AUUGN

Australian Unix systems User Group Newsletter

Volume 9 - Number 2

April 1988

Registered by Australia Post Publication No. NBG6524

The Australian UNIX* systems User Group Newsletter

Volume 9 Number 2

April 1988

CONTENTS

* UNIX is a registered trademark of AT&T in the USA and other countries.

AUUG General Information

Memberships and Subscriptions

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

All correspondence concerning membership of the AUUG should be addressed to:-

The AUUG Membership Secretary,
P.O. Box 366,
Kensington, N.S.W. 2033.
AUSTRALIA

General Correspondence

All other correspondence for the AUUG should be addressed to:-

The AUUG Secretary,
Department of Computer Science,
Melbourne University,
Parkville, Victoria 3052.
AUSTRALIA

ACSnet: *auug@munnari.oz*

AUUG Executive

President	**John Lions** *johnl@cheops.eecs.unsw.oz* School of Electrical Engineering and Computer Science, University of New South Wales, New South Wales	Secretary	**Robert Elz** *kre@munnari.oz* Department of Computer Science, University of Melbourne, Victoria
Treasurer	**Chris Maltby** *chris@softway.sw.oz* Softway Pty. Ltd., New South Wales		
Committee Members	**Chris Campbell** *chris@comperex.oz* Comperex Pty. Limited, New South Wales		**Piers Lauder** *piers@basser.cs.su.oz* Basser Department of Computer Science, Sydney University, New South Wales
	Tim Roper *timr@labtam.oz* Labtam Limited, Victoria		**Peter Wischart** *pjw@anucsd.oz* NEC Information Systems, Canberra

Next AUUG Meeting

The next meeting will be held in Melbourne at the Southern Cross Hotel from the 13th to the 15th of September 1988. Futher details will be provided in the next issue.

AUUG Newsletter

Editorial

This is my ten issue as Editor of the Newsletter and the last one I will produce while working for the Monash University Computer Centre. By the time you have received this issue, I will have started in my new postion at Webster Computer Corporation. I happy to report my new employers have expressed the desire to support me in my role as Editor of this Newsletter. Please note, my new address given below.

I wish to thank the Computer Centre and Department of Computer Science at Monash University for the support they have given to enable me to produce the Newsletter.

As I have reported in previous issues, I have moved the printing and envolope packing of the Newsletter to Pink Panther in Melbourne. After a few minor hickups this seems to be now being working smoothly.

I am disappointed that the last few issues of the Newsletter have been dominated by reprints from *;login:* and *EUUGN*. I would be pleased if we could increase the Australian content of the Newsletter. I again ask you to think seriously about contributing an article to the AUUGN.

You should also consider producing a paper to present at the Winter Conference which will be held at the Southern Cross Hotel in Melbourne during September. The Call for Papers appears in this issue, and the deadline for abstracts is mid-June.

I hope you enjoy this issue and look forward to producing many more.

REMEMBER, if the mailing label that comes with this issue is highlighted, it is time to renew your AUUG membership.

AUUGN Correspondence

All correspondence reguarding the AUUGN should be addressed to:-

John Carey
AUUGN Editor
Webster Computer Corporation
1270 Ferntree Gully Road
Scoresby, Victoria 3179
AUSTRALIA

ACSnet: *john@wcc.oz*

Phone: +61 3 764 1100

Contributions

The Newsletter is published approximately every two months. The deadline for contributions for the next issue is Friday the 17th of June 1988.

Contributions should be sent to the Editor at the above address.

I prefer documents sent to me by via electronic mail and formatted using *troff -mm* and my footer macros, troff using any of the standard macro and preprocessor packages (-ms, -me, -mm, pic, tbl, eqn) as well TeX, and LaTeX will be accepted.

Hardcopy submissions should be on A4 with 35 mm left at the top and bottom so that the AUUGN footers can be pasted on to the page. Small page numbers printed in the footer area would help.

Advertising

Advertisements for the AUUG are welcome. They must be submitted on an A4 page. No partial page advertisements will be accepted. The current rate is AUD$ 200 dollars per page.

Mailing Lists

For the purchase of the AUUGN mailing list, please contact Chris Maltby.

Disclaimer

Opinions expressed by authors and reviewers are not necessarily those of the Australian UNIX systems User Group, its Newsletter or its editorial committee.

AUUG Institutional Members

Altos Computer Systems Pty Limited

Australian National University

Australian Telescope Computer Group (CSIRO)

Australian Wool Corporation

Ballarat Base Hospital

Department of Industry, Technology and Resources, Victoria

Digital Equipment Corporation (Australia) Pty. Limited

Fujitsu Australia Limited

Hewlett Packard Australia Limited

Hewlett-Packard, Australian Software Operation

Honeywell Information Systems

Intercept Computers Pty. Limited

James Cook University of North Queensland

Macquarie Bank Limited

Macquarie University

Nixdorf Computer Pty Limited

Olivetti Australia Pty Ltd

Prime Computer Research & Development

Pyramid Technology Australia

Q. H. Tours Limited

Queensland Government Computer Centre

Sanyo Office Machines Pty Limited

Sigma Data Corporation Pty Ltd

South Australian Institute of Technology

Sun Microsystems Australia

Swinburne Institute of Technology

Tattersall Sweep Consultation

University of Adelaide

University of Melbourne

University of New England

University of New South Wales

University of Sydney

Call For Papers

AUUG '88

Australian Unix systems User Group

Winter Conference and Exhibition 1988

September 13–15 1988, Melbourne, Australia

Summary

The 1988 Winter Conference and Exhibition of the Australian UNIX† systems User Group will be held on Tuesday 13th – Thursday 15th September 1988 at the Southern Cross Hotel in Melbourne, Australia.

The conference theme is *Networking – Linking the UNIX World.*

AUUG is pleased to announce that the guest speakers will include:

Ken Thompson	Bell Laboratories
Michael Lesk	Bell Communications Research
Mike Karels	University of California at Berkeley

Papers

Papers on topics related to computer networks and UNIX are now invited. Some suggested topics include but are not restricted to:

- Operating system and programming language support for networks
- Distributed file systems and their application
- Networked window systems
- ISO/OSI and UNIX
- Security aspects of computer networks
- Legal and social aspects of computer networks
- Protocol specification methods
- Harnessing new technologies
- Network applications under UNIX

Papers on other (non networking) aspects of the UNIX system are also sought.

Authors of papers presented at the conference will receive complimentary admission to the conference and the dinner. AUUG will again hold a competition for the best paper by a full time student at an Australian educational institution. The prize for this

† UNIX is a trademark of Bell Laboratories.

competition will be an expense paid return trip from within Australia to the conference to present the winning paper. A cash prize in lieu of this may be paid at the discretion of AUUG. Students should indicate with their abstract whether they wish to enter the competition. AUUG reserves the right to not award the prize if no entries of a suitable standard are forthcoming.

A special issue of the group's newsletter AUUGN containing the conference proceedings will be printed for distribution to attendees at the conference.

Acceptance of papers will be based on an extended abstract and will be subject to receipt of the final paper by the due date. Abstracts and final papers should be submitted to the programme committee chair:

Tim Roper	Phone:	International	+61 3 5871444
AUUG 88		National	03 5871444
Labtam Limited	Fax:	International	+61 3 5805581
PO Box 297		National	03 5805581
Mordialloc	Telex:	LABTAM AA33550	
Victoria 3195	ACSnet:	timr@labtam.oz	
Australia	UUCP:	uunet!munnari!labtam.oz!timr	
	ARPA:	timr%labtam.oz@uunet.uu.net	

Final papers may be sent via electronic mail and formatted using *troff* and any of the standard UNIX macro and preprocessor packages (–ms, –me, –mm, pic, tbl, eqn) or with TeX or LaTeX. Alternatively, final papers may be submitted as camera ready copy on A4 paper with 35mm margins left at the top and bottom. Intending authors unable to produce either of these forms are requested to contact the programme committee chair.

Timetable

Receipt of Extended Abstracts	Monday 13th June
Letters of Acceptance Sent	Monday 4th July
Receipt of Final Papers	Monday 8th August
Conference and Exhibition	13th–15th September

Adelaide UNIX Users Group

The Adelaide UNIX Users Group has been meeting on a formal basis for 12 months. Meetings are held on the third Wednesday of each month. To date, all meetings have been held at the University of Adelaide. However, it was recently decided to change the meeting time from noon to 6pm. This has necessitated a change of venue, and, as from April, meetings will be held at the offices of Olivetti Australia.

In addition to disseminating information about new products and network status, time is allocated at each meeting for the raising of specific UNIX related problems and for a brief (15-20 minute) presentation on an area of interest. Listed below is a sampling of recent talks.

D. Jarvis	"The UNIX Literature"
K. Maciunas	"Security"
R. Lamacraft	"UNIX on Micros"
W. Hosking	"Office Automation"
P. Cheney	"Commercial Applications of UNIX"
J. Jarvis	"troff/ditroff"

The mailing list currently numbers 34, with a healthy representation (40%) from commercial enterprises. For further information, contact Dennis Jarvis (dhj@aegir.dmt.oz) on (08) 268 0156.

Dennis Jarvis,
Secretary, AdUUG.

Dennis Jarvis, CSIRO, PO Box 4, Woodville, S.A. 5011, Australia.

PHONE: +61 8 268 0156

UUCP: {decvax,pesnta,vax135}!mulga!aegir.dmt.oz!dhj
ARPA: dhj%aegir.dmt.oz!dhj@seismo.arpa
CSNET: dhj@aegir.dmt.oz

Softway Pty Limited (Incorporated in NSW)
PO Box 305, Strawberry Hills, NSW 2012, Australia
First Floor, 120 Chalmers St, Strawberry Hills NSW
☎ (02) 698 2322 Fax (02) 699 9174

Softway

SOFTWAY PRODUCTS

- ☑ SUN-III (ACSnet)
- ☑ BBBackup
- ☑ C++ Translator
- ☑ UNIX† System V
- ☑ Technical Backup
- ☑ Courses:
  - Beginner's Workshop
  - Fast Start to UNIX
  - System Administrators' Workshop
- ☑ Documenter's Workbench 2.0
  - and various back-end drivers
  - PostScript support of plain text
  - support for graphs and images
- ☑ Ports & Device Drivers
- ☑ Intelligent Benchmarking
- ☑ Biway - Bi-directional modem software

† UNIX is a trademark of AT&T Bell Labs.

A Note on Security and UNIX

M. Anderson
Department of Computer Science
Monash University

ABSTRACT

Demand for secure systems is rising. UNIX in its basic form is not adequate to meet this demand. This note discusses very briefly two modification methods, capabilities and access control lists, which may be used to provide a better class of protection for UNIX files.

1. INTRODUCTION

With the increasing demand for more secure systems and the rising popularity of UNIX in the commercial arena, pressure will be placed on the system to provide a secure operating environment. Already, some UNIX like systems exist where various security policies can be implemented. The purpose of this note is to discuss briefly some of the aspects which must be considered when going about providing a more secure environment than that available on "vanilla" UNIX systems. Note that there is no intention of going into detail over well known security problems. Rather, we call attention to two particular methods which can be used to provide a better level of security in UNIX. Several references are supplied where details of implementation and a discussion of effects can be garnered.

2. PROTECTION

Protection of files in UNIX is specified by three sets of permission bits, one for owner, one for group, and one for everyone else. Directories specifying files and the permission sets associated with them are arranged in the well known hierarchy.

The basic protection mechanism for UNIX is quite inadequate for any serious security requirements. This is not meant as a harsh criticism as the mechanism was never meant originally to stand up to vigorous protection issues and was implemented more to prevent users from tripping over each other. A typical example of a protection problem is the inability to specify fine grained access control such as indicating which particular users may access a file irrespective of system defined groups.

A UNIX like system, Secure Xenix (SX) (Chapman et al, 1987), is able to get around the problem by implementing "access control lists" (ACL). The APOLLO domain (Leach et al, 1985)

system also supplies ACL's in its UNIX implementation. Another solution to the problem is to implement "capabilities" which are functionally equivalent to ACL's (Laur, Needham, 1979). Both are described below.

In an ACL system, each file has an associated ACL. Each list contains tuples. Each tuple, typically, comprises a username and a set of permission bits. ACL's allow owners of files to specify what type of access can be granted to any particular user for any particular file of theirs. Hence it is possible to specify access for groups, and particular users who may not be in the group. SX still implements the basic UNIX protection mechanism but the owner of the file has the choice to convert the conventional permission format of a file into the ACL format.

Capabilities are like "tickets" which authorise some type of access to files. A capability specifies a file and a set of access permissions. There can be more than one capability refering to the same file and the capabilities needn't have the same access permissions. For example, there may be a capability which gives read and write access to a file, and another which gives read only access to the file.

There is more than one way to implement capabilities and their properties can have numerous sideffects. It is outside the scope of this note to go into any real detail concerning all the properties of capabilities. In the context of interest regard capabilities as tickets which can be "held" by processes, duplicated, destroyed, and passed on to other processes. Naturally capabilities must be unforgeable so only the system itself should be able to manufacture them.

If a process does not have a capability to a file, then it cannot gain access to that file. Hence finegrained access control can be achieved by a user when starting up a process by supplying it with capabilities it may need. There are two important sideffects when capabilities are present. First, how are they disseminated when a program is run? If the program asks for a filename in order to gain access to the corresponding file then there must be some method of indicating the transmission of an appropriate capability as well as the name of the file. Hence some easily managed protocol must be developed. Such a protocol might involve some form of control character appearing with a filename indicating to a compiler or command interpreter that a capability should accompany the filename. One could also envisage optional characters specifying what access permissions were allowed in the transmitted capability.

The second sideffect is that "setuid" programs are no longer necessary. Capabilities to files can be, depending on the implementation, embedded in the code body.

The implementation of capabilities or ACL's to provide a better class of protection is user visible and makes redundant the original UNIX protection mechanism. Whether users can accept such a change remains to be seen. Access control lists are probably a more "natural" implementation for users to deal with as a list can disseminate user names along with access permissions. Capabilities, while granting access to a file, do not necessarily identify users. However, we will see in the next section why it is not good simply to abandon the capability route.

3. VIRUSES

Viruses are code fragments implanted into programs in order to perform a function other than that intended by the program's owner. Typically, these functions are harmful to the executor

of the program. One other function of a virus is to reproduce by implanting a copy of itself in some other executable program.

UNIX is particularly vulnerable to viruses. Consider a program which contains a virus. Assume it is executed by some unwitting user. On execution, the virus gains all the privileges of the user. Probably the first thing it does is go about reproduction. This is achieved by the following. The virus searches for write accessible, executable files. There is no point for a user to remove write access privilege to executables they own in the hope that the virus will be stymied as it can always manipulate the privileges temporarily to gain access to the files. Having found an executable the virus copies itself into a suitable part of the code body section. It then changes the program entry point in the header to point to the start of the virus code. At the end of the virus code a jump to the original entry point is implanted. The next time that executable is run by the user, or better still by some other poor user, the process of reproduction continues. At some predetermined "trigger" condition, e.g. a certain date, the virus performs some function which leaves the user and probably the system for that matter, in some highly undesirable state.

The basic problem in vanilla UNIX is that when a user executes someone else program the writer of the program gains, temporarily, all the executor's privileges. ACL's as described above are of not much use as the virus has the same privileges as the executor and thus can gain access to their files.

