

# AUUGN

The Journal of AUUG Inc.

Volume 17, Number 1  
February 1996

More about mmap()!

Computer room archaeology

AUUGN Vol. 16 Index

... plus reviews, Chapter news, and more

AUUG  
Inc.

UNIX & OPEN SYSTEMS USERS



# AUUGN

Volume 17, Number 1  
February, 1996

## AUUG Membership & General Correspondence

The AUUG Secretary  
PO Box 366  
Kensington NSW 2033

Tel: (02) 361 5994  
Fax: (02) 332 4066  
Freephone: 1-800-625-655  
E-mail: [auug@auug.org.au](mailto:auug@auug.org.au)

## AUUG Executive

President: Michael Paddon  
[mwp@acca.net.au](mailto:mwp@acca.net.au)  
Kodak  
173 Elizabeth St.  
Coburg VIC 3058

Vice President: Glenn Huxtable  
[glenn@fs.com.au](mailto:glenn@fs.com.au)  
Functional Software  
PO Box 192  
Leederville WA 6903

Secretary: Peter Wishart  
[pjw@auug.org.au](mailto:pjw@auug.org.au)  
EASAMS/GEC Marconi Systems  
PO Box 4806  
Unit 7, 10 Kennedy St.  
Kingston ACT 2604

Treasurer: Stephen Boucher  
[stephen@mtiame.mtia.oz.au](mailto:stephen@mtiame.mtia.oz.au)  
MTIA  
509 St. Kilda Road  
Melbourne VIC 3004

## Committee Members:

Phil McCrea (Past President):  
[pmc@syd.dit.csiro.au](mailto:pmc@syd.dit.csiro.au)  
Division of Information Technology  
CSIRO  
Building E6B  
Macquarie University NSW 2113

Frank Crawford  
[frank@atom.ansto.gov.au](mailto:frank@atom.ansto.gov.au)  
ANSTO  
Private Mail Bag 1  
Menai NSW 2234

Lucy Chubb  
[lucyc@sw.oz.au](mailto:lucyc@sw.oz.au)  
Softway Pty. Ltd.  
PO Box 305  
Strawberry Hills NSW 2021

Chris Maltby  
[chris@sw.oz.au](mailto:chris@sw.oz.au)  
Softway Pty. Ltd.  
PO Box 305  
Strawberry Hills NSW 2021

David Purdue  
[David.Purdue@aus.sun.com](mailto:David.Purdue@aus.sun.com)  
SunSoft  
119 Willoughby Rd.  
Crows Nest NSW 2065

## AUUGN Business Manager

Elizabeth Egan  
[auug@auug.org.au](mailto:auug@auug.org.au)  
PO Box 366  
Kensington NSW 2033

# Table of Contents

## Editorial 3

### President's Report 3

*Michael Paddon*

### Software review:

#### Website 1.0/1.1 for Windows NT 4

*Arthur Marsh*

### Preface to

#### "Why aren't you using mmap() yet?" 6

*Kevin Sheehan*

### Paper: Why aren't you using mmap() yet? 7

*Kevin Sheehan*

### Advice: A Moving Story 19

*Andrew van der Stock*

### Background: Linux - The Choice of a New Generation 20

*Frank Crawford*

### Opinion: Will the real Information SuperHighway please stand up! 21

*Phil McCrea*

### Background: What is a Network? 23

*Frank Crawford*

### Advice: Computer Room Archaeology 24

*Frank Crawford*

AUUG Inc. acknowledges the  
generous support of its  
corporate sponsors...



**TELLURIAN**  
PTY. LTD.

## UNIX Tricks & Traps 25

Janet Jackson (sub editor)

## Book Reviews 27

Frank Crawford (sub editor)

## WAUG news: From the Western Front 34

## AUUG Canberra news: Canberra chapter 35

## Rules of AUUG Incorporated 36

## AUUGN Volume 16 Index:

### Book Reviews (Sorted by author) 42

## AUUGN Volume 16 Index:

### Reports (Sorted by author) 44

## AUUGN Volume 16 Index:

### Papers (sorted by title) 45

## AUUG Institutional Members 47

## AUUG Membership applications & change notification

### AUUGN Correspondence

Please send all correspondence regarding AUUGN to:

AUUGN Editor  
PO Box 366  
Kensington NSW 2033

E-mail: [auugn@auug.org.au](mailto:auugn@auug.org.au)

### Submission Guidelines

Submission guidelines for AUUGN contributions are regularly posted on the [aus.org.auug](http://www.aug.org.au) news group. They are also available from the AUUG World Wide Web site at <http://www.aug.org.au>

Alternately, e-mail 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 details, 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, at \$50 each for members, and \$60 for non-members. Contact the AUUG Secretariat for details.

### Mailing Lists

Inquiries regarding the purchase of the AUUGN mailing list should be directed to the AUUG Secretariat. Telephone (02) 361 5994 during business hours, or Fax (02) 332 4066.

### Disclaimer

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

### Copyright Information

Copyright ©1995 AUUG Incorporated. All rights reserved. AUUGN is the journal of AUUG Incorporated, an organisation with the aim of promoting knowledge and understanding of Open Systems, including, but not restricted to, the UNIX<sup>®</sup> system, networking, graphics, user interfaces and programming and development environments, and related standards.

Copying without fee is permitted, provided that copies are made without modification, and are not made or distributed for commercial advantage. Credit to AUUGN and the author(s) must be given. Abstracting with credit is permitted. No other reproduction is permitted without prior permission of AUUG Incorporated.

### Trade marks

UNIX is a registered trade mark of X/Open in the United States and other countries.

---

# Contribution deadlines for AUUGN in 1995/96

Vol 17, #2 (April '96): April 15th

Vol 17, #3 (June '96): May 24

Vol 17, #4 (August '96): July 19

Vol 17, #5 (October '96): September 20

Vol 17, #6 (December '96): November 22

## Editorial

Phil Anderson <phil@osa.com.au>

Welcome to the first issue of AUUGN for 1996. Well, the first *official* issue; as you all know, the last issue for '95 only sprang forth in late January, and even then many of you received the special "bound on the right issue" ... maybe our bindery was subconsciously presaging our change in federal government! ; ^)

Seriously though, I'd like to apologise for the binding snafu; our printers do a sterling job of not only printing, but also coordinating the binding and mail-out of each issue. Most of the time, everything goes extremely well, but once in a while, gremlins sneak into the operation and voila!, a whole new way of reading AUUGN.

The delay in getting vol.16#6 out the door has pushed this issue out by about a month, worst luck, but I'm already assembling the April issue, so we should be back on track pretty soon. Thanks for your patience, and keep submitting material ... or in the case of *most* of you, *start* submitting material!

Things of note in this issue include the last word on mmap(), courtesy of Kevin Sheehan (whose .sig describes him as a "Consulting Poster Child" ...) and for all you lawyer wannabes, the most recent incarnation of the AUUG constitution.

•••

Liz has promised me some photos from the Summer conferences for the next issue; ooooh! Some fun! AUUG will gladly accept donations *not* to print those incriminating social pix (us? tabloid? nev-eerr!).

Y'all be good now!❖

## President's Report

Michael Paddon <mpaddon@acca.net.au>

As I write this column, the majority of the AUUG Summer Conference Series is over. I have been receiving excellent reports, and occasionally incriminating photos, from all the completed events. This year's series appears to have been very successful, with high quality programmes and fault free organisation all round. Several of the conferences have been multi day affairs, with some including strong tutorial streams as well as papers.

Our Business Manager, Liz Egan, has been taking this opportunity to visit all the chapters, so hopefully many of you have had a chance to meet her by now. One of Liz's key roles is to assist the chapters develop their activities programmes (whatever they may be), and to assist the chapters coordinate both their events and with each other. One of the fascinating things Liz has noticed in her travels is just how diverse an organisation AUUG is. No two chapters do things alike, and they all have a distinct feel.

This poses some unique challenges for a national user group, but it also is the source of great opportunity. I don't think that we have really taken advantage of all the resources hidden away in our chapters, particularly in terms of the talented and insightful speakers that would find a ready audience in other parts of the country.

The AUUG committee has decided that another roadshow (similar to the ones arranged with Kirk McCusick and Brent Chapman) will be run midway through this year. As well as international exports, we'd also like to include some local content (bring out some of our own hidden talent, so to speak). We have had some very animated discussion about what an appropriate topic would be, with Java being a hot favourite subject to concerns about bandwagons. I'd like to take this opportunity to ask: what, or who, would you like to see?

All this talk about conferences lead us naturally to the next big one. A lot more of the early organisational work has been done towards the Winter Conference, which will be held at the World Congress Centre in Melbourne between the 18th and 20th of September.

Those of you that have been web surfing will have already noticed that the AUUG 96 page bears both the AUUG and the Charles Sturt University logos. After much deliberation, your committee has decided to combine AUUG 96 with the Asia Pacific World Wide Web Conference for the second year running.

This was not a straightforward decision because our first experiment along these lines suffered some

serious teething troubles. Under our post mortem scrutiny, however, we decided that the majority of these occurred because the two conferences were melded too late in the piece. Four streams was too much, and there were major hassles with too much being on at once and a perception that quality was being swamped by quantity.

On a personal note, I'll add here that I thought there were some brilliant papers at last year's conference. And there were, as always, some that just did not work out on the day. Nevertheless, it is no small thing to produce a paper, have it accepted by our programme committee, and then stand up and present at Australia's largest open systems conference. I always have the highest respect for all of our presenters, and I urge you to consider contributing a paper of your own, especially if you have felt dissatisfied with the content in the past.

Getting back to the joint conference, it was felt that there really is an enormous synergy between AUUG 96 and the APWWW. Now, more than ever, the worlds of Unix, programming, system administration and the Internet are overlapping as the Web becomes the delivery "tool de jour" for just about any new computing or information service. Languages like Java, in particular, are making the skills and lessons found in the open systems world freshly relevant.

Given this commonality of interest, it seemed foolish not to let each conference build on the strengths of the other. Certainly, we expect to attract a number of high quality papers from academia which would not usually be submitted to an AUUG conference. We have, however, taken certain steps to ensure that the "AUUG 96 & Asia Pacific World Wide Web 2nd Joint Conference and Exhibition" (the official name) is even better than last year.

Firstly, we are currently planning for three streams, covering the (rather broad) areas of Management, Technical and Web. The reduction in concurrent programming is expected to yield a better focused, more enjoyable event while still leaving enough room for the diversity of interest that characterises AUUG. Of course, there will still be a strong tutorial programme run on the days before the conference proper.

Secondly, there is one conference committee and one programme committee right from the outset. AUUG's key representatives on these bodies are Erno Davids and Adrian Booth (as I mentioned in my last column), although many other AUUG members will be serving as well. In fact, this is a great time to call for volunteers who can donate a chunk of their time towards these and other conference related duties. Please feel free to email me if you are interested...

Check out the web page (it's linked off <http://www.auug.org.au/>) and let us know what you think. You'll also find the call for papers there (hint, hint).❖

---

Software review:

## Website 1.0/1.1 for Windows NT

*Reviewed by Arthur Marsh <arthur@gateway.dircsa.org.au>*

*O'Reilly and Associates Inc  
1995, Software  
ISBN 1-56592-143-7*

In case anyone is wondering, yes I still run UNIX (-:). For the purposes of this review I'm assuming that AUUG members considering Website also have UNIX in-house.

When this software was advertised for review by an AUUG member, I was interested in seeing if Website could integrate well with an existing UNIX and Netware environment. The answer turned out to be yes, but to begin at the beginning:

### What you get

I'd heard of Website through Boardwatch and Byte magazines as a World Wide Web server for Windows NT and Windows 95, and wondered what it could offer someone who was running a freely available HTTPD on UNIX. It turns out that the Website package includes with the server the server admin program, plus Webview, a utility for checking the links in HTML documents, Webindex, a program for building search capabilities into Website, an Image-map editor, the WWW viewer Enhanced Mosaic, and new to version 1.1 the Hot Dog HTML editor. Physically, version 1.0 came on two 3.5" floppy disks and was accompanied by a 320-odd page book of the quality we've come to expect from O'Reilly and a Website T-shirt with the Website map logo instead of an animal. Note that at present Website cannot act as an HTTP proxy server.

### Installation

Windows NT Work-station 3.50 was installed on a 486DX-33 pc with a 200 Megabyte IDE hard disk and 16 Megabytes of RAM and an NE2000 clone network card without trouble once the video card was specified as plain VGA. The Netware client capability had to be explicitly installed, which was unexpected considering that the IPX/SPX transport had already been specified. A loose 10 Base-T connection was a surprisingly subtle problem to diagnose, but was easily fixed (-:).

Following the instructions in the Website documentation for setting up multi-homing, the NT machine was configured for 2 IP addresses, 202.14.187.111 and 202.14.187.121. Using Samba on an existing UNIX machine (202.14.187.122) and the Windows NT file manager, the UNIX machine's /opt/lib/httpd directory was mapped to a drive letter on the NT machine. Drives were also mapped from the Netware server.

Website itself was then installed, with the document root pointing to the existing document root on the UNIX machine. The documentation provides a good checklist of tests to run, which worked, with the installation of Website taking under an hour. Website can be started by a double click on the server icon, or can be installed as a service under NT so that an account other than administrator can use the NT machine whilst Website is running. One of the first places one should connect to is Website Central, <http://website.ora.com>, which has up-to-date information on Website and third party utilities.

The server admin program enables one to map between the directory format in HTML files and the drive letters on the NT machine. At the simplest level the location of the document root is specified. I added mappings for the Netware server directories, and could also individually map directories that had the format ~username to /home/username/public\_html on a UNIX machine.

## Multi-homing

With version 1.1 of Website (which was made available to me from a passworded section on Website Central), a multi-homing Wizard is available to assist in creation of multiple identities for the WWW server. As version 1.0 was running when multi-homing was set up, I had to follow instructions on Website Central, which involved using the NT Registry Editor. One problem I encountered was in setting up a sub-directory off the document root as the document root for a second identity - some tricky mapping was required and Webindex refused to run. If you are considering multi-homing on Website, the recommended method is to have separate hierarchies under the document root and use the multi-homing wizard.

## Logging

Website provides access logs similar to CERN httpd on UNIX, but requires cycling of the logs - running a program that closes the existing log file, renaming it and opening a new log file - in order to view recent log

activity. DNS resolution of the IP addresses of connecting systems is a configurable option.

## CGI-bin

I did not migrate the CGI-bin programs from UNIX over to Website, but if one wants to run CGI-bin scripts under Website, some sample programs are available for running under Website's CGI-Win, CGI-shl, and CGI-DOS interfaces. Visual Basic is recommended if you wish to use CGI-Win, and one needs to obtain the NT POSIX shell from the Windows NT Resource Kit or an NT version of Perl to use the CGI-shl.

## Webview

Version 1.0 of Website included Webview, a program that can show you the document tree under any reachable URL, and perhaps more importantly, indicate which HTML documents are unreachable from the machine running Website and Webview. Version 1.1 adds the very welcome feature of allowing the use of an HTTP proxy, so that offsite references don't have to be reached every time Webview is run.

## Conclusion

Website 1.1 is an easy to set up and stable WWW server package, which can easily added to an existing LAN, and provides a useful collection of tools that are worthwhile not just for the Website server, but also for the existing WWW servers at a UNIX-equipped site. If you need to move data from sources accessible to Visual Basic in response to on-line queries, Website is a logical solution. Website can also be used to hand over the local WWW page creation and serving tasks to a relatively low cost machine on the LAN, whilst not removing the usefulness of having UNIX on-site for a proxy httpd and general internet connectivity.

Check <http://website.ora.com> (running Website, of course) for a detailed list of Website's features.❖

# Preface to "Why aren't you using mmap() yet?"

Kevin Sheehan  
Uniq Professional Services  
PO Box 70 Paddington, NSW, 2021 Aust.  
<kevin@uniq.com.au>

Some time ago a quick article originally written as a managerial overview of VM facilities was published here in AUUGN. It wasn't really intended for the audience of AUUGN, and a letter to the editor from Andrew Tune made a number of criticisms, some worth noting.

First, that it was non-technical I concede - I had no idea it was to have been published in AUUGN, as it was originally written for Open Systems Review. That it was inaccurate I'll dispute, but in credit to the critic, some of the points were necessarily hazy in an overview and may have been misunderstood.

Mr. Tune's assertion that twice the memory will not be used in a read() is incorrect. In SVR4, the buffer cache is only used for file system metastate. Under VM, all other pages (such as file pages) compete equally for physical memory. A read() of 16 MB will cause all 16MB to be "faulted into" memory at some point. While these pages will be read-only, the system copies the data to the application's address space, which \*does\* dirty those pages - causing them to be candidates for a page out to the swap device at a later time.

My greatest concern with the feedback is that it might lead the less experienced to believe they have been sold a load of goods :-). The article (and certainly this paper) made it clear that the advantages being discussed were those of regular file access, which is by far the most common in application programming. I agree that filters are a good model for many types of application - but their use for large files (e.g. imaging applications) is prohibitive in terms of resources required. As an another example, if 100 processes all read() in /etc/passwd, there are 100 dirty copies in private memory - if they all used mmap() instead, there would be one read-only copy in memory.

As to the issue of standards and implementation, I'd say that if you are sticking strictly to POSIX standards, then yes, you do not have mmap() as a tool. POSIX is a fairly comprehensive standard, but does not include many facilities of great use. If you pick SVR4 as your programming model, then you are guaranteed to have

it available. All of the major workstation vendors implement mmap() and, contrary to Mr Tune's assertion, sequential access is not worsened by use of mmap(), it is indeed enhanced by the ability to notify the system explicitly of the pattern of access you intend. In general, the savings in used memory are impressive.

fread() and fwrite() have their place too, but to assert that mmap() is not useful because it is not part of POSIX is to miss the point of the paper. mmap() \*is\* a very useful system call that is vastly underutilized - in the author's opinion because it is not well understood what the effects of using it are. The following is a paper presented originally at SUG '92. It has been presented a number of times since then in various forms in an attempt to raise the level of awareness.

If all you have is a hammer, then everything looks like a nail. The intention of this paper is to present application programmers with another tool in the box, not to advocate the loss of the filter paradigm or to ignore issues such as portability. In that spirit, I present "Why aren't you using mmap() yet?", and apologize not a whit for the evangelical nature of the paper :-)

# AUUGN

## IS YOUR JOURNAL!

Without you, there is no AUUGN: if you've knowledge to share, share it through AUUG's bimonthly journal.

You'll be reaching over 700 individuals, and more than 300 organisations involved in the UNIX/Open Systems world.

### WE'RE LOOKING FOR:

Talk to your local Chapter contact for ideas, and see elsewhere in this issue for submission guidelines.

- Papers
- Reviews
- Articles
- News
- Comment



Paper:

# Why aren't you using mmap() yet?

Kevin Sheehan  
Uniq Professional Services  
PO Box 70 Paddington, NSW, 2021 Aust.  
<kevin@uniq.com.au>

## Abstract

Since the inclusion of the Virtual Memory (VM) facilities in SunOS 4.0, the `mmap()` system call has provided a powerful and very efficient alternative to `read()` and `write()` to access regular files. Unfortunately, due to somewhat abstract and obscure documentation, developers have not been making great use of the interface.

With the inclusion of the VM facilities in SVR4, it is time for developers to start using `mmap()` more often as the preferred interface. It is no longer a Sun-specific interface, and the benefits demonstrated on the Sun platform will almost certainly apply to applications used on other platforms.

`mmap()` provides many benefits. Ease of use is certainly one, but the major benefit of using `mmap()` is increased performance. This is partly due to the more efficient use of system resources, partly due to the fact that `mmap()` allows the application developer more control over use of resources, and also that the system can be given more information in how to deal with those resources.

This paper is also intended to become an application note supplementing the obscure documentation that exists. In that spirit, this paper and the examples will be made freely available. It is the author's hope that others will supply examples and comments to ease the transition for those not yet fortunate enough to be using `mmap()`.

## A Brief Look at VM

### Paging

In order to facilitate the concept of virtual memory and demand paging, the address space of each process is divided up into a number of fixed size pages. Each process is assigned a particular memory management unit (MMU) context in which to run. The job of the MMU is to translate the references of virtual addresses to physical addresses of one kind or another.

