involving database management systems (DBMS) running in a UNIX environment. It is these systems that is the focus of this paper.

As it is expensive to setup and conduct suitable system tests, they are usually restricted to large scale systems, which means databases in the gigabyte range with hundreds of active users and average response times of less than five seconds. The main purpose of the tests is usually to determine the upper limit of acceptable performance of the system and to prove that the specified system could meet performance objectives.

The approach taken in conducting the performance experiments will first be outlined then some results from the experiments will be given and analysed.

System Testing Approach

A useful approach to system testing is to treat it as a series of (scientific) experiments. This implies that the tests should be documented, controlled and reproducible, with all results analysed to provide a better understanding of the system's behaviour. At the completion of a test, it should be possible to say why the results differed from a previous test, or why the system behaved in the manner observed.

There are two basic approaches to system testing. One is to have actual users of the system running through a series of scripted transactions and the other relies on the simulation of those users through the use of programs. The first approach is not reproducible or practical for large-scale systems, which leaves the second which is also known as remote terminal emulation (RTE) as the only viable option. Setting up an RTE simulation is expensive and time consuming as the workload of the users must be accurately determined and representative work patterns and mixtures determined.

Not all user transactions need to be simulated, only those that will have an impact on the system performance and behaviour. Thus weekly or monthly updates to, say, control data would be excluded, but a monthly batch update and reporting job, for instance, that runs for several hours might be simulated. An existing system may be able to provide examples of typical transactions but care needs to be taken that these transactions will still be relevant in the new system. If tests are being conducted on an existing system, care needs to be taken that all extraneous system activity is removed to ensure that results obtained are valid.

In addition to defining the transaction types, the correct mix of transactions needs to be specified as well as the "think time" defined. This is the time lapse between each transaction submitted by the user.

There are four variables involved in user simulations and are related by the formula shown in Figure 1. To a large extent, the goal of the tests determines which of the four variables will be the dependent variable.

Having defined the user workload, an ASCII data file containing the user transactions is created. The transactions in the data file are in random order so that caching, either by the operating system or the database, is somewhat representative of the real system. The number of transactions in the file is such that the entire file can be processed by the system in around 30 minutes. During a test, the system must be able to reach "steady state" processing. If longer run times are required, the test can cycle through the data file.

$$tps = \frac{Nbr\_of\_Users}{(Avg\_Cmd\_Time + Pause\_Time)}$$

where:

tps =	Transactions per Second
Nbr_of_Users =	number of users to be simulated
Avg_Cmd_Time =	Average time, in seconds, taken for each transaction to complete
Pause_Time =	Time, in seconds, between each transaction submitted by a user.

Note: all times are rounded up to the nearest second.

Figure 1 - User Simulation Formula

The data file has a simple, tab-delimited, structure as shown in Figure 2. The first field contains the transaction type, while all remaining data on the line are specific to the transaction type and is in the order required by the transaction as implemented in the simulation program.

```
1   Canberra
5
3   10    AUUG  14 Feb
2   ANU   Canberra    ACT
1   Sydney
```

Figure 2 - Sample Transaction Data File

A *driver* program reads transactions from the data file and writes that data, after any necessary formatting, to *stdout*. This output is piped to a second program, *tester*, which reads *stdin*, and is responsible for:

- any system initialization processing, eg. connecting to a transaction processing (TP) monitor;
- connecting to the database once at startup;
- applying the transaction to the system;
- responding to any error conditions;
- recording the time taken for each transaction;
- writing the results of each transaction, after any necessary formatting, to *stdout*.
- disconnecting from the database at the completion of the test; and
- any termination processing.

For example:

```
sys[tst]% driver < data.file |
tester > output.file 2> error.file
```

The output format used by *tester* is either fixed width columns or tab delimited, with an alphanumeric code for each record type, to allow the output data to be processed by one or more of the following; UNIX utilities such as *grep*, *sed* and *awk*, custom developed *results* program or a spreadsheet, such as Microsoft Excel. Using a spreadsheet provides easy access to statistical functions for data analysis and graphical presentation of results.

The *driver/tester* program pair is replicated once for each user to be simulated, eg. a 600 user simulation would require 600 instances of the *driver/tester* program pair to be running. Depending on the system being tested, they could either be run on a small test harness machine or on the test machine itself. Using a smaller machine as a test harness does not appear to cause any problems with network traffic, as the amount of data exchanged between the machines is fairly small.

Either approach requires careful thought to be given to the placement of files used by the test process. These files must be on disk(s), and preferably controllers, that are not used by the system under test and that do not interfere with the system during the test.

In order to manage the test process, a *control* script is used that will:

- document the parameters used for the test run;
- record the system configuration at the start of the test, eg. database configuration file(s) and operating system version and patch level;
- start the specified number of *driver/tester* instances (number of users);
- record the state of the system at the start and end of the test run;
- start any necessary monitoring, eg. *iostat*, *vmstat*, *ps*;
- stop the test run after a specified time (typically 30 minutes); and
- initiate any necessary post-processing of results.

A code fragment from a *control* script is shown in Figure 3.

```
#!/usr/ksh

# Input parameter processing code omitted

# Initialization code omitted

# System state recording code omitted

# Monitoring code omitted

loop=1
users=500
while (( loop <= users ))
do
    driver < data.file | tester  > output.file.${loop} \
                               2> error.file.${loop}  &
    sleep 1
done

# driver/tester process shutdown code omitted

# Post-processing code omitted

# Script termination code omitted
```

Figure 3 - *control* script code fragment

To avoid potential problems associated with several hundred processes starting in rapid succession, eg. contention within the database management system, and/or the operating system, a sleep statement is used to slow down the rate of process creation and database connection requests from the *tester* program. The length of time taken to start all *driver/tester* program pairs should be taken into account during any results processing.