The use of capabilities can prevent the propagation of and subsequently aid in the detection of viruses. Remember that a process can only access those files for which it has capabilities. Unless the process is running a compiler, in which case the user has taken great care to ensure that it is the system compiler being used, it is unlikely that the process requires capabilities with write access to executable files. If a virus attempts reproduction, the process will incur an access violation and thus alert a user to a possible virus. The disadvantage of having to implement a protocol to distribute capabilities on executing a program is shown to confer an advantage here. Another advantage is that any damage from malicious procedures can be limited to specific files.

Access control lists can be implemented to provide the same finegrained protection. However, a method of identifying the process and what context it is running in (called a subject) must be provided. Hence the tuples listing just usernames are not sufficient. The end result can be an explosion in the list sizes and a confusing problem for the user in distributing access.

If a virus infects system utilities, especially a compiler, then real problems become apparent. Many different users, including the super user, execute utilities and system compilers. Hence a virus could spread to all parts of the system and effectively gain super user privileges. The resultant carnage would be catastrophic. The only recourse to a heavily infected system is to restore a "clean" version of the system from some medium such as tape. Prevention is better than cure and for relatively static system utilities it is better to store them on write protected disks or in roms.

4. CONFINEMENT

While capabilities and ACL's provide fine grained, flexible access control, there is as yet no support for security management. That is, support for controlling the flow of information.

Consider the following: A user has a program capable of operating on some type of data. Another user wants the use of the program to operate on their data. Unfortunately, both users are suspicious of each other and A wishes to ensure that B cannot ascertain the method used by the program let alone examine the source or binary code while B wishes to ensure that the program is unable to report to anyone, including its owner, anything about the data it operates on. Such is the nature of the confinement problem.

The desired solution exhibits properties whereby a user cannot tell anything about a server program and the server has its output channels specified by the client (executor of the server). The ability for a client to specify what information outlets are available to a server on each use of the server is said to be the imposition of a "discretionary security policy" on the part of the client.

UNIX systems are capable of implementing part of the solution in that they can prevent clients from obtaining information on the server by setting the execute only permission bit. Unfortunately it falls far short of implementing a full solution as a program can always find a means to communicate with its owner. The same can be said of a system implementing capabilities or ACL's as described in the previous sections.

In SX, each file has a security "clearance" and "category" associated with it. When a user logs on, he specifies at which clearance and any processes created during the login session inherit that clearance. Together, the clearance and category satisfy the constraints of the Bell and La Padula lattice model. Rather than explain in detail what a lattice security model is and the definition of categories, keep in mind the simple lattice formed by the well known military model comprising of several clearances ranging from confidential to top secret. A person with top secret clearance can read documents at that clearance or lower. S/he can only write documents at the their current clearance or higher. The rule preventing downgrading a file or writing into one with a lower clearance than your own is called the * property. Campbell (1985) provides a simple, and readable description of lattices in her report on computer security. The military model, without categories, forms a one dimensional lattice with the flow of information towards the top secret clearance level.

A system such as SX contains a partial solution to the confinement problem in that programs can operate on high clearance data files and not be able to divulge their contents to their owners if the owners can only read files of a lower clearance. Unfortunately, the solution does not cater to all situations as the client and server may have unsuitable clearances for the client. The security policy being imposed here is "non discretionary" in the sense that the client must play by its rules and systemwide in the sense that all users in the system are under its jurisdiction.

While a systemwide non discretionary security policy may satisfy some users, other users and groups will demand security policies applicable to their group. This may or may not include lattice models. What is needed is for the system to provide support for discretionary security policies and a framework for building coexisting non discretionary policies. It would also be desirable that users which can satisfy certain constraints, to be able to move from the jurisdiction of one policy defined by a group to another.

Something more than simple ACL's or capabilities is required to implement discretionary security policies.

There is a capability based multiprocessor system at Monash with a confinement mechanism able to implement confinement in the fashion defined at the beginning of the section. While the actual kernel of the multiprocessor is capability based and the confinement mechanism is part of that kernel, the UNIX system being implemented as a set of ordinary processes above the kernel will be able to use both the capability and confinement mechanisms to supply its users with strong finegrained protection and security management. Groups of users are able to construct their own security policies and use the system's confinement mechanism to enforce them.

4.1. COVERT CHANNELS

The security mechanisms have only been described in the context of controlling the flow of information through overt channels. However, it may be possible for a program to disseminate information to unauthorised users through covert channels. Covert channels exist in many forms and their existence is dependent on the system. Rather than a formal definition of a covert channel, a simple example will highlight their salient characteristics. A timing channel, one which exists in any timeshared computer, is CPU modulation. A server can modulate the system load by the creation and subsequent deletion of many processes. The server's owner can demodulate the signal by sampling the load. Thus information can be transfered. Of course other processes in the system can cause problems by introducing the equivalent of "noise" into the system. Information theory techniques exist whereby no matter the amount of noise placed on a covert channel, information can be passed. However, the more noise, the more sophisticated the algorithm required and the lower the bandwidth of the channel.

Covert channels are graded according to their bandwidth. Some are only one or two bits per minute and thus generally of no real consequence but others could be several thousand baud. There is no real assurance that the covert channels discussed by Chapman et al in SX cannot be usefully exploited.

Various mechanisms exist in the multiprocessor constructed at Monash to limit covert channel bandwidth (Anderson, 1987).

For a description of a systematic method for identifying covert channels in a system see Kemmerer (1982).

5. CONCLUSION

The demand for more secure systems is rising. Vanilla UNIX systems are not adequate to meet this demand. However, relatively simple, user visible modifications can go some way to transforming a UNIX system into a relatively secure one. Probably the most important single modification that should be considered is the introduction of a new access control mechanism via capabilities or access control lists. These mechanisms are valuable tools in building non discretionary security policies for the system. It should be noted however, that it is a non trivial task to provide support for discretionary policies which come from confinement given the definitions of capabilities and access control lists in many systems.

It is stated without proof and somewhat provocatively that capabilities in the long run will prove more flexible than ACL's. A hint as to why is tied up with the probability that users will demand more privacy. Hence a need for systems which are highly secure but are still able to deal with anonymous individuals and groups can be envisioned. Some capability based systems such as the multiprocessor built at Monash are already capable of meeting such a criterion. ACL's, with their inherent authentication and identification mechanism, may not be suitable for such an environment. In this case it is assumed that implementing relatively static pseudonyms is not acceptable as it may lead to user identification. However, until users can grapple with the characteristics of capabilities ACL's specifying access on a user basis are probably more suitable in the interim.

REFERENCES

Anderson, Pose, Wallace 1986:
A Password Capability System,
The Computer Journal.
VOL 29, 1.

Anderson 1987:
A Password Capability System,
Ph.D. Thesis,
Monash University.

Campbell 1985:
Computer Security: A Status Report,
Proc. of the 18th Hawaii Int. Conf.
on System Sciences.

Chapman et al 1987:
Design and Implementation of Secure Xenix,
IEEE Trans. on Soft. Engineering.
VOL SE-13, 2.

Kemmerer 1982:
A Practical Approach to Identifying
Storage and Timing Channels,
1982 IEEE Symp. on Security and Privacy.

Laur, Needham 1979:
On the Duality of Operating System Structures,
SIGOPS VOL 13, 2.

Leach et al 1985:
The File System of an Integrated Local Network,
Proc. of the 1985b ACM Computer Science Conference.

Parallel Programming Facilities on the Sequent Series of Machines†

Frank Crawford (frank@teti.qhtours.oz)

Q.H. Tours
PO 630, North Sydney 2060

and

Jagoda Crawford (jc@atom.oz)

Australian Nuclear Science and Technology Organisation
Private Mailbag 1, Menai 2234

1. INTRODUCTION

One of the most exciting machines in the USA at present, at least as far as the UNIX™ community is concerned, is the Balance™ and Symmetry series by Sequent. These are all multiprocessor systems having between 2 and 30 processors per machine. Further, they provide libraries to allow the programmer to explore the multiprocessing capabilities by allowing a single application to consist of multiple closely cooperating processes. Utilities such as *make*, *sh* and *apply* have been enhanced to take advantage of the parallel processing facilities.

The operating system, DYNIX, is a fairly standard 4.2 BSD UNIX, at least at the user interface, with only a few minor enhancements for multiprocessing, however the kernel has been greatly modified in the area of the scheduling algorithms, to provide the multiprocessing support. There is a System V Application Environment (SVAE), but this currently has no direct support for parallel processing.

As with most commercial UNIX systems compilers are available for C, Pascal and Fortran 77, all of which can make use of the Parallel Programming Library. Further, as will be detailed later, third party products such as an Ada Development System and a Restructuring Fortran Complier are available.

2. HARDWARE

The main feature that makes the Sequent systems unique is the hardware, which is designed to support multiple processors. In fact there is no single processor version available, the basic *building block* is a dual processor board.

Sequent markets two ranges, the Balance series and the Symmetry series, which mainly differ in the CPU. For the Balance series the CPU is a National Semiconductor Series 32000 microprocessor (currently a NS32032, giving a performance of about 0.7 MIPS/CPU), with a NS32082 Memory Management Unit and a NS32081 Floating Point Unit. The Symmetry Series consists of Intel 80386 microprocessor (about 3 MIPS/CPU), with a 80387 Floating Point Unit and an optional Floating Point Accelerator based on the 1167 chip from Weitek Corp. Also included is a proprietary data cache.

The remainder of the system is as would be expected for a multiprocessor system, it has a high speed bus connecting the CPU's to a shared memory subsystem and Dual Channel Disk Controllers. It also supports a SCSI and a Multibus interface.

The Balance Series includes hardware support for atomic locks, which is crucial for synchronisation within a multiprocessing environment, whereas the Symmetry Series handles this by a special instruction

---

† The work presented in this paper was made possible with the support of Sigma Data Corp., by making available documentation and system time.

within the microprocessor instruction set.

2.1 Balance Series Locking Mechanisms

The hardware support within the Balance Series consists of a set of locks (called Atomic Lock Memory or ALM) on each Multibus adapter board. Each board has a section of memory that consists of 32 × 2 KB regions which are accessed through special devices in the directory */dev/alm* (*alm00..alm31*). These devices are opened by a process and then mapped into its virtual address space using the system call *mmap()*.

A lock is a 32-bit double word in the ALM, of which only the least-significant bit is used[1]. The state of the lock is either *locked* (1) or *unlocked* (0). Reading a lock returns its current state, and at the same time setting it automatically to *locked*. This operation is indivisible. Writing a 0 to a lock *unlocks* the lock.

A simple approach based on the above (see Fig 1) can result in high bus usage. To reduce this a programmer can implement a *shadow lock*, *i.e.* keep a copy of the ALM in shared memory (see Fig 2). As reads from shared memory are cached, subsequent reads are satisfied from this cache until the cache controller detects a write to the shadow variable on the system bus.

```
/*
 * Lock the ALM lock whose address is lockp.
 */
lock (lockp)
    char *lockp;
{
    while (*lockp & 1)
        continue;
}

/*
 * Unlock the ALM lock whose address is lockp.
 */
unlock (lockp)
    char *lockp;
{
    *lockp = 0;
}
```

Figure 1. Simple ALM Spin-lock.

2.2 Symmetry Series Locking Mechanisms

The locking mechanism used by the Symmetry Series is based on the System Bus, and is implemented in the Symmetry assembly language. To set the bus lock any instruction can be proceeded by the prefix *lock*. This blocks any other bus access for the duration of the instruction. Further the *xchg* instruction is always locked whether it is prefixed by *lock* or not. The basic difference between the Balance and Symmetry is that in the Symmetry any byte can be used as a lock, this greatly simplifies caching problems at the expense of having to resort to assembler to write locking procedures.

3. PARALLEL PROGRAMMING

DYNIX provides a number of low-level facilities to allow parallel programming, e.g. *fork()* and *shared memory* facilities. To simplify their use, a library of procedures, called the Parallel Programming Library (PPL), has been provided, however before studying this, a summary of the low-level facilities is given.

1. Reads and writes to other bits within the word are possible but the results are undefined.

```
struct lock_t {
    char *lk_alm;          /* address of ALM lock */
    char lk_shadow;        /* shadow in memory */
};

/*
 * Lock the ALM lock whose address is lockp.
 */
lock (lockp)
    register struct lock_t *lockp;
{
    /* Go for the ALM lock. */
    while ( *(lockp->lk_alm) & 1 ) {
        /*
         * Didn't get it.  Spin until shadow
         * is unlocked and try again.
         */
        while (lockp->lk_shadow)
            continue;
    }
    /* Got the ALM lock.  Lock the shadow. */
    lockp->lk_shadow = 1;
}

/*
 * Unlock the ALM lock whose address is lockp.
 */
unlock (lockp)
    struct lock_t *lockp;
{
    lockp->lk_shadow = 0;
    *(lockp->lk_alm) = 0;
}
```

Figure 2. Shadow Locking for ALM Spin-lock.

3.1 fork()

The basis for process creation in DYNIX is the same as for any other version of UNIX, the *fork()* system call. With this it is possible to create multiple copies of the current process to perform a given function in parallel. As *fork()* is a relatively expensive call it is usually done at the beginning of a parallel application and the child process kept until the end of the application. If a process is not needed at certain times it can be kept in a busy or spin-loop or put to sleep until it is needed. By default a process will wait in a spin-loop unless some other action is taken.

3.2 Shared Memory

To understand the operation of the PPL it is necessary to understand how DYNIX has implemented the sharing of memory. There are two separate areas, the first is shared common memory.

In a normal UNIX module there are three program segments, loosely these are:

i. Text,

ii. Data, and

iii. Stack.

DYNIX has subdivided the data segment into *private data* and *shared data*. Variables placed in the *shared segment* at compile or link time are automatically mapped into memory that can be shared among multiple processes. When a process forks, its child process inherits access to the parent's shared

segment. Thus processes accessing this shared segment must have a common ancestor.

To make use of this shared segment the DYNIX C compiler supports two new storage class modifiers, *shared* and *private*, the default being *private* unless the compiler option -Y is given.

The second method of creating and expanding shared memory regions is using the system call *mmap()*. This call was specified but not implemented in 4.2 BSD and Sequent have implemented these specifications in DYNIX. With this call any file or region of physical address space can be mapped into a process's virtual address space. A process creates a shared memory region by opening an ordinary file and then using *mmap()* to map the file into the process's address space. Using this method processes without a common ancestor can share memory by "rendezvous" on a common file. Again this shared region is inherited by the process's children.

DYNIX also makes it possible to divide the shared segment into *shared data*, *shared heap* and *shared stack* as shown in Fig 3.