In order to have a process address space larger than physical memory, the concept of virtual memory was implemented. In this scheme, not all of the pages a process is using need be mapped at one time. In order to get access to a page that is not mapped, the process references the unmapped page, and takes a "page fault". This is a trap that tells the kernel that an unmapped page has been referenced. The kernel responds by checking to see that it is a valid reference, and arranging for the proper contents to be mapped. The process is then transparently restarted, seeing nothing from the fault to the restart, hence the term virtual memory.

In short, the process address space is treated as an array of pages, each of which is associated with some physical page. In the next two sections, we will discuss how this was done in pre-VM systems, and how it is now done under VM.

A topic that is well discussed in *4.3 BSD Unix [1]* is fetch, placement, and replacement policies. Fetch policy is what determines when a page is loaded, placement policy determines where you place a page, and replacement policy determines when you select a page to be replaced by another (when do I throw it out?).

Pure demand paged systems typically fetch on demand only (when a page is referenced), and replacement is achieved by a Least Recently Used caching scheme that uses the working set as an approximation to the optimal set of pages for each process. These policies arise because the system generally has no a priori information about page usage. We'll discuss the facilities that VM provides to alleviate this shortcoming, but the author highly recommends the above book for further reading by those seriously interested in performance issues.

### Classic Unix Memory Usage

In pre-VM UNIX, physical memory was divided into two pools, one for the text and data pages, and another (called the buffer pool or buffer cache) for file I/O. The reasoning behind the buffer cache was that disks are much slower than memory, and keeping recent file blocks in core meant much less disk I/O. This meant that processes doing `read()` and `write()` calls to regular files had to deal with a (necessary) memory copy, but for blocks used by more than one process, or used more than once by a process, no I/O was done. The only type of filesystem that UNIX had to deal with was UFS, and the buffer cache was used for "block" I/O devices, which typically meant disks.

## Paper: Why aren't you using mmap() yet?

For a demand paged process (the only kind we consider here) running the process caused the systems to 1) allocate space on the swap/paging device (in a virtual memory system, you have to put the page someplace when it is not in memory) 2) copy the appropriate pages to the paging device (in the case of "pure" text programs, the text pages may have already been on the device and another copy is not required) and 3) loading some subset of those pages and starting execution of the process. As time goes on, the process will fault in pages it needs, and pages that are not referenced will age and be replaced (written out to swap if dirty and made available) if the system needs physical memory.

Great, we have virtual memory, a demand paging system that does a pretty good job of figuring out which pages (on average) are needed, we're caching disk blocks to minimize I/O done to the disks. Can we do better?

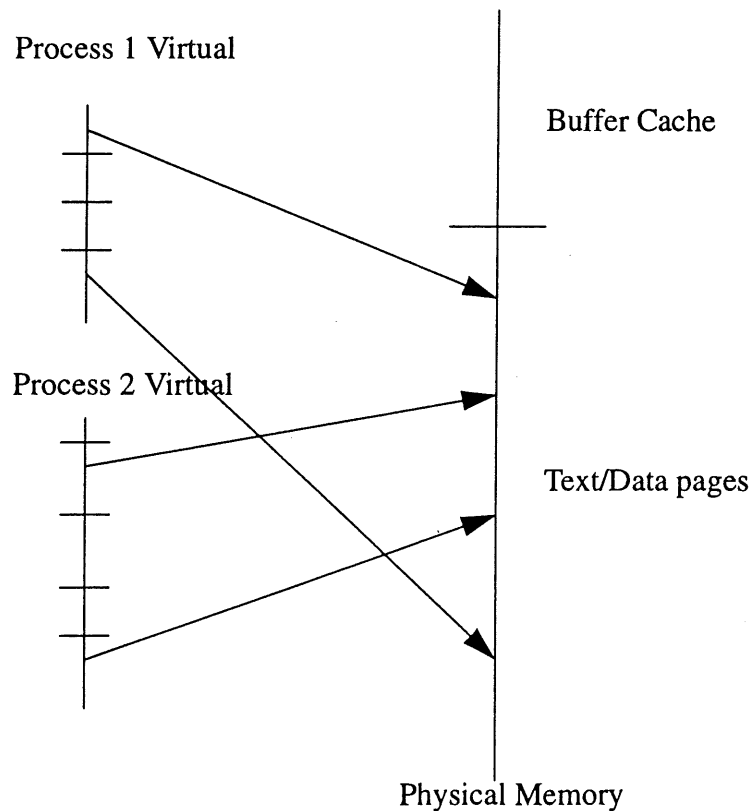
As UNIX grew from a time sharing system to meet the needs of a diverse number of platforms, the pre-VM facilities for mapping and management of the physical memory pool no longer proved sufficient.

Perhaps the biggest problem from a performance point of view was the static boundary between the buffer cache and the text/data pages. For a system doing large amounts of file I/O, the buffer cache is too small, and the text/data pages go to waste. For a system that needs many pages for text/data, the memory allocated to the buffer cache is not available to ease the load. More importantly, memory was being managed by two separate and non-cooperating sub-systems.

The requirements of mapping devices such as frame buffers were not well provided for - in order to map that memory, an allocation of virtual address space (backed by the swap device) was required, followed by the replacement of that mapping by the physical mapping to the device.

No method of interposing some action on the part of various subsystems in response to a page fault was provided, which meant that little or no management of the objects being mapped was really possible.

But perhaps the most telling argument for the VM rewrite was that the management of the mappings was heavily machine dependent. When the only model you have to worry about is the VAX or Sun2/3, that is



Physical Memory Usage in Pre-VM UNIX

one problem. When you have to start worrying about multiple architectures, caches, I/O caches and the like as well as the problem of many consumers of both virtual and physical memory, unifying and abstracting your management of virtual and physical memory resources becomes imperative.

### VM Model

Essentially, VM is an abstraction of the three levels needed to manage the mapping from a process' virtual address space to a physical page. The three layers are the Address Space (AS) Layer, the Segment (SEG) Layer, and the Hardware Address Translation (HAT) Layer.

The Address Space layer is responsible for managing the allocation of the process virtual space, and keeping track of the associations between the virtual addresses and the objects underlying them. Each mapping contains a virtual address, a length, the segment involved in the mapping, and an offset within that segment.

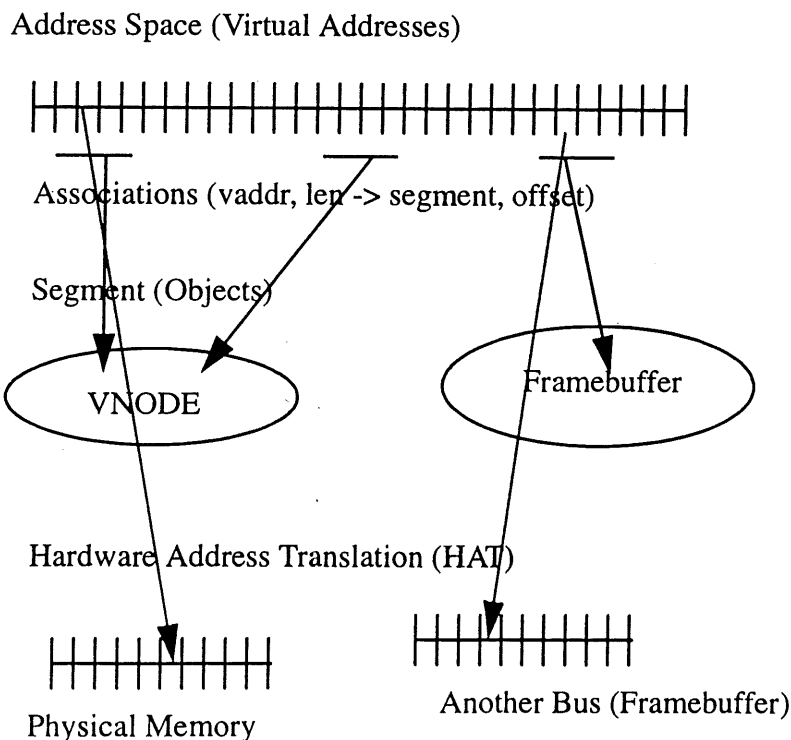
The segment layer is where the work of managing the objects is done. Examples of segment types are vnodes (the abstraction of files used in the Virtual File System), generic device drivers (such as frame buffers), and devices which require management of the context associated with each mapping, such as graphics accelerators.

The Hardware Address Translation layer is where the machine dependent work of loading and unloading mappings is done. Allocation of MMU resources, cache consistency and the like happens here.

What does this buy us? Instead of having a page fault routine that has to know everything about the possible ways to fill in a page, the problem is now factored into nice neat chunks. The address space operations allocate and free associations with various types of segment, and call the proper segment ops when a page fault is encountered. The segment ops make sure the proper page is prepared and calls the HAT layer to load (or unload) a translation when appropriate. The HAT layer handles all the grotty details of getting the page mapped and consistent.

So instead of having to rewrite the page fault handler to extend mapping facilities, all you have to do is provide a new segment driver. Instead of encapsulating knowledge of the mapping hardware throughout the OS, the HAT layer handles it. With the abstraction of the Virtual File System, only one driver was needed for "regular" files, the segment driver for vnodes

Simply put, `mmap()` is the interface that creates the associations between the process address space and various objects in the system. With the factoring of functionality provided by VFS and VM, the range of objects available for mapping is considerably extended. In fact, `mmap()` is now used extensively by



## Paper: Why aren't you using mmap() yet?

the system for management of resources in the lifetime of a process.

In addition to the familiar read, write and execute permissions that `mmap()` allows you to associate with a page, there is the concept of shared and private mappings. In a shared mapping, *all* mappings of a particular object are shared. In other words, if two different processes both map the same page of an object, they will have the same physical page mapped in for both address spaces at some point. In a private mapping, if the page is marked writable, the shared copy is mapped in read-only. On the first write reference to the page, an anonymous copy is made for that address space to use privately. This is also known as a Copy On Write (COW) mapping.

Since you are now able to explicitly manage your address space with `mmap()`, it is now possible to give the system more information about your usage of a particular mapping with the `madvise()` system call. In a classic demand paged system, the kernel knows nothing about your pattern of access, and has no a priori information about which pages you will or won't need at any given time. `Madvise` provides for three patterns of access to modify the fetch and replacement policies, explicit notification of whether a range of pages is needed or not, the ability to flush blocks back to secondary storage synchronously or asynchronously, and the facility to invalidate a mapping (cause the pages to be re-read on next fault).

The three access patterns are normal, sequential and random. The normal access pattern is to do read ahead, i.e. to read page N+1 when a page fault for page N is received. This presumes that page N+1 will be needed soon, and that page N-1 (if it is valid) may be needed and the resources should not be freed. Sequential access does read ahead as well, but to also do free behind. This assumes that page N+1 will be needed, but that page N-1 will not, and that the resources from page N-1 will be needed soon. Random access presumes nothing, i.e. it only fetches the page required by the fault, and allows the LRU algorithm to age pages as appropriate.

For cases where more explicit control is desired, `madvise` provides `WILLNEED` and `WONTNEED` advice. You can tell the system that a certain range of pages will be needed, and it will attempt to make sure they are available. When you no longer need them, and want the system to start making them re-usable, you simply tell it.

`Msync()` also allows you to explicitly flush pages either asynchronously or synchronously if you need to

guarantee consistency without affecting the mapping. It also allows you to invalidate the mapping without undoing it if you wish to reload the pages on the next reference.

Combined, these facilities offer the application developer a set of powerful tools to control the use of system resources. Since everything is handled by a page fault (or the simulation of a page fault) the various segment operations that need physical memory compete on an equal basis, and system resources can in theory be optimally allocated.

## Examples and Benefits

Let's look at some examples of how the system and various applications use `mmap()`, and the benefits over the older techniques used.

### Executables and Shared Libraries/Objects

As we saw in pre-VM UNIX, executables were loaded onto the paging device and paged in from there. There existed the possibility of sharing pure text, but all else was copied over. In particular, all executables had to be statically linked so they could be copied over, as no provision for mapping for dynamic linking existed.

In post VM UNIX, the vnode associated with the executable is simply `mmap()`ed Copy on Write, and the uninitialized data is mapped COW to `/dev/zero`. Let's see what this means for the lifetime of an executable called `foo`.

When you `exec()` `foo`, the system does a lot of what it used to - find `foo`. The first difference is that it will be dealing with a vnode, and not a collection of blocks in a particular flavor of filesystem. A small run time loader does two `mmap()` calls. The first creates a private mapping for the text and initialized data in the executable. The second creates a private mapping for the bss (uninitialized data) area to `/dev/zero` (`/dev/zero` is a device that returns zeros when read or mapped, as opposed to `/dev/null`, which returns nothing).

At this point no I/O (other than the page needed to get the exec header) has been done, so we have our first advantage - we don't need to copy the text or data to the paging area on start-up. All we have is an association and enough of the process actually in core to start it running.

Let's assume that it doesn't write to the text pages, but just faults them in as a result of fetches. In other words, when we touch a new page in text, we take a page fault, which calls the segment ops for vnodes, which allocates a page, schedules I/O to fill the page from

the vnode, calls the HAT layer to load a translation, and then restarts our process.

The advantage here is that the text need never appear on the paging device, the pages can always be fetched from the vnode.

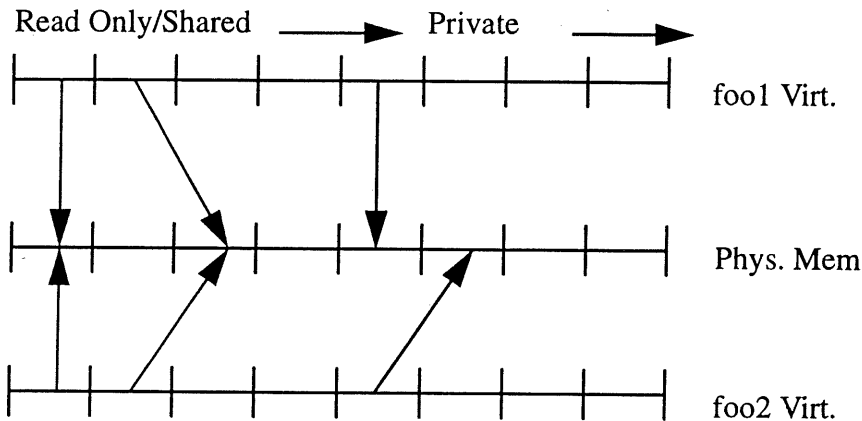
What happens when we reference the data or bss pages? If it is a read reference, the same thing happens as for text. In other words, if we have a lot of read-only data, we can read that out of the vnode as well, and don't need to have it on the paging device either. If it is a write reference, then a private copy is made. We take a page fault (either because it is unmapped, or because it is mapped read-only at this point) and the system notices that it is a write reference. The segment ops allocate another page, and copy the contents of the original page to it. This page is then "anonymous" i.e. only we can ask for it, and it is now backed by the paging device, not the vnode.

Well, not a huge advantage for written pages, but at least the I/O is done when we need it, and the previous savings apply.

What happens when we run another copy of foo? The actions are the same, but the effects differ because some of the pages are already in core, read-only and sharable. When the vnode segment ops resolve the page fault for the text pages (or read-only data pages) they will find the pages already mapped in for foo1.

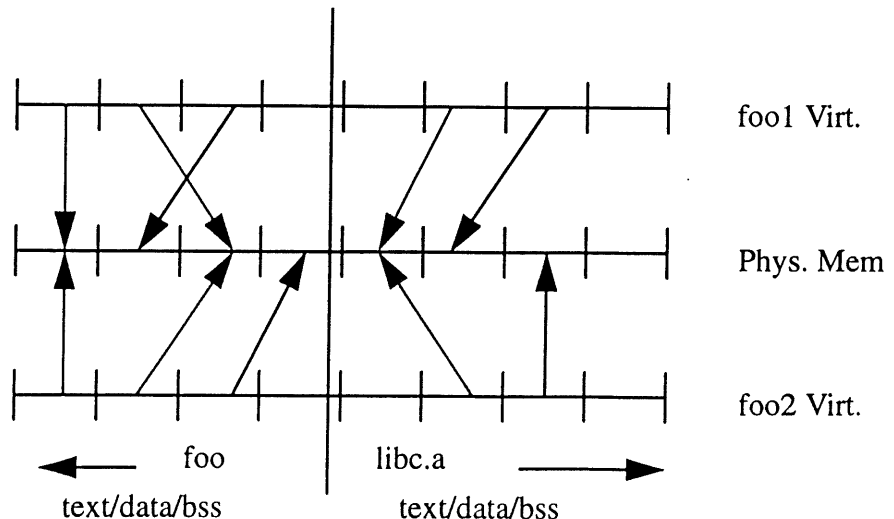
Now we can see one of the major advantages of the VM approach - data will be shared by different processes whenever possible. We don't have to keep a sticky bit and clog up the paging device, we just re-map a page when we can. The segment ops look up a page in physical memory by vnode and offset (where appropriate) and simply map it into another process address space.

If we can share pages in an executable, can we find a way to share things that every process in the system is likely to use? Yes, if we can map one executable into a process space and share portions, then it is a straightforward step to map another executable into the same address space. This is exactly the mechanism that SunOS 4.x and SVR4 use to implement shared libraries and shared objects.



Two processes sharing pages.

Two processes sharing pages and multiple text/data/bss mappings.



## Paper: Why aren't you using mmap() yet?

In this figure you can see that a process address space can be made up of multiple text, data, and bss segments. Instead of one each all lumped together, dynamic linking allows libraries and objects to be mapped and resolved as needed. More importantly, it allows common pages like library text to be shared among all processes, significantly reducing the amount of required memory resources.

Another huge advantage to dynamic linking and shared objects is that since the object is not linked statically into the executable, you can change it without relinking. In the case of libraries, it means developers can release and distribute portions of the product instead of doing full releases.

As an example, picture a C++ class implemented as a shared object. If you update the implementation of the class, but do not change the class definition, you only need to release the new class implementation, not your whole application.

### Reading Files

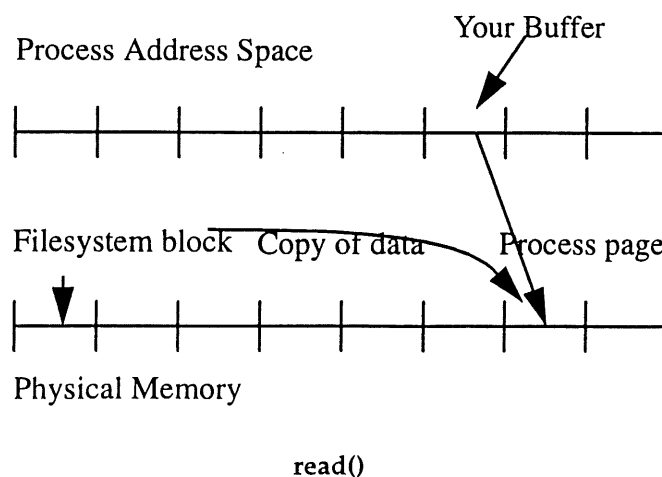
Now that we have a rough idea of how VM works, and how it benefits the system, let's take a look at how the facilities can be used by the application developer. The simplest example is reading a file. We'll use a small read by one process to see how `read()` differs from mapping a file, then look at the benefits for large reads, and more than one process using a file.

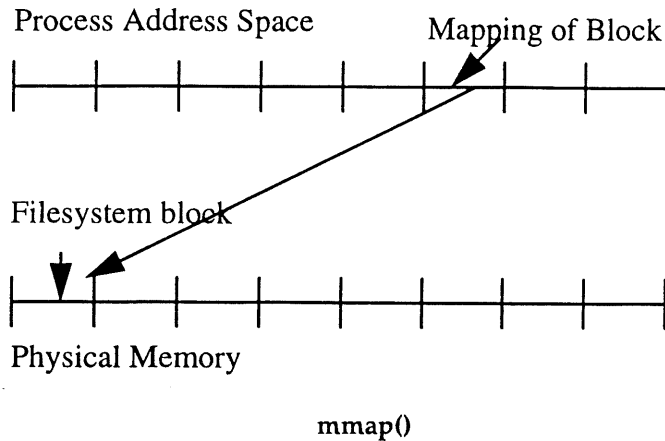
While the system does not have a pre-allocated portion of physical memory in which to store filesystem I/O, it uses a similar technique as before to perform reads to user buffers. Assume that we are reading the first 10 bytes from a file. The system first simulates a page fault, which allocates a page and

does the disk I/O in order to get the required disk blocks into memory, then makes sure that the portion of your process space that contains the buffer is in memory, then it copies the requested bytes into that memory. The resources consumed are two pages of memory (one of which is now dirty and may need to be written out), the space on the paging device to back the private page for the process, an I/O operation to get the filesystem block into memory, and a copy to move the info from one page to the other. We haven't even looked at the data yet...