A self documenting directory structure is used to hold all files involved in the tests. This allows for easy backup when the tests have been completed.

The documentation produced by the *control* script is captured in an output directory specific to each test. The directory contains:

- a copy of all input data file(s) used for the test;
- a copy of the *control* script used for the test;
- system configuration files;
- all output files, including error files, generated during the run;
- a copy of the output produced by the *control* script;
- a process listing from *ps*, at the start and end of the run. This can be used to determine what processes where running at the start and end of the test run, and to check if any *driver/tester*

program pairs terminated abnormally during the run;

- all system monitoring files from *iostat*, *vmstat*, *ps*, *etc.*; and
- any post-processed results files.

Taking copies of the data files and the *control* script used for each test allow the data and *control* script to be changed during the conduct of the tests should this become necessary.

Testing Process

Before serious testing commences, some preliminary tests should be conducted to ensure that the test process is working correctly and that any gross configuration issues have been resolved. For instance, the database under test may not be configured correctly and could be using the operating system swap disk for temporary sort space leading to severe disk I/O problems.

The use of a visual tool such as *Virtual Adrian* in the Solaris environment or *Performance/6000 toolbox* in the AIX environment provide a useful high level view of the system's performance and are a good guide to potential performance bottlenecks. They also help in gaining an insight into the system's behaviour during a test. Once a potential problem has been identified further, detailed, investigation can be carried out using the data collected by the monitoring tools started by the control script.

Test Example

The following illustrates a series of tests that were conducted on a database running on a small machine with around 256 megabytes of memory. The purpose of the tests was to validate the test process, including results analysis, before testing a larger scale system. The tests simulated two different user loads, both with the same transaction rate of about 10 transactions per second (tps). From previous tests, it was determined that the average time for all transactions was less than one second.

From the formula given in Figure 1, it can be seen that as the number of users is varied, the think time must be varied also in order to keep the transaction rate. For the test runs shown in Figure 4 and Figure 6, a think time of 20 seconds was used for 200 users and an 80 second think time for 800 users.

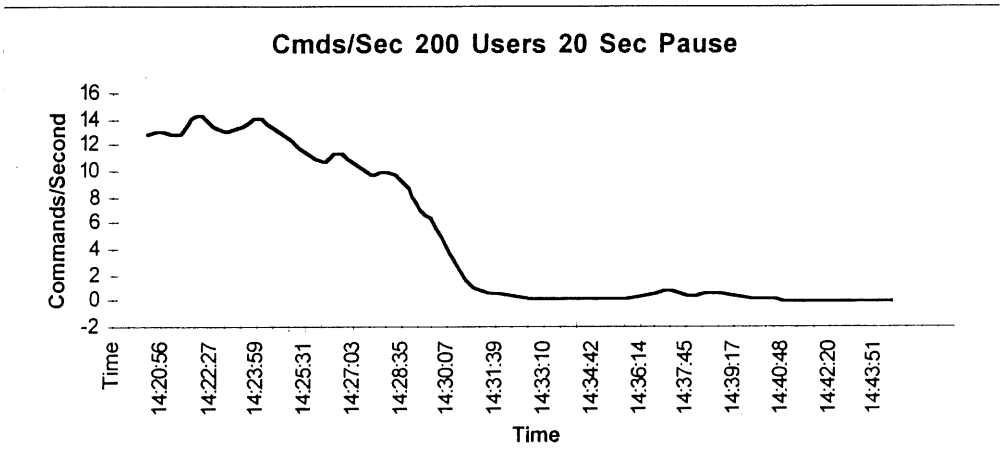
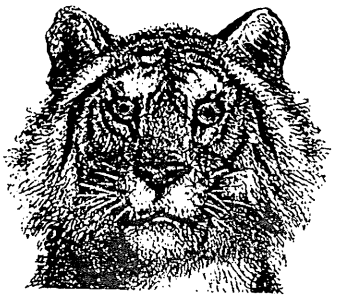


Figure 4 - Database TPS at 200 Users with 20 Second Think Time

From Figure 4 it can be seen that the transaction rate started to drop off after about time 14:23. From Figure 5, it can be seen that at time 14:23, the database management system process size had increased in size to around 150 megabytes. At this point paging commenced as the machine's physical memory of 256 megabytes was completely allocated and the database process was being paged out to disk.

Leap forward with O'Reilly



O'Reilly is the key to helping you master Internet publishing and programming topics, going above and beyond mere mechanics.
O'Reilly books explore subjects in-depth and are written and reviewed by experts.
For all levels of professional users, consider O'Reilly as the only solution.

1565920163
Vol 6A Motif Programming Manual
Heller
O'Reilly & Associates $69.95
A source for complete, accurate, and insightful guidance on Motif application programming. Learn how to write applications using the Motif toolkit from the Open Software Foundation (OSF).

1565922689
Oracle Design
Ensor
O'Reilly & Associates $79.95
A thorough look at the field of Oracle relational database design including the project life cycle, data models, and such classic design problems as keys, data integrity, stored procedures, and denormalisation.

093717582X
TCP/IP Network Administration
Hunt
O'Reilly & Associates $59.95
Here is a complete guide to setting up and running a TCP/IP network for administrators of networks of systems or lone home systems that access the Internet. This book covers troubleshooting and security, several important packages available from the Net, and BSD and System V TCP/IP implementations.

1565922786
Sendmail Desktop Reference
Costales
O'Reilly & Associates $9.95
This quick reference guide to the sendmail program provides a complete overview of sendmail, from command line switches to configuration commands, from options declarations to macro definitions, and from m4 features to debugging switches - all packed into a convenient, carry-around booklet.