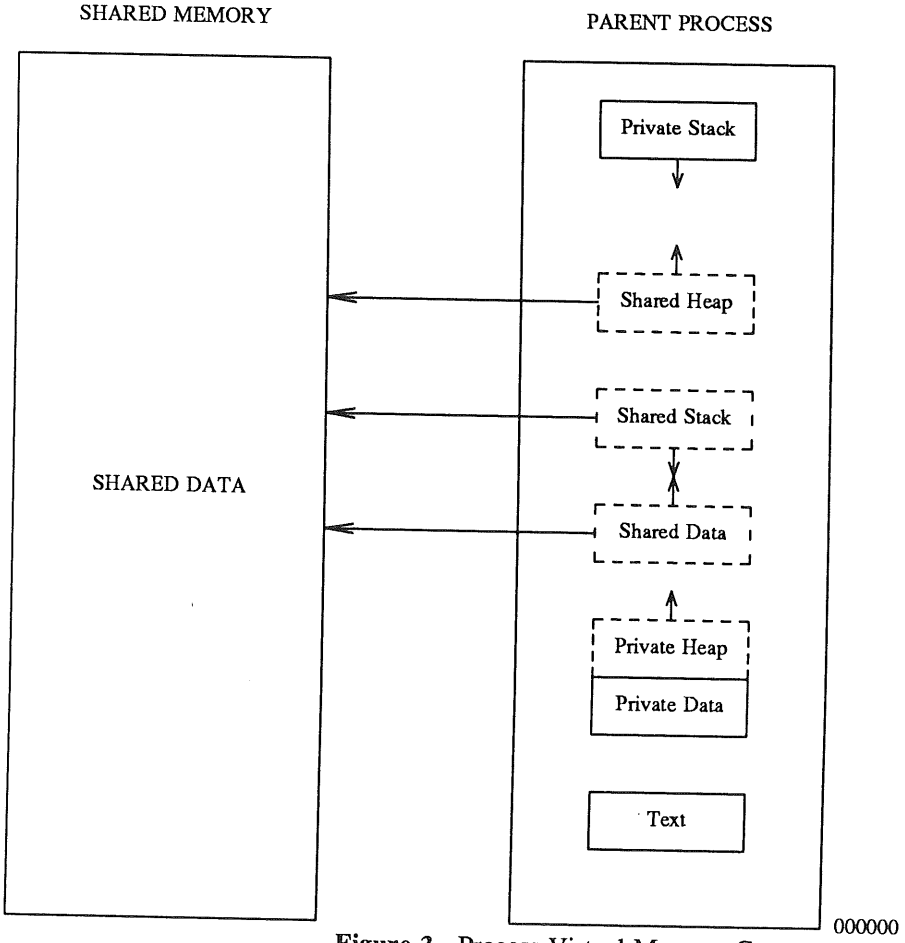


Figure 3. Process Virtual Memory Contents.

3.3 The Parallel Programming Library

The Parallel Programming Library (PPL) simplifies the use of shared memory and the ALM (for Balance Series), and supports the most commonly used parallel programming mechanisms. There are

two broad areas covered by the PPL: multitasking and shared memory operations. Operations supported by the PPL include:

- initialisation of the parallel environment,
- process creation, scheduling and termination,
- process synchronisation and loop scheduling, and
- dynamic memory allocation.

3.3.1 Initialisation of the parallel environment. In the PPL a routine *_ppinit()*, is called before *main()* and performs the following:

- maps the program's shared data segment into shared memory,
- allocates a block of ALM, if necessary, and
- initialises other PPL routines.

3.3.2 Process creation, scheduling and termination. The routine *m_fork()* is used to execute parallel phases of the application. It creates child processes if needed and initiates their parallel execution. The number of processes created depends on how many processors are available in the system and can be controlled, within limits, by the routine *m_set_procs()*. Once all processes have completed execution control returns to the parent process and the children wait for the next *m_fork()* call. The routine *m_kill_procs()* is provided to kill the child processes when no longer needed, and *m_park_procs()/m_rele_procs()* for suspending/resuming child process execution.

3.3.3 Process synchronisation and loop scheduling. A number of routines are provided to synchronise parallel execution. These include such things as *m_next()* to administer a shared counter, lock and unlock routines to access shared data structures in an orderly manner and two forms of barriers (*i.e.* synchronisation points). For example:

- *m_lock()* and *m_unlock()* for simple locks,
- *s_init_lock()*, *s_lock()*, *s_clock()* and *s_unlock()* to create, set, test and release more complex locks, and
- *m_sync()*, *s_init_barrier()* and *s_wait_barrier()* to initialise and wait at a barrier.

3.3.4 Dynamic memory allocation. Parallel C programs can use the library routines *shmalloc()*, *shbrk()* and *shsbrk()* to allocate additional shared memory. These routines make extensive use of the *mmap()* call. The library also includes exception handling code to ensure that all processes maintain a consistent view of shared memory.

4. A PARALLEL APPLICATION

The functions within the PPL can be called from programs written in any of Sequent's supported languages (*i.e.* C, Pascal or Fortran). Commonly, examples of parallel processing involve numerical applications, particularly matrix multiplications, but this is not the only field that can benefit from the use of these techniques. The example given here is based on the quick sort algorithm and compares a parallel version to the standard version supplied in DYNIX.

The algorithm implemented (see Appendix A) modifies the quick sort algorithm by using separate processes to deal with each partition, as it becomes available, in place of the recursive calls normally implemented. However, as this would require an exponential number of processes, each partition is placed on a first-in, first-out (FIFO) queue which processes access when they are free to do more work.

The queue is stored in shared memory and access to it is controlled by the functions:

- *add_queue()* (line 84) to add an element (containing the bounds of the partition) to the queue,
- *get_queue()* (line 111) to select an element to be processed, and

- *del_queue()* (line 130) to free an element once processing is finished.

As these functions are critical regions it is necessary to use the locking facilities (*s_lock()* and *s_unlock()*) to control access to the queue variables.

One of the difficulties with this method is that the required number of processes is not known in advance and further in all practical cases exceeds the number of processors available. The algorithm as implemented requires initially one process, which then divides the work into two, four, etc, thus requiring an increasing number of processes as the process continues. The PPL is designed to allocate a fixed number of processes at the start of the run which cannot be changed without killing all the current children. This means that child processes not currently active have to be blocked until work becomes available. Also it is necessary to distinguish between no work being currently available and all work being completed. This again is handled in the routine *get_queue()* which blocks processes on a semaphore, *wait_queue* (line 125), which is released when either another element (line 109) has been added to the queue or there is no further work to do (line 127).

TABLE 1. Time in Seconds for Modified and Standard Quick Sort Algorithm.

# Proc	*m_qsort*		*qsort*	
	User	Real	User	Real
1	4.60	5.20	2.70	3.20
2	4.97	3.20	2.70	3.20
3	5.40	2.53	2.70	3.20
4	5.83	2.20	2.70	3.17
5	6.37	2.03	2.70	3.20
6	7.03	1.90	2.70	3.20
7	7.87	1.90	2.70	3.20
8	8.90	1.90	2.70	3.20
9	10.23	1.93	2.70	3.20
10	11.73	2.00	2.70	3.17
11	13.70	2.07	2.70	3.20

Table 1 gives some performance results for the standard *qsort()* function versus the modified version *m_qsort()* given in Appendix A for various number of processors participating in the sorting of 10000 randomly generated numbers. It should be noted that no effort was made to optimise the method implemented in *m_qsort()*, aside from compiling with the C optimisation option. As can be seen from the results the **real** time taken to sort the list improves as more processors participate in the sort up to a limit where the overhead of scheduling the additional processes outweights the benefits. Also the total CPU time used by all the processes increases as more processors are participating, with the real time improving. The performance for a single processor being less than the standard method is due mainly to the overhead of maintaining the queue.

5. OTHER PARALLEL PROCESSING FACILITIES

Aside from the PPL described above, a number of other method are available to make efficient use of the parallel architecture. These include a Fortran compiler and an Ada Development System.

5.1 Fortran Parallel Processing Facilities.

The standard Fortran compiler available from Sequent accepts a set of directives which identify:

- loops to be executed in parallel,
- shared and private data within each loop, and
- critical sections of loops.

These directives are expanded by a preprocessor to produce a program from which the compiler can generate parallel code.

Aside from the preprocessor a number of third party products are available to further facilitate the production of parallel code. These include:

- KAP/Sequent - a product which "discovers parallelism within Fortran code". It takes standard (sequential) Fortran code and inserts Sequent directives to enable the compiler to generate parallel code.
- The Force - a set of Fortran macros which provide automatic process creation and termination, declaration of shared and private data, and synchronisation of critical code sections.

5.2 VERDIX Ada Development System.

Ada is the first standardised commercial language to provide support for parallel programming and is available on the Sequent series. Aside from the compiler, a parallel runtime system and an extensive tool set is available. The VERDIX Ada Development System appears to be a modification of a number of standard utilities such as *a.make*, *a.du* and *a.tags*.

5.3 Pdbx Parallel Debugger.

Symmetry also support a modified version of the BSD symbolic debugger, *dbx* to handle parallel programs called *pdbx*. The extensions over *dbx* are that it has a number of commands to handle separate processes, e.g. *%n* to change process and *create* to create a new process. It can be used with any of the languages supported by Sequent and can display program source as well as allowing debugging at assembler level.

6. CONCLUSION

The Sequent series of computers provide a number of different facilities to support parallel programming many of which have been developed by outside groups and adapted by Sequent. However there is still scope for further development both in the libraries and within the application area.

The basic system, while being the first commercial system to be built entirely on parallel processors is not significantly different from other system, and shows how flexible the original concepts were. The basic call within the kernel for creating parallel processes is still *fork()* as it has been from the beginning. The extensions in the area of shared memory have long been needed, but even here Sequent only developed ideas that were sketched out previously.

Despite being very much a standard "box" it is an interesting machine and offers a number of areas that can be developed by those willing to put in the effort.

Appendix A: PARALLEL QUICK SORT FUNCTION

```
 1 /*
 2  * Quick sort implementation using Sequents Parallel Programming Library
 3  */

 4 #include <stdio.h>

 5 #include <parallel/microtask.h>
 6 #include <parallel/parallel.h>

 7 typedef struct queue_entry {            /* Definition for queue elements */
 8     char               *base;          /* Base address */
 9     unsigned           nel,            /* Number of elements */
10                        sizeof_base;    /* Size of base type */
11     struct queue_entry *next;          /* Pointer to next element in list */
12 } queue_entry;

13 typedef struct sem_t {                  /* Semaphore structure */
14     slock_t             sem_lock,      /* Lock for semaphore */
15                         sem_wait;      /* Lock to block on */
16     unsigned int        sem_cnt;       /* Semaphore value */
17 } sem_t;

18 static shared queue_entry
19                         *queue_head = NULL,
20                         *queue_tail = NULL;

21 static shared slock_t   queue_lock,    /* Lock for queue access */
22                         work_lock;     /* Lock for work count */

23 static shared int       work_count = 0; /* Count of partitions */

24 static shared sem_t     wait_queue;

25 static private char     *median = NULL, /* Median value for quick sort */
26                         *swap = NULL;   /* Temporary */

27 static queue_entry      *get_queue();
28 static void             add_queue(),
29                         del_queue(),
30                         sort(),
31                         sort_elements();

32 static void             init_semaphore(),
33                         inc_semaphore(),
34                         dec_semaphore();
```

```
35 void m_qsort(base, nel, sizeof_base, compar)
36     char        *base;
37     unsigned    nel,
38                 sizeof_base;
39     int         (*compar)();
40 {
41     if ((swap = (char *) malloc(sizeof_base)) == NULL
42             || (median = (char *) malloc(sizeof_base)) == NULL) {
43         fprintf(stderr, "m_qsort: memory allocation failed0);
44         exit(1);
45     }

46     s_init_lock(&queue_lock);
47     s_init_lock(&work_lock);
48     init_semaphore(&wait_queue);
49     add_queue(base, nel, sizeof_base);  /* Setup first partition */
50     m_fork(sort, compar);               /* Start the children */
51     m_kill_procs();                     /* All done - clean up */
52     free(median);
53     free(swap);
54 }

55 static void init_semaphore(semaphore)
56     register sem_t *semaphore;
57 {
58     s_init_lock(&semaphore->sem_lock);
59     s_init_lock(&semaphore->sem_wait);
60     semaphore->sem_cnt = 0;
61 }

62 static void inc_semaphore(semaphore)
63     register sem_t *semaphore;
64 {
65     s_lock(&semaphore->sem_lock);
66     semaphore->sem_cnt++;
67     s_unlock(&semaphore->sem_wait);     /* Release anything waiting */
68     s_unlock(&semaphore->sem_lock);
69 }

70 static void dec_semaphore(semaphore)
71     register sem_t *semaphore;
72 {
73     while (1) {
74         s_lock(&semaphore->sem_lock);
75         if (semaphore > 0) {
76             semaphore->sem_cnt--;
77             s_unlock(&semaphore->sem_lock);
78             return;
79         }
80         s_unlock(&semaphore->sem_lock);
81         s_lock(&semaphore->sem_wait);   /* Wait here if all used up */
82     }
83 }
```

```
 84 static void add_queue(base, nel, sizeof_base)
 85     char        *base;
 86     unsigned    nel,
 87                 sizeof_base;
 88 {
 89     register queue_entry *queue_element;

 90     if ((queue_element =
 91               (queue_entry *) shmalloc(sizeof(queue_entry))) == NULL) {
 92         fprintf(stderr, "add_queue: shared memory allocation failed0);
 93         exit(1);
 94     }

 95     queue_element->base = base;
 96     queue_element->nel = nel;
 97     queue_element->sizeof_base = sizeof_base;
 98     queue_element->next = NULL;

 99     s_lock(&queue_lock);
100     if (queue_head == NULL)
101         queue_head = queue_element;
102     if (queue_tail != NULL)
103         queue_tail->next = queue_element;
104     queue_tail = queue_element;
105     s_unlock(&queue_lock);

106     s_lock(&work_lock);
107     work_count++;
108     s_unlock(&work_lock);

109     inc_semaphore(&wait_queue);         /* Let go any waiting for work */
110 }

111 static queue_entry *get_queue()
112 {
113     register queue_entry *queue_element;

114     while (work_count > 0) {            /* Any work still to do ? */
115         s_lock(&queue_lock);
116         if (queue_head != NULL) {       /* Check if any ready now */
117             queue_element = queue_head;
118             queue_head = queue_head->next;
119             if (queue_head == NULL)
120                 queue_tail = NULL;
121             s_unlock(&queue_lock);
122             return(queue_element);
123         }
124         s_unlock(&queue_lock);
125         dec_semaphore(&wait_queue);     /* Nope - just sleep */
126     }
127     inc_semaphore(&wait_queue);         /* None left - wake someone else */
128     return(NULL);
129 }
```

```
130 static void del_queue(queue_element)    /* Free completed partitions */
131     queue_entry *queue_element;
132 {
133     shfree(queue_element);
134     s_lock(&work_lock);
135     work_count--;
136     s_unlock(&work_lock);
137 }

138 static void sort(compar)
139     int         (*compar)();
140 {
141     register queue_entry *queue_element;

142     while ((queue_element = get_queue()) != NULL) {
143         sort_elements(queue_element->base, queue_element->nel,
144                 queue_element->sizeof_base, compar);
145         del_queue(queue_element);
146     }
147 }

148 /* One quick sort step */
149 static void sort_elements(base, nel, sizeof_base, compar)
150     char        *base;
151     unsigned    nel,
152                 sizeof_base;
153     int         (*compar)();
154 {
155     char        *left,
156                 *right,
157                 *end;

158     left = base;
159     end = right = base + (nel - 1) * sizeof_base;
160     bcopy(base + (nel / 2) * sizeof_base, median, sizeof_base);

161     while (left <= right) {
162         while ((*compar)(left, median) < 0)
163             left += sizeof_base;
164         while ((*compar)(median, right) < 0)
165             right -= sizeof_base;
166         if (left <= right) {
167             if (left != right) {
168                 bcopy(left, swap, sizeof_base);
169                 bcopy(right, left, sizeof_base);
170                 bcopy(swap, right, sizeof_base);
171             }
172             left += sizeof_base;
173             right -= sizeof_base;
174         }
175     }
176     /* Only difference - add to queue instead of doing recursion */
177     if (right > base)
178         add_queue(base, (right - base) / sizeof_base + 1, sizeof_base);
179     if (end > left)
```

```
180         add_queue(left, (end - left) / sizeof_base + 1, sizeof_base);
181 }
182
```

;login:

The USENIX Association Newsletter

Volume 13, Number 2 **March/April 1988**

CONTENTS

The closing date for submissions for the next issue of *;login:* is April 29, 1988

USENIX THE PROFESSIONAL AND TECHNICAL UNIX® ASSOCIATION

Fifth Workshop
on
Real-Time Software and Operating Systems

Omni-Shoreham Hotel
Washington, DC

May 12-13, 1988

Sponsored by

The IEEE Computer Society

The USENIX Association

This year's workshop broadens the scope to include general real-time systems. This workshop will bring together researchers, designers, and implementers of real-time operating systems and software. There will be a substantial emphasis on practical experience, so workers from industrial organizations are encouraged to attend. Topics of specific interest include:

- Primary requirements of real-time systems
- Distributed real-time operating systems
- Application-specific operating systems
- Practical experiences and implications
- Exotic applications: medicine, music, etc.
- Architectural support for real-time
- Language, programming support, and reusability
- Types of real-time constraints
- Scheduling and resource management
- Predictability, adaptability, and maintainability
- Reliability and fault tolerance
- Instrumentation and performance measurement
- Case studies

The format of the workshop will be geared to encourage intense technical interactions and focussed discussions.