Now let's use `mmap()` to get to the same 10 bytes. When we call `mmap()`, no I/O or allocation is done, all that happens is a record is made at the Address Space level of the association between a virtual address (which the system typically picks) and the vnode and offset (which in this case is 0). When we reference those 10 bytes for the first time, we generate a page fault, the same one that was simulated in the `read()` example. The system allocates a page, does the I/O, but instead of copying the data to our buffer, it maps the block used directly to our process address space. The resources consumed are one page of memory (and not dirty, so no further I/O is necessary) and one I/O operation. Half as much memory, half as much disk, and no copying.

Now let's take a look at what happens for a larger file. Let's assume that we are looking at a 2 MB file. When the `read()` is issued, the system has to allocate the pages for the file systems blocks, and do the I/O, then allocate pages for the private copy of the file, and copy the pages over. 2 MB of (in this example) useless pages, 2 MB of dirty pages, a big chunk of I/O, and we haven't even started yet. In this case the dirty pages in





the process address space are likely to be written back to disk, creating further I/O.

For `mmap()`, the association is made, and no I/O is done. As each page is referenced, a page fault is taken, a page is allocated, and the required I/O is done when needed (actually, in the normal case, read-ahead would halve the fault rate). Instead of a glut of I/O, and twice as much memory and disk consumed, we have used exactly what was needed when it was needed.

As a further example, imagine the case of a file half the size of physical memory or larger. For `read()`, by the time we are done we have used twice as much memory as needed, so the pages we probably want at the beginning are guaranteed to be out of memory, and will have to be read in again. For `mmap()`, references to the page we want cause it fault into memory as we need it. Consider a huge file - `mmap()` doesn't need the backing of swap for the private copy, so as long as the file exists, and we have enough address space, the file can be mapped. For `read()`, we need as much swap as there is file, and this only to read the file!

Now add multiple processes all using the file. For `read()`, we have each process needing a copy of the private pages, all the I/O done for the pages up front (to be fair, if the page is already in memory from another page fault, just the copy is done) memory to memory copies done for each of the processes, and swap allocated to cover all those pages. In other words, for  $N$  processes, we increase the system resource usage  $2*N$  times.

For `mmap()`, since the mapping looks directly for the page allocated by the segment layer, the pages are shared once the I/O is done. If every process is using the same pages at the same time, there is only one

copy in memory, and no swap backing it. In other words, the worst case for `mmap()` is 1 copy on disk (the original) and 1 copy in memory.

The cases for programs that read a file, modify it, and then write it back are even more striking. In addition to the work done to read the file, the whole file has to be doubly paged again to write it back as well! For `mmap()`, the original pages can be modified, resulting in exactly the amount of work that needs to be done on the pages.

The write only case is closer to the read only case - one copy of the file is in core, one copy of the file takes up disk space.

Last but not least, the user has no control over what the system does with the pages in their address space. With an `mmap()` ed file, you can use the facilities of `madvise()` and `msync()` to manage the resources, and give the system all the information it needs to manage optimally on your behalf.

Less filling, tastes great - why aren't you using `mmap()` yet?

### Cp and Cat

A good example of `mmap()` used to improve a simple program is the method `cat` and `cp` now use for input. If a file can be `mmap()` ed, both these programs will do so. What this means is that no user pages need be dirtied in order to do the I/O. When the `mmap()` is done, only the association is made for the process. When the `write()` call is issued to the output file, the kernel allocates the necessary pages to read the file in, and then does the write directly from there, halving the memory usage, eliminating an unnecessary copy of the data to user space and the use of swap backed private memory.

## Paper: Why aren't you using mmap() yet?

In the trivial case of "cp x /dev/null" or "cat foo >/dev/null", no I/O is done at all. The association is made (no I/O is performed) and then the write to /dev/null does nothing (in particular, it does not request the pages to be read in). While some consider this cheating, I'd submit that it is simply doing what it has been told to do - nothing.

## Examples

### map\_file() and make\_file()

The steps needed to map a file are 1) open the file 2) find the length of the file 3) mmap() the file and 4) close the file descriptor. There is no need to keep an open file descriptor for a mapping, which is very convenient for applications using many files.

In these two examples, we assume that the file is smaller than the address space, and map the whole file for use. For most applications, this is a reasonable assumption. For applications such as cp and cat, a more general method of mapping a chunk at a time is necessary.

```
struct mapping {
    void *addr;
    unsigned long len;
};
void *
map_file(mapping, name, permit)
    struct mapping *mapping;
    char *name;
    int permit;
{
    struct stat s;
    void *p = 0;
    int fd, prot;
    u_long len;

    if ((fd = open(name, permit)) < 0) {
        fprintf(stderr, "open of %s failed (%s)\n",
            name, sys_errlist(errno));
        return (p);
    } else if (fstat(fd, &s)) {
        fprintf(stderr, "stat of %s failed (%s)\n",
            name, sys_errlist(errno));
    } else {
        if (permit == O_RDWR) prot = PROT_READ |
            PROT_WRITE;
        else if (permit == O_WRONLY) prot =
            PROT_WRITE;
        else prot = PROT_READ;
        if ((p =
            (void *) mmap(0, len, prot, MAP_SHARED, fd,
            0)) == (void *) -1) {
            fprintf(stderr,
                "mmap of %s failed (%s)\n", name,
                sys_errlist(errno));
            p = 0;
        }
    }
}
```

```
    }
}
close(fd);
if (mapping) {
    mapping->addr = p;
    mapping->len = s.st_size;
}
return (p);
}
```

The mapping structure can be used to record the length of the file. If a null pointer is passed in for the structure, the routine just hands back the address of the mapping. If no mapping is performed, then a null pointer is returned. map\_file() uses the file permissions from the open() system call declarations - this was an aid to the original conversion for which this routine was written, it should probably use the flags that mmap() uses for specifying the protection desired. You will also note that this routine uses void \* as a generic pointer, where mmap() is declared as returning char \* (caddr\_t). Another routine caught in the ANSI C transition.

It is not difficult to create a file using mmap(), as the following (very similar) routine shows. It is important to note that no allocation of disk space is done until a given page is written, so there is no penalty for creating a huge file to map, and then making it smaller with truncate() once the final size is known.

```
void *
make_file(name, len)
    char *name;
    int len;
{
    void *bits = 0;
    int fd;

    if ((fd = open(name, O_RDWR | O_CREAT, 0666)) <
    0) {
        fprintf(stderr, "make_file failed for %s
        (%s)\n",
            name, sys_errlist(errno));
    } else if (ftruncate(fd, len)) {
        fprintf(stderr, "sizing failed for %s
        (%s)\n",
            name, sys_errlist(errno));
    } else {
        len = (len + getpagesize() - 1) &
            ~(getpagesize() - 1);
        if ((bits =
            (void *) mmap(0, len, PROT_READ |
            PROT_WRITE,
                MAP_SHARED, fd, 0)) == (void *) -1) {
            fprintf(stderr, "mmap failed for %s (%s)\n",
                name, sys_errlist(errno));
            bits = 0;
        }
    }
}
```



```

close(fd);
return (bits);
}

```

The size of the mapping is rounded up to the machine page size, but this *does not* change the size of the file. If a page fault occurs at the end, and the offset is not beyond the real size of the file, the page fault succeeds, and a whole page with the end portion is mapped. If a page fault sees an access beyond the real end, a segmentation violation is signalled.

### Simple cp

A simple copy program using the above routines would then be something like this:

```

main(int argc, char **argv)
{
    int fd;
    struct mapping in;

    /* check arguments */

    if ((fd = open(argv[2], O_WRONLY, 0666)) < 0) {
        /* error processing for write side */
    } else if (!map_file(&in, argv[2], O_RDONLY)) {
        /* error processing for read side */
    } else if (write(fd, m.addr, m.len) != m.len) {
        /* error processing for copy */
    }
}

```

In the above example, the pages will not be read in until the `write()` call asks that the pages be read in as part of I/O. More importantly, no user pages are dirtied. Also, all of the I/O will be done in the kernel, allowing it to choose the optimal size chunk, instead of making the application decide on a buffer size.

### Accounting Summary

As another example of an application reading a file more efficiently, let's look at a quick accounting summary. The accounting file (typically `/var/adm/pacct`) is organized as a series of 32 byte fixed length records as defined in `/usr/include/sys/acct.h`. For our example, the sum of all characters transferred will be computed. `ctoi()` is a routine that converts between the `comp_t` format used by accounting to an integer.

```

main(int argc, char **argv)
{
    struct mapping af;
    struct acct *ap;
    u_longnap, nchars = 0;
    char *filename = (argc > 1) ? argv[1] : "/var/adm/pacct";

```

```

    if (!map_file(&af, filename, O_RDONLY)) {
        /* error processing */
        exit(1);
    }
    ap = (struct acct *)af.addr;
    nap = af.len/sizeof(*ap);
    while(nap--) {
        nchars += ctoi(ap->ac_io);
        ap++;
    }
    printf("%d\n", nchars);
    exit(0);
}

```

The equivalent example using `read()` and `fstat()` would read the records one at a time into a private copy or read the entire file into an allocated array of `acct` structs. The advantage of mapping the file in the first case is that no `read()` calls are done - on a machine with a 4K pagesize, this eliminates 128 system calls and copies per page. In the second case, as we have seen before, we eliminate reading the entire file at once and the swap backed private pages used to hold our copy.

### Read Modify Write

We've seen the advantages for simple reader programs, let's look at an application that does an in-place modification of data. In this example, let's assume we have an 8-bit pseudo-color Sun rasterfile with a ramp in gray, and we want to make it a bit brighter. We will assume that the rasterfile header correctly describes the size of the file. In practice, an application should check this assumption, otherwise accesses to invalid parts of the mapping could create segmentation violations.

```

main(int argc, char **argv)
{
    u_char *p; /* pointer to pixels */
    u_char pixel; /* pixel value */
    struct rasterfile *rp; /* pointer to raster header */
    u_longsize;

    /* check arguments */

    if (!(p = (u_char *) map_file( (struct mapping *)0, argv[1], O_RDWR))) {
        /* error processing */
    }

    rp = (struct rasterfile *)p; /* pointer to raster header */
    /* check raster header */
    p += ((sizeof(*rp) + rp->ras_maplength); /* skip header and cmap */
    size = rp->ras_width * rp->ras_height;
    while(size--) {
        pixel = *p;

```

## Paper: Why aren't you using mmap() yet?

```
    ii (pixel != 255) *p++ = ++pixel;
    else p++;
}
exit(0);
}
```

Fortunately, this is not a paper on C programming or image enhancement :-). Let's look at the benefits of an in-place mapping for applications that do read-modify-write. Using the normal `malloc()/read()/process/write()` model, we would have to read the whole file into a private copy. This would dirty our swap backed private pages and cause all of the file I/O to be done at once. We then have to manipulate that copy, and write it back, causing all of our private pages and all of the file pages to be loaded one more time. Two memory copies, use of a swap copy, and two complete runs thru the file.

For the mapped version, we access the pages actually used by the file, so no private copy is required, the system is far more likely to be able to keep the pages in memory, and when we are done, the actual pages that comprise the file have already been modified, so no write is necessary. The pages will age as appropriate and be written when necessary.

### Large and Huge Files

Presuming that the previous example was a typical 1152x900 rasterfile, we were only dealing with approximately 1MB worth of data. What if the size of the file approaches the size of physical memory? What if it exceeds it? What if it exceeds the size of available swap space?

As already noted, using `read()` and `write()` presumes that the file pages will be read in and copied to or from a private copy. When the size of the desired manipulation starts approaching half of physical memory, `read()/write()` applications will start to "turn the bend" in performance, because the limiting factor will be disk I/O as the system will be bound by the paging rate - and this happens before you access the data. For a mapped application, the bend is at worst about the size of physical memory, and depending on how fast an application moves thru the pages, it may never become paging limited.

As the size of the chunk goes beyond physical memory, the `read()/write()` application still has to thrash thru the whole file before accessing the first byte, where the mapped application is still in the same situation as before - pages are drawn in as needed, and with use of `madvise()`, the application can help the system do exactly the right thing.

As the size of the chunk approaches the size of the swap device, the `read()/write()` application has to break up the problem into smaller chunks, and it is using a somewhat precious system resource (and presumably I/O bandwidth) to do so. The mapped application needs no backing from swap, so the first limit it runs into is the size of the virtual address space.

It was in fact the manipulation of large image files that first led the author to use `mmap()`. The benefits in performance for 1MB files is noticeable. When you are dealing with 225MB files, the path quickly becomes obvious.

### XImages

A brief note here - most raster display formats can be described by XImage with the right settings. It is a big performance win (especially in systems like Sun that have a Direct X interface) to `mmap()` the image data, and use a pointer to the mapping in the XImage structure instead of reading a private copy. A recent animation example showed a 6X performance enhancement using `mmap()` to access all the data, and just switching the pointer instead of reading different files all the time...

### Shared Data as Memory

The benefits of sharing pages being used by many cooperating processes has been discussed earlier in this paper. An interesting application for `mmap()` is that of using a commonly mapped file as shared memory. In addition to the simpler interface and the benefits of being able to rendezvous using the file system namespace, the model has some less obvious benefits as well when file locking is used.

As an example, consider a print spooler that has to process files in an input queue, and then move the processed files to an output queue to be processed by an output manager. RPC calls could be used to move the files about, but in a resilient system, the information must be kept around in order to restart the jobs.

The input queue would have a header describing the number of entries, the next free slot in the table, and the output queue would have the same. Manipulation of the queue entries can be accomplished by a common set of routines. Before updating an entry, the routines would acquire a write lock using the `flock()` system call (this allows a portion of the file to be locked) in order to make the access atomic. When information from the file is read, a read lock is acquired. If another process is in the midst of writing or reading that portion of the file, then competing

process will block. If non-exclusive access (such as two reads) is attempted, both processes will succeed.

The processes share the data (which can be extensive), there is effectively no limit of the size of the shared memory that can be used, and the files can use advisory locking to guarantee atomic access to the file contents. The programming model is that of shared memory, instead of having to `read()` and `write()` entries as well as performing locking.

In the author's experience, mapping files as shared data is much easier, much more extensible in size, and more efficient (when you consider the control that `flock()` and `advise()` give you over the file) than using shared memory and/or semaphores.

### Shared Objects

We've discussed the implementation of shared libraries and objects before, but it is a small step from using shared data with `mmap()` to using shared objects as well. The point to be made here is that with the benefits of access methods and data hiding provided by languages such as C++, classes can use `mmap()` in their implementation for performance, and the application programmer need never know.

### Caveats

While VM and the facilities it provides have a number of advantages, there are some caveats to consider.

The first is that `mmap()` won't work on stream based files such as sockets and pipes, so applications that have to deal with these will have to have a separate access method. It is worth considering re-writing the application to use regular files. As an example, image processing systems that use a series of filters to effect transformations might better be implemented as routines that `mmap()` (possibly temporary) files in order to eliminate the private pages and better control use of memory. In some cases, it just isn't possible.

The second is the use of NFS files for mapping. NFS files will `mmap()`, but only if file locking (using `lockf()` or `fcntl()`) is *not* being used. The typical way to get around this is to use a secondary file for the locking, and `mmap()` the original. Use of `msync()` to guarantee that you have a coherent copy in conjunction with ancillary locking works, but is obviously not as easy as mapping UFS files locally. It is also worth noting that network locks are slower than `flock()`, and have been known to have the odd problem.

The last caveat is that you are using a new method of access for files. You will make mistakes, and there will be some differences in the way the system behaves.

You'll notice update more often (consider turning it off for large numbers of written pages running a long time), you'll get core dumps with addresses in strange places, you'll forget the offset parameter, you'll forget to use both `PROT_READ` and `PROT_WRITE`, you'll forget to open the file with the right permissions. Be patient, it's worth it.

### Summary

VM and the `mmap()` and `advise()` facilities are generally much more efficient and allow you to give the paging system explicit information on how to deal with the use of memory for applications. It is present in SVR4, which means it can no longer be considered a SunOS specific interface.

Examples of all the applications mentioned above are available, as well as electronic copies of this paper. The documentation for `mmap()` has never been extensive, but it is the hope of the author that this paper will be the start of a communal application note, and that the examples made available will come back to form a good toolkit for those seeking to move quickly to `mmap()`.

With all the advantages, I can only ask: *Why aren't you using mmap() yet?*

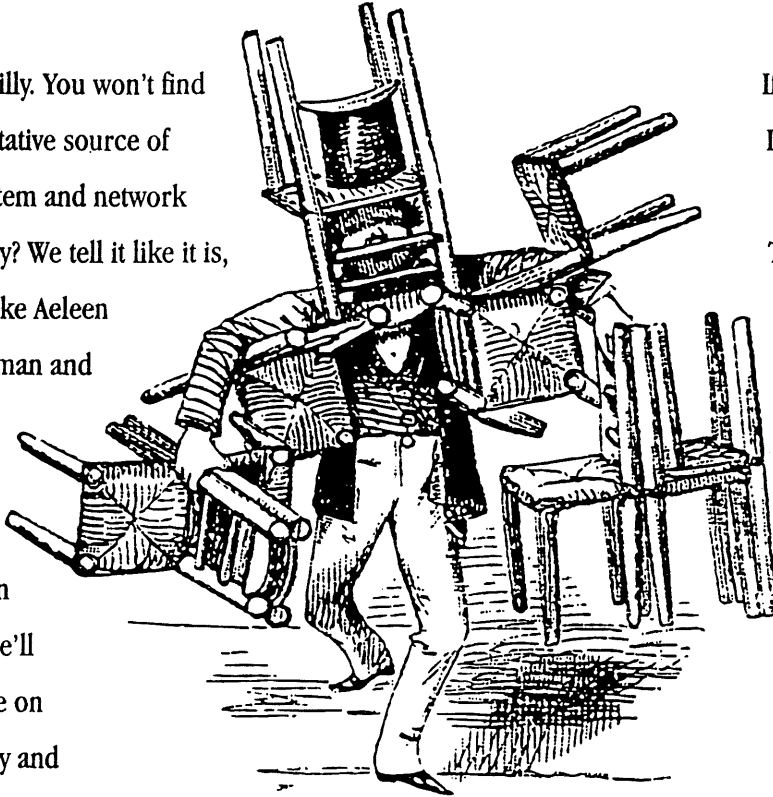
### References

Leffler, McKusick, Karels, Quarterman; *The Design and Implementation of the 4.3BSD Operating System*, Addison Wesley, ISBN 0-201-06196-1 (1989)

Gingell, Moran, Shannon; *Virtual Memory Architecture in SunOS*, USENIX Association Conference Proceedings, pp 81-94, (1987)

# Where do system administrators go for workarounds, tips and tricks and networking solutions?

In a word, O'Reilly. You won't find a more authoritative source of information on system and network administration. Why? We tell it like it is, thanks to experts like Aileen Frisch, Brent Chapman and Gene Spafford. You'll learn the nuts and bolts of UNIX administration and security. And we'll keep you up-to-date on the latest technology and industry trends.



If you're like most sys admins, Internet administration is taking up more and more of your time. That's why you should check out the O'Reilly networking classics shown here. And don't forget the second edition of our Classic Essential System Administration. For detailed information about all of our books and software products, check out

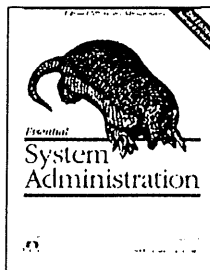
<http://www.ora.com/>

When ordering please refer to code ASYS



Explains motivation behind DNS and how to set up the BIND software. More advanced topics such as how to become a "parent" are also covered.

418 pages \$59.95  
ISBN: 1565920104



Introduces you to the care and feeding of UNIX systems in the real-world. New edition has been updated to reflect latest versions of all major UNIX variants.