1565921275
Essential System Administration 2/e
Frisch
O'Reilly & Associates $65.00
This guide is for those who use a stand-alone UNIX system, those who routinely provide administrative support for a larger shared system, or those who want an understanding of basic administrative functions.

1565920015
UNIX In A Nutshell System V & Solaris 2 2/e
Gilly
O'Reilly & Associates $19.95
A complete reference containing all commands and options, along with generous descriptions and examples that put the commands in context. For all but the thorniest UNIX problems this one reference should be all the documentation you need. Covers System V Releases 3 & 4 and Solaris 2.0.

1565921518
Running Linux 2/e
Welsh
O'Reilly & Associates $59.95
This second edition of Running Linux covers everything you need to understand, install and start using your Linux system, including a comprehensive installation tutorial, complete information on system maintenance, tools for document development and programming and guidelines for network and Web site administration.

156592262X
Java in a Nutshell 2/e
Flanagen
O'Reilly & Associates $39.95
This second edition of Java in a Nutshell has been updated to cover Version 1.1 of the Java Development Kit (JDK). It contains descriptions of all the Java classes in the Java Core API, with a definitive listing of all methods and variables. This is the only quick reference that a Java programmer will need.

1565922352
HTML: The Definitive Guide 2/e
Musciano
O'Reilly & Associates $65.00
This revised and updated guide helps you become fluent in HTML, fully versed in the language's syntax, semantics, and elements of style. It covers the most up to date version of the HTML standard, plus all the common extensions, especially Netscape extensions, and how all elements interact with each other.

1565921496
Programming Perl 2/e
Wall
O'Reilly & Associates $79.95
This heavily revised second edition of Programming Perl contains a full explanation of the features on Perl. It covers version 5.0 Perl syntax, functions, debugging, efficiency and the Perl library. It also includes a Perl cookbook and a quick reference card.

O'REILLY™

Distributed in Australia by WOODSLANE
Phone (02) 9970 5111 Fax (02) 9970 5002 Email techbooks@woodslane.com.au

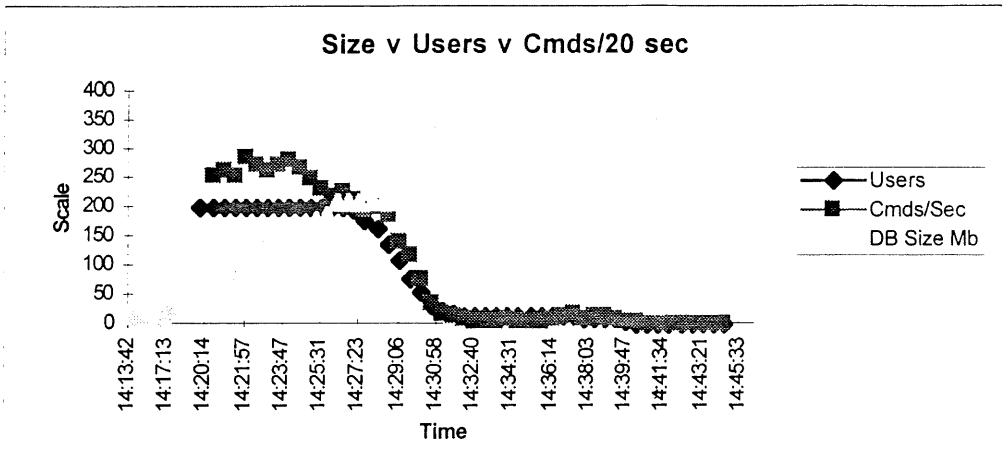


Figure 5 - Database Size verses TPS at 200 Users with 20 Second Think Time

The same test was repeated with 800 users as shown in Figure 6, but instead of the reasonably well defined transaction rate profile of Figure 4, this test shows only an irregular slow down in the transaction rate caused by the large amount of memory, allocated within the database process, for the user. This, in turn, causing early, severe process paging.

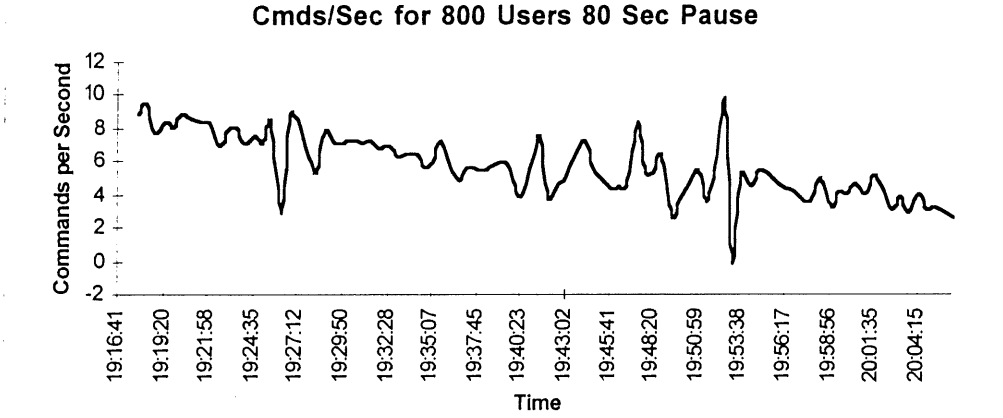


Figure 6 - Database TPS at 800 Users with 80 Second Think Time

These two simple examples show the type of analysis that can be undertaken of test results in order to understand the behaviour of a system. Clearly, there is much additional work that can be done, and to a large extent this is governed by the goal of the tests being conducted.