Those wishing to attend this workshop should contact Lui Sha at the address below immediately. Attendance will be limited and there are few spaces available.

Program Co-chairs:

Dr. Marc Donner
IBM Research
P.O. Box 218
Yorktown Heights, NY 10598
(914) 945-2032
donner@ibm.com

Dr. Lui Sha
Computer Science Department
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213
(412) 268-7668
sha@k.gp.cs.cmu.edu

Call for Papers
UNIX Security Workshop

Portland, Oregon

August 29-30, 1988

Matt Bishop is the chair for the UNIX Security Workshop to be held in Portland, Oregon, on Monday and Tuesday, August 29th and 30th, 1988. This workshop will bring together researchers in computer security dealing with UNIX and system administrators trying to use UNIX in environments where protection and security are of vital importance. It is believed these people battle many of the same problems repeatedly and can share their unique solutions to some problems in order to avoid duplication of effort in making UNIX secure enough for their needs. It is intended that each participant will present briefly unique attributes of his/her environment and/or research and contribute a short (five minute) discussion (and paper) detailing some solution from their environment or work.

Some topics to be considered include: password security (password file integrity, enforcing choice of a safe password, spotting and handling crackers), network security (problems arising from logins over an unprotected Ethernet, containing a break-in to one machine in a networked environment), file system security (auditing packages, security in an NFS environment), new designs to obtain C-level (or better) certification, making existing UNIX systems more secure, and locating and fixing UNIX security problems.

This gathering will follow a "workshop" format rather than a "paper presentation" format. Each participant will submit electronically to the chair a one or two page summary describing a solution to some problem. The summary should contain a description of the problem and a description of the solution detailed enough that fellow researchers and administrators can implement or use it. Also, include with your submission five (or so) topics that you'd like to hear about.

The workshop chair will collate the papers to schedule sessions for appropriate audiences. It is anticipated that some sessions will include all participants; some will be for smaller groups. Send your submissions to the chair by noon, EST **July 1, 1988**.

For further details on the workshop:

Matt Bishop
Dept. of Mathematics & Computer Science
Bradley Hall
Dartmouth College
Hanover, NH 03755

(603) 646-3267
{ihnp4,decvax}!dartvax!bear!bishop
bishop%bear.dartmouth.edu@relay.cs.net

For details about registration, contact:

USENIX Conference Office
P.O. Box 385
Sunset Beach, CA 90742

(213) 592-1381 or 592-3243
{uunet,ucbvax}!usenix!judy

Call for Papers
Workshop on UNIX and Supercomputers

Westin William Penn Hotel
Pittsburgh, Pennsylvania

September 26-27, 1988

Sponsored by the USENIX Association

A large number of supercomputers are now or will in the future be running UNIX as their primary operating system. This is the first workshop to consider the general problems of running UNIX on supercomputers, and will cover topics both practical and abstract. Areas of specific interest include but are not limited to:

Systems administration
Archiving
Scheduling
File systems
Networking and network protocols
Job batching systems
Monitoring performance/parallelism
Programming languages and environments
Fast file I/O
Shared memory management
IPC
Very large files
Checkpoint-restart

The workshop will include both shorter presentations and full-length papers, and there will also be tours of Pittsburgh Supercomputing Center and Westinghouse Energy Center facilities and a reception at the Pittsburgh Supercomputing Center. Workshop proceedings will be available at the Workshop.

If you are interested in presenting either a full paper or a brief discussion of your current work, please send an abstract of your paper or presentation to Melinda Shore by **July 15, 1988**. If you are sending your submission by US Mail, please send three copies. All submissions will be acknowledged.

Program Co-chairs:

Lori Grob
NYU Ultracomputer Research Lab
715 Broadway, 10th Floor
New York, NY 10003

(212) 998-3339
grob@lori.ultra.nyu.edu

Melinda Shore
Pittsburgh Supercomputing Center
4400 Fifth Avenue
Pittsburgh, PA 15213

(412) 268-5125
shore@reason.psc.edu

EUUG Spring 1988 Conference

London

April 11-15, 1988

The UKUUG will host the Spring '88 European UNIX systems User Group Technical Conference at the Queen Elizabeth II Conference Center in London. Technical tutorials will be held on April 11 & 12, followed by the three day conference.

For further information, contact the EUUG Secretariat at the address below.

Call for Papers: EUUG Autumn Conference

Portugal

October 3-7, 1988

The Autumn '88 European UNIX systems User Group Technical Conference will be held in southern Portugal. Technical tutorials will be held on October 3 & 4, followed by the three day conference.

The theme of the conference is "New Directions for UNIX." The EUUG invites abstracts from those wishing to present their work. Submissions from students are particularly encouraged under the EUUG Student Encouragement Scheme, details of which are available from the EUUG Secretariat. All submitted papers will be refereed. Abstracts must be submitted by **post** to the EUUG Secretariat.

The Programme Chair will be pleased to provide advice to potential speakers.

Deadlines are:

Receipt of abstract	30 April
Acceptance notification	15 May
Final paper received	1 August

Those interested in offering a tutorial should contact the EUUG Tutorial Officer as soon as possible.

For further information about this and future EUUG events, contact the Secretariat.

Secretariat

EUUG
Owles Hall
Owles Lane
Buntingford, Herts. SG9 9PL
United Kingdom

Phone: (+44) 763 73039
Fax: (+44) 763 73255 (G2)
Email: euug@inset.uucp

Tutorial Officer

Neil Todd
IST
60 Albert Court
Prince Consort Road
London SW7 2BH
United Kingdom

Phone: (+44) 1 581 8155
Fax: (+44) 1 581 5147 (G3)
Telex: 928476 ISTECH G
Email: neil@ist.co.uk

Programme Chair

Peter Collinson
Computing Laboratory
University of Kent
Canterbury, Kent CT2 7NF
United Kingdom

Phone: (+44) 227 764000, x7619
Email: pc@ukc.ac.uk

Future Events

EUUG Spring Conference
London, April 11-15, 1988

See page 19.

Real-Time Operating Systems Workshop
Washington, DC, May 12-13, 1988

Sponsored by USENIX and the IEEE. The Program Chairs are Marc Donner and Lui Sha. See page 16.

USENIX 1988 Summer Conference and Exhibition San Francisco, June 20-24, 1988

The 1988 Summer Conference will be held at the Hilton Hotel in San Francisco. There will be a conference, tutorials, and vendor exhibits. You will receive a registration packet in late April.

UNIX Security Workshop
Portland, OR, Aug. 29-30, 1988

The Program Chair is Matt Bishop of Dartmouth College. See page 17.

UNIX and Supercomputers Workshop
Pittsburgh, PA, Sept. 26-27, 1988

The Program Chairs are Melinda Shore of the Pittsburgh Supercomputer Center and Lori Grob of New York University. See page 18.

EUUG Autumn Conference
Portugal, Oct. 3-7, 1988

See page 19.

C++ Miniconference
Denver, CO, Oct. 17-20, 1988

The Program Chair is Andy Koenig of AT&T. Information will be available in the next issue of *;login:*.

Large Installation
System Administration II
Monterey, CA, Nov. 17-18, 1988

The Program Chair is Alix Vasilatos of MIT's Project Athena. Information will be available in the next issue of *;login:*.

Fifth Annual Computer GO Tournament

The fifth annual USENIX Computer Go Tournament and Championship will be held on Wednesday, June 22, during the USENIX conference in San Francisco. All interested parties are invited to submit programs. The tournament rules will be essentially those established for the first USENIX Computer Go Tournament.

Sequent Computer will provide a Symmetry S81, which results in two major changes in the rules for this year. First, program time will be measured by clock time, not CPU time. This allows entrants to take advantage of the Sequent's multiple processors. Second, contestants may provide their own hardware such as IBM PCs; such entries must communicate with the Sequent machine over a 9600 baud RS-232 link, using the usual ASCII protocol to talk to the referee.

Conference attendees may bring programs to submit with them as long as they get in touch with Rob Pike NO LATER THAN noon on June 21 (preferably earlier). He can be reached through the Conference Office. Those unable to attend the conference who would like to enter programs can do so by sending a compilable source to one of the addresses below.

Comments, programs, or requests for more information (including details of the Sequent computing environment) can be sent via electronic mail to UUCP!research!rob or ARPA!rob@att.com. U.S. Mail should be sent to:

Rob Pike
Bell Labs 2C524
Murray Hill, NJ 07974 USA

An Update on UNIX and C Standards Activities

Shane P. McCarron, NAPS Inc.

January 21, 1988

Overview

The Standards community isn't necessarily a closed entity, but it is one that is hard to look into. There are so many different activities going on all over the place that it is difficult for most people to get involved. I suppose this is as it should be, since if everyone were involved, nothing would ever get accomplished. However, it is always good to know what is going on at a macro level, even if the details pass you by.

That is where this report comes in – I am going to try to summarize what has transpired in the UNIX and C standards areas during the past three months. As anyone who has been involved in a standards committee can tell you, not a lot will happen in a quarter in any one committee, but over several committees the cumulative effect can be daunting.

Before I start summarizing what went on in the last quarter of 1987, I should define the scope of this report. I am not going to try to touch on all of the technical discussions that go on. These are often boring, and if you have that level of interest, you should really be on the mailing list for the group in question. Instead, I am going to give an overview of some of the key issues that were raised and the important milestones that were reached or passed.

In addition to the activity at the December meetings of P003, a few other things happened that are worth noting:

P1003.1 Final Ballot

On November 15th the P1003.1 document went out for its full use ballot. The balloting period was 30 days, and closed around December 15th. When ballot resolution is completed, the first full use standard from a 1003 group will have been ratified. This should be around March, 1988.

[Unfortunately, the first 1003.1 ballot resolution procedure failed due to logistical difficulties partly caused by a too-short time period. A new resolution period is currently planned for April, meaning the IEEE 1003.1 Full Use Standard will probably be ratified in June. –jsq, 4 March 1988]

New P1003 Working Groups

There are three new working groups under the P1003 committee (.0, .5, and .6). Since I haven't talked about all of these before, here is a list of all of the POSIX working groups:

1003.0 – POSIX Guide
1003.1 – Systems Interface
1003.2 – Shell and Tools Interface
1003.3 – Verification and Testing
1003.4 – Real Time
1003.5 – Ada Binding for POSIX
1003.6 – Security

IEEE Standards Board

At the December meeting of the IEEE Standards Board, the Board approved the IEEE Technical Advisory Group Procedures document. This was a major event in that it allowed the first meeting of the United States TAG on POSIX to take place "in wedlock."

US Technical Advisory Group on POSIX

The first meeting of the US TAG on POSIX was held in conjunction with the P1003 meetings in December. A TAG is a group that exists in each International Standards Organization (OSI) member country that is interested in a particular ISO working group (in this case, WG15 of Subcommittee 22). The TAG recommends to the ISO standards body for that topic in that country what the country's position should be on the issue. In this case the standards body is the IEEE, and the issue is POSIX. In a future report, I hope to spend more time talking about what it means to be in the International Standards Organization, and how it affects POSIX.

Since it was the first meeting, the members present elected a chair and secretary, and learned about what it means to be a TAG. In addition to this, the TAG established what the US position on POSIX should be. Basically this boils down to "The US recommends that POSIX be accepted as a Draft Proposed Standard, but any changes made to the standard by IEEE P1003.1 should be incorporated into the ISO document." It would be very bad form not to recommend our own standard.

C Language

C Language Standard

In addition to the P1003 standards activities, the work of the X3J11 standards committee holds particular interest for people in the UNIX community. This is the group that is defining the ANSI X3.159 C Language Standard. They have been working on this for quite a while now, and are very close to resolution. They went into their first public review period last spring, and have just recently finished responding to all of the comments that were submitted at that time.

Based on information I have about the December meeting of X3J11, here is what is happening in the future:

- Around January 8th, 1988 the second public review draft will be completed.
- Soon after that, the second (2 month) public review period will begin. As with last time, the standard will be available to the public through Global Press in Washington, DC.
- This public review will close in time for the comments to get out to the committee members before the April meeting.
- At that meeting, the committee will break down into subgroups and review the comments. There will be great resistance to making any substantive (non-editorial) changes to the standard. If there are any substantive changes made, it will result in another public review period, which will delay the standard for at least one calendar year.
- Assuming that there are no substantive changes to the standard after the next public review period, there should be a ratified standard before the end of 1988.

If the C Language Standard can be completed before the end of the year, it could mean a lot for POSIX system implementors. Since the .1 standard will not be really a standard until June, it is unlikely that vendors will be able to complete an implementation before the end of 1988 in any event. If they could release a system that supported both Standard C and POSIX, it would be a real shot in the arm for application developers. A delay of another year on Standard C would mean that application developers must write code under POSIX that could very well be broken under an ANSI C conforming compiler.

NBS FIPS

NBS POSIX FIPS

One other item that is of concern to system implementors and application developers alike is that the National Bureau of Standards (NBS) is going to announce a POSIX Federal Information Processing Standard (FIPS) this month. This FIPS will be used by most federal agencies when drafting Request for Proposals (RFPs) for many classes of applications.

Just what is NBS going to require? Well, the NBS POSIX FIPS is based on POSIX D12, the draft that went out to the balloting group. The final POSIX standard may be considerably different than this, but NBS has assured the .1 working group that they will incorporate the substantive changes in the standard into their FIPS when the standard is complete.

So, if NBS is going to specify POSIX as the FIPS, what are we worried about? Well, in order to increase consensus and support as many existing implementations as possible, POSIX has a lot of "options" in it. NBS felt that these "options" made it difficult for applications developers to write applications that used the nice facilities of POSIX (they are right), so they are requiring that many of these options be included in a FIPS conforming implementation. For systems implementors, this means that you had better include all of these options if you want to sell to the federal government. For applications developers, it means that if your customer base is the federal government, you can use these facilities without fear – they will be there.