788 pages \$65.00  
ISBN: 1565921275



Far away the most comprehensive book on sendmail. Includes a complete tutorial.

830 pages \$65.00  
ISBN: 1565920562



How to set up Internet servers and become a publisher on the Net.

668 pages \$59.95  
ISBN: 1565920627



Complete guide to setting up and running a TCP/IP network. Includes basic setup and how to configure important network applications.

502 pages \$59.95  
ISBN: 093717582X

Available from your local bookshop.



Distributed by Woodslane Pty Ltd. A.C.N. 004677519  
Unit 7/5 Yuko Place Warriewood NSW 2102  
Phone (02) 9970 5111 Fax (02) 9970 5002

O'REILLY

Advice:

# A Moving Story

Andrew van der Stock

Moving offices happens to everybody sooner or later. If you happen to be part of the Systems Administration area, your role in any move is more important than ever before. It is vital that you are involved in every stage of any office move. Moves are also a great opportunity to revitalise your network and your unit's image - if you do it right.

It's vital to plan your move - this cannot be stressed highly enough. You should be involved in the selection of new offices, ensuring that sufficient data and phone points are available to staff, and that any machine room resources you need are adequate. There is no point moving into a shiny new office if your legacy servers cannot be installed because they do not fit through the door! Think about climate control, data risers, hub positioning, cabling contractors and so on, even before you have selected the building. If you don't have the expertise, don't be afraid to use consultants, as you have to live with the results for years to come.

Moving is an ideal time to totally redraw your network, installing modern switches, changing from coax to 10Base-T cabling, and so on. You should work out how to re-partition your network based upon real life statistics, so start collecting the traffic data now. You should also take into account a great number of users are mobile not only in terms of equipment but also in terms of job description. They may work in one department three days a week, and another two days a week. Do not build in dependencies that assume a locked down desktop workstation.

After the building is chosen, get cabling contractors or in-house expertise to wire the entire place as the office is being outfitted. Work with the building electricians - the result will be much cleaner port placement, and much less cross-talk between your data and the power runs. Ensure that the cable contractor tests each and every port and cable run, and that they supply the cables - this will reduce the heartache of being bounced between two competing contractors when the cables don't work in real life. Get the climate control in your machine room installed and tested to your satisfaction before installing a single piece of equipment. Once you are happy with the machine room, it's probably worth moving your legacy servers in staged moves. You'll need to set up a WAN link, as the risk in moving the servers at the same time as everyone else is just too high.

The icky bit is more than likely the packing and unpacking of workstations. There are far more workstations than servers, and more than likely your group will be called on to do the dirty work. To trick here is to minimise the dirty work, whilst still remaining in control of the process. Hire computer movers, who know how to pack a workstation and transport it in one piece. Let the user decide where they want their workstation placed before the move even takes place - so that desks and the workstation are placed in offices in the correct place to start with. Users can waste an extraordinary amount of time moving a workstation around if you let them. Figure out which users are going to be affected by sun-glare and make suggestions before they move. If new furniture is being bought, stipulate that the office furniture must not have full backs, and have ports already pre-drilled for cabling. Full backed furniture makes it impossible for the desk to be moved flush to the wall, and stops you from plugging the phone and workstation in.

It is essential that you under-promise and over-deliver during moves. Do not promise that everything will be working from day one - a very career limiting move. Make sure that you have adequate staff for the move - do not allow holidays or training to fall during the move. Make sure that you have the authority to pay extensive overtime to ensure that you have enough staff to handle the weekends before and after the move. Talk to other section heads to ensure their staff don't take rostered days off in the aftermath of the move - you need them to find stuff in their boxes and to prove to them that you do indeed work hard - image is everything, especially if you don't visit the users all that often.

If you have planned well, the workstations will be in the new building and you'll need to pop around to each and every one and get it going. It is very hard to justify coming to each and every workstation all the time, so if you moved the servers previously, this is the last time that you need to come by and reconfigure workstations. It is very worthwhile making sure that the asset register is accurate, and that equipment is properly secured during this phase. Moving can be a very trying experience (trust me on this one :), but if you do it right and in a timely fashion, the rewards will be well deserved. Moving can help improve your group's image with the users, as they finally get to see your group in action, and management finally realise that your group is very important in the day to day operations of their company. Keep this in mind, especially if your company doesn't have a CIO (or equivalent) yet. :-)\*

Background:

# Linux - The Choice of a New Generation

Frank Crawford

UNIX has long been known as a system that could handle data from almost anything, including non-Unix systems. For example, the ability to translate EBCDIC to ASCII or blocked records to unblocked records. Much of this is because Unix has long been an excellent development platform and programmers have been keen to transfer to from other systems.

Today, Linux is extending that interoperability to modern systems, such as DOS, Windows95, NT and MacOS. In fact this interoperability goes much further than was previously considered. Certainly, the functionality supplied is not unique to Linux, many of the individual packages are also available in other versions of Unix, but Linux has a wider range than any other.

The reason for this is fairly simple, many of the packages require changes to the operating system, and the availability of the kernel source for Linux provides ample opportunity for this to occur. The only commercial operating system that approaches Linux in this area, is SunOS 4.1.X, mainly due to its large penetration in the Unix market place, particularly in organisations developing applications. It is interesting that the other free Unix's, such as FreeBSD and NetBSD, are also well represented by such packages.

The packages available cover a wide range of areas, starting from simple data format translation, which includes facilities to manipulate DOS floppy disks and files and to translate between DOS and Unix format text files as a standard produce. For more seamless use of DOS data, there is the "msdos" file system, which allows Linux to directly access, or mount, a DOS partition and treat it like a normal Linux file. In fact, most Linux systems automatically mount the DOS C-Drive (if present) as part of their standard startup.

Seamlessly accessing data is one side of interoperability, running DOS programs is another. The freely available package, dosemu, being developed for Linux allows most DOS programs to be executed directly under Linux. This package goes so far as to even support IPX and CDRom access, and allows DOS programs to be run in a separate X window. Obviously, this is only for occasional use, as heavy use of DOS programs will always be more efficient under native DOS.

For those who aren't satisfied with just DOS, work is underway to allow dosemu to support MS-Windows 3.1. While this is in it's infancy, it is possible to run simple applications, such as calendar, at the present time.

A "rival" group is also working on adding support for MS-Windows to Linux. However, this project is taking the WABI approach, i.e. translating the Windows ABI to X11 calls. While this also is in its early stages, it can currently handle more advanced programs like "solitaire".

Another major package available for Linux is SAMBA, which performs the same functions as LanManager to support the SMB protocol. This is the same protocol as supported by Windows for Workgroups and Windows95, for sharing files and printers over the network. SAMBA allows the Linux system to export its file systems and printers for use by Windows PC, and also to access printers and file systems shared by Windows PCs. Support includes network login and authentication, and even allows the use of NT servers. In fact, SAMBA highlighted some problems with Microsoft's implementation of network logins, which they subsequently fixed.

To round out the interoperability with NT, Linux also supports the hpfs in a read-only mode. Again, this has caused some concerns for NT, as it is possible to set up a Linux boot disk with the hpfs file system enabled, boot an NT server and read files, bypassing NT's normal security. This is just another example of the old security adage, that once you have physical access to a machine you can do anything to it.

Linux access isn't restricted to Microsoft products, there are also numerous packages available for MacOS. For a start there is a module available for mounting, in read-only mode, MacOS file systems. As well, there are other packages to handle MacOS specific data formats such as binhex.

Going further, packages are also available for supporting Macintosh file and printer sharing. There are two common packages, CAP and netatalk. Both provide similar functionality, although netatalk is more closely tied to Linux, as the kernel modules required are now being distributed as a part of the development kernel (Linux 1.3.X). Both these packages allow Linux file systems to be exported to Macintosh systems, and for printers on either the Linux or Macintosh system to be used across the network.

With this wealth of tools, it is simple to set up an environment that will support MacIntosh's, DOS and Windows systems, all communicating through a common server, without having to resort to expensive third-party packages. All you need is a Linux system and someone who knows how to use it. ❖

Opinion:

# Will the real Information SuperHighway please stand up!

Phil McCrea

I never cease to be fascinated by the numbers emanating from the Optus Vision and Foxtel public relations offices concerning how many homes their respective cables have now past - Foxtel in the ground, and Optus Vision strung from pole to pole. And the amount of column inches this all receives in the business press: hardly an issue goes by without a feature article on pay TV.

Telstra and Optus seem convinced that Australians will rush headlong into pay TV, much in the same way as we have rushed into portable telephones over the past few years. It is unlikely that this writer will succumb to the temptation of pay TV, as long as we have quality programming on ABC and SBS. (Mind you, if it transpires that the game they play in heaven - rugby union - were to be available *only* on pay TV, this policy may have to be re-considered...)

What I object to is the notion that Foxtel and Optus Vision are in some way bringing us the 'Information Super-highway' by means of their cable. Have you noticed that people who use terms such as the dreaded ISH generally do not understand the concept of on-line services? - and probably have never gone near a keyboard in their lives. The ISH is *not* recycled television programs available at the drop of a hat (or smart-card for that matter). The ISH - if we can use the term - is about *interaction*, and furthermore we mean *symmetrical* (or near symmetrical) interaction, where the forward and return data paths are of similar carrying capacity. In other words, a user of the ISH can create as much information as (s)he receives.

Let's drop the term ISH, and use the more appropriate term 'Internet' - after all the Internet is by far the closest thing we have at present to a real ISH. Problem is, the term Internet strikes fear in the over 45s, who were brought up before keyboards became prolific. Given that most people in this age range know how to watch TV, but not how to use computers, it is not difficult to see why our communications companies

are outlaying such a huge investment in pay TV infrastructure.

I'm being a little cynical, I know, but it is certainly true to say that the cable layers did not envisage the increased popularity of the World Wide Web, which is being delivered over narrow band - ie telephone lines - rather than over cable. Whilst the data capacity of telephones is not as high as cables, it can be quite acceptable for many business applications. Data rates are 28.8kbps using current technology modems, although ISDN can provide 128Kbps over the same telephone lines, using an all-digital approach which does not require modems at all (recall that modems modulate and de-modulate - hence the name - from analogue to digital and vice-versa).

The ISDN rate of 128K is more than adequate for most Web services that are currently used by business. And furthermore ISDN uses the current telecommunications infrastructure. Several years ago Australia was a leader in adopting ISDN, but sadly we have lost this lead. This can partly be attributed to the pricing policy for ISDN, which has placed it outside the reach of most organisations - and certainly homes.

Telephone lines are the primary means of communications for business for both voice and data - not coax cables. We can even send letters over telephone lines using facsimile machines. Web delivery over ISDN is not only possible, it is achievable right now.

So what is our Government-owned telecommunications company doing to encourage Australian industry to use the Internet for competitive advantage? If the price of ISDN is the main guide, then not a lot. Rolling out cable for television-watching armchair norms at home should not be a priority for our government-owned telecommunications company.

AUUG is committed to the view that IP based networks (ie the Internet) will become the default communications protocol for all on-line services. We cannot afford to wait for several years till cables appear behind every PC in the country, both at home and in the office. Let us use our existing infrastructure - i.e., telephone lines - more effectively for on-line services in the short to medium term. It's time to reduce the price of ISDN significantly - i.e., by an order of magnitude. Business will be given an almighty boost as a result.❖

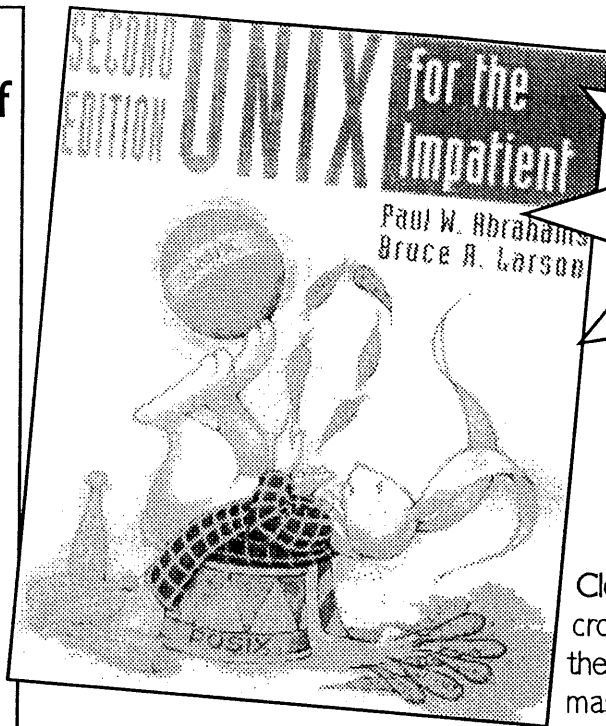
New edition of  
one of our  
bestselling  
UNIX books  
now  
completely  
revised &  
updated.

## UNIX FOR THE IMPATIENT

Second Edition

Paul Abrahams and  
Bruce Larson

0-201-82376-4  
650 pages/ Paperback



20% off for  
AUUG  
members

Clear, concise, and readable. Thoroughly cross-referenced and indexed, UNIX for the Impatient is the power-user's guide to mastering UNIX quickly.

Organized functionally and useful for any UNIX implementation on any platform, including System V, BSD, LINUX, Solaris, POSIX.2, ULTRIX, UNIXWARE (Novell), and others.

Covers all essential information including:

- File operations
- Data manipulation using filters
- The awk programming language
- Shells and shell scripts, now using the KornShell
- Editors — vi, ex, ed, and GNU Emacs
- Mailers, remote addressing, telnet, ftp, uucp, WWW, other Internet issues
- GUIs, including X Windows

I would like to order \_\_\_\_\_ copy/copies of **UNIX for the Impatient** by Paul Abrahams and Bruce Larson (82376-4) at the special AUUG price of \$35.95 (RRP \$44.95)

I enclose a cheque for \$ \_\_\_\_\_ **OR**  Please charge my credit card No: \_\_\_\_\_

BANKCARD     VISA     MASTERCARD     AMERICAN EXPRESS ID No: \_\_\_\_\_

EXPIRY DATE:    /    /    AUUG MEMBERSHIP NO: \_\_\_\_\_    PHONE NO: \_\_\_\_\_ (Wk Hrs)

NAME: \_\_\_\_\_    SIGNATURE: \_\_\_\_\_

STREET ADDRESS: \_\_\_\_\_

STATE: \_\_\_\_\_    POST CODE: \_\_\_\_\_

Send your  
order to >

Addison  
Wesley  
Longman

Unit A1, 6 Byfield Street,  
North Ryde, NSW 2113, Australia.  
Telephone (02) 878 5411  
Customer Service Fax (02) 888 9404

 LONGMAN

 ADDISON-WESLEY



## Background:

# What is a Network?

Frank Crawford

The only thing that matters in a network is the bandwidth, right? After all the only thing that matters is getting the data from place A to place B as fast as possible. This is the attitude that is taken with regard to the Australian Internet, and is also taken by most organisations for the internal networks. Unfortunately, it is very wrong and can lead to disastrous and expensive mistakes.

As an example, at a recent conference I stated that the Internet in Australia will probably have a very rocky road in the next six to twelve months, and was promptly told that an increase in bandwidth would fix it. However, I didn't see the problems being with the current bandwidth, but with the current administrative procedures and network management. This is not to say that Telstra Internet Services are at fault, they just aren't responsible for all the details.

As a different example, I was recently discussing how to speed up a network backup and the immediate reaction was that we should replace the Ethernet connection with an FDDI connection. This statement was made with no evidence or even thought as to where the bottlenecks might be. Just as an aside, I believe that the major problem is with the software configuration and not with any network performance, however, I'll wait for some performance statistics before looking for a solution.

So what does constitute a network? Obviously, the most visible component is the cable plant, whether it be your local Ethernet or fibre cables, or leased bandwidth from Telstra. To make use of this plant are the devices to access it, be it cards in a PC, high speed routers (which also perform other functions) or an ISDN NTU. This is just the start, there are also bridges, splitters, repeaters and a myriad of other hardware components, all of which are visible.

However, more and more these days, much of what is needed for the network is not visible. Like everything else, much of any network is driven by software, from simple things like PC card configuration programs to dedicated specialist operating systems such as found in CISCO routers. This software is an integral part of any network, as without it all you have is some electrically connected devices with no purpose.

Much of the software is also expected to perform a wide variety of functions, from selecting the next path for data to traverse, to splitting, join or reformatting data to conform to requirements for the next step and

on to updating other devices on current best routes and possible destinations for data packets.

At the layer above this you reach more abstract but no less important items. This includes things like names to address mappings, such as performed by DNS and Novell's NDS but also mapping of IP addresses to Ethernet addresses. Without these type of services again no network would exist, as it would be impossible to specify a destination for data, even if you could physically communicate with that destination.

Most of modern implementation of name to address mapping are using dedicate servers, where as previously it was common for the information to be replicated on every host. This can be seen in the Internet in the change from a flat file (/etc/hosts) to the distributed database system used for the Domain Naming System (DNS) where every domain must have a (possibly shared) server.

At the level above this is the need to have some valid destination for your data, i.e. a server of some sort. For example, on a local area network, you would probably need to access a file or print server, while on the Internet today you are more likely to want to access a Web or FTP server. Without something for you to access, there is no point to designing and building a network.

A final essential part of a network is the people to manage and administer it. Without them, no network would continue to function for very long. As an example, consider the rapid growth in the Internet in Australia, particularly in the domain ".com.au". The time taken to just keep the DNS entries up to date has been estimated at one to two man days per week, and that is with extensive work on automating the process! Without such human support the domain would rapidly become unusable, as new sites would not be reachable and old invalid addresses would become common.

When I say that the Australian Internet is going to run into trouble, it is not because of bandwidth, but rather because of problems with many of the other areas related to the network. In particular, the area of concern is the people involved in managing and administering the network, many of whom do it on either a voluntary basis or as a secondary function to their real job. However, when the time required exceeds the time available something has to give, and that will most likely be their support of the network.

Finally and just as important, when you discuss the network within your organisation, it's cost and it's support do you take into account all the items listed above or do you just look at the most visible?❖

Advice:

# Computer Room Archaeology

Frank Crawford

Christmas has come and gone, and most people in the computing industry have taken the chance to have a holiday. But there is one group within your organisation that probably didn't get a holiday and that is your operations staff.

However, there is a chance that their work was different to their normal routine. Whereas normally your operations people have to react to the day to day issues and demands, this quiet period is a chance to catch up.

This catching up often takes the form of reorganising and relocating equipment for either better maintenance, future growth or just for easier location. All these activities can often be seen as either trivial or non-essential especially to those outside of the operations area.

In fact, this "non-essential" work is very important to the continued successful operations of any installation. Even more, it can be a very interesting part of operations work.

In an operational setup that has been established for a long time, there is a chance to study the history of computer and networking technology. Because the installations of new equipment and networks usually occurs in a "hap-hazard" fashion, which means that any older cables or equipment gradually get buried. If the computing centre has a false floor, it becomes worse, as it is easy to hide the mess where it isn't obvious. The study of this "spaghetti" could be described as "computer room archaeology".

During this down time some interesting things can be found. For example during a recent chance to reorganise a computer room, I had a chance to do a bit of archaeology. The room had a long history going back to the early 1960's, although much of it had been cleaned out in the 1980's.

Starting at the lowest layer, there were a very large number of RS232 cables, often running for considerable distances. Almost all of these were no longer connected to any equipment. On top of (or often tangle with) these was a thick ethernet cable, snaking over the floor, however, this time there were

still a few units connected to it. The RS232 cables were the prime means of communications in the early 1980's for Unix systems, with ethernet beginning to be widely deployed. However, the type of cables used were far different to today.

In a layer above this was found thin ethernet coax cables, in nearly the same quantity as the RS232 cables, obviously being the modern equivalent. Along with these were some twisted pair cabling, showing some of the new technology that is now coming on line. As well, there was one fibre optics cable, as an example of things to come.

However, not everything has changed over time. Within the mix of cables were also IBM channel cables, which were found at all levels. These seem to indicate that the IBM architecture follows the old saying "if it works, don't change it". One other interesting point was that even unused channel cables could not be removed, because they also snaked around the floor, but were also too tightly bundled with other and could not be separated from cables currently in use.