Results Analysis

Each test produces a single measurement statistic and it is important to realise that there are random variations between each run that occur for reasons outside the control of the person conducting the test. It is unwise to rely on single point values without doing sensitivity analysis of the results and some statistical analysis of the data.

Results should be reported as a mean with a standard deviation, and where appropriate, a 95% confidence interval around the mean. The calculation of the confidence interval can be done simply using the CONFIDENCE function provided by a spreadsheet or from the formula given in a statistics text book.

It is possible to use a simple rule of thumb that says there is 95% probability that the real value will lay within the mean plus and minus 3 times the standard deviation. For example, assume that the average

transaction time was two seconds with a standard deviation of 0.25 seconds, then there is a 95% probability that the real transaction time will be somewhere between 1.25 seconds and 2.75 seconds. Although this seems imprecise, there is a long history statistical theory and analysis, and it is unwise to ignore it.

Microsoft Excel has a wide range of statistical analysis functions that are easy to use, however, it is worthwhile becoming familiar with elementary statistical theory to ensure that the correct statistical functions are used to analyse the data. The more complex the systems being tested, the more important becomes the need to have good statistical knowledge. For instance, in some cases it becomes necessary to determine whether two results obtained from two different tests are statistically equivalent.

Conclusion

The approach outlined in this paper for user simulation has been used successfully in testing several large-scale systems, and can be adopted for testing most systems. The keys to the successful testing of systems lie in the accurate modelling of the user transaction profile and the systematic recording of tests and their results.

It is necessary to undertake rigorous statistical analysis of test results to ensure that results obtained are meaningful and can be used with confidence when determining system configurations. As the size and complexity of systems increases, the correct analysis of test data becomes more critical, as decisions based on incorrect test results can have a severe financial penalty.

❖

AUUG BOOK CLUB & PRENTICE HALL AUSTRALIA

Oracle Electronic Resource Kit contains 11 searchable books on CD-ROM - a collection of the best Oracle tools and utilities plus Oracle 7.3 Developer's Guide. Here is complete coverage on everything from administration to performance tuning to designing interfaces to server/OS maintenance.

Oracle Database Administration on UNIX is the first comprehensive guide for both new and experienced DBAs and managers. The book presents a thorough overview of Oracle database architecture, step-by-step guidance for planning, sizing and installation and expert guidance on UNIX-specific configuration and tuning issues.The accompanying CD-ROM includes a multimedia installation and configuration tutorial.

Oracle & UNIX Performance Tuning addresses tuning of both current Oracle releases and the underlying Solaris HP-UX or Sequent Dynix systems and is also the first Oracle database tuning book with detailed, specific coverage of optimising both OLTP and OLAP decision support systems. The CD-ROM which is included contains a collection of tuning tools.

Maximum Security is designed for systems administrators and managers who need to find out how to protect their computers, networks and Internet sites from unauthorised intrusions.Written by an experienced hacker, this unique guide to Internet and networks security identifies the security holes and faults and then describes how to go about fixing them. The CD-ROM includes an assortment of evaluation versions of third-party products.

LDAP: Programming Directory-Enabled Applications is for those who design or program software for network computing or are interested in directory services. Learn how to understand the LDAP API, to write LDAP programs, to LDAP-enable an existing application and to use a set of command-line LDAP tools to search and update directory information.

Fire in the Computer Room, What Now? has the collected wisdom of an international team of IBM experts who share their combined decades of experience in simple, everyday terms. The book will show you how to assess your risks and requirements so you can implement and maintain a recovery solution that suits your needs.

20% DISCOUNT TO AUUG MEMBERS

Please send me a copy/copies of the following books

☐	Oracle Electronic Resource Kit 067231097X	~~$159.95~~	$127.95
☐	Oracle Database Administration 0132446669	~~$69.95~~	$55.95
☐	Oracle & UNIX Performance Tuning 0138491674	~~$69.95~~	$55.95
☐	Maximum Security: A Hacker's Guide to Protecting Your Internet Site and Network 1575212684	~~$89.95~~	$71.95
☐	LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol 1578700000	~~$69.95~~	$55.95
☐	Fire in the Computer Room, What Now? 0137543913	~~$49.95~~	$39.95

AUUG members receive 20% discount below recommended retail price . Please send me the selected book/s on 30-Day approval

Name: _____ Position: _____

Company: _____

Address: _____

_____ Telephone: _____

PLEASE SEND MY BOOK/S ON 30-DAY APPROVAL

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

☐ Charge to me **OR** ☐ Company purchase Order No. _____

Please charge my: ☐ Bankcard ☐ Visa ☐ MasterCard ☐ AMEX

Expiry Date: _____ Credit Card No: ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

Signature: _____

PRENTICE HALL AUSTRALIA
14 Aquatic Drive, Frenchs Forest NSW 2086
Tel: (02) 9454 2200 • Fax: (02) 9453 0117 A.C.N. 000 383 406

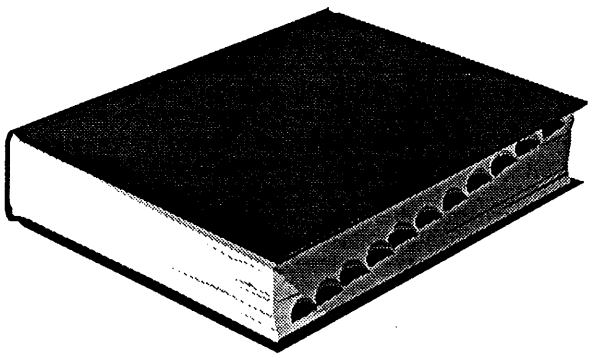
Book Reviews

Frank Crawford <Frank.Crawford@auug.org.au>

Another issue of AUUGN rolls around, and it is time to look at some of the latest books. This issue is a bit smaller than others because, while lots of books are out for review, the return rate was a bit low. Come on folks, you agree to review these things, get cracking. Anyway, for this issue we have books on Java and Java (popular isn't it), Communications and the Unix Shell. Something for most technical people.