What are these options? Well, the following is an excerpt from the NBS POSIX FIPS draft specification.

As an aside, it is important to note that many of these so-called "options" are not really options at all, but rather cases in which there was some ambiguity as to how the system would function. I will indicate in the following list some examples of real options and their opposites for clarity.

- The term "appropriate privileges" shall be synonymous with the term "super-user."

This is not really an option, but rather a clarification being introduced by the NBS people. The term "appropriate privileges" was introduced into the standard to provide for secure implementations of POSIX. By indicating that certain facilities of POSIX require "appropriate privileges," the door was left open for implementations where processes could have subsets of the power normally granted to a monolithic "super-user." In fact, the above requirement is incorrect. You could not simply replace the term "appropriate privileges" with the term "super-user" throughout the standard and have it make any sense. However, we get the idea.

- A null pathname shall be considered invalid and generate an error (2.10.3, lines 894-896).

- The use of the `chown()` function shall be restricted to a process with super-user privileges (2.10.4, lines 924-926).

This is an example of a real option in POSIX. If the macro _POSIX_CHOWN_RESTRICTED is defined, it means that only a process with "appropriate privileges" can change the owner of a file. This is in conflict with the current System V definition of how `chown` works, but is more in line with trusted implementations. Users should not be able to "give away" files.

- Only the super-user shall be allowed to link or unlink directories (2.10.4, lines 938-939).

Another useful option. A portable application may need to know whether it requires "appropriate privileges" to move directories around.

- The owner of a file may use the `utime()` function to set file timestamps to arbitrary values (2.10.4, lines 943-945).

- The implementation shall support a value of {NGROUPS_MAX} greater than or equal to eight (8) (2.9.2). An implementation may provide an option for setting {NGROUPS_MAX} to a value other than eight (8).

The POSIX standard is still in the ballot resolution process. When it went to ballot it defined the BSD-style supplementary groups feature. This says that there is a group-id associated with a process, but that there may be additional, supplementary groups also.

As of this writing, the definition has been changed to a more flexible definition. There will now be an array of group IDs associated with a process. Although this change has not been accepted by the full balloting group yet, I think that it will be.

- The implementation shall support the setting of the group-ID of a file (when it is created) to that of its parent directory (2.10.4, lines 934-937). An implementation may provide a programmable selectable means for setting the group-ID of a file (when it is created) to the effective group-ID of the creating process.

This is another example of a true option. Here the FIPS is specifying the BSD method of creating files. This method makes a lot of sense in a multiple group per process environment. However, they also allow the System V behavior.

- The use of `chown()` shall be restricted to changing the group-ID of a file to the effective group-ID of a process or when {NGROUPS_MAX} > 0, to one of its supplementary group-IDs (2.10.4, lines 927-930).

- The `exec()` type functions shall save the effective user-ID and group-ID (2.10.3, lines 902-903).

This mirrors the System V behavior.

- The `kill()` function shall use the saved set user-ID of the receiving process instead of the effective user-ID to determine eligibility to send the signal to a process (2.10.3, lines 891-893).

This is also similar to System V.

• When a session process group leader executes an `exit()` a SIGHUP signal shall be sent to each member of the session process group (2.10.3 lines 880-883).

• The terminal special characters defined in Sections 7.1.1.10 and 7.1.2.7 can be individually disabled by using the value specified by _POSIX_V_DISABLE (2.10.4, lines 946-949; 7.1.1.10; 7.1.2.7).

• The implementation shall support the _POSIX_JOB_CONTROL option 2.10.3, lines 884-886).

Although I have not described how job control works under POSIX, suffice it to say that it is confusing at best. The ballot resolution group is still trying to decide how to resolve the problems pointed out during balloting.

• The implementation shall provide a single utility for reading and writing POSIX data interchange format files (10.). This utility shall be capable of reading USTAR and CPIO data interchange formats without requiring the format to be specified. The implementation shall write CPIO data interchange format when no option on format type is specified.

• Pathnames longer than {NAME_MAX} shall be considered invalid and generate an error (2.10.4, lines 940-942).

• When the `rename()`, `unlink()` or `rmdir()` function is unsuccessful because the conditions for [EBUSY] occur, the implementation shall report the [EBUSY] errno (5.5.1.4, lines 481-482; 5.5.2.4, lines 523-524; 5.5.3.4, lines 593-594).

• When the `rename()` function is unsuccessful because the conditions for [EXDEV] occur, the implementation shall report the [EXDEV] errno (5.5.3.4, lines 593-594).

• When the `fork()` or `exec()` type function is unsuccessful because the conditions for [ENOMEM] occur, the implementation shall report the [ENOMEM] errno (3.1.1.4, line 54; 3.1.2.4, lines 175-176).

• When the `getcwd()` function is unsuccessful because the conditions for [EACCES] occur, the implementation shall report the [EACCES] errno (5.2.2.4, lines 148-149).

• When the `chown()` or `wait2()` function is unsuccessful because the conditions for [EINVAL] occur, the implementation shall report the [EINVAL] errno (3.2.1.4, line 272; 5.6.5.4, line 857).

• The implementation shall detect an [EFAULT] errno condition (2.5, lines 554-558). The implementation must state as part of the required documentation: (1) the conditions when an [EFAULT] is detected and an [EFAULT] errno is generated, and (2) those conditions, if any, when [EFAULT] may not be detectable.

• The `tcsetattr()` function shall only set the parameters supported by the underlying hardware associated with the terminal (7.2.1.2, line 502).

• An interrupted `write()` function shall return a count of the number of bytes successfully transferred from the application program to the system (6.4.2.2, lines 195-196; 6.4.2.4. lines 240-242).

• An implementation may provide errno [ENOEXIST] in place of errno [EACCES].

• A POSIX FIPS implementation shall successfully PASS the NBS-PCTS validation suite.

From all of these options, I am sure that it is obvious that there is room for considerable variation in the POSIX standard. The FIPS goes a long way towards firming up an otherwise wishy-washy document. Since many system implementors want to sell to the US Government, it is probable that all of the above requirements will be available on a majority of POSIX conforming systems. This is excellent news for application developers who want to take advantage of some of the additional facilities introduced in POSIX as optional.

Status of the IEEE P1003 Working Groups

1003.1 – System Services Interface

The .1 working group has reached an interesting point in its life. Since the standard they have produced is now in final ballot and ballot resolution, the working group in effect has nothing more to do. At the December meeting they tried to decide what, if anything, should be done by this body in the future. Although no decision on this was made, many good options were suggested.

Most promising among these is the design of a language independent description of POSIX. One of the requirements that ISO made of POSIX when it was adopted as a Draft Proposed Standard last fall was that at some point in the future it be described in such a way that the functionality could be understood without an understanding of the C language. ISO recognized that it was unrealistic to make this a requirement before adopting the standard, but felt that it was reasonably important. I feel that this is something the working group will be taking on soon after the Full Use Standard is approved by IEEE.

1003.2 – Shell and Tools Interface

The Shell and Tools group is operating under a very ambitious schedule. The National Bureau of Standards (NBS) has indicated that they are going to declare a Federal Information Processing Standard (FIPS) based on the command set in the .2 standard, and that they are going to do so in the summer of '88. This working group only started serious work one year ago, and has already produced a larger document than the .1 group did in four. The group is working hard to make sure that the command set is locked down before the deadline being imposed by NBS.

Unfortunately, this has the consequence that many decisions are being made as rapidly as possible. I am afraid that the resulting standard may be flawed, if only because the group is moving forward too fast. On the other hand, the .1 group was guilty of exactly the opposite, and NBS pressure has forced that group to really get its act together. It has proven to be a boon there, and it may do so here as well.

The Shell and Tools group has a milestone schedule something like:

Date	*Milestone*
Mar '88	Command Selection frozen; 75% described.
Jun '88	100% commands described; functional freeze.
Oct '88	Clean-up, slack; produce "mock ballot" for draft (#8); international signoff.
Jan '89	Resolve mock objections; produce balloting draft (#9).
Apr '89	Resolve ballot objections; produce final standard.
Jul '89	Final standard approved by IEEE.

This may not appear to be all that hectic a pace, but I can assure you that it is. When I say that the commands are 100% described, it means that the current functionality of each command that has been included in the standard (a substantial part of the current UNIX command set) is described in painful detail. The goal of the standard is to describe each command in such a way that a person who has never seen a UNIX machine can write the commands from scratch. It's a lot of text. With about 75% of the commands in, and those being about 75% described (albeit incorrectly in some cases) the document is now approaching 400 pages. In a future report I will tell you just what is involved in a command description. We don't have the space this time.

1003.3 – Testing and Verification

This is another group that has been very active in the last year or so. They have the dubious honor of figuring out how to test whether implementations of the .1 standard are actually conforming. Although the IEEE is not going to be providing any validation services or rating and systems, P1003 thought that it was important that they define what parts of the system should be tested in what ways.

The .3 group seems to be on track for balloting within the next 6 to 9 months. Their work is very far along, and a verification suite is already being worked on by the NBS based on the .3 assertion list about POSIX. Although the .3 document will not be as earth-shattering as POSIX, it is a still a very important step – actually showing how to test conformance to a standard at the same time you are defining one.

1003.4 – Real Time

Until recently, all the real time considerations in POSIX were being looked into by a /usr/group technical committee. Last fall that committee decided that their research was mature enough that they could actually start the work of producing a standard about it. The real time work promises to add much of the functionality that I and many others feel is absolutely necessary in POSIX; things like semaphores, shared memory, event processing, and other inter-process communication mechanisms that were left out of the .1 standard because they just did not have the time.

Unfortunately, there is quite a bit of dissension as to how all of these things should be implemented. Not just IPC, but also contiguous files, timers, and those things that a real time application would need to really be real time. After talking to some of the people who attended the December meeting, I would guess that this group has a long way to go.

However, what will happen when they get there? At this time I'm guessing that the .4 document will be positioned as a supplement to the .1 standard. It should require no changes to the .1 standard, and will probably be a set of optional facilities, as job control and some others are already. When this standard is finally produced, it will answer many of the objections we have heard to POSIX all along. I am sure that it will be well received. Let's hope that it can be timely enough to be useful.

Publications Available

The following publications are available from the Association Office. Prices and overseas postage charges are per copy. California residents please add applicable sales tax. Payments **must** be enclosed with the order and **must** be in US dollars payable on a US bank.

The EUUG Newsletter, which is published four times a year, is available for $4 per copy or $16 for a full-year subscription.

The July 1983 edition of the EUUG Micros Catalog is available for $8 per copy.

Conference and Workshop Proceedings

Meeting	Location	Date	Price	Overseas Mail Air	Surface
USENIX	Dallas	Winter '88	$20	$25	$5
C++ Workshop	Santa Fe	November '87	$15	$25	$5
Graphics Workshop IV	Cambridge	October '87	$10	$15	$5
USENIX	Phoenix	Summer '87	$20	$25	$5
USENIX	Wash. DC	Winter '87	$10	$25	$5
Graphics Workshop III	Monterey	December '86	$10	$15	$5
USENIX	Atlanta	Summer '86	$10	$25	$5
Graphics Workshop I	Monterey	December '84	$ 3	$ 7	$5

NOTICE: There was a paper inadvertently omitted from the Dallas Proceedings. It will appear in the May/June issue of *;login:*. –Ed.

Want to get Published?

O'Reilly & Associates is looking for knowledgeable authors to write new titles for its series of Nutshell Handbooks.

The handbooks are generally short, focused treatments of a specific topic. Current titles include:

Reading and Writing Termcap Entries
Programming with Curses
Using UUCP and Usenet
Managing Projects with Make
Learning the vi Editor

ORA is interested in just about any topic, but here are some of the things that are high on their list:

device drivers
SCCS
interprocess communication
mailers
shell programming (sh, csh, ksh)
debugging (adb, sdb, dbx)

It is essential that titles be user-oriented, whatever the audience level. ORA is looking for books that present and solve real problems, rather than ones that simply present the facts.

You need not be a first-rate writer; ORA is prepared to edit your manuscript heavily if necessary. They'd rather work with experts who know their stuff than hack writers who can put together a smooth but superficial treatment.

ORA is interested in buying all rights for a one-time fee (with the amount depending on size and salability of the manuscript, as well as how much editing they think it needs).

If you have a handbook in you, and would like to see your name in print, send them a description and outline of the book you have in mind. Phone calls are ok, but e-mail or USPS queries are preferred.

Tim O'Reilly
O'Reilly & Associates, Inc.
981 Chestnut Street
Newton, MA 02164

(617) 527-4210
uunet!ora!tim
tim@ora.uu.net

–PHS

Interested in China?

Communication International is a venture involved in personnel exchanges between the Peoples Republic of China and the US. CI approached the USENIX Association for assistance.

The Chinese personnel exchange committee is interested in American firms and educational institutions that might be interested in promoting relationships by hosting an individual for varying periods of time. CI services and fees vary with length and type of arrangement.

Many of the Chinese have expertise in computer science and are interested in placements in both large and small companies or educational institutions. After a period of up to a year in an American setting, the Chinese visitors will return home and their US counterparts may accompany them for either short or long-term exchanges.

In addition to encouraging business contacts and relationships, these exchanges build trust for further development and business and academic relationships. The Chinese are often interested in long-term arrangements, e.g. for periods like 30 years.

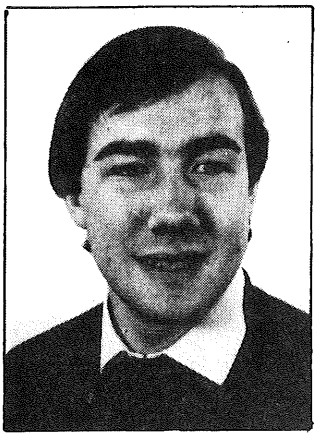
For further information, contact:

Communication International
1545 Green Street
San Francisco, CA 94123
(415) 499-5772

EUROPEAN UNIX SYSTEMS USER GROUP NEWSLETTER

Volume 8
Number 1
Spring 1988

Security of Ethernet Under UNIX and Internet Protocol



Marcin Skubiszewski
skubi@ens.ens.fr
skubi@frulm63.bitnet

Ecole Normale Supérieur
45 rue d'Ulm
75005 PARIS
FRANCE

Marcin Skubiszewski is a computer science student at the Ecole Normale Superieure and the University of Paris – Orsay, and this year is the last before the beginning of his doctoral studies.

Last summer he worked on the Internet drivers of Berkeley UNIX in the MASI laboratory.