Finally, within the mix were also numerous power cable, power boards and telephone cables, many of which were unused, showing the continuing need for these. Again there were some interesting issues in the power cables, in that previously extension cords and 20 amp plugs were common, today power boards and 10 amp supplies are the norm. One other issue with power boards, it appears that any new piece of equipment demands the installation of at least one, if not more power boards, often daisy chained from previous boards.

One outcome of this general reorganisation, was an increase in the reliability and maintainability of the equipment in use. No longer will it be necessary to hunt through a number of different connectors, many of which are not connected to anything, to replace something that has been dislodged.

If the operations staff within your organisation don't regularly retrieve cables and reorganise their computer room, you should ask why. If they claim that it is hard work for no reward, just describe to them both the interesting things you can learn, and the necessity to run an efficient operation. If they still object why not find a local archaeology student and see if they are willing to do it?❖

# UNIX Tricks & Traps

Edited by Janet  
 Jackson<janet@dialix.oz.au>  
 Phone/Fax (09) 295 4753

I still need more submissions for this column. Take a look at your startup files and your `bin` directory and see if you've invented anything that might interest other UNIX users. Or write a short article on how to get the most out of some piece of UNIX software. And next time someone asks you a question, CC the answer to me. ❖

## Getting the exit status of any command in a pipeline

Glenn Huxtable, Functional Software <glenn@fs.com.au>

After the Bourne shell runs a command, the shell variable `?` contains the command's exit status. If the command was a pipeline, `?` contains the exit status of the last program in the pipeline.

The following example shows how to get the exit status of the other programs in the pipeline. The example is adapted from a backup program that uses `find` piped to `cpio` piped to `compress`. If `find` or `cpio` fails (for example, because of a disk error—this happened to one of our customers) we want to know!

In the Bourne shell command

```
exec 4>&1
```

`exec` is not being used for its normal purpose, which would be to replace the current process. Instead, it says to redirect file descriptor 4 to the same place as file descriptor 1 (standard output) from now on.

```
#!/bin/sh
# this script works by writing out the exit status of each component of
# a pipeline as a variable assignment and later "eval"ing the variable
# assignments to set the actual variables.
# to do this requires a good deal of messing about with file
# descriptors, mostly to do with preserving the scripts STDOUT while
# internally abusing STDOUT for passing the variable assignments
# out of the pipeline.
#
# Glenn Huxtable (glenn@fs.com.au) Functional Software
#
Dir=$1

# FD 0, 1 and 2 are STDIN, STDOUT and STDERR as usual
# redirect echo xxxx_status to FD 3 as STDOUT is used for pipe
COMMAND="(find $Dir -mount -print; echo find_status=\$? >&3) \
          |(cpio -ocB -Hodc; echo cpio_status=\$? >&3) \
          |(compress; echo compress_status=\$? >&3) "

# what was sent to FD 3 needs to go to STDOUT so back-ticks substitution can
# catch it and so the real STDOUT has to be sent somewhere else (FD 4)
# we want to use this script as a filter, so FD 4 has to go to the script's
# STDOUT

# make FD 4 go to the same place as STDOUT (FD 1)
exec 4>&1

# evaluate the pipeline. returning status assignments into 'stati'
# stati will look something like
#   find_status=0 cpio_status=0 compress_status=0
#   or perhaps, if a find error occurred...
#   find_status=1 cpio_status=0 compress_status=0
stati='eval "($COMMAND >&4) 3>&1"'
wait

# initialise known status
find_status=0; cpio_status=0; index_status=0; compress_status=0

# evaluate status assignments
eval $stati

exit_status=0

# the variables "find_status" etc are now set.
if [ $find_status != 0 ] ; then
  echo "$0: find failed with exit status $find_status" >&2
  exit_status=1
fi

if [ $cpio_status != 0 ] ; then
  echo "$0: cpio failed with exit status $cpio_status" >&2
  exit_status=1
fi

if [ $compress_status != 0 ] ; then
  echo "$0: compress failed with exit status $compress_status" >&2
  exit_status=1
fi

exit $exit_status
```

❖

# AUUG BOOK CLUB & PRENTICE HALL AUSTRALIA

## 20% DISCOUNT TO AUUG MEMBERS

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

***Panic! System Crash Dump Analysis***

ISBN: 0131493868 Bk & Disk \$73.95\*

\*Deduct 20% from listed retail price

### CONTACT DETAILS

Mr/Mrs/Ms/Dr: .....

First name: .....

Surname: .....

Position: .....

Company: .....

Address: .....

City/Suburb: .....

State: .....

Postcode: .....

Telephone: ( ) .....

Fax: ( ) .....

### PAYMENT DETAILS

Enclosed is a Cheque for \$ .....

(Payable to 'Prentice Hall Australia Pty Ltd')


Charge my  Bankcard  Visa  MasterCard


Card Number: .....

Card Expiry Date: .....

Signature: .....

### TO ORDER

 Fast Phone Service: Liz Guthrie (02) 9907 5648

 Fax: (02) 9905 7934

Mail to Prentice Hall Australia,

7 Grosvenor Place BROOKVALE NSW 2100

**GUARANTEE:** *If you are not completely satisfied you may return the book for a full refund within 30 days.*



Prentice Hall Pty. Ltd.

7 Grosvenor Place, Brookvale NSW 2100.

Tel: (02) 9939 1333 Fax: (02) 9905 7934

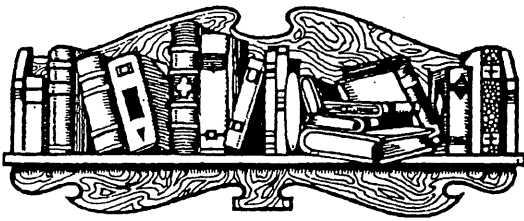
A V I A C O M C O M P A N Y

# Book Reviews

Frank Crawford <frank@ansto.gov.au>

Well here we are, into the new year, with many interesting and informative reviews covering such topics as software project management, the World-Wide-Web (multiple times), software porting, understanding and fixing PCs and crash dumps and graphical shells. This shows the wide variety covered by AUUG members interests and needs, and more will follow in the future.

As many of you have noticed, we currently have lots of books coming for review. The current practice is to post a note to the mailing list [auug-books@ansto.gov.au](mailto:auug-books@ansto.gov.au) and the newsgroup [aus.org.auug](mailto:aus.org.auug) when we have new books available. 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) 717 9273, with your contact details and preferences. ❖



## The Mythical Man-Month: 20th Anniversary Edition

by Frederick P. Brooks, Jr.  
Addison-Wesley 1995,  
322 pages, \$42.95  
ISBN 0-201-83595-9

Reviewed by Adrian Booth Tandem Computers Pty Ltd  
<[abcc@DIALix.oz.au](mailto:abcc@DIALix.oz.au)>

The Mythical Man-Month - first published 20 years ago - was the first influential book on software project management. What on earth could it possibly have to tell us today? An awful lot! TMMM comprises 177 pages of distilled experience and common sense from someone with vast experience in the management of large software projects.

Depressingly, much of its content is still as applicable today as it was when it was written; an interesting statement in itself on how far we haven't progressed in our ability to create complex software. The individual statements in TMMM read like simple common sense

in many cases. This is simply because so many of our accepted ideas on large software projects derive directly from TMMM. For example, TMMM saw the first mention of *Brooks' Law* (Adding people to a late software project makes it later).

This book, however, is not just a reprint of TMMM - it is about half again as long as the original, with four new chapters. One of these is a reprint of the author's 1986 paper, *No Silver Bullet*, in which he argues that there is no new programming technique visible in the next 10 years that will lead to an order of magnitude improvement in programming productivity. As we enter the tenth year after the publication of that paper, it appears that he was right.

Subsequent chapters offer a retrospective on *No Silver Bullet*, summarise the propositions of the original TMMM, and finally give a detailed retrospective on the original TMMM in the context of software development today. The main lesson to draw from TMMM - and in particular from *No Silver Bullet* - is that software development is a two-stage process - conceptualisation followed by implementation.

The implementation phase would have to comprise 90% or more of the total effort - and we would be able to make that phase take zero time (!) - to see any order of magnitude improvements in productivity from such fads as object-oriented programming, development environments, CASE tools and so on. No-one with any experience with large software projects would argue that implementation is as significant a component as the conceptualisation and design required to deliver a system.

So the only way we are going to see substantial improvements in software development productivity is by targetting the specification and design of software. Today's software developers remind me of the person who lost a watch in the street, but looked for it under the street light "because it was easier to see" there. It is easier to see (and tackle) the problems of implementing software, and so that is the area currently the subject of the most research. But to see truly revolutionary improvements, we need to tackle the fundamentals of software conceptualisation, design and specification. The author discusses several techniques by which such improvements might be accomplished in the future.

Whether you program or manage programmers for a living, I suggest that this book is an absolute must for you to have read, and that once you have read it you will want it on your bookshelf permanently! Highly recommended. ❖

## World Wide Web Secrets

by Paul J. Perry  
IDG Books 1995,  
710 pages + CDROM, \$79.95  
ISBN 1-56884-456-5

*Reviewed by Andrew Wenn Victoria University of Technology  
(Footscray) <awenn@westgate.vut.edu.au>*

What a wonderful opportunity the opening up of Internet and the establishment of the World Wide Web has been to book publishers. Never have I seen so many books published on a topic in such a short time, some bad, many average, some good and the occasional excellent one. Unfortunately, "World Wide Web Secrets" is not in the excellent category but it is better than average if you are a beginner in this area. It is, however, difficult to see what relevance it will have to the audience of this publication as it has been written for DOS users and only rarely mentions UNIX and then the inference is that UNIX is something to be avoided unless you are technically competent, mad or both.

The 25 chapters of the book are divided into 4 sections with a fifth section containing 4 appendices.

Part 1 contains three chapters that give a brief introduction to the Web and the Internet and it is here that we strike the first of the so called secrets that this book promises to reveal - hence the use of the word Secret in the title. One such gem is that "it is important to remember what technology is: anything that makes something else more efficient" (page 46) something that I find neither particularly revealing nor agree with. Unfortunately most of the secrets are of a similar nature - another one revealed on page 49, is the meaning of the word acronym. However, the book should not be denigrated just because of the nature of the secrets it reveals, just be aware of the level of them if you consider purchasing.

Part 2 outlines methods of connecting to the WWW covering a variety of commercial Web access services, many of which will be of little use to Australian users, such as America Online, CompuServe and Prodigy. There are useful chapters on Netiquette and Searching the Internet, although, I felt the section on searching would benefit from an in depth discussion of the more advanced features offered by some search engines such as the use of regular expressions and full text searching. Ironically, and perhaps reflecting the authors true preferences, the chapter discussing Netscape is longer than the one discussing the use of the NetCruiser software that is supplied on the enclosed CD ROM.

To me, Part 3 was pretty much a waste consisting as it does of 100 pages of screen shots of web sites, mainly american, from organisations of various types - government, computer related business, other business and so on. A few examples of home pages would have sufficed to give readers a feel for the type of information available not 100 pages!

Part 4, entitled Publishing on the World Wide Web, has a chapter on setting up the Web server software supplied with the book (Windows HTTPD), a chapter discussing the type of hardware needed, then it launches into HTML. Whilst the first chapter provides some discussion of the basics of HTML, I have seen better presentations elsewhere and you should note that it does not cover Version 3.0 or the Netscape extensions to HTML. Another chapter covers HTML Forms - again at a very elementary level and does not mention the problem of case sensitivity with form methods. I mention this in particular because it is one problem that I struck when I first started using forms and can be very difficult to track down. A further chapter talks about design guidelines for home pages providing some useful hints for beginners but I felt some attention could have been paid to the various graphics formats available for displaying images and their pros and cons. The section on maintenance of home pages is woefully short considering how much attention has to be paid to ensure validity of links and keeping information up to date. Another chapter provides some ideas for various types of home page layouts and discusses the HTML templates that are provided on the CD. These are fairly elementary but should enable a tyro to get started.

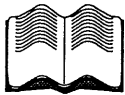
The penultimate chapter is on commercial uses of the Web and security considerations and gives a brief outline of Netscape's SSL and the W3 Consortium's SHTTP security developments. The final chapter in this section gives a very technology driven view of the Web's future really only providing the author with a chance to do some crystal ball gazing adding nothing to the actual content of the book.

As mentioned previously four appendices are contained in Part 5. The first of these provides information about the software on the CD - mainly a discussion of the menu options - and is for some inexplicable reason completely devoid of screen shots thus making it hard for anyone using the software to relate to the on-screen appearance of the products. The second is a fairly short list of U.S. based resources (magazines and vendors). A four page HTML quick reference comprises the third appendix and a fairly simplified glossary the fourth.

The index is quite long and detailed (Australia rates one mention!) the editor having seen fit to index even such things as the Netscape accelerator keys. There are a number of typos in the text especially in some of the URL's, which tends to detract from the overall feel of the text - none however present major problems.

The CD ROM contains a variety of software, all PC based, some of which is limited to use in North America. Since this is a journal specifically for UNIX users I see no point in reviewing the contents of the CD here.

If you are new to the World Wide Web and the Internet and a dedicated DOS user then this book may suit you. The style of writing did not appeal to me but then each to his/her own. Many of the secrets revealed are fairly trivial and I would question the use of the word secret in the title. It does provide a reasonable introduction for the beginner but I feel that to many people it would soon outlive its usefulness. Definitely a browse-before-you buy text. ❖



## "IRQ, DMA and I/O" (resolving and preventing PC system conflicts)

by Jim Aspinwall

MIS Press 1995,

268 pages + Diskette, \$49.95

ISBN 1-55828-456-7

Reviewed by Dave Horsfall <dave@fgh.oz.au>

This is the book I wished I'd had a few weeks back, when I experienced my first hardware conflict (my Unix box had been perfectly stable, until I added an Ethernet card that had ideas of its own); note that I have next to no experience with MS-DOS and its ilk, so the book will be reviewed in that context.

It is written in a somewhat breezy style, and it gathers together information that is usually scattered far and wide, such as a complete listing of the preferred addresses used by various hardware. Did you know that many Ethernet cards use 32 bytes for memory addresses, instead of the "normal" four? Well, I didn't ...

The early history of PCs is explained, showing us why things are the way they are, and the "Plug and Play" ("PnP") specification receives a fair coverage. No doubt this is old hat to DOS people, but in the

meantime the rest of us still wonder why something that is so fundamentally broken could be so successful (the author himself admits that the PC was designed by and for engineers). Common causes of conflict are covered, and there are many hints for DOS-oids (but of little use to Eunuchs).

Naturally, DOS, Windows, NT, and OS/2 receive coverage, but the only mention of Unix occurs in this howler on page 106: "We've had access to the Unix operating system for years. Unix is very powerful and works on more types of computers than any other operating system, but it is also very costly and complex to implement and use. There are many very attractive and effective graphical user interfaces for Unix, but they, too, are costly and complex. Even though Unix is almost universal across various systems, it has never been designed, packaged, or supported for use by the general public. Yet we've wanted to be able to use something with the high-performance, multitasking, interconnection features of an operating system like Unix." Even though the book was published in 1995, plainly the author has never heard of "costly" Linux, FreeBSD, or X ...

A 90mm floppy disk is included with the book, containing both "QAInfo" (a hardware configuration system) and "What's in that Box" (a cute animated tutorial). I had trouble with QAInfo; it hung regularly (yes, I tried the hints), and it did not see the Western Digital network card, so some more work is needed here. The installation instructions for "What's in that Box" need to be looked at, as it said to remove the self-extracting file without mentioning running it first ...

The book could have benefitted from some further proof-reading; for example, on page 23 we see modem commands "AT13" and "AT17" (instead of "ATI3" and "ATI7" respectively), and on page 226 the myth is perpetuated that a DX4/100 is a clock-quadrupled chip (it's not; it's a tripled 33MHz or a doubled 50MHz chip; Intel must have used a Pentium to assign those numbers).

Did I find it useful? Certainly, but with reservations (I wasn't looking for errors; they just leapt out at me). Would I buy it? Dunno; the list price is US\$24.95, which given the stranglehold on the Australian market by the publishing cartel, it could end up as anything here, and I certainly would not be happy paying AU\$50 for a book with errors and software that doesn't work. Would I recommend it? Yes, but with reservations. ❖

## PANIC! Unix System Crash Dump Analysis

by Chris Drake and Kimberly Brown  
SunSoft Press, Prentice Hall 1995,  
492 pages, Soft Cover with CD-ROM.  
ISBN 0-13-149386-8

*Reviewed by David Denton ANSTO <dtd@ansto.gov.au>*

Ever got the message "Segmentation fault (core dumped)" ? Was your very next command "rm core"? Ever wondered what was in those "core" files? Even worse, have you ever had a Unix system actually "Panic:" on you and felt powerless to do something about it?

If so, then this is a book for you. In my own case, I had just installed a new Sun Server and was preparing for the worst (Which did not eventuate I'm happy to report).

The text deals with how to proceed when a UNIX System, specifically a SUN system running SUNOS 4.1.x or Solaris 2.x, goes belly up. A semi tutorial approach is used with the initial chapters presenting issues such as: How to identify when a system has crashed, the difference between a panic and a hang, how to use savecore and how to generate a savecore, a wonderful introduction into how to crash your own system with appropriate warnings about "not doing this at home kids" and how to carry out basic problem determination after an initial crash.

The next couple of chapters cover the basics of how to drive some of the Unix debugging tools. The main tool of interest is adb. Other debuggers are briefly reviewed but the main emphasis is on adb. Having mastered the basic commands of adb we proceed to what the book refers to (quite correctly) as "the gory details". This includes advanced commands and macros, the macro chapters are used to identify and explore some of the structures that are manipulated within the kernel address space in the course of debugging. These chapters form part #1 of the book.

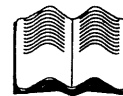
During this discussion some of the more interesting areas of the kernel are discussed and you feel yourself being drawn toward the real purpose of the book i.e. crash dump analysis.

Unfortunately, just at the point where you think you will be diving into a crashed Solaris kernel you take a major detour into part #2 of the book which entails: Assembly Language, SPARC architecture, the SUNOS/Solaris operating systems and SPARC hardware. This is necessary to understand the various case studies which round out the book.

The case studies form part #3 of the book and are the whole reason for this text. The authors take a look at some example problems in the style of a tutorial. The background detail found earlier now comes into its own as we delve into network problems, kernel logic errors, I/O problems and hardware faults.

The book comes with a CD-ROM containing useful programs and data. I did not have a chance to investigate this but it looks like it could be of considerable interest.

This book is very readable, considering the dryness of the subject matter, with the authors providing plenty of tips on when it would be a good idea to have a cup of coffee. The introductory material provides a good starting point for anyone interested in the insides of SUNOS/Solaris and the actual dump analysis may prove invaluable when one encounters a system that is not behaving the way you would like. ❖



## Porting UNIX Software

by Greg Lehey  
O'Reilly and Associates 1995  
ISBN 1-56592-126-7

*Reviewed by Glenn Huxtable Functional Software  
<glenn@fs.com.au>*

The book is subtitled 'From Download to Debug' and it's not kidding. It deals in great depth with everything you might want to know about unpacking, patching, compiling, installing, testing and debugging Unix software - and then some.

One thing this book does not deal with is writing portable software. The author does not specifically discuss design issues for portability, that's a whole other topic. This book focuses more on UNIX features affecting portability, and how to map from one feature to another when crossing platforms.

When I first chose to review a draft copy, I was actively involved in porting my company's product and was looking for helpful information on 'portability', and probably more specifically good design principles for portability.

My initial reaction to the first few chapters was that this book was about compiling software from the net, which I've been doing that for years. I put the book down. Coincidentally my focus was shifted from porting to support for a time.



A while later a real copy of the book arrived in the mail, prompting me to have another go. Don't do what I did - don't put the book down and write it off as just a book about building free software from the net. It starts out talking about software from the net, but that's only because the net is the most common source of UNIX application source code. There's more to this book than that.

The book is an in-depth introduction and reference for the newcomer who wants to understand all the in's and out's of how to build UNIX software on already ported platforms, and is a great reference for the more experienced user wishing to actually port the software to new platforms.

At 516 pages, it's not a light read, however it can easily be tackled in chunks. The book is in three sections, each section comprising a number of mostly self contained chapters that the reader can browse, or return to as a reference at any time.