At present, the supply of books is a bit low, both because of the number outstanding, and because of some changes within the publishers. This will pickup soon, and when it does I'll post a note to the mailing list <auug-books@ansto.gov.au> and the newsgroup aus.org.auug. Unfortunately, this disadvantages members without network connections, or on the end of a low speed link. For people in such a position, either mail, via the AUUG PO Box, or fax me on (02) 9717 9273, with your contact details and preferences.

❖



EXPLORING JAVA

by Patrick Niemeyer & Joshua Peck
O'Reilly and Associates
1996, 407 pages,
ISBN 1-56592-184-4

Reviewed by
Jagoda Crawford <jc@ansto.gov.au>
Australian Nuclear Science and Technology Organisation

Exploring Java is an introduction to the Java programming language and is aimed for people intending to build applications using Java. The authors state that the book could also be considered a crash course in object-oriented programming, and it certainly was used as such in my case.

A background of Java and the virtual machine concept is introduced in chapter 1, including a comparison of Java with languages such as C++, Smalltalk and Tcl. With the introduction and background presented in the first chapter, the rest of the book deals with Java programming.

The second chapter goes directly to guiding the reader into the creation of the first applet, "Hello Web!". By using, and building on this simple example applet, the authors introduce and explain terms such as class, method, constructors, object, instance, package, etc.

Tools needed to compile and run Java applications are covered in chapter three, including information required for running Java applets in Web pages.

The next three chapters cover the Java language, Java's object-oriented features and threads. Attention is given to areas where Java differs from other languages, with a good coverage of classes, subclassing and inheritance, object creation and distraction, compilation units and exception handling.

The rest of the book is devoted to the Java Application Programming Interface (API), the collection of classes that comes with every Java implementation, for example the basic language classes in java.lang, input and output in java.io, the Abstract Windowing Toolkit, java.awt, etc. Since the publication of the book, Java 1.1 has been released with a number of changes in AWT, however, Java 1.0.2 is still being run in many sites. If you are heavily into the development of user interfaces with

AWT, the O'Reilly book, Java AWT Reference, by John Zukowski, covers both 1.02 and 1.1.

The book was my first introduction to the world of Java. I found it easy to read and the numerous examples and explanations got me well on the way to implementing my application. I am recommending this book to people intending to use Java in my organisation, and I'd encourage others to do the same.

❖

JAVA IN PLAIN ENGLISH

by Brian Overland
MIS Press
1997, 552 pages, $29.95
ISBN 1-55828-503-2

Reviewed by
David Williams <mgdmw@u2.newcastle.edu.au>
Quality Coal Consulting

This book could have been a very convenient reference for Java. Unfortunately it suffers from the recent API changes of the 1.1.1 standard, and so significant portions of the calls described in this book would be marked as "deprecated" features by the latest version of "javac". These include reasonably trivial things such as the use now of setVisible (boolean) rather than the calls show () and hide (), which are described in this book, but more serious matters such as the completely redesigned event handling model of Java 1.1.1. Really, this book is a reference for 1.0.2 and as such has just missed out in its timing.

Further, despite the front cover blurb stating "Concisely explains cutting-edge features of Java 1.1" the only real topic in addition to 1.0.2 is some coverage of JDBC. There certainly are no details of new 1.1.1 components such as pop-up menus or scrollpanes.

The book is divided into five essential parts, with an appendix and index.

The first part is a straightforward overview of Java detailing how it works and its distinctions. This is presented in a direct and concise manner. The second part is a comparison between Java and C++, occupying just over a hundred pages. This in particular makes the book a good quick reference for those moving from a C++ base.

The third part of the book is simply an index mapping topics to particular API references, leading on to the fourth part of the book which is a comprehensive API reference. This is the bulk of the book, at over 250 pages, and is most likely that which will be of more importance.

Java in Plain English is not at all intended as a tutorial or introductory sort of work. Rather it is a most useful and concise desktop reference for the Java developer.

I very much like the format and style of the book, but I cannot recommend it unhesitatingly due to its lack of 1.1.1 coverage. Certainly, the language itself remains unchanged, but the API is too significant to avoid consideration. For the developer working with 1.0.2 it would be very useful, but to such a developer I would recommend migrating to the new standard anyway. Still, should Overland manage to rework his book to be up to date, I would be more than willing to consider it as a necessary item.

❖

COMMUNICATION SYSTEMS & NETWORKS

by Ray Horak and Mark A Miller
M&T Books, New York
1997, 486 pages, $69.95
ISBN 1-55851-485-6

Reviewed by
Francis Liu <Francis.Liu@uts.edu.au>

In a conversational style the author explains the technologies that are found in modern telecommunications networks. In fact, this book is written in language so clear that it could be read aloud to a class of first year students in datacomms. It has an excellent chapter on Broadband Network Services. It covers ATM, B-ISDN, Frame Relay, and SMDS (a derivative of DQDB) in enough detail for the novice to grasp the essentials. Yet at the same time, it is a reference manual for the professional.