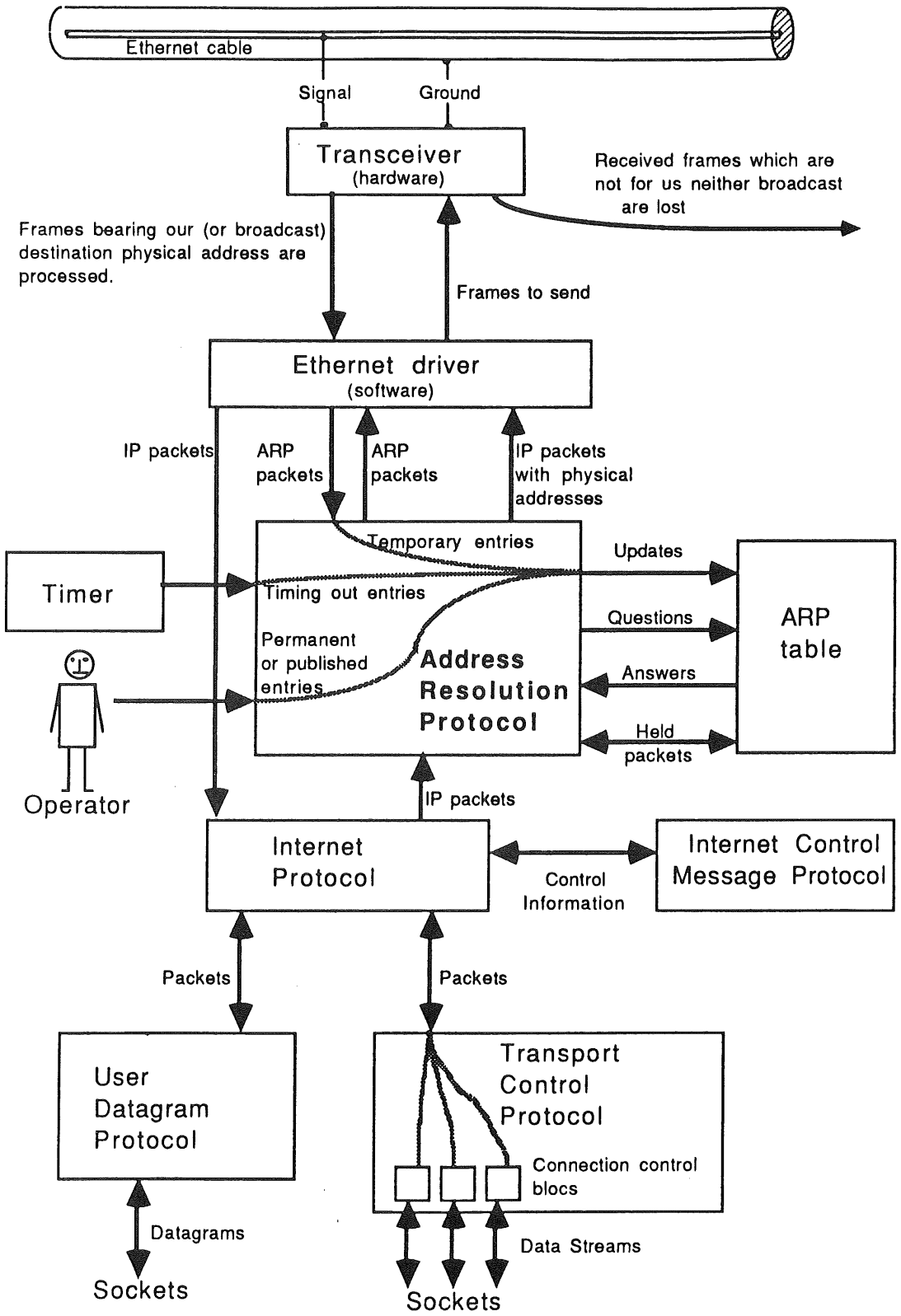
1. Introduction

An Ethernet is simply a cable (similar to TV aerial feeders) able to carry electric signals at the speed of 10 Mbits/second; all the stations (i.e. hosts) are connected to it *in parallel*. This organisation makes the network naturally insecure for two reasons. First, every frame can be illegally read by any station on the network (to be kept secret, communications must be encrypted). Second, there is no way to check which host is the actual sender of a given frame: the sender has to identify itself by filling the *source address* field in the frame and it is impossible for the destination host to verify whether this information is true or not. The possibility to lie about one's identity is an important defect of Ethernet, because this imposture enables one to obtain privileges granted to another host. On many hosts, it is possible to log in as root (without password) by this means. Fortunately, it appears to be possible to modify protocols used on Ethernet in order to identify stations with reliability. Proposing such modifications is the main goal of this paper.

This paper is the result of a research made in the laboratory MASI (Méthodologie et Architecture des Systèmes Informatiques), attached to the University Paris 6. I experimentally proved that the most widespread existing implementation of Internet, the 4.3 BSD UNIX, is insecure.

2. Who Are The Potential Ethernet Hackers?

Under well designed systems (like Berkeley UNIX) ordinary users have access to Ethernet only through sophisticated protocols implemented in the kernel.



The Ethernet software
(the ARP is shown in detail)

So, their access to the network is strictly controlled and, in my mind, they have absolutely no way to break down its security. The only ones who are capable of Ethernet hacking are super-users of machines connected to the network and users of some exotic (e.g., experimental or old-fashioned) systems. A super-user is not submitted to any limitation when using Ethernet: his machine gives him a large freedom and, when this freedom is insufficient, a super-user has the possibility of modifying and re-compiling the relevant part of the kernel (this operation is easy for a system programmer). Users of exotic machines have the opportunity to hack Ethernet only if their systems give them insufficiently controlled (or even raw) access to the network: in this case they are similar to super-users. For instance, an IBM PC with an Ethernet transceiver is suitable for hacking if one has access to the necessary software, which can be bought.

The number of potential Ethernet hackers of this kind is low on most networks and the most popular Ethernet implementation, the 4.3 Berkeley UNIX, has no protection against them. However, implementing some protection algorithms seems to be useful, mainly for two reasons: first, super-users are becoming numerous on some networks due to the development of work-stations; second, when the network is insecure, a hacker who becomes super-user on one machine gets the opportunity to become super-user on all the machines.

3. Problems with Keeping Communications Secret

As it has been already mentioned, there is no way to stop an Ethernet station from receiving any data sent on the network. Encrypting all data would be the only fully satisfying solution to the problem. However, encrypting and decrypting involves so much computing power that this solution is unacceptable. Another, more realistic, approach consists in encrypting only critical information. This solution would be acceptable from the point of view of the involved computing power, but it would be rather complicated. For instance, users would have to mark their data as critical or not (the same connection can contain critical and ordinary data, e.g. a password followed by a large file); this would involve changes in existing application programs and a great attention on the part of users. An extra encryption scheme (using publishable keys) would then be necessary in order to exchange encryption keys.

Since encrypting is so complicated, we will not propose in this paper any method of protection based on encryption. Besides, no encryption schemes are being currently used. This implies that passwords *should not be sent on an Ethernet*. Instead, the privileges a remote user can obtain on a host should only depend on his identity (verified by his own host). This requires only few changes in the existing UNIX systems when all the involved hosts use UNIX. However, when UNIX machines communicate with other machines, some substantial changes are necessary.

When all hosts use UNIX, the first thing to do would be to maintain complete files of equivalent remote hosts and accounts: `/etc/hosts.equiv` (the file of equivalent hosts) and `.rhosts` (the file of equivalent remote accounts; there is one such file per local account).[1] If these files are well maintained, remote users who can legitimately log in on a host will always have right to do it without password. However, some unaware users will continue using passwords instead of maintaining their `.rhosts` files. I suggest to make it impossible: small changes in `/bin/login`[2] and `/etc/ftpd`[3] can prevent remote users from using passwords. These modifications could be made easily, either by UNIX vendors or by individual system administrators (N.B.: the `telnet` daemon, which always requires a password and is unnecessary in a UNIX environment, should be suppressed).

When some hosts on the network do not use UNIX, there is no universal method of avoiding passwords. However, it is possible to solve the problem in many single cases. For instance, Ethernet terminal concentrators do not identify their users; therefore, when one logs in from a concentrator, one must give a

1. For `/etc/hosts.equiv` and `.rhosts`, see `man 1 rlogin` on your machine.
2. `/bin/login` is called by the `rlogin` daemon with option `-r <remote host name>`
3. `/etc/ftpd`: File Transmission Protocol Daemon.

password. It would be easy to change this procedure: these devices could identify their users in the same way as UNIX hosts do and, this being done, they would follow the ordinary UNIX `rlogin` protocol (without password) to log them in on UNIX computers.

4. Hacking by use of False Identity

4.1 Preliminary remarks

When somebody is hacking a host via Ethernet, the following situation arises: the hacker tries to obtain some privileges (for example, to log in on an account) on a host, called below his *peer*. For this purpose, the hacker establishes, from his own machine, a connection with his peer. He makes his machine to use the identity of another host, which is trusted by the peer and which is granted interesting privileges by the peer. This host will be called the *victim*.

In most cases an Ethernet is used together with the Internet Protocol (IP).[4] This implies that application programs use either the User Datagram Protocol (UDP) or the Transport Control Protocol (TCP). Let us recall some details about Internet.

Under IP each host is granted a 4 bytes long *internet address*. This address enables to identify it under IP: if a station is able to use your internet address instead of its own, it can get all of your privileges.

Concurrently with internet addresses another address family is used on Ethernet: each host has a 6 bytes long *physical address* (there is generally no similarity between the internet address and the physical address of a given machine). When receiving, the host's Ethernet transceiver uses these addresses to determine whether a given frame has to be processed by the host: if the destination address of the frame is equal to the host's physical address, or if it is broadcast, the frame is sent to the host; otherwise it is lost. This is the unique purpose of physical addresses.

A hacker who wants to obtain the privileges normally granted to a given victim V can choose between:

- using the internet address of V, together with its own physical address (let us call this "*wise mode*"). For this purpose, the hacker needs to convince his peer that the victim's physical address has changed. This mode prevents the victim from receiving packets related to the hacker's connections.
- or using both internet and physical addresses of V, i.e. becoming indistinguishable from V (let us call this "*ordinary mode*"). This method works well if the victim is down: when the hacker starts working, everybody believes that V is up again. When the victim is up, its reaction can hinder this sort of hacking.

Both possibilities are discussed below.

4.2 Hacking when the victim is up

4.2.1 "Ordinary mode" and "wise mode"

Imagine somebody illegally using both internet and physical addresses of V (the "ordinary mode"). We assume that V is up. The hacker can then use IP together with UDP or with TCP.

If he uses UDP, he can succeed in making everybody believe that he is V. But UDP is used only in few cases and a hacker using only UDP (and not TCP) is not really dangerous (for instance, he cannot log in on any host).[5]

If the hacker uses TCP, he will fail because of the victim's reaction. The communication will look like this:

- the hacker sends a `syn` (i.e. connection request) packet to his peer X. The source internet address of this packet is V (the victim's address);

4. We do not discuss other cases in this paper.

5. However, the NFS (Network File System) uses UDP. I think that it is a bad choice.

- host X sends a `syn+ack` (i.e. connection accepted) packet to the hacker. Both destination addresses of this packet (internet address and physical address) are those of the victim, so the victim receives the packet;
- the victim finds the `syn+ack` packet strange, because this packet accepts a connection which the victim never requested; so it sends a `rst` (error indication) packet to X;
- having received a `rst` packet, X closes the connection.

This shows that, under TCP, the hacker needs to prevent his victim from receiving packets related to the connection. For this purpose, he needs to make his peer X send packets to a physical address which is not the victim's one: he must use the victim's internet address together with his own physical address., i. e. he must use the "wise mode". This is possible by sending false information via Address Resolution Protocol.

4.2.2 A description of the Address Resolution Protocol (ARP)[6]

Under IP users supply internet, and never physical, addresses of hosts to which they want to send data. However, in order to send a frame, the kernel needs to know both internet and physical addresses of the destination. For this purpose, it must be able to map the internet address of the destination into its physical address. This is done thanks to a table called below the *ARP table*. This table contains, for each internet address concerned, an entry holding the corresponding physical address together with various flags.

ARP tables are automatically maintained. For this purpose, hosts exchange information through the network, via the Address Resolution Protocol (ARP). An ARP packet is either a request or a response. The meaning of a request is: *Which physical address corresponds to the internet address X?* A response says: *The internet address X corresponds to the physical address Y.* Normally, a host responds when it receives a request asking for its own address.

A request contains the sender's physical and internet addresses and the requested internet address. It is always broadcast because the sender does not yet know where it should send it. A response contains its sender's internet address together with the corresponding physical address and both addresses of the destination (the destination is normally the sender of a previous request).

When the ARP module receives a packet, the following checks are made before it is processed:

Test 1

If the packet is from us (i.e. the sender internet and physical addresses are both ours), we loose the packet and return.

Test 2

We check that the packet does not come from a host pretending to have the same internet address as we have (in that case the packet's sender internet address would be the same as ours while the sender physical address would be different from ours). If an impersonator is discovered by this mean, we loose the packet, print an error message to the operator and return.

Once these checks accomplished, the incoming packet is processed under no further conditions; we update our ARP table according to the information it holds.

4.2.3 Extra tests improving consistency of ARP

The tests described above are generally sufficient against the frequent operator error which consists of giving two different machines the same internet address: when one of these machines broadcasts its first ARP request, the other one reports immediately an error found by Test 2.

However, this test does not take into account the following feature of ARP. It is possible not to implement ARP in a host H. In this case, another host must respond every time the physical address of H is requested.

---

6. The source code for ARP is in the files `/sys/net/if_arp.h`, `sys/netinet/if_ether.h` and `sys/netinet/if_ether.c`, 4.3 BSD UNIX.

The host charged with responding to ARP requests about H is called below its *publisher*.[7] And when the same internet address is assigned (by error) to two different hosts, the hosts on which the ARP is not implemented, none of them will discover it and strange things will happen. It is necessary therefore to complete Test 2 by the Test 2a as follows:

Extra test 2a: protecting hosts without ARP

We check that the packet does not come from a host falsely pretending to have one of the internet addresses we have to publish. In that case the packet's sender internet address would be one of these which we have to publish while its sender physical address would not be the one which corresponds to it following our ARP table. If this check fails, we loose the packet, print an error message to the operator and return.

Another feature of ARP which should be taken into account in these tests is the existence of permanent ARP table entries. Ordinary entries are created and modified automatically (according to the incoming ARP packets) and they are destroyed when idle for 20 minutes; the permanent ones can be created and suppressed only by the local system administrator and never by an incoming ARP packet. Thus, the fact that these entries can, like any other ones, be updated by an incoming ARP packet, seems inconsistent. To change this, let's add the following:

Extra test 2b: protecting permanent entries

If the received ARP packet would modify a permanent entry to the ARP table (i.e. if its sender internet address corresponds to a permanent entry when the sender physical address does not correspond to the same entry) we loose the packet, print an error message and return.

4.2.4 Extra tests against hacking

The "wise mode" hacking implies sending false information about the victim's internet address. It remains undetected as long as the victim does not receive packets related to it. This happens when the hacker sends false ARP information only to his peer, without broadcasting it. The peer can avoid this situation by making sure that all information it uses is broadcast. For this purpose, it may perform the following:

Security operation 1: broadcasting ARP information

If a modification of our ARP table results in receiving an ARP packet which was not broadcast, we broadcast a copy of that packet.

The broadcast copy of the packet is then received by the victim and hacking is discovered by Test 2 (if the ARP is not implemented in the victim, hacking is discovered by its *publisher*, Test 2a).

Now let us discuss what should be done when Test 2 (or 2a) finds an abnormal ARP packet. Under 4.3 BSD UNIX, in that case an error is reported to the operator. This is useful but, as long as the operator has not read the information, hacking can continue. It would be better to stop hacking immediately and in a fully automatic way. For this purpose, the victim needs to correct the false ARP information. The action to take could be the following one:

If test 2 (or test 2a) shows that an ARP packet contains false information about my internet address (or an internet address which I have to publish), I rectify this information by broadcasting an ARP response containing the true information about my address.