The first section 'The Story of a Port' deals with just about everything you need to know about extracting, configuring, building, testing and debugging source archives. By the end of this section, there isn't much that an inexperienced reader won't know about the process of taking a source archive from the net, or off a CD-ROM, and processing it right the way through to a compiled and tested program ready to be used.

A sample of topics covered includes, archive file formats, common compression programs, identifying archive types, unpacking them, updating them with 'patch', configuring sources, controlling the build with make, understanding the compiler and compiler errors and warnings, common compiler problems, documentation, documentation formats and conventions, testing and debugging with gdb, truss and friends and finally, advice on how to report modifications so that you don't have to repeat the hard bits next time you download the same software.

The second section, the 'UNIX Flavor Guide' is a comprehensive study of the differences between UNIX systems that effect portability of programs, including comparisons of SCO, SVR3, SVR4, BSD, SunOS and IRIX functionality.

Starting with the obvious hardware dependencies of data types, pointer size, byte order and data alignment the author then moves into general kernel dependencies, such as IPC mechanisms, sockets, FIFO's, process groups, setuid functionality and the many flavours of 'wait'.

The chapter on 'Signals' is a most detailed coverage of SVR4, BSD and POSIX signals and signal handlers.

The next chapter is 'File Systems' which describes the common file system types and most, if not all, system calls and function libraries to do with accessing and manipulating file-systems.

This is followed by 'Terminal Drivers'. a comparison of Old, SVR4 and BSD terminal drivers, ioctl's, termio and termios.

'Timekeeping' looks at the problems of daylight savings times and international time formats, as well as time keeping and manipulation functions.

Just when you head is about to burst, the author dips into 'Header Files' for a brief look ANSI and POSIX header files, and some of the problems of different platforms using different include files, or different structures for the same effect.

If that isn't enough, the chapter on 'Function Libraries' is a fairly complete reference on just about every C function library, and their various differences across platforms.

The second section wraps up with chapters devoted to comparisons of K&R and ANSI C compilers, and object file formats, assembler and linker conventions.

The third section consists of appendices of comparative UNIX data types, C compiler options and assemble directives and options.

Throughout, the author compares features available under SCO, SVR3, SVR4, BSD, IRIX with occasional reference to Linux. The second section in particular contains numerous tables showing which functionality is available under which system. This is a serious text for anyone wishing to port software to new platforms and an invaluable reference for comparative functionality.

This book is certainly up to the expected standard of O'Reilly books. The author clearly knows his stuff, not only providing a wealth of factual information, but also many years of experience and tips to the reader.

'Porting UNIX Software' has certainly earned it's place on my bookshelf. Highly recommended if you're into portability across UNIX platforms. ❖



## Desktop KornShell Graphical Programming

by J. Stephen Pendergrast, Jr.  
Addison-Wesley Professional Computing Series 1995,  
840 pages, \$56.95  
ISBN 0-201-63375-2  
*Reviewed by Jamie Honan <jhonan@mpx.com.au>*

This book describes dtksh (Desktop KornShell), an amalgam of KornShell and tools for the X Window system, specifically Motif, the Xlib and Xt (X Toolkit). The author has added another X layer, he calls it XU for X Utilities, in order to make using this tool simpler!

The author's rationale for such a beast is simple: it has been blessed by a standards body. "... a new initiative swept across the Unix industry - the Common Open Software Environment (COSE)". Part of this set of specifications is called Common Desktop Environment or CDE. Dtksh was specified as part of CDE as the scripting language to use for writing X Window system applications.

Rival scripting systems are dismissed early on. Tcl/Tk is a rival toolkit that, unlike dtksh, is free and freely available (not to worry, every Unix vendor will have dtksh. Use Linux or Net/Free BSD? Bad luck!). Tcl/Tk gets a nod for its "simplicity" and the fact its source is "of high quality and freely available". Tcl/Tk doesn't include the kitchen sink, "so problems arise when using Tcl/Tk for industrial strength programming in the CDE environment".

Taking the author at his word, is this a useful book if you had to use dtksh? The answer is yes, with some qualifications.

You'd have to be an experienced C or shell programmer, and the second section of the book goes heavily into Xlib and Xt calls that really would require prior experience or access to other material.

The KornShell is explained quite well for programmers who are capable in other shell languages.

The first section of the book deals very well with Motif basics, although again some shielding is done by the author through his XU layer. Traditional Motif concepts, widgets and gadgets are explained in detail.

The author develops two applications, a cheque book system and a stock chart display application, the full source for these takes 124 pages.

There are a number of chapters that concentrate on common programming tasks, rather than explain

underlying widgets or concepts. Examples include - feedback during long processes, message catalogues.

The chapter on using the interface to C is not really fully fleshed out. For a very large or specialised application, the use of C code will be necessary either for tweaking performance at a weak spot or for specialised interface (if not both).

In summary, if you're forced to use dtksh you'll probably need this book. If you're looking for an easy way to do GUI programming under X, you might find Tcl/Tk and some of the myriad extensions (for example Tcl/Motif if you have to / want to use Motif) will be more useful in both the short and long terms.❖



## The Student's Guide to Doing Research on the Internet

by Dave and Mary Campbell  
Addison Wesley 1995,  
349 pages, \$26.95  
ISBN 0-201-48916-3  
*Reviewed by Jon Wright Guru Software Services  
<jon@UNS.com.au>*

The title of this book attracted immediate attention, probably as a result of the media hype surrounding the Internet combined with an obviously acceptable excuse for spending additional time (and money). Colleagues that saw the book arrive in the post wanted to borrow it immediately. Unfortunately, the title is probably the best thing about the book.

Approximately half of the book is a simple introduction to the various common internet tools. There are specific sections on: FTP, Gopher, Veronica, Jughead, WAIS, and the WWW.

Generic introductory chapters also discuss telnet and how to get an Internet connection. The authors assume that there are basically three choices for a connection: an educational link, a freenet service or a commercial provider (Delphi, NetCom are mentioned but not AOL or CompuServe). This is a very US-centric view of the Internet and it really ignores the huge non-American audience. Australian readers after a connection should replace the first eight chapters with an alternative such as "OzInternet" (Goodheart & Crawford).

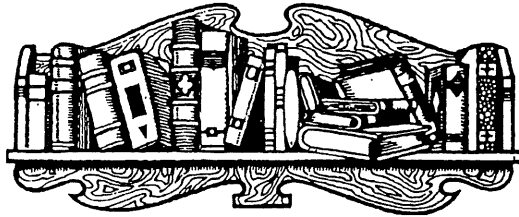
The remainder of the book consists of chapters covering various study areas including:

- The Humanities
- Management
- School Resources
- Computer Science
- Education and General Reference
- Engineering
- Geography and Travel
- History and Political Science
- Legal Resources
- Literature
- Medicine and Health
- The Sciences and Mathematics

Each of these chapters contains a series of paragraphs with a URL for the specific resource and then a few sentences describing the content at that site. The problem with this approach is that the information can become dated very quickly. The authors tackle this problem by suggesting that updates are available from

the Addison-Wesley web page. (This assumes that you have managed to get a connection....). Readers interested in a broader range of topics would probably find "The Whole Internet" (Krol) more useful.

Overall, I was very disappointed in the book. It did not convince me to change from my standard recommendation of Internet books for the expert or beginner: "OzInternet" and the "Internet Resource Guide". Overall I cannot see a real place for this book. It is not detailed enough for novices and it is too quickly outdated for experts. In conclusion, I cannot really recommend this book to an Australian audience. ❖



WAUG news:  
**From the  
Western Front**

Edited by Tom Hallam  
<thallam@geol.uwa.edu.au>

---

### .Meeting information

WAUG meets at the Freeway Hotel, 55 Mill Point Road, South Perth. We meet at 6:15pm on the third Wednesday of each month.

Our meetings are advertised in the Diary column of the Computers section of Tuesday's *West Australian*.

If you need further information about the next meeting, please contact Mark or one of the committee.

#### **SPEAKERS ARE NEEDED**

especially ones who can actually commit to giving a talk on a certain date! So if you can give a talk, or know someone who can, please let us know. Mark (our meeting organiser) cannot produce them out of thin air.

---

### WAUG Email Aliases, Newsgroups and Web Page

WAUG has the following mail aliases on uniwa.uwa.edu.au:

**waug-membership** - for membership enquiries

**waug-chair** - our Chairperson

**waug-meetings** - our meeting organiser

**waug-secretary** - our Secretary

**waug-newsletter** - for newsletter contributions or enquiries

**waug** - for general correspondence (will be read by the Secretary, as a paper letter would be).

So, for example, you may send general correspondence to [waug@uniwa.uwa.edu.au](mailto:waug@uniwa.uwa.edu.au).

Check us out on the World Wide Web at:

<http://www.auug.org.au/auug/waug/waug.html>  
(thanks Canberra AUUG).

Also see the newsgroups [wa.waug](mailto:wa.waug) and [aus.org.auug](mailto:aus.org.auug) for announcements and discussion.

---

### Committee Contact Details

#### Office-bearers:

- Chair : Adrian Booth 321 9111  
[booth\\_adrian@tandem.com](mailto:booth_adrian@tandem.com)
- Treasurer: Patrick Ko 483 8111  
[pko@DIALix.oz.au](mailto:pko@DIALix.oz.au)
- Secretary: Major 357 5076  
[major@yarrow.wt.com.au](mailto:major@yarrow.wt.com.au)

#### Ordinary committee members:

- Mark Baker (Meeting Organiser) 491 6081  
[baker@telecomwa.oz.au](mailto:baker@telecomwa.oz.au)
- David Buck  
[dbuck@ncc.telecomwa.oz.au](mailto:dbuck@ncc.telecomwa.oz.au)
- Luigi Cantoni 474 3700  
[lui@DIALix.oz.au](mailto:lui@DIALix.oz.au)
- Don Griffiths 351 7691  
[griffith@cs.curtin.edu.au](mailto:griffith@cs.curtin.edu.au)
- Tom Hallam 380 2665 (AUUGN Sub-editor)  
[thallam@geol.uwa.edu.au](mailto:thallam@geol.uwa.edu.au)
- Glenn Huxtable 328 8288  
[glenn@fs.com.au](mailto:glenn@fs.com.au)
- Janet Jackson 272 5061  
[janet@DIALix.oz.au](mailto:janet@DIALix.oz.au)
- James Patton (Meeting Reporter)  
[jrp@mrwa.wa.gov.au](mailto:jrp@mrwa.wa.gov.au)

A publicity officer has not yet been appointed. Please volunteer. (Adrian is acting in the interim, but don't you think he has enough to do already?)

---

### For Systems Administrators: Local SAGE-AU Meetings

The WA Regional Group of the Systems Administrators Guild of Australia (SAGE-AU) meets on the *first Tuesday of each month at 6pm, in room G3 at the Alexander Library* (Note the change).

For more information, please contact any of the following:

- regional group chair,  
Janet Jackson <[janet@DIALix.oz.au](mailto:janet@DIALix.oz.au)>,  
(09) 272 5061
- meeting organiser  
Mike Horton <[mikandfi@DIALix.oz.au](mailto:mikandfi@DIALix.oz.au)>,  
(09) 479 8424
- Tom Hallam <[thallam@geol.uwa.edu.au](mailto:thallam@geol.uwa.edu.au)>,  
(09) 380 2665

For information about SAGE-AU in general, you may also look at

<ftp://ftp.sage-au.org.au/pub/SAGE-AU> and  
<http://www.sage-au.org.au:8080/> ❖

---

AUUG Canberra news:

## Canberra chapter

John Barlow, 019 935477,  
 <cauug.secretary@auug.org.au>  
 Secretary, Canberra Chapter of AUUG Inc.

### March Monthly Meeting

Topic: An Introduction to Java

Presenter: Jan Newmarch

When: 7:30 pm for 8:00 pm, Tuesday, 12th March 1996

Where: Manning Clark Theatre 5 ANU Campus

Jan Newmarch will be introducing us to Java. All going well, this will include real, live demos (there you go, I've put a curse on it already :-)

Java is a language that enables portable program performing some very neat graphic work and some actually useful functionality. It is rumoured to be the next wave that will wash over the Internet. It should firmly embed the World Wide Web into many organisations (and homes).

### The INTERNET Project

If you are an AUUG member in the Canberra Chapter you can get email and news access to the Internet, and there is no cost (so long as you remain a financial member). This is provided by a dialup service which has been recently updated and has a direct IP connection to the Internet. Full IP access to the Internet can be obtained at extra cost (\$90 for 300 hours, approximately). If you are interested in this service, please contact John Barlow (mobile: 019 935477) to have a chat about it.

We have a Linux box and a FreeBSD box available via the same dialup service, so if you want to examine these great (free, full source) UNIX implementations, here is a prime opportunity ! Existing users who wish to use these boxes need to email [linux@canb.auug.org.au](mailto:linux@canb.auug.org.au) or [freebsd@canb.auug.org.au](mailto:freebsd@canb.auug.org.au) to have an account setup.

### Meetings organiser wanted:

If you want a specific topic discussed at a future meeting, or want a specific UNIX presentation made, please contact John Barlow (contact details at the end of this message).

### Volunteers needed:

We need people to volunteer to handle some of the workload. You may have noticed that some of the committee have become very scarce recently, and we need extra people to help out. One example of a job looking for a good home is somebody to look after membership details for the dialup project, so people don't get their accounts accidentally suspended ...

### Coming Events:

March 12

Java talk, 7:30 pm Manning Clarke Theatre 5 ANU

April 9

General meeting TBA

May 14

General meeting TBA

June 11

Annual General Meeting

❖

# Rules of AUUG Incorporated

As amended at 1 July 1995

## Name

- 1 The incorporated association shall be known as the AUUG Incorporated, abbreviated hereinafter to AUUG.

## Definitions

- 2 (1) In these rules, unless otherwise stated: "he", "him" and "his" shall also be construed to mean "she", "her" and "her" respectively; "The Act" means the Associations Incorporation Act 1981 (Vic); "Financial year" means the period from 1 June to 31 May; "General Committee Member" shall mean a general member of the Management Committee; "mail" shall imply the transmission of information in written or printed form, first-class pre-paid, via the general post or public or private courier service; "unfinancial member" shall mean any member whose most recent term of membership has expired and who has not yet paid the subscription for the next twelve month period; "voting member" shall mean any member entitled to cast a vote.  
(2) In these Rules, a reference to the secretary of the AUUG is a reference: (a) where a person holds office under these Rules as Secretary of the AUUG, to that person; and (b) in any other case to the Public Officer of the AUUG.  
(3) Words or expressions in these rules shall be interpreted in accordance with, and subject to, the Act as in force from time to time.  
(4) If any doubt arises as to the proper construction or meaning of any clauses in these Rules, the decision of the Management Committee thereon shall be final and conclusive provided such decision be reduced to writing and recorded in the minutes of a meeting of the Management Committee.

## Aims

- 3 The aims for which the AUUG is established are to promote knowledge and understanding of Open Systems including but not restricted to the UNIX system, networking, graphics, user interfaces and programming and development environments, and related standards.

For the furtherance of these aims and to achieve its purposes, the AUUG may carry out any or all of the following activities: conduct technical meetings, conferences, discussion groups, panels, lectures and other types of meeting; prepare and distribute a newsletter and other publications; collect software and distribute said software to its members for their use; verify licenses of members for the purposes of administering the services of the AUUG; subscribe to or cooperate with or affiliate with or amalgamate with other associations formed elsewhere with similar aims; accumulate assets; and establish and promote other activities not included in the above list consistent with its aims for the benefit of its members.

## Eligibility for membership

- 4 Any individual or organisation who subscribes to the aims of the association, and who agrees to be bound by its rules and regulations and who has not been previously expelled from the association shall be eligible to join the AUUG.
- 5 An application for membership shall be in writing on the form approved by the Management Committee and shall provide such information as shall from time to time be prescribed by the Management Committee.
- 6 (1) Membership shall become current from the date on which a valid membership application accompanied by payment of the appropriate entrance fee plus annual membership subscription is received by the Secretary. The initial membership period will extend from this date until the first renewal date (June 30 or December 31) no less than twelve months from the beginning of the membership period.  
(2) Upon completion of the initial membership period and any subsequent periods, membership may be renewed for a further period of twelve months by payment of an additional annual membership subscription.
- 7 (1) There shall be four classes of members: Ordinary members, Institutional members, Student members and Honorary Life Members.  
(2) Any natural person who is eligible to be a member may become an Ordinary Member.  
(3) Any person or organisation who is eligible to be a member may become an Institutional Member.  
(4) Any full-time student who is eligible to be a member may become a Student Member.  
(5) Any person who is an Ordinary Member of at least five years standing and who has rendered special services to the AUUG may be elected via a ballot of the members as an Honorary Life member.

(6) If before the first day of May the Secretary receives a petition from at least twenty voting members requesting the election of a member of the AUUG to the position of Honorary Life Member, then he shall arrange a ballot of the membership on this question to be conducted in conjunction with the annual election of Officers and General Committee Members.

- 8 All Ordinary, Institutional and Honorary Life Members whose membership is current shall be entitled to cast a vote.

## Membership subscriptions and fees

- 9 The Management Committee shall determine before the commencement of each financial year a scale of fees for entrance to the AUUG, and for annual subscriptions for each class of members to be applied during that financial year.

## Register of members

- 10 (1) The Secretary shall keep and maintain a register of members in which shall be entered the full name and address of each member and the register shall be available for inspection by members at the address of the Public Officer.
- (2) Nothing in the previous subsection shall entitle any member to make a copy of the register of members, except with the permission of the Management Committee, and on such terms and conditions as the Management Committee shall from time to time determine.

## Termination of membership

- 11 (1) A member may resign his membership at any time by giving notice in writing to the Secretary. No member who resigns shall have any claim for a refund of subscriptions paid.
- (2) A member who has been unfinancial for more than two calendar months shall be deemed to have resigned his membership, and shall no longer be entitled to any privileges enjoyed by members.
- (3) Former members who have resigned will be entitled to rejoin the AUUG on the same basis as new members joining the AUUG.

## Expulsion of members

- 12 Upon receipt of a petition so requesting from twenty or more members, or half the membership, whichever is less, the Management Committee shall call upon any member to explain any alleged misconduct, and the Management Committee shall have power to suspend or expel any member who in its opinion has either been guilty of misconduct

or has acted prejudicially to the interests of the AUUG or who has wilfully infringed any of the Rules of the AUUG.

## General meetings

- 13 The Annual General Meeting shall be held within the second half of each calendar year. The date and general location of each Annual General Meeting shall be determined at the preceding Annual General Meeting but either the date or location or both may be changed by the Management Committee if it proves impossible or highly inconvenient to meet at the location previously selected or on the date previously selected.
- 14 An ordinary general meeting of the AUUG shall be called by the Management Committee in conjunction with any technical meeting or conference or other function where attendance by a quarter or more of the voting members is expected by the Management Committee.
- 15 (1) Written notice of the time and place for each meeting and its agenda shall be mailed to each voting member of the AUUG at least four weeks before the date of the meeting.
- (2) Business conducted at such meetings shall be confined to matters included in the written agenda, reports from Officers, and resolutions instructing the Management Committee to conduct a formal ballot of the membership on matters of substance. Such resolutions shall not be binding on the Management Committee unless the meeting was attended by at least twenty voting members, or half the membership, whichever is less, and the resolution was supported by at least three-quarters of the members voting.
- (3) All voting members shall be entitled to cast one vote.
- (4) Any voting member may award his proxy to another voting member for the period of a single General meeting providing he so notifies the Secretary in writing at least 24 hours before the appointed time of commencement of the meeting.
- 16 (1) Upon receipt of a petition so requesting from twenty or more members, or half the membership, whichever is less, the Secretary shall call an Extraordinary General meeting of the AUUG for a date no later than two calendar months after receipt of the petition.
- (2) The business of the meeting shall be confined to matters described in the petition and to other matters specifically provided for in these rules and recorded in the written agenda sent to all members by mail at least four weeks before the date set for the meeting.

## Rules of AUUG Incorporated

(3) If the Management Committee does not cause a special general meeting to be held within two months after the date on which the petition is sent to the address of the Secretary, the members presenting the petition or any of them, may convene a special general meeting to be held not later than four months after the date of that petition.