The book comprises sixteen chapters. Communication Systems & Networks starts with no assumptions about the reader and introduces the analog/digital concept. This of course, is for the absolute beginner, It progresses into the physical components, then the various application systems that sit on top. A chapter on voice, a chapter on fax, a chapter on the PSTN. Then we get to the

interesting bits. Data communications. Public Data Networks, Local Area Networks, Broadband Networks, Wireless Networks, the Internet, all get a pretty discussion. I had a problem with the consistency of the writing. Some sections were written in an argumentative style, that is, the advantages and disadvantages were listed and discussed, and other sections were just slabs of text.

The chapters on the physical elements of the communications infrastructure are patchy. There are two pages dedicated to discussing the pros and cons of coaxial cable, but six pages are used for satellite. I think that most readers would be using twisted pair or coax, not satellite, and an indepth on satellite transmission is really unnecessary.

Even though the book is clearly written, it falls into the almost unavoidable trap of using acronyms, excessively. As you progress through the book, the number of acronyms increases. Until we reach a page where you find "GTE and each of the RBOC's and IXC's have developed their own version of IN in support of 800 services, calling card verification and other services" (p325).

There is a good discussion on legislative issues relating to telecommunications. Interconnections, Universal Service, Number Portability are all dealt with briefly in the US context. Convergence of telephony, computing, and television has a chapter all to itself. It is more an eye opener than a real analysis, but it is very useful reminder to networking people that the real world is more than hardware and standards. Unfortunately, almost all of this is in an American environment.

I think the style of the book disagreed with me. That's why this review is hot and cold about it. The technical points, when I could ignore the conversational style, were precise, if American. The explanations were lucid, the text clear and easy to read. The book is good, except for the television style writing.

❖

UNIX SHELLS BY EXAMPLE

by Ellie Quigley
Prentice-Hall
1997, 644 Pages + CDROM
ISBN 0-13-460866-6

Reviewed by
Craig Macbride <craig@rmit.edu.au>

The most obvious thing to say about "UNIX Shells by Example" is that it's a big book, as well you might expect. For the most part, it covers its subjects, which are the C, Bourne and Korn shells and the utilities awk, sed and grep, quite thoroughly.

Ellie Quigley, in the preface, explains that this book was written to give students one reference which covers these particular areas. Unfortunately, this means that neither the shell area nor the utilities area is covered as completely as I would have liked. As reference or even tutorial material, I would have much preferred to see one volume on shells, including all the common shells, and one volume on text processing commands, including more than just awk, sed and grep.

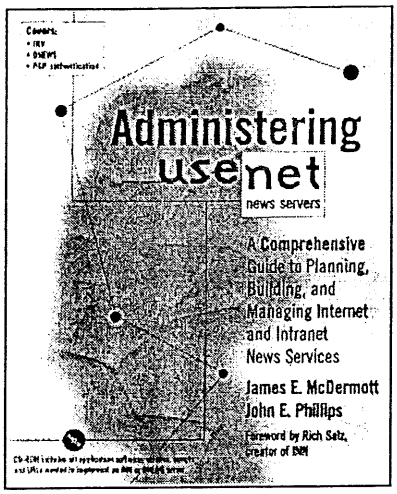
The book begins well, by covering the UNIX fundamentals that a shell programmer needs to know about, such as parent and child processes, standard I/O steams, working directories and the like. It give the reader a good description of what is going on in the UNIX environment.

Almost one third of the book is then spent on sed, awk and grep. The detail is good, including details of differences between awk and nawk. Many examples are provided, along with step by step explanations of why the example routines work as they do. The rest of the space is devoted to shells, with slightly more space given to the Korn shell than the other two. Again, there is a massive amount of detail and more excellent detailed explanations of why things work as they do.

Even without the comments in the preface, it is obvious that this is a set of course notes that have grown into a text book, with both the good and bad points that might be expected from that heritage. As a tutorial on UNIX command interfaces and how to write shell scripts, it works well. The very last appendix is one thing I definitely liked: a feature by feature comparison of the different elements of syntax in the Bourne, C and Korn shells. And, there are even lab exercises in an appendix at the back, boys and girls!

Unfortunately, there are some technical matters that are not as well covered. While it mentions that the arguments to ps are different in SysV and BSD-derived systems, and gives appropriate variants on examples that use ps, it really doesn't go any further than that in telling a reader what differences there might be. How to make scripts setuid is explained at one point, and yet no mention whatsoever is made of the security risks inherent in doing so.

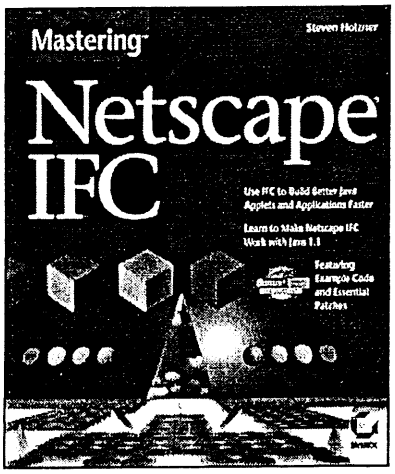
NEW! from Addison Wesley Longman



Administering Usenet News Servers

This book is a comprehensive guide to planning, building and managing Intranet and Internet news services. The accompanying CD-ROM includes all application software, utilities, scripts and URL's needed to implement an INN or DNEWS server.

ISBN: 0 201 41967 X RRP $45.95



Mastering Netscape IFC

The Internet Foundation Classes are pre-built class libraries that dramatically extend the efficiency, power and consistency of Java programming, and this book helps you to take full advantage of them to build better Java applets and applications. Comes with CD-ROM.

ISBN: 0 7821 2116 0 RRP $79.95

Also available: Quarter Century UNIX, a look at the first 50 years of UNIX.