This algorithm is good against hackers, but it may be disastrous when an administrator's error occurs. If two administrators give their machines the same internet address (but different physical addresses), each machine will correct each other's ARP packets by sending other ARP packets, and they will enter an infinite loop. Because ARP packets sent during this loop would be broadcast, all hosts on the network would spent a lot of time on processing them. For this reason, I suggest another solution. Instead of simply correcting false ARP information, the potential victim informs other hosts about irregularities and its address becomes

7. By setting the appropriate flag in the ARP table, an administrator can order his machine to publish an address. See `man 1 arp`.

unusable for a while. Such an information is not an ordinary ARP packet, so it is never corrected again by another host and no loop can arise. After a short period, all the hosts suppose that the error has been corrected (e.g. the hacker is gone) and they become able again to use the potential victim's address.

Security operation 2: broadcasting a warning by the victim

When a host discovers (by mean of Test 2 or 2a) an ARP packet holding false information, a flag is set in the ARP table entry corresponding to the invoked address V (let us call this flag the *address error flag*). It means that the address is fallacious (i.e. a hacking attempt or an ARP error has taken place); the flag prevents the use of this address. To warn the other hosts, an ARP packet with the right physical address for V is broadcast. Its *type* field is set with the special value `ARP_ERROR` instead of the standard `ARPOP_REPLY`. This packet causes other hosts to set the *address error flag* in their ARP tables.

The *address error flag* is automatically reset when, in a given amount of time T (say 30 seconds), no error related to the given address is reported.

The operation which we described is possible even when some hosts on the network still use the ordinary 4.3 BSD Address Resolution Protocol: such hosts, being unable to process `ARP_ERROR` packets, accept them as if they were ordinary ARP responses (this fact does not result from standards, but it is fortunately true).

4.3 Preventing hacking when the victim is down

4.3.1 Introduction

The security operations and extra tests described above make it impossible for a hacker to illegally obtain any privilege on a machine. They work when the victim (or its *publisher*) collaborates, either by sending TCP `rst` packets (''ordinary mode'' hacking) or by performing Test 2 (or 2a) (''wise mode''). So, when the victim is down, hacking is left undetected. To prevent this, it would be necessary to design a reliable algorithm to detect which hosts are down. Obviously, stations will always refuse to communicate with a host marked as being down.

4.3.2 The main idea of the algorithm

The algorithm described in this chapter is based on encrypted passwords. The encryption scheme is of the same kind as the well known one used in encrypting users' passwords in `/etc/passwd`[8]: while encrypting is easy, decrypting is virtually impossible.

When a host comes up again after a shut-down, it notifies this fact to other stations. Such notifications are authenticated by passwords. This is apparently in contradiction with the fact that any hacker can read any password sent on Ethernet. However, the scheme works because the *unique* meaning of a password is: *host X is up again* and the password is not sent on the network until this fact becomes true. N.B.: if the same host goes down again later, a different password will be necessary to confirm its coming up again.

4.3.3 The basic algorithm

Publishing a password.

On every boot, we generate a random password P. We encrypt P using the `crypt` library function[9]. The original password P is held in a file (readable only by root) called above `/etc/goodbye`. Then, as long as we are up, the encrypted copy of P is broadcast every minute by our `rwho` daemon[10]. It is received and kept by other host's `rwho` daemons.

---

8. See `man 3 crypt`

9. `crypt` is used to encrypt passwords in `/etc/passwd`.

10. This daemon exists already, it broadcasts various information every minute.

Shutting down.
Just before we go down, we broadcast a *shut-down notification packet* via UDP. This packet contains our internet address. It repeats the encrypted password P. When receiving it, every other host creates an ARP table entry for our address and sets a flag (let's call it the *host down flag*) in this entry. The encrypted password P is kept by all other hosts.

When we go down incorrectly (e.g. on a "panic trap" or a power failure), we cannot send the *shut-down notification packet*. But our `rwho` daemon stops broadcasting packets every minute and this fact is detected few minutes later by the other `rwho` daemons, which process this as an implicit *shut-down notification*. In this case hacking remains possible: the hacker can start simulating our `rwho` daemon immediately after we go down, and other hosts will never note that we are down. However, such an operation is difficult for the hacker because it must start at the moment when we are going down incorrectly (under ordinary 4.3 BSD UNIX, hacking remains possible all the time when the victim is down).

Coming up again after a shut-down.
When we come up again, we check the existence of `/etc/goodbye` in our file system. If the file exists, we broadcast a *boot notification* packet with the password P (both original and encrypted versions) inside. Every host receiving this packet first authenticates it thanks to the password, then resets the *host down flag* in the ARP table entry corresponding to our address.

4.3.7 Discussion and improvements to the basic algorithm

Lost packets and protocol errors
The algorithm proposed above is not satisfying because it is suitable only for "normal" situations, i.e. as long as no protocol error has been reported. A simple solution would consist in reporting every anomaly to the operator, but it is better to deal with it automatically as long as possible.

Assuming that software contains no bugs, protocol errors can be due to the loss of a packet or to a hacker's interference.

The loss of a *shut-down notification* packet is acceptable as long as it remains exceptional: it simply makes imperfect the protection against hackers. Therefore, nothing should be done to re-send such packets. But when a *boot notification* is lost, its sender is still supposed down and cannot communicate. There must exist a way to recover from such a situation and I hope that the two improvements proposed below will be sufficient.

Improvement 1: Multiple passwords

Data structures: When an ARP table entry indicates that a host is down, we must be able to remember more than one encrypted password related to this host (even if, under normal circumstances, we remember just one password at a time). The `/etc/goodbye` file of each host should contain not only the last broadcast password, but the list of all passwords recently broadcast by it together with their encrypted versions.

Procedures Assume we receive a *shut-down notification* packet from a host H already marked down. If the password contained in the packet is one of these we already remember as corresponding to this host, we assume that the packet is duplicate and we simply loose it. Otherwise, we *add* the password held in the packet to the list of passwords corresponding to the host. Before being marked as up, that host will have to send us one boot notification per password hold by us: each notification will remove the corresponding password from the list; the host will be marked as up again when the last password is removed.

Improvement 2: lost packets

When we receive a packet (which is not a *boot* or *shut-down notification*) from a host H which is marked down in our ARP table, we ask H to send its *boot notification* packet again. In our question, we specify the corresponding encrypted password. If we remember more than one encrypted password concerning H, we ask H one question per password.

The boot

When a host boots, it cannot know which other hosts are down and which passwords should authenticate their *boot notification* packets. The host must read this information on the network in a way which resists any hacker's interference.

Improvement 3: inquiry about which machines are down

When we boot, we broadcast via UDP the question: *Which hosts are down and which passwords will authenticate their boot notifications?* All hosts on the network[11] answer this question by sending UDP packets with the relevant information. We mark a host in our ARP table as being down if *any* of the received answers says it is down. In the same way, if any answer says that a given password is required to authenticate a host's reboot, we believe it. This algorithm is secure because a hacker could only add a false answer to our question, i.e. add, not remove, a host marked down or a password required to authenticate a *boot notification*.

4.4 Ideas for implementation of these operations

Unfortunately, I had no the opportunity to implement the ideas explained here. I would like, however, to formulate a few proposals.

4.4.1 Modifications of ARP

The ARP table should be extended to contain the new flags which I proposed in Section 4.2. All the proposed procedures can be included into the ARP module of the kernel because they are quite simple and they logically belong to it.

4.4.2 Detection of hosts being down

I propose to modify the kernel as little as possible and to implement almost all of the required procedures in the `rwho` daemon.[12] Only the following features need, in my mind, to be added to the kernel:

- The ARP table: a *host down* flag should be included in each entry (the lists of passwords related to hosts being down can be held by the `rwho` daemon, not in this table).
- Reception control: if the received packet comes from a host marked down, it is not forwarded through Internet (if we are a gateway), neither it is sent to any socket. Exceptions to this rule must exist: the `rwho` daemon needs to receive such packets, for instance in order to receive the boot notifications. For this reason, an IP level option should exist to enable a UDP socket to bypass the control. Another option should enable a raw IP socket to receive packets *only* when they come from machines marked down. This would enable the daemon to detect such packets, i.e. to find out anomalies.
- An `ioctl` should exist to enable the daemon to mark hosts as down (or up) in the ARP table.

To inform the `rwho` daemon that the local host is going down, the command performing the shut-down (or the reboot) can use a UNIX domain socket[13] (or a named pipe on System V). The actual shut-down would then take place after `rwho` confirms that the *shut-down notification* has been broadcast.

---

11. On big networks (e.g. more than 100 stations) the answer should be sent by a number of hosts preselected as servers and not by all the hosts; servers should be numerous enough to ensure that at least some of them are up at any time.

12. This is similar to the implementation of routing under 4.3 BSD UNIX using the route daemon.

13. Because only root should be able to reboot a machine and the read-write-execute protection does not work for UNIX domain sockets, this socket should be put in a directory searchable only by root.

An Adaptation of Spell To French

Pascal Beyls
mcvax!inria!echbull!beyls
beyls@echbull

Bull
FRANCE

His hobbies are Internationalisation
and twins of the young generation.
The networking nightmare
is in his company's care.

H.D.

UNIX is now being Internationalised. It is natural that in France we consider what a French UNIX could be. That implies usage of accented letters, and cedillas.

Some utilities are totally transparent such as `cp(1)`*: a copy of a file is independent of its content. On the other hand, some utilities are reluctant to work in a non-English environment:* `spell`, `look`, `style`, *and* `diction` *need modifications. For example, hyphenation provided by* `nroff/troff` *works differently in French.*

We must also remember the problems specific to hyphenation; in French syllables (and thus words) are divided according to different rules than in English.

The present paper describes the problems encountered while re-working `spell(1)` *to work correctly in French.*

This is a part of a paper published, in French, in Tribunix 87.

1. Usefulness of Spell

Who in France uses `spell(1)`? Not many people, because it only works for English. This has obviously limited its use in France considerably. It took us quite a while to start using it, even though we quite often write papers in English. (It's a good tool to run your papers through before submitting them to the EUUG.) After having found the tool quite useful, we decided to "port" it to France.

2. Usage

Here is a short review of how `spell` works. `Spell(1)` analyses text written in English, ignoring any `n/troff` commands, and writes any "non-valid" words on its standard output. `deroff(1)` is used to

strip off commands used by `troff`, `pic`, and `tbl`; in addition, it follows chains of files (`.so` and `.nx` `nroff` macros). Each word is analysed:

- Is the word in the *stop list* (which contains non-acceptable words)?
- Is the word in the list of acceptable words?
- Is the word American or English, and could an English spelling be derived from the American, examples are: *color/colour* or *center/centre*. This does not apply to: *boot/trunk*.
- Is the word derived from another one by adding appropriate prefixes and/or suffixes ?

Thus if *frequent* is in the dictionary, then `spell` will accept *frequents*, *frequently*, *frequenting* and other variants. Because some words do not follow the normal rules for prefixes and suffixes there is a separate dictionary listing all of the exceptions. This mechanism has a counterpart in that the dictionary does not indicate the category for a word (noun, verb, adjective,...). So there are mis-spellings which are not detected, such as : *neats*, *neating*...

`Spell` also complains about a large number of correct words because its dictionary does not contain every word, especially not technical ones. As an example, the word *internationalisation* is detected as an error unless you add it in the file `local-file`. This file allows extra technical words.

3. *Some Limits*

`Spell` is unable to detect grammatical errors: *They speaks* is not an error for `spell`. You must remember that `spell` is only a spelling tool.

On the other hand, adjectives are accepted with a final *s*, as in *differents things*. To a certain extent, you can fill the `stop` list in order to detect *differents* but you will slow down `spell` in so doing.

However, even though `spell` is not perfect, it is a valuable aid for locating many spelling errors.

4. *Extensions*

Some versions of the text editor `vi` optionally run in conjunction with `spell`. So, you can type your text with an immediate spell check.

5. *Difficulties in Converting Spell*

Conversion of `spell` for managing French text is not trivial.

At the beginning of this project, there were two choices: either to completely rewrite `spell`, or to adapt it to the French language. We decided on the second option, because a lot of source code could be reused and it was the best way to avoid divergence with the English `spell`, in terms of functionality.

But this solution involved the resolution of many difficulties:

1. The dictionary
   To our knowledge, there is no free French dictionary on magnetic support. Generally, they are copyrighted.

   Our French dictionary includes more than 74,000 words.

2. The codeset
   Unfortunately, this dictionary was not in the ISO 8859 codeset, which is suitable for European characters. A conversion was made, and now this dictionary can be handled by standard utilities.

   However, we modified `spell` (that means `spell`, `spellprog`, etc...) to handle the 8th bit correctly (this is called the *8th bit clean up*). By and large, the cleaning `spell` represents less than 10% of this project.

3. Number of letters
   The French language uses about 80 letters instead of 63 for English. This difference involves

modifications inside the hash algorithm and raises some mathematical problems: Is this conversion still effective? ; Is the Hoffman algorithm still the most efficient?

4. The verbs
   As it was, `spell` worked, but without any support for conjugations. We had to extend the capabilities offered by suffixes to support the 40 ways of writing a verb. In addition, there are about 132 different cases of conjugations.

   You can see that the difference between French and English (which has the 3 suffixes: *s*, *ed* and *ing*), is considerable.

   Rules for suffixes have been included inside `spell` only for the first and second class of verbs. For verbs of the third class, there are so many rules that we chose to update the dictionary with certain verbs rather than increase and slow `spell` down.

   As an example, we have included all the rules for *venir* because some other verbs obey the same rule (*tenir*).

5. Strange rules
   How do you solve the following: Verbs ending in *eler*, *eter*; double the *l* or *t* before a silent *e*? For example; *appeler*, *j'appelle*; *jeter*, *je jette*.

6. Apostrophe
   The French language has different rules for shortening words with apostrophes. For instance, we say: *Lorsqu'on* or *Lorsque l'on* instead of *Lorsque on*.

7. Capital letters
   According to exact typography, lower case letters do not loose their accents after a conversion into upper case. The ISO 8859 codeset includes all the capital accented letters.

   For example, in Paris, *LE PALAIS DES CONGRES* would be an aquarium (*congre* = fish).

   Ambiguities are solved by using the accented letters: For example: how do you understand this header:
   L'AUGMENTATION DES RETRAITES ?
   Does it mean :

   — l'augmentation des retraites

   — l'augmentation des retraités

   Unfortunately, current usage avoids the accented letters. This has an impact on `spell`. If you begin a sentence with a capital letter which should be accented, `spell`, by converting all the text into lower case, detects a mistake which does not exist. For example: *Etrange...*, after conversion `spell` sees only *etrange* which is incorrect (the correct word is *étrange*).

8. Internal rules
   The rules used for prefixes and suffixes are applicable only for the hashed list and not for the supplementary list. This is very inconvenient.

9. Integration of French `spell`
   It is a pity, but we have to cooperate with the English language. That means the new `spell` has to work differently according to the nature of the text (English, French). This is done by consulting the environment variable `LANG`. For example, the file `/usr/lib/spell/hlist` becomes `/usr/lib/spell/hlist/$LANG` etc...

6. *Conclusion*

During this project, we found many unforeseen difficulties, which were solved, one by one.