(4) A special general meeting convened by members in pursuance of these rules shall be convened in the same manner as nearly as possible as that in which those meetings are convened by the Management Committee and all reasonable expenses incurred in convening the meetings shall be refunded by the AUUG to the persons incurring the expenses.

17 (1) For each general meeting, the quorum shall be fifty members personally present and entitled to vote.

(2) If within an hour after the appointed time for the commencement of a general meeting, a quorum is not present, the meeting if convened upon the requisition of members shall be dissolved.

(3) In other cases the meeting shall be deferred to a place and time determined by the Management Committee. If that meeting is to be at the same location on the following day then notice of the meeting may be given by posting a notice at the location specifying the time of the meeting and the business to be conducted no less than four hours before the time of the meeting. In any other case notice shall be given as for any other General Meeting.

18 At all general meetings of the AUUG the Chair shall be taken by the President, or in his absence, the Vice President, or in his absence by a member elected by the meeting.

### Officers

19 The Officers of the AUUG shall be: the President; the Vice-President; the Secretary; the Treasurer; the Returning Officer; and the Assistant Returning Officer.

### Management committee

20 (1) The management and control of the business and general affairs of the AUUG shall be vested in a Management Committee consisting of nine elected members and the Immediate Past President.

(2) The elected members of the Management Committee are the President, the Vice-President, the Secretary, the Treasurer, and five General Committee Members.

(3) The position of Immediate Past President shall be held by the person who held the position of President at the end of the previous term of office, providing they have not been elected to any other position and agree to accept this position. Otherwise, the position of Immediate Past President remains vacant for the entire term of office.

### Elections

21 (1) The election of Officers and General Committee Members shall be by a postal ballot held annually.

(2) Nominations for each position shall be received by the Secretary up until the fourteenth day of April each year. Each nomination must be in writing, must name the position or positions sought, must be signed by at least three voting members, and must be countersigned by the nominated member who must be a financial voting member of the AUUG.

(3) Where only one valid nomination is received for a particular position by the close of nominations, the nominee shall be declared elected forthwith, and no ballot for that position shall be held.

(4) Any position for which no nomination is received, or which remains unfilled after the election has been conducted, shall be considered as a vacancy on the Management Committee, and handled as specified in these rules.

(5) On or before the first day of May, the Secretary shall advise the Returning Officer of all valid nominations received, and if a ballot is required shall advise him of a date no later than the fifteenth day of May for the ballot for all contested positions, and shall provide him with a list of voting members.

(6) While any Ordinary Member may be nominated to more than one office or position, no person shall be elected to more than one position. Ballots shall be determined in the following order: for President, for Vice-President, for Secretary, for Treasurer, for General Committee Members, for Returning Officer, and lastly for Assistant Returning Officer.

(7) All voting members shall be entitled to cast one vote.

22 The term of office for all Officers, General Committee Members and the Immediate Past President shall be for one year, from July 1 to June 30.



## Vacancies on the management committee

- 23 (1) The position of any General Committee Member shall be vacated if the member fails to attend any Management Committee meeting without furnishing a satisfactory explanation as to the cause of his absence, and if the Management Committee resolves that his office be vacated, or if the member ceases to be a member of the AUUG
- (2) Should the office of President be vacant, the Vice President shall become President, and the office of Vice President shall become vacant instead. If for this reason, or for any other, at any time any of the other principal Officers (Vice President, Secretary or Treasurer) be unable to continue in office for any reason, then the Management Committee shall appoint one of their number to the vacant office.
- (3) Should the position of Immediate Past President become vacant, it remains vacant for the remainder of the term of office.
- (4) Should a vacancy occur among the other Officers, or among the General members of the Management Committee, then the Management Committee shall appoint an Ordinary Member of the AUUG to fill the vacancy.
- (5) Should a vacancy occur as the result of the creation of a new position, the vacancy shall be filled as specified in these rules.
- (6) The Management Committee shall make the approval of such appointments an order of business for the next General Meeting of the AUUG if any such meeting will be held before the next election of Officers and General Committee Members.

## Management committee meetings

- 24 (1) The Management Committee shall meet formally at least twice per year.
- (2) Notification of time, place and agenda for each meeting shall be made in writing to each member of the Committee by the Secretary at least four weeks in advance.
- (3) All members of the AUUG are entitled to be present at such meetings, and may speak when invited by the Chairman, but only members of the Management Committee may vote.
- 25 At meetings of the Management Committee the President shall take the chair, or in his absence, the Vice President, or in his absence a member of the Management Committee elected by the meeting.
- 26 The quorum for such meeting shall be five. If a quorum is not present at the nominated time for the start of the meeting, the commencement of the

meeting may be delayed for up to one hour, and if at that time a quorum is still not present the meeting shall be dissolved.

- 27 Resolutions of the committee shall require a simple majority of the members present and voting. The chairman shall have a casting vote in the event of a tie.

## Distribution of income

- 28 The income and property of the AUUG however derived shall be applied solely towards the aims and purposes of the AUUG as set out in these Rules, and no portion thereof shall be paid or transferred directly or indirectly by way of dividend to any member of the AUUG at any time.
- 29 The AUUG shall not appoint a person who is a member of the Management Committee to any office in the gift of the association to the holder of which there is payable any remuneration by way of salary, fees or allowances.
- 30 Notwithstanding the previous section the AUUG may compensate the reasonable expenses actually incurred by any member in the conduct of the business of the AUUG under the direction of the Management Committee.

## Chapters

- 31 (1) Ten or more members of the AUUG may petition the Management Committee to form a chapter of the AUUG.
- (2) General rules for the organisation, operation, obligations and privileges of chapters shall be as resolved by the Management Committee or the membership as a whole from time to time.
- (3) Each chapter shall appoint a chapter committee consisting of at least a Chapter Chairman and a Secretary/Treasurer.
- (4) The chapter committee may convene meetings consistent with the aims of the AUUG, but may not enter into any financial commitments on behalf of or in the name of the AUUG except with the written approval of the Management Committee.

## Affiliation or amalgamation with other organisations

- 32 The Management Committee may at any time seek or discuss the possibility of affiliation or amalgamation with any other organisation whose aims are similar to or compatible with those of the AUUG. No agreement for affiliation or amalgamation may be finalised until the matter has received the assent of three-quarters of the members voting in a postal ballot.

## Dissolution of the AUUG

- 33 (1) Upon receipt of a petition requesting the dissolution of the AUUG from twenty or more members, or half the membership, whichever is less, the Secretary shall arrange for the question to be put to the membership by ballot no later than one month after the date that he receives the petition.
- (2) If three-quarters of the members voting agree, the AUUG shall be dissolved.
- (3) If upon the dissolution of the AUUG there remains after satisfaction of all its debts and liabilities any property whatsoever, the same shall not be paid to or distributed among the members or Chapters if any, but shall be given or transferred to some public educational institution, or other institution to be determined at or before the time of dissolution by resolution of the membership.

## Changes to the rules

- 34 Changes to these Rules may be initiated at the request of a General meeting, or by the Management Committee. All proposed changes must be approved by a three-quarters majority of the votes received in a postal ballot of the members before having effect.

## Rights of members

- 35 (1) Each member shall be entitled to attend all meetings of the AUUG, including meetings of the Management Committee, provided any prescribed attendance fee is paid.
- (2) Each member shall be sent a copy of the association's newsletter.
- (3) Each member entitled to vote in a ballot shall be sent notice in writing of all ballots and copies in writing of the annual reports of the Secretary and Treasurer.

## The Secretary

- 36 (1) The Secretary shall furnish to the Returning Officer a complete list of all voting members whenever this is required for the conduct of a ballot.
- (2) The Secretary shall keep or cause to be kept full and correct minutes of all resolutions and proceedings at General meetings and Management Committee meetings of the AUUG.
- (3) The Secretary shall conduct correspondence on behalf of the AUUG.

- (4) The Secretary shall, during his last month of office, prepare a written report on the state of the affairs of the AUUG for distribution to the membership.

## The Treasurer

- 37 (1) The Treasurer shall keep or cause to be kept correct accounts and books and records showing the financial affairs of the AUUG.
- (2) The Treasurer shall notify the President and Secretary in writing of the usual location of said accounts, books and records whenever this location is changed.
- (3) The Treasurer shall receive all fees and subscriptions and all other monies on account of the AUUG and provide receipts for the same. The Treasurer shall deposit all monies received into a bank account maintained by the AUUG.
- (4) The Treasurer shall receive accounts for payment for services rendered to the AUUG, and as directed by the Management Committee arrange for payment from the AUUG's account.
- (5) The Treasurer shall, during his last month of office, prepare or cause to be prepared a written report on the financial affairs of the AUUG for distribution to the membership.
- (6) The accounts and books referred to in sub-clause (1) shall be available for inspection by members.

## Funds

- 38 The funds of the AUUG shall be derived from entrance fees, annual subscriptions, donations and such other sources as the Management Committee determines.
- 39 (1) Signing Officers for the AUUG's accounts shall be the President, the Vice President, the Secretary, the Treasurer and one other General Committee Member chosen by the Management Committee.
- (2) All cheques, drafts, and other orders for payment of money out of the funds of the AUUG, if for less than a limit established by the Management Committee, may be signed by only one Signing Officer.
- (3) For other amounts, each such instrument must be signed by at least two Signing Officers.

## Seal

- 40 (1) The Common Seal of the AUUG shall be kept in the custody of the Secretary.

(2) The Common Seal shall not be affixed to any instrument except by authority of the Management Committee and the affixing of the Common Seal shall be attested by the signatures either of two members of the Management Committee or of one member of the Management Committee and the Public Officer of the AUUG.

## Execution of Contracts

41 The Management Committee, except as otherwise provided in these Rules, may prospectively or retroactively authorise any Officer or member of the AUUG to enter into any contract or execute and satisfy any instrument, and any such authority may be general or confined to specific instances, except that any contract whose dollar value exceeds an amount predetermined by the Management Committee must be specifically authorised in advance by the Management Committee.

## Voting

42 (1) All voting by the members with respect to the election of Officers and General Committee Members, with respect to the election of Honorary

Life Members, with respect to changes to these Rules, and all other substantive matters shall be conducted by postal ballot.

(2) Every voting member of record as of the date of entry of a ballot into the mails shall be entitled to vote in the ballot.

(3) On all questions to be put to a ballot, the Secretary shall designate a date for the ballot to be placed in the mails, and the due date shall be four weeks after that date.

(4) The Returning Officer shall nominate the address to which voters shall return completed ballot papers by mail.

(5) A ballot will not be counted if it is received after the due date or if the ballot paper does not comply with the instructions printed on it.

(6) The ballots will be received by the Returning Officer, and counted by him and the Assistant Returning Officer.

(7) The Returning Officer shall report the result of the ballot in writing to the Secretary no later than two weeks after the due date.

(8) The formal procedures of voting shall be determined from time to time by the Management Committee.

AUUGN Volume 16 Index  
**Book Reviews**  
**(Sorted by author)**

Compiled by Stephen Prince <S.Prince@clcs.com.au>

Author	Title	Reviewer	Iss.
<unknown>	SCO Open Desktop / SCO Open Server - Graphical Environment Administrator's Guide	Tony Beresford	2
Ables, Robert King	The Keys to Successful UNIX System Management	Frank Crawford	4
Angell, David and Brent Heslop	The Internet Business Companion	Giles Lean	3
Chapman, Brent and Elizabeth Zwicky	Internet Security Firewalls	Warren Tomey	5
Cline, Marshall P. and Greg A. Lomow	C++ FAQs	Greg Bond	3
Comer, Douglas E.	The Internet Book	Craig Macbride	1
Comer, Douglas E. and David L. Stevens	Internetworking with TCP/IP Volume II: Design, Implementation, and Internals (Second Edition)		2
Corrigan, Peter and Mark Curry	ORACLE Performance Tuning	Mark White	2
Curry, David	Systems Programming for UNIX	Michael Haldey	5
Dowd, Kevin	High Performance Computing	Jagoda Crawford	2
DuBois, Paul	Using csh & tcsh Work Faster, Type Less		4
Ebbs, Geoff and Jeremy Horey	The Australian Internet Book	Phil McCrea	4
Flanigan, David	Motif Tools	David J. Hughes	1
Frisch, Aeleen	Essential System Administration - 2nd Edition	David Baldwin	6
Garfinkel, Simson	PGP: Pretty Good Privacy	Lawson Hanson	1
Goodheart, Berney and Frank Crawford	OzInternet: A guide to connecting to the Internet in Australia	Jon Wright	5
Goodheart, Berny and James Cox	The Magic Garden Explained Solutions Manual	Adrian Booth	5
Hu, Wei	OSF Distributed Computing Environment: DCE Security Programming	Michael Paddon	4
Hunt, Craig	Networking Personal Computers with TCP/IP	Craig Macbride	4
Johnson, Eric and Kevin Reichard	Advanced X Window Applications Programming, Second Edition	Michael Haldey	3

Kirch, Olaf	The Linux Network Administrator's Guide	David Conran	1
Kurani, Bharat	Applied UNIX Programming Volume 1	Greg Black	3
Libes, Don	Exploring Expect	Janet Jackson	2
Libes, Don	Exploring Expect	Jon Wright	4
Liu, Cricket, Jerry Peek, Russ Jones, Bryan Buus, and Adrian Nye	Managing Internet Information Services	Lawrie Brown	3
Loukides, Mike	Programming with GNU	Alex Kowalenko	6
Montgomery, John	The Underground Guide to UNIX: Slightly Askew Advice from a UNIX Guru	Wayne Bell	3
Mui, Linda and Valarie Quercia	X User Tools	Andrew Wenn	1
Nemeth, Evi, Garth Snyder, Scott Seebass, and Trent R. Hein	UNIX System Administration Handbook, Second Edition	Frank Crawford	3
Neou, Vivian	Internet CD	David J. Hughes	1
Neumann, Peter G.	Computer Related Risks	Adrian Booth	1
Newham, Cameron and Bill Rosenblatt	Learning the Bash Shell	Kate Lance	5
R.Cheswick, William and Steven M. Bellovin	Firewalls and Internet Security: Repelling the Wily Hacker	Danny Yee	
Radin, David	Building a Successful Software Business	Zoltan Somogyi	1
Salus, Peter H.	A Quarter Century of UNIX	Danny Yee	1
Salus, Peter H.	Casting the Net: From ARPANET to INTERNET and Beyond...	Michael Usher	5
Till, Dave	Teach Yourself PERL in 21 Days	J. Wright	5
Walsh, Norman	Making TeX Work	Jagoda Crawford	6
Welsh, Matt and Lar Kaufman	Running Linux	Ian Crankanthorp	6

AUUGN Volume 16 Index

# Reports (Sorted by author)

Compiled by Stephen Prince <S.Prince@clcs.com.au>

Author	Title	Issue	Page	Month
<unknown>	Canberra Chapter	2	25-26	April
Barlow, John	Canberra Chapter	6	29	December
Dauids, Enno	AUUG Inc - Victorian Chapter	1	35-36	February
Hallam, Tom	From the Western Front	6	26-28	December
Jackson, Janet	From the Western Front	5	34	October
Jackson, Janet	From the Western Front	4	26-28	August
Jackson, Janet	From the Western Front	3	17-18	June
Jackson, Janet	From the Western Front	2	25	April
Jackson, Janet	From the Western Front	1	35	February
Patton, James	WAAUG Meeting reports	5	35-37	October
White, Mark	Qld. Chapter Report	5	38	October
White, Mark	Queensland Chapter Update	4	28-29	August
White, Mark	Queensland Chapter: Summer Technical Conference	2	26	April

## AUUGN Volume 16 Index

**Papers (sorted by title)**

Compiled by Stephen Prince <S.Prince@clcs.com.au>

- Paddon, Michael, "APWWW/AUUG '95 - The Presidents View," AUUGN, vol. 16, no. 5, pp. 11-12, AUUG Inc., Melbourne, VIC, October 1995.
- Jackson, Janet, "Attention to detail!," AUUGN, vol. 16, no. 6, pp. 17-21, AUUG Inc., Melbourne, VIC, December 1995.
- Crawford, Frank, "AUUG95-APWWW95 wrapup," AUUGN, vol. 16, no. 5, p. 16,21, AUUG Inc., Melbourne, VIC, October 1995.
- Ching, Keith Lewis, Kathy, "AUUG Conference Report," AUUGN, vol. 16, no. 2, pp. 16-21, AUUG Inc., Melbourne, VIC, April 1995. \*\*\*\* Reprint from AUUGN v9n5 \*\*\*\*
- Bishop, Jeremy, "Behind the Web at <http://www.auug.org.au>," AUUGN, vol. 16, no. 3, pp. 7-11, AUUG Inc., Melbourne, VIC, June 1995.
- Sheehan, Kevin, "A Better Way to Access Files - Mapping," AUUGN, vol. 16, no. 3, pp. 19-20, AUUG Inc., Melbourne, VIC, June 1995.
- Maltby, Chris, "The Cryptography Debate in Australia," AUUGN, vol. 16, no. 5, p. 18, AUUG Inc., Melbourne, VIC, October 1995.
- Maltby, Chris, "Death of the Net?," AUUGN, vol. 16, no. 1, p. 23, AUUG Inc., Melbourne, VIC, February 1995.
- Crawford, Frank, "The Desktop of the Future," AUUGN, vol. 16, no. 4, pp. 19-21, AUUG Inc., Melbourne, VIC, August 1995.
- Honan, Jamie, "Economics," AUUGN, vol. 16, no. 4, p. 24, 29, AUUG Inc., Melbourne, VIC, August 1995.
- Tune, Andrew, "Feedback (mmap())," AUUGN, vol. 16, no. 4, p. 13, AUUG Inc., Melbourne, VIC, August 1995.
- Crawford, Frank, "Firewalls," AUUGN, vol. 16, no. 4, pp. 23-24, AUUG Inc., Melbourne, VIC, August 1995.
- Long, Tim, "Formatting C," AUUGN, vol. 16, no. 1, pp. 43-55, AUUG Inc., Melbourne, VIC, February 1995. \*\*\*\* Reprint from AUUGN v4n1 \*\*\*\*
- Paddon, Michael, "A Free Word Processor for UNIX," AUUGN, vol. 16, no. 6, pp. 5-7, AUUG Inc., Melbourne, VIC, December 1995.
- Saarelainen, Markku J., "Information Security System responsibilities, structure and development," AUUGN, vol. 16, no. 6, pp. 8-9, AUUG Inc., Melbourne, VIC, Decemeber 1995.
- Honan, Jamie, "Information Services," AUUGN, vol. 16, no. 4, pp. 21-22, AUUG Inc., Melbourne, VIC, August 1995.
- McCrea, Phil, "Internet-based Electronic Commerce will give us 'eye-level' access to world markets," AUUGN, vol. 16, no. 4, p. 11,15, AUUG Inc., Melbourne, VIC, August 1995.
- Crawford, Frank, "The Internet - will your kids know more than you?," AUUGN, vol. 16, no. 2, p. 11, AUUG Inc., Melbourne, VIC, April 1995.
- Bell, Gorden, "Internet 3.0 Envisioned," AUUGN, vol. 16, no. 4, p. 19, AUUG Inc., Melbourne, VIC, August 1995.
- Booth, Adrian, "Interview: John Lions," AUUGN, vol. 16, no. 5, p. 15, 18, AUUG Inc., Melbourne, VIC, October 1995.
- Crawford, Frank, "Interview: Tim O'Reilly at AUUG'95," AUUGN, vol. 16, no. 5, p. 13, AUUG Inc., Melbourne, VIC, October 1995.
- Honan, Jamie, "Long Live Text," AUUGN, vol. 16, no. 4, pp. 22-23, AUUG Inc., Melbourne, VIC, August 1995.
- McCrea, Phil, "Microsoft Network," AUUGN, vol. 16, no. 6, pp. 11-12, AUUG Inc., Melbourne, VIC, December 1995.
- Purdue, David, "Netscape Break-in," AUUGN, vol. 16, no. 5, p. 19, AUUG Inc., Melbourne, VIC, October 1995.
- Huxtable, Glenn, "One System Administrator's Story," AUUGN, vol. 16, no. 6, pp. 10-11, AUUG Inc., Melbourne, VIC, December 1995.
- Hanson, Lawson, "OOPS and GUI: Object Oriented Programming Systems and Graphical User Interfaces," AUUGN, vol. 16, no. 6, pp. 9-10, AUUG Inc., Melbourne, VIC, December 1995.
- McCrae, Phil, "Open Systems and Open Networks," AUUGN, vol. 16, no. 2, p. 23, AUUG Inc., Melbourne, VIC, April 1995.