ISBN: 0 201 54777 5 RRP: $28.95

AUUG Members receive a 20% Discount off the RRP

Please post or fax to Alison Barr, Trade Marketing Coordinator at Addison Wesley Longman
95 Coventry Street, South Melbourne Victoria 3205 Ph (03) 9697 0664 Fax: (03) 9699 2041.

ORDER FORM

Title _____ Price _____

Title _____ Price _____

Total less 20% _____

Postage and handling charges **1 book** $4.00 **2 or more books** $7.00 Plus Postage _____

Send your order with a cheque, or bill your credit card. ❑ Bankcard ❑ Mastercard ❑ Visa ❑ Amex

Card No ❑❑❑❑ ❑❑❑❑ ❑❑❑❑ ❑❑❑❑

Expiry Date _____ Amount _____

Name _____ Signature _____

Address _____

Suburb _____ State _____ Postcode _____

Prices are correct at time of printing, but are subject to change without notice

The book includes a CD-ROM, which contains a strange collection of material. Despite BASH not being mentioned at all in the book, there are copies of it for Atari, Linux, HP/UX and OS/2 on the CD-ROM. A range of other utilities are provided for different platforms, including any mixture of the above, with or without MSDOS and/or WindowsNT. The index file on the CD-ROM lists directories and their contents for gdb, html, perl and some others, despite them not actually being on the CD-ROM! If you look for a program to run under a particular operating system, it may be present, or it may not. If it is, it may be available for a wide range of operating systems, except the one you are using!

As a text for those learning about UNIX, command interpreters, how to do shell programming, etc, this book covers its ground well. Even in that case, however, I would have to question the wisdom of not including anything on perl. While it may not be possible to shoehorn detailed information of other shells, such as BASH, or other utilities, such as perl, into this book as well as its current topics, without turning it into an enormous tome, it is a shame that they get no mention at all. As a technical reference on the Bourne, C and Korn shells, it's not bad. They really should do something about the CD-ROM, because it's basically a mish-mash.

❖

UNIX Traps & Tricks

Sub-Editor:
Günther Feuereisen
<gunther@ibm.net>

We're back! Thank you to those of you who wrote dropped me a note!

As always, I'm looking for contributions from all you gurus out there .. so sit down, and email me an idea, a concept or a piece of code .. I'd love to hear from you!

❖

To start off this months issue, we have the following contribution from Peter Sundstrom <peters@daedalus.com.au> for improving web server performance:

The one single thing that speeds up web server performance is to turn off DNS logging. On a high volume site, this will reduce performance quite drastically. The DNS lookups are usually only used for logging purposes, so it makes sense to parse the web log file at the end of each period and resolve all the IP addresses.

The following perl script will work for any log which has an IP address as the first field (which happens to be most web server logs). I use it as a pre process step each week before I generate the web statistics, so that I have host and domain information.

Typical use is:

```
$ cat access | ip2name >access.converted
```

```
#!/usr/bin/perl
#==================================================
#
# NAME: ip2name
#
# AUTHOR: Peter Sundstrom (peters@daedalus.com.au)
#
# DATE: 8 April 1997
#
# PURPOSE: Convert a log file with IP entries to FQDN
# entries.
#
# Will work for any log file that has an IP address as
# the first field.
#
# UPDATED: 8 Apil 1997
#
#==================================================

while (<>) {
  @line=split;

  if ($line[0] !~ /\[a-zA-Z]/) {
    if ($fqdn{$line[0]}) {
      $name=$fqdn{$line[0]};
    }
    else {
      @ip=split(/\./,$line[0]);

      $name=gethostbyaddr(pack("C4",$ip[0],$ip[1],
            $ip[2],$ip[3]),2);
      $fqdn{$line[0]}=$name;
    }

    if ($name) {
      shift @line;
      print "$name ";
    }
  }


  print "@line\n";
}
```

❖

The next script, is straight out of my own stable of scripts. One of the problems a System Administrator has is identifying system utilisation trends. Sometimes you will be presented with that management catch phrase "The system is too slow!". Now it might be the system is slow, and needs more resources, but it could also be that everyone is firing off huge reports just before lunch.

The following perl script runs in the background, and logs the system load (obtained from uptime(1)) vs. time of day. This then allows you to produce pretty pictures, to see if your machine is under resourced, or is idle all day, except for 4.55pm ;-)

```
#!/usr/local/bin/perl
#
# loadd - a daemon to capture system load every minute
# to produce load trends reports.
#
# CONCEPT
#
# Use uptime(1) to get average load over a
# minute, every minute.

$logdir = "/usr/local/logs/load";

while ( 1 )
{
  $date = `date +%y%m%d`;
  chop $date;
  $logfile = "$logdir/$date";
  $time = `uptime | cut -d ' ' -f 1`;
  chop $time;
  $load = `uptime | cut -d , -f 5`;
  chop $load;
  open(LOGFILE,">> $logfile");
  print LOGFILE "$time\t$load\n";
  close(LOGFILE);
  sleep(60);
}
```

Of course there are other wys to write this. The footprint of the script is farily small, with minimal impact on the system. This could be written in some Shell, C or any other language that tickles your fancy.

There are lots of ways to measure performance - this is but one of a suite of tools I have written over the years to collect statistics for later analysis. Everyone has their own set of tools, and vendors from all companies generally provide some sort of performance tool, with the pretty pictures and history mechanisms. Often, the best tools, are the ones you write yourself.

If anyone out there has a favourite tool of their own, a publically available one they use, or have had experiences with packaged performance software, drop me a line, and I'll present the views in upcoming installments of "Traps & Tricks".