**AUUGN Volume 16 Index Papers (sorted by title)**

McCrae, Phil, "The Operating System they use in Heaven...," AUUGN, vol. 16, no. 1, p. 6, AUUG Inc., Melbourne, VIC, February 1995.

Crawford, Frank, "Overview of the AUUG95 Network," AUUGN, vol. 16, no. 5, p. 17, AUUG Inc., Melbourne, VIC, October 1995.

Vanderstock, Andrew, "Productivity through better user interface design," AUUGN, vol. 16, no. 2, p. 12, AUUG Inc., Melbourne, VIC, April 1995.

Taylor, Ian, "Professions On-Line: Australian IT consultants on the Web," AUUGN, vol. 16, no. 4, pp. 14-15, AUUG Inc., Melbourne, VIC, August 1995.

Main, Alan, "Sex, Lies and Policy Manuals - Developing and effective policy and procedures manual for Open Systems," AUUGN, vol. 16, no. 2, pp. 13-18, AUUG Inc., Melbourne, VIC, April 1995.

Honan, Jamie, "Software Minimalism," AUUGN, vol. 16, no. 5, p. 20, AUUG Inc., Melbourne, VIC, October 1995.

Honan, Jamie, "Sun, Tk + Java," AUUGN, vol. 16, no. 4, pp. 15-16, AUUG Inc., Melbourne, VIC, August 1995.

McCrea, Phil, "Thoughts of an Outgoing President," AUUG Inc., vol. 16, no. 4, pp. 12-13, AUUG Inc., Melbourne, VIC, August 1995.

Purdue, David, "Transparency and Performance Issues in Sun RPC," AUUGN, vol. 16, no. 1, pp. 17-21, AUUG Inc., Melbourne, VIC, February 1995.

McRae, Andrew, "UNIX. Live free or die!," AUUGN, vol. 16, no. 2, p. 19, AUUG Inc., Melbourne, VIC, April 1995.

Jackson, Janet, "UNIX Tricks & Traps," AUUGN, vol. 16, no. 2, pp. 21-22, AUUG Inc., Melbourne, VIC, April 1995.

*How to stir up a storm with email forwarding*  
*Remembering options, executing your prompt*  
*Finding out with Telnet*

Jackson, Janet, "UNIX Tricks & Traps," AUUGN, vol. 16, no. 4, pp. 25-26, AUUG Inc., Melbourne, VIC, August 1995.

*Greg's Perl Iota*

Jackson, Janet, "UNIX Tricks & Traps," AUUGN, vol. 16, no. 6, pp. 14-15, AUUG Inc., Melbourne, VIC, December 1995.

*Tracing System Calls*  
*Trap: interpreters can't be scripts*

Jackson, Janet, "UNIX Tricks & Traps," AUUGN, vol. 16, no. 3, pp. 11-16, AUUG Inc., Melbourne, VIC, June 1995.

*Some Simple Directory Tools*  
*A Program to Debug and Edit Pipelines*  
*Centering text, with and without vi*

Jackson, Janet, "UNIX Tricks & Traps," AUUGN, vol. 16, no. 5, pp. 22-24, AUUG Inc., Melbourne, VIC, October 1995.

*Correspondence about "grepdir"*  
*Timing out a process from the shell*

Leonard, David, "UNIX Tricks & Traps - Consistent Binary Support for Multiple Architectures Across a Common Filesystem," AUUGN, vol. 16, no. 1, p. 22, AUUG Inc., Melbourne, VIC, February 1995.

Crawford, Frank, "Who is doing your System Administration?," AUUGN, vol. 16, no. 4, p. 17, AUUG Inc., Melbourne, VIC, August 1995.

Chubb, Lucy, "Yuletide Packets," AUUGN, vol. 16, no. 1, p. 34, AUUG Inc., Melbourne, VIC, February 1995.



# AUUG Institutional Members

as at 22/01/96

AAIL  
 ACAY Network Computing Pty.Ltd.  
 Actrol Parts  
 Adept Software  
 Alcatel Australia  
 Amalgamated Television Services  
 Amdahl Australia  
 Andersen Consulting  
 ANI Manufacturing Group  
 Ansett Australia  
 ANSTO  
 Anti-Cancer Council of Victoria  
 ANZ McCaughan  
 AT & T GIS  
 Attorney-General's Department  
 Ausnet Services Pty. Ltd.  
 AUSOM Inc.  
 AUSTA Electric Qld Minerals & Energy Centre  
 Australian Archives  
 Australian Bureau of Statistics  
 Australian Centre for Remote Sensing (ACRES)  
 Australian Customs Service  
 Australian Defence Industries Ltd.  
 Australian Electoral Commission  
 Australian Film Television and Radio School  
 Australian Information  
     Processing Centre Pty. Ltd.  
 Australian Medical Enterprise  
 Australian Museum  
 Australian National Audit Office  
 Australian National University  
 Australian Submarine Corporation  
 Australian Taxation Office  
 Australian Technology Resources (ACT) Pty. Ltd.  
 Australian Technology Resources Pty. Ltd.  
 AWA Defence Industries  
 B & D Australia  
 Barwon Water  
 Bay Technologies Pty Ltd  
 BHP Information Technology  
 BHP Information Technology  
 BHP Minerals Exploration  
 BHP Research - Melbourne Laboratories  
 BHP Research - Newcastle Laboratories  
 Burdett Buckeridge & Young Ltd.  
 Bureau of Meteorology  
 Bytecraft Pty. Ltd.  
 Cape Grim B.A.P.S  
 Capricorn Coal Management Pty. Ltd.  
 CelsiusTech Australia  
 Central Queensland University  
 Central Sydney Area Health Service  
 Centre for Open Systems Pty. Ltd.  
 CITEC  
 Clegg Driscoll Consultants Pty. Ltd.  
 Coal & Allied Operations  
 Cognos Pty. Ltd.  
 Com Net Solutions  
 Com Tech Communications  
 Comcare Australia  
 Commercial Dynamics  
 Commercial Industrial  
     Computer Services Pty. Ltd.  
 Communica Software Consultants  
 Composite Buyers Ltd.  
 Computech Pty. Ltd.  
 Computer Associates  
 Compuware Asia-Pacific  
 Continuum Australia  
 Copper Refineries Pty. Ltd.  
 Corinthian Engineering Pty. Ltd.  
 CSC Australia Pty. Ltd.  
 CSIRO Division of Information Technology  
 CSIRO Division of Manufacturing Technology  
 Curtin University of Technology  
 Cyberdyne Systems Corporation Pty. Ltd.  
 Cyberscience Corporation Pty. Ltd.  
 Cybersource Pty. Ltd.  
 Daedalus Integration Pty. Ltd.  
 Data General Australia Pty. Ltd.  
 Datacraft Technologies  
 Dawn Technologies  
 DB Bain Group Services Pty. Ltd.  
 Deakin University  
 Defence Housing Authority  
 Defence Service Homes  
 Department of Communications and the Arts  
 Department of Conservation  
     & Natural Resources  
 Department of Defence  
 Department of Defence (TC Section)  
 Department of Education QLD  
 Department of Family Services &  
     Aboriginal & Islander Affairs  
 Department of Gaming & Racing  
 Department of Lands Housing & Local  
     Government  
 Department of the Treasury  
 Department of Urban Services  
 Dept. of Industrial Relations Employment  
     Training & Further Education  
 DEVETIR  
 Dialix Internet Services27  
 Digital Equipment Corp. (Australia) Pty. Ltd.  
 Dominion Systems Pty. Ltd.  
 DSTO Lab 73  
 EASAMS (Australia) Limited  
 Edith Cowan University  
 Electricity Trust of South Australia  
 Electro Optics Pty. Ltd.  
 Engineering Computer Services Pty. Ltd.  
 Environmental Resources  
     Information Network (ERIN)  
 Department of Environment Sport and Territories  
 Equity Systems Pty. Limited  
 Ericsson Australia  
 ESRI Australia Pty. Ltd.  
 Execom Consulting  
 Executive Computing Group  
 FFE / James Hardie Bldg. Serv.  
 FGH Decision Support Systems Pty. Ltd.  
 Financial Network Services  
 First State Computing  
 Flinders University  
 Fremantle Port Authority  
 G.James Australia Pty. Ltd.  
 GEC Alsthom Information Technology  
 Genasys II Pty. Ltd.  
 Great Barrier Reef Marine Park Authority  
 Haltek Pty. Ltd.  
 Hamersley Iron Pty. Ltd.  
 Hannan Group Computer Services  
 Heath Insurance  
 Hermes Precisa Australia Pty. Ltd.  
 Hitachi Data Systems  
 Honeywell Australia Ltd.  
 Honeywell Ltd.  
 Hong Kong Jockey Club Systems  
     (Australia) Pty. Ltd.  
 I.P.S Radio & Space Services  
 IBM Australia Ltd.  
 Ideas International Pty. Ltd.  
 Independent Systems Integrators  
 Informatel Online  
 Information Technology Consultants  
 Insurance & Superannuation Commission  
 Integration Design Pty. Ltd.  
 Intelligent Network Development  
 James Cook University  
 Joint House Department  
 JTEC Pty. Ltd.  
 Keays Software  
 Knowledge Engineering Pty. Ltd.  
 Laboratory Systems Pty. Ltd.  
 Labtam Australia Pty. Ltd.  
 Land Information Centre  
 Land Titles Office  
 Leeds & Northrup Australia Pty. Limited  
 Logica Pty. Ltd.  
 Lotus Development  
 Lyons Computer Pty. Ltd.  
 Macquarie University  
 Main Roads Western Australia  
 Mayne Nickless Courier Systems  
 Mayne Nickless Information Tech. Services  
 Medical Benefits Funds of Australia Ltd.  
 Memtec Limited  
 Mentor Technologies Pty. Ltd.  
 Mercedes-Benz (Australia) Pty. Ltd.  
 Message Handling Systems  
 Metal Trades Industry Association  
 Mincom Pty. Ltd.  
 Minenco Pty. Ltd.  
 Mitsubishi Motors Australia Ltd.  
 Mitsui Computer Limited  
 Moldflow Pty. Ltd.  
 Motorola Communications Australia  
 Motorola Computer Systems

## AUUG Institutional Members

Multibase Pty. Ltd.  
Multiline BBS  
National Library of Australia  
National Resource Information Centre  
NCOM Services  
NEC Australia Pty. Ltd.  
Northern Territory Library Service  
Novell Pty. Ltd.  
NSW Agriculture  
NSW Teachers Federation Health Society  
Object Design Pty. Ltd.  
Object Technology International Pty. Ltd.  
Office of the Director of Public Prosecutions  
Open Software Associates Ltd.  
OPSM  
OSIX Pty. Ltd.  
Pacific Star Communications  
Peter Harding & Associates Pty. Ltd.  
Petrosys Pty. Ltd.  
Philips PTS  
Port of Melbourne Authority  
Powerhouse Museum  
Primary Industries & Energy  
Process Software Solutions Pty. Ltd.  
Prospect Electricity  
Pyramid Data Centre Systems  
Qantek  
QLD Department of Transport  
Quality By Design Pty. Ltd.  
Redland Shire Council  
Renison Golffields Consolidated Ltd.  
Rinbina Pty. Ltd.  
Royal Melbourne Institute of Technology  
SCEGGS Redlands Ltd  
Sculptor 4GL+SQL

Security Mailing Services  
SEQEB Business Systems  
Siemens Nixdorf Information Systems Pty. Ltd.  
Smallworld Systems (Aust.) Pty. Ltd.  
Snowy Mountains Authority  
Software Plus (Australia) Pty. Ltd.  
South Australian Co-operative Bulk Handling  
Specialix Pty. Ltd.  
St. Gregory's Armenian School  
St. John of God Hospital  
St. Vincent's Private Hospital  
Stallion Technologies Pty. Ltd.  
Standards Australia  
Stanilite  
State Library of Victoria  
State Revenue Office  
Steelmark Eagle & Globe  
Sterling Software  
Storage Technology of Australia  
Sydney Electricity  
Sydney Ports Corporation  
System Corporation Pty. Ltd.  
Systems Development Telecom Australia  
TAB Queensland  
TAFE NSW Information Systems Division  
Tandem Computers  
Tattersall Sweep Consultation  
Technical Software Services  
TechNIX Consulting Group International  
Telecom Australia  
Telecom Payphone Services  
Telstra Applied Technologies  
Telstra Research Laboratories  
The Far North QLD Electricity Board  
The Fulcrum Consulting Group

The Roads & Traffic Authority  
The Southport School  
The University of Western Australia  
Thiess Contractors Pty. Ltd.  
Thomas Cook Ltd.  
TNT Australia Information Technology  
Toshiba International Corporation Pty. Ltd.  
Tower Technology Pty. Ltd.  
Tradelink Plumbing Supplies Centres  
Transport Accident Commission  
Triad Software Pty. Ltd.  
Unidata Australia  
University of Adelaide  
University of New South Wales  
University of Queensland  
University of South Australia  
University of Sydney  
University of Tasmania  
University of Technology Sydney  
Vanguard Computer Services Pty. Ltd.  
Victoria University of Technology  
VME Systems Pty. Ltd.  
Walter & Eliza Hall Institute  
Water Board  
WCS Australia Pty. Ltd.  
Wesfarmers Limited  
Western Mining Corporation  
Westrail  
Woodside Offshore Petroleum  
Workers' Compensation Board of QLD  
Workstations Plus  
XEDOC Software Development Pty. Ltd.  
Zircon Systems Pty. Ltd.

# MEMBERSHIP APPLICATION

## INDIVIDUAL



UNIX® AND OPEN SYSTEMS USERS

To apply for AUUG membership, complete this form and return it with payment in Australian Dollars to:

**REPLY PAID 66, AUUG MEMBERSHIP SECRETARY,  
P.O. BOX 366, KENSINGTON, NSW 2033, AUSTRALIA  
Tel: +61 2 361-5994 or 1 800 625 655 • Fax: +61 2 332-4066**

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

NOTE: Please do not send purchase orders - perhaps your purchasing department will consider this form to be an invoice. Foreign applicants please send a bank draft drawn on an Australian bank.

### INDIVIDUAL OR STUDENT MEMBERSHIP:

I, \_\_\_\_\_ do hereby apply for:

- Renewal/New membership of AUUG  \$ 90.00
- Renewal/New Student membership  \$ 25.00 (please complete certification portion)
- International air mail  \$ 60.00

*I agree that this membership will be subject to the 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.*

TOTAL REMITTED: AUD\$ \_\_\_\_\_ (Cheque, or money order, or credit card)

\_\_\_\_\_  
Signature  
\_\_\_\_\_  
Date

### LOCAL CHAPTER DESIGNATE:

You can participate in the activities of a local AUUG Chapter. Part of your fee will be given to the chapter to support those activities. By default a regional chapter will be selected for you. If you would rather nominate a chapter, please specify here \_\_\_\_\_ (indicate NONE for no chapter).

### TO BETTER SERVE YOU, PLEASE PRINT YOUR CONTACT INFORMATION:

Name/Contact: \_\_\_\_\_  
 Position/Title: \_\_\_\_\_  
 Company: \_\_\_\_\_  
 Address: \_\_\_\_\_  
 \_\_\_\_\_ Postcode \_\_\_\_\_  
 Tel: BH \_\_\_\_\_ AH \_\_\_\_\_  
 Fax: BH \_\_\_\_\_ AH \_\_\_\_\_  
 email address: \_\_\_\_\_

### STUDENT MEMBER CERTIFICATION: (to be completed by a member of the academic staff)

I, \_\_\_\_\_ certify that  
 \_\_\_\_\_ (administrator)  
 \_\_\_\_\_ is a full time student at  
 \_\_\_\_\_ (name)  
 \_\_\_\_\_ and is expected to  
 \_\_\_\_\_ (institution)  
 graduate approximately \_\_\_\_\_ (date)

\_\_\_\_\_  
Signature  
\_\_\_\_\_  
Title Date

*Over for Institutional Membership*

Please charge \$ \_\_\_\_\_ to my  
 Bankcard,  Visa,  Mastercard

Account number: \_\_\_\_\_  
 Expiry date: \_\_\_\_\_  
 Name on card: \_\_\_\_\_  
 Signature: \_\_\_\_\_

### AUUG Secretariat Use

Chq: bank \_\_\_\_\_ bsb \_\_\_\_\_  
 a/c \_\_\_\_\_ # \_\_\_\_\_  
 Date: \_\_\_\_\_ \$ \_\_\_\_\_  
 Initial: \_\_\_\_\_  
 Date processed: \_\_\_\_\_  
 Membership # \_\_\_\_\_

AUUG Inc. as a user group, exists to provide UNIX® and open systems users with relevant and practical information, services, and education through cooperation among users.

# MEMBERSHIP APPLICATION

## INSTITUTIONAL



UNIX® AND OPEN SYSTEMS USERS

To apply for AUUG membership, complete this form and return it with payment in Australian Dollars to:

REPLY PAID 66, AUUG MEMBERSHIP SECRETARY,  
P.O. BOX 366, KENSINGTON, NSW 2033, AUSTRALIA  
Tel: +61 2 361-5994 or 1 800 625 655 • Fax: +61 2 332-4066

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

NOTE: Please do not send purchase orders - perhaps your purchasing department will consider this form to be an invoice. Foreign applicants please send a bank draft drawn on an Australian bank.

We, \_\_\_\_\_  
(Company Name)

do hereby apply for:

Renewal/New\* Inst. membership of AUUG  \$350.00  
International air mail  \$120.00

TOTAL REMITTED AUD\$ \_\_\_\_\_  
(Cheque, money order, or credit card)

We agree that this membership will be subject to the 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.

We understand that we will receive two copies of the AUUG newsletter, and may send two representatives to AUUG sponsored events at member rates, though we will have only one vote in AUUG elections, and other ballots as required.

Signed \_\_\_\_\_ Date \_\_\_\_\_

Title \_\_\_\_\_

### INSTITUTIONAL MEMBER DETAILS:

To be completed by institutional members only.

Following are our specified contacts. The primary contact holds the full member voting rights. The two designated reps will also be given membership rates to AUUG activities including chapter activities. By default a regional chapter will be selected for you. If you would rather nominate a chapter, please specify in space provided (indicate NONE for no chapter). (Please print clearly or type)

Primary Contact \_\_\_\_\_

Position/Title \_\_\_\_\_

Address \_\_\_\_\_

Postcode \_\_\_\_\_

Bus. Tel: \_\_\_\_\_ Bus. Fax: \_\_\_\_\_

e-mail address \_\_\_\_\_

Local Chapter Pref. \_\_\_\_\_

1st Rep. \_\_\_\_\_

Position/Title \_\_\_\_\_

Address \_\_\_\_\_

Bus. Tel: \_\_\_\_\_ Bus. Fax: \_\_\_\_\_

e-mail Address \_\_\_\_\_

Local Chapter Pref. \_\_\_\_\_

2nd Rep. \_\_\_\_\_

Position/Title \_\_\_\_\_

Address \_\_\_\_\_

Bus. Tel: \_\_\_\_\_ Bus. Fax: \_\_\_\_\_

e-mail address \_\_\_\_\_

Local Chapter Pref. \_\_\_\_\_

Please charge \$ \_\_\_\_\_ to my

Bankcard,  Visa,  Mastercard

Account number: \_\_\_\_\_

Expiry date: \_\_\_\_\_

Name on card: \_\_\_\_\_

Signature: \_\_\_\_\_

### AUUG Secretariat Use

Chq: bank \_\_\_\_\_ bsb \_\_\_\_\_

a/c \_\_\_\_\_ # \_\_\_\_\_

Date: \_\_\_\_\_ \$ \_\_\_\_\_

Initial: \_\_\_\_\_

Date processed: \_\_\_\_\_

Membership # \_\_\_\_\_

AUUG Inc. as a user group, exists to provide UNIX® and open systems users with relevant and practical information, services, and education through cooperation among users.