❖

Application for Institutional Membership

Section A: MEMBER DETAILS

The primary contact holds the full member voting rights and two designated representatives will be given membership rates to AUUG activities including chapter activities. In addition to the primary and two representatives, additional representatives can be included at a rate of $80 each. **Please attach a separate sheet with details of all representatives to be included with your membership.**

NAME OF ORGANISATION: _____

Primary Contact

Surname _____ First Name _____

Title: _____ Position _____

Address _____

Suburb _____ State _____ Postcode _____

Telephone: Business _____ Facsimile _____

Email _____ Local Chapter Preference _____

Section B: MEMBERSHIP INFORMATION

Renewal/New Institutional Membership of AUUG *(including Primary and Two Representatives)*	☐	$390.00
Surcharge for International Air Mail	☐	$120.00
Additional Representatives	Number ☐	@ $80.00

Rates valid as at 07/97

Section C: PAYMENT

Cheques to be made payable to **AUUG Inc** *(Payment in Australian Dollars only)*

For all overseas applications, a bank draft drawn on an Australian bank is required. Please do not send purchase orders.

-OR-

☐ Please debit my credit card for A$_____

☐ Bankcard ☐ Visa ☐ Mastercard

Name on Card _____

Card Number _____

Expiry Date _____

Signature _____

Please mail completed form with payment to:

Reply Paid 66
AUUG Membership Secretary
PO Box 366
KENSINGTON NSW 2033

Or Fax to:

AUUG Inc
(02) 9332-4066

Section D: MAILING LISTS

AUUG mailing lists are sometimes made available to vendors. Please indicate whether you wish your name to be included on these lists:

☐ Yes ☐ No

Section E: AGREEMENT

I/We agree that this membership will be subject to rules and by-laws of AUUG as in force from time to time, and that this membership will run from time of joining/renewal until the end of the calendar or financial year.

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.

Signed: _____

Title: _____

Date: _____

AUUG Secretariat Use

Chq: bank _____ *bsb* _____

A/C: _____ *#* _____

Date: _____ *$* _____

Initial: _____ *Date Processed:* _____

Membership#: _____

AUUG Inc

UNIX® AND OPEN SYSTEMS USERS

Membership Application

AUUG Inc Secretariat
PO Box 366, Kensington NSW 2033, Australia

Tel: (02) 9361 5994
Free Call: 1 800 625 655
Fax: (02) 9332 4066

email: auug@auug.org.au

ACN A00 166 36N (incorporated in Victoria)

http://www.auug.org.au

AUUG Inc is the Australian UNIX and Open Systems User Group, providing users with relevant and practical information, services and education through co-operation among users.

AUUG OFFERS SOMETHING FOR YOU!!

AUUGN

Technical Newsletter

AUUG's quarterly publication, keeping you up to date with the world of UNIX and open systems.

Education

Tutorials
Workshops

Events.....Events.....Events

- Annual Conference & Exhibition
- Overseas Speakers
- Local Conferences
- Roadshows
- Monthly Meetings

DISCOUNTS

to all AUUG events and education.

Reciprocal arrangements with overseas affiliates.

Discounts with various internet service providers, software, publications and more...!!

Connections

- Newsgroup aus.org.auug

Application for Individual or Student Membership

Section A: PERSONAL DETAILS

Surname _____ First Name _____

Title: _____ Position _____

Organisation _____

Address _____

Suburb _____ State _____ Postcode _____

Telephone: Business _____ Private _____

Facsimile: _____ E-mail _____

Section B: MEMBERSHIP INFORMATION

Please indicate whether you require Student or Individual Membership by ticking the appropriate box.

RENEWAL/NEW INDIVIDUAL MEMBERSHIP

Renewal/New Membership of AUUG ☐ $100.00

RENEWAL/NEW STUDENT MEMBERSHIP

Renewal/New Membership of AUUG *(Please complete Section C)* ☐ $25.00

SURCHARGE FOR INTERNATIONAL AIR MAIL ☐ $60.00

Rates valid as at 07/97

Section C: STUDENT MEMBER CERTIFICATION

For those applying for Student Membership, this section is required to be completed by a member of the academic staff.

I hereby certify that the applicant on this form is a full time student and that the following details are correct.

NAME OF STUDENT: _____

INSTITUTION: _____

STUDENT NUMBER: _____

SIGNED: _____

NAME: _____

TITLE: _____

DATE: _____

Section D: LOCAL CHAPTER PREFERENCE

By default your closest local chapter will receive a percentage of your membership fee in support of local activities. Should you choose to elect another chapter to be the recipient please specify here:

_____

Section E: MAILING LISTS

AUUG mailing lists are sometimes made available to vendors. Please indicate whether you wish your name to be included on these lists:

☐ Yes ☐ No

Section F: PAYMENT

Cheques to be made payable to **AUUG Inc**
(Payment in Australian Dollars only)

For all overseas applications, a bank draft drawn on an Australian bank is required. Please do not send purchase orders.

-OR-

☐ Please debit my credit card for A$_____

☐ Bankcard ☐ Visa ☐ Mastercard

Name on Card _____

Card Number _____

Expiry Date _____

Signature _____

Please mail completed form with payment to:

Reply Paid 66
AUUG Membership Secretary
PO Box 366
KENSINGTON NSW 2033
AUSTRALIA

Or Fax to:

AUUG Inc
(02) 9332-4066

Section G: AGREEMENT

I agree that this membership will be subject to rules and by-laws of AUUG as in force from time to time, and that this membership will run from time of joining/renewal until the end of the calendar or financial year.

Signed: _____

Date: _____

AUUG Secretariat Use

Chq: bank _____ *bsb* _____

A/C: _____ *#* _____

Date: _____ *$* _____

Initial: _____ *Date Processed:* _____

Membership#: _____