The French `spell` is now in use and the very first results indicate the following errors:

— dyslexia

— missing accents over letters

— use of non-French words (such as *implementation*)

Grammatical rules are not part of `spell`, but this lack is the most important complaint ...

"One Great Dane meeting another"

Keld Simonsen on a recent visit to Owles Hall

Benchmarking in the AFUU

Nhuan DODUC
inria!ftc!ndoduc
ndoduc@TEKNOWLEDGE.ARPA

Christophe Binot
inot@afuu.fr
BINOT@FRCITL71

Framentec

Universite de Valenciennes

This paper is derived from an e_mail announcement a short while ago. We take the opportunity to detail some points deemed essential to the subject of BENCHMARKING, and also to put emphasis on some aspects of our activities.

1. Benchmarking...

1.1 Why should we Benchmark?

Benchmark interests arise from permanent needs about evaluating hardware in line with a purchase, a process, fortunately more and more frequently thanks to the wide accessibility (and low price) of the *xxx*-computer, where *xxx* may mean personal, departmental, near-super, or even personal-super... Losses due to under- or over- loaded systems are becoming unbearable and a solution has to be found.

1.2 How should we Benchmark?

Theoreticians will want to find Mathematical Models of Information Systems that can predict the behaviour of such systems when used in real conditions. While this is promising no significant results are actually usable.

Another way, more driven by practice, is derived from our daily working environment: testing new equipment with an existing workload, (supposed to be representative), in order to simulate, with as much fidelity and accuracy as possible, the workload in new environment: this is all about Benchmarking.

2. The Past

Benchmarking was certainly an obscure aspect of data processing: when there were only a few computing centres, whose existence was self-satisfying, which were surrounded by a (happy) few computer worshippers, there was definitely no rationale for that exotic idea whose trend invariably shows that the system doesn't or won't perform as predicted. At that time, computer architectures were relatively straight forward, and since data processing essentially meant numerical computation, some firm conclusions have been drawn: "the king of the shop" being the CPU, the only-worthwhile program being simulation code for nuclear engineering (...) written of course in FORTRAN... All these factors may be relatively easily condensed, at least theoretically, into a Gibson mix or Whetstone kiloflops.

Among the first signs of evolutionary maturity is the Dhrystone benchmark whose name is self-explanatory. By the end of the 70's, many benchmarks came around, the LLNL with its 14 loops, the Linpack from ANL, the Productivity from US-Steel, to mention only a few publicly well known ones. By then, along with the recognition of the supercomputer (Cray-1, 1978) and IBM-plug-compatible (Amdahl 470, 1977) phenomena, benchmarking became not only respectable but even useful: a needed ingredient, most essential, in the awful cooking recipe that should help in delivering to customers the right hardware that is supposed to satisfy as exactly as possible their workloads.

The second revolution followed at an accelerated pace, as soon as the end of the first half of the 80's: the Personal Computer brought with it the Byte benchmark (sorry, I nearly missed the magazine with that same name !), and since an illustration is never useless, allow me to bet against any odds that each computer magazine has its own PC benchmark (OK: I win, but that's too easy a victory !).

The emergence of (graphics) work stations, RISC architecture, AI near-realty, the very-soon-computer-on-a-chip, ... only worsens (or brightens ?) the situation.

3. The Present

By now, Benchmarking has gained an recognised status, and it's no longer original nor hilarious to show interest in it: beside Benchmark groups from most prestigious laboratories (Los Alamos, LLNL, ANL...), beside Benchmark specialists from big (Fortune xxx) companies, beside the endless discussions on any e-mail network, beside many System Performance sessions in any conference (God only knows how many are they!), beside all these uncompromising signs, we shall mention only the three most significant:

— Commercial benchmarking activities: there were internal specialists or paid-consultants, but right now one may find such publicly available commercial services. We'll note only a few occurrences: AIM-II series, Neal Nelson, Infotreck ... you may want to know that you can subscribe to some Benchmark Testing Service for the symbolic amount of 6000 to 15000 US$ per year, for any purpose

— Up till now, through computer magazines, we're flooded with units (Vups, IBM Mips, Bull internal scale...) but we, the public, are as ignorant as can be of these as well as of the exact meaning of, say, the word "elementary" in Particle Physics (sorry to some of my best friends!). But things have changed and we are contemplating hundreds and hundreds of officially released documents, comparing each and every machine in much detail. I'm not going to do a free advertisement for some small red company that doesn't deserve or need it, but to you, benchmarkers in this Realm, I can point to some references in our ever increasing list that can surely satisfy your legitimate curiosity and/or knowledge.

— The benchmark of the century, to be executed by EDS on behalf of GM, that small company, which has to buy 5000 work stations (and about 10000 more if we include the procurements from GM's subcontractors) within the next 2 years.

4. The Problems

Benchmarking was perhaps an obscure aspect of data processing but then there were not many things to compare. Only a few machines, some languages and almost no operating system (sorry to those worshippers of MVS or VME...)

The good news was that, when there were no methodologies to make a start at benchmarking, things were simple: just design a synthetic set of some language's constructs or take any simplified-and-sanitized big-money-spender-program and run it privately in the basement of your organisation, with or without your boss's agreement and that's IT.

That good news was also bad news: everybody created his/her own benchmark, and while this was a good way to evaluate his/her need(s), it was not so useful generally. Even worse, people (myself included) discuss Mips, Megaflops and KWIPS although they are not talking about the same thing!

The situation is quite understandable when it occurs in marketing presentation to naive or in-a-hurry buyers; it becomes less pleasant when it can, and will surely, lead to a personal computer or a departmental

mainframe acquisition. We all, each of us, have a copy of the *xxxStone* that is definitely unique in the world, don't we?

5. and the AFUU

Recognising this situation, a few zealous worshippers in AFUU, the biggest of the European national groups, gathered and created the BENCHMARK subgroup in March 1987 simultaneously with the WORK STATION's, after the NET's but before the two last offsprings: UNIX CULTURE's and SECURITY's. We think that we shall not invent or re-invent the wheel, but try to have it run smoother: a very simple task!

We think that our role:

— should be a coordinating one. We are not going to create any new benchmark for the pleasure of having a new one; instead, we are going to collect all available benchmarks, trying to gather every bit of information about them, rationalise them in order to make them available to everyone through the AFUU channel, for the sole purpose of Benchmarking.

— has to be an informative one. We shall try to understand them as best as we can; we are going to evaluate the most significant of these, in order to be in a position where we can willfully and soundly assert on the value of the selected benchmarks.

We have been meeting monthly since March 1987 to "prepare the battlefield". Here is a summary of our activities as of today:

— We first have taken care of CPU benchmarks, promising to put next on our agenda subjects such as I/O, or graphics, or (UNIX) system benchmarks. Next to get our attention will be real-time, transaction type...ones. Right now, we're looking at the MUSBUS from McDonnell from Australia.

— We've selected for the CPU part benchmark the following: Whetstone, Dhrystone, Linpack, Doduc for some reasons detailed in the minutes of our working sessions. This means that one version of those benchmarks is frozen, documented and already successfully tested over some machines to become a good basis for future comparison and discussion.

— We are not going to be over ambitious, conscious of our situation of a (not-so-small) subgroup inside a (not-too-small) national group. We've somehow finished the first part of our work: the CPU benchmark is near its release, targeted any time now.

— We are pursuing more contacts with other organisations or individuals with the same purpose as ours and so, through this article in the EUUG magazine, we announce that we welcome any collaboration or benchmark submission from any source or origin, especially if accompanied with thorough documentation: explanation, examples, procedures... all the things that differentiate consistent and coherent activities from personal hobbies or occupations even if well-commented.

Our activity within the Benchmark subgroup of AFUU is meant to further the Benchmarking activity that is thriving in many places around the world, a proof that the activity is a well-founded and sound one that deserves much attention and works.

We are not going to succeed alone, by ourselves: for whom are built the numerous benchmarks? Let's this starting path be a good one, and this is only possible with collaboration of all of you.

6. Announcement

The group will appreciate your help to make its existence known to members of your communities and welcomes any inquiry, comment, or collaboration, participation from each and all of you.

Contacts:
Christophe BINOT
Universite de Valenciennes
binot@afuu.fr (uucp) & BINOT@FRCITL71 (Earn)
+33 27 42 41 00 X 1226

Nhuan DODUC
Framentec
Tour Fiat
Cédex 16
92084 PARIS LA DEFENCE
France
inrialftclndoduc (uucp) & ndoduc@TEKNOWLEDGE.ARPA
+33 1 47 96 46 00

News from the Netherlands

Frances M. T. Brazier
frances@psy.vu.nl
vupsy!frances

Department of Cognitive Psychology,
Vrije Universiteit,
Amsterdam

Frances is the secretary of the board of the NLUUG, and is their representative in the EUUG's Governing Board.

The EUUG celebrated its 10th anniversary this year – the NLUUG its 5th. Things have changed since the initiation of both groups, giving cause to serious consideration and thought to matters concerning the goals and objectives of the groups. This occurred in Paris at the strategy workshop for the EUUG – the NLUUG is currently engaged in comparable discussions.

An overview of the results of our most recent activities should give an impression of what we've been up to.

1. PR Activities

The NLUUG initiated activities to stimulate active collaboration between companies involved in UNIX promotion. This resulted in the instigation of a subdivision of the existing VIFKA organisation, a member of the Federation Europeene des Importateurs de Machine de Bureau. The NLUUG is an honorary member of this subdivision, known as the VIFKA/X/OPEN-UNIX group.

2. Backbone

At the moment we are still in negotiation with a number of companies for the acquisition of a Dutch backbone. For historical reasons mcvax is both the European backbone *and* the Dutch backbone. This has resulted in capacity problems for both the machine and the organisation. Something has to change within a short period of time and will! The results of these activities will reported in the next newsletter.

3. Our Last Conference – November 10, 1987

Our last conference was on UNIX and 4th generation tools. A short description of the contents of the sessions should give you an idea of what was presented. If any further information is required, (for example the address of one of the speakers), please contact the NLUUG.

Relational Database Management Systems on UNIX,
drs. ir. J.A.J. Numan, Unify Europe

After a short introduction on relational database systems and their components, a number of typical problems encountered with the use of relational database systems and UNIX as an operating system were discussed. A number of (renewed) hot topics were addressed:

1. access and storage structures,
2. benchmarks and their value,
3. areas in which RDBMS can be applied,
4. the increasing importance of 4GL in RDBMS.

A 4th Generation development environment,
J. de Jong, Baan Info Systems B.V.

Good 4th generation software does not depend on one specific database but can work with a number of independent databases. The existing data-dictionary (of an independent database) can be embedded into the dictionary of the 4th generation software. The facilities provided within such 4th generation systems were identified.

The standard database management language is SQL. A new "shell" has been developed to act as an interface between UNIX and the applications. The applications themselves concern product-control and material and needs planning.

Mimer & 4GL Data Management Software.,
Hub Bouwens, Software Enterprises Europe B..V

In many organisations fourth generation products are materialising. One of the most successful products in this area is the Swedish MIMER. Mimer fourth generation database software is available on a large number of mainframes, IBM PCs and PC compatibles, and has been installed on approximately 1000 sites in Europe. It is applicable in mixed hardware environments and consists of a fast relational database, prototyping tools, query languages, and a report-generator. The practical side of implementation, experiences and applications of users were addressed as were future developments and expectations.

Information management,
S. de Boer, Hewlett Packard Nederland B.V.

The topics:

1. information storage,
2. reports and presentation,
3. applications development,

4. integration and portability,

define the information management of an organisation, and as such are all essential. An extensive set of information management tools that offer a no-nonsense solution to these problems was described.

INGRES – the evolution of distributed databases on UNIX,
Rene Bonvanie, Relational Technology International B.V.

The evolution of the workstation philosophy via distributed processing to distributed database systems, was described in detail.

Semantic Databases,
Ir. J.H. Ter Bekke, Technical University Delft.

In modern moduling techniques much attention is paid to the design phase and the relations between the data. The models have therefore gained in significance resulting in more efficient use. This was illustrated with an example.

Prototype CLOVIS,
Ruurd Beerstra, CMG Informatietechniek B.V.

Experiences gained during the implementation of the prototype of CLOVIS (Real Property System) were discussed. Diverse graphical tools, a RDBMS and a 4GL were used during the development of this system which runs on UNIX. The following topics were addressed:

1. What is CLOVIS?,
2. The graphical administrative coupling,
3. GPM Graphical Presentation Module (schematic cartography and business graphics),
4. Why UNIX was chosen for this project (demands on flexibility and portability),
5. Why a 4GL was used (the advantages and disadvantages), and
6. The techniques employed.

A Description and Discussion of a Hierarchical Database, or Monsters, Mattes and Movie Databases,
Ed Gronke, CWI

A hierarchical database previously used at Industrial Light and Magic for tracking projects internally was the basis for this talk. A description of the front-end user process (Oracle) and the back-end database maintainer (sibyl) was described..

An overview of the system was followed by an in-depth description of how the front-end and the back-end of the system were linked, and the functionality provided by the system. This was followed by a description of the performance of the system and a critique of various parts of the system and what could be done to improve them.

ORACLE, Strategy for the 4th generation environment,
Theo van de Leuv, Oracle Benelux

ORACLE is an advanced RDBMS well suited to the UNIX environment and system. The tools for both the professional DP specialist and for the end-user in a fourth generation environment were described.

Database, fourth generation tools and what can go wrong,
Ir. N. Prangsma, Centre for Micro-Electronics

Fourth generation tools together with RDBMS systems make it possible to develop applications more quickly and more flexibly. The ease with which changes can be made in such systems may result in slackening attention during the design phase.

Programmers accustomed to the procedural approach in third generation languages are not automatically capable of adapting to the demands and limitations of fourth generation tools. Although

fourth generation tools are capable of revolutionising software development, disappointmented users and developers can be the result of insufficient consideration of the factors mentioned.

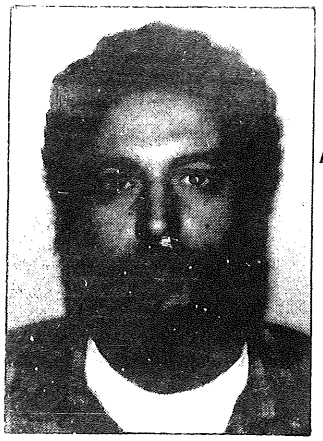
ISEE a complete graphical driven development environment for information systems,
drs. O. Schiperus, West Mount Technology B.V.

On the basis of information obtained via graphical and text editors ISEE generates third and fourth generation languages that can be processed by the INGRES fourth generation development environment, resulting in a working application.

4. Our Next conference - May 10, 1988

Our next conference will be on IPSEs – Integrated Project Support Environments. A call for abstracts has been published. Those who are interested or know of interesting speakers on research, technical aspects or UNIX related experiences pertaining to IPSEs are requested to contact the NLUUG as soon as possible. Although the main language will be Dutch contributions in either German or English are also very welcome.

I2u is alive and good-looking



Joy Marino
joy@ugdist.uucp

DIST-Universita' di Genova I2u Board Member

Ecole Normale Supérieur

Joy Marino is associate professor of Computer Science at DIST (Dipartimento di Informatica, Sistemistica e Telematica), University of Genoa. He started using UNIX in 1980, when he was interested in ADA: now he is still using (and teaching) UNIX, C and C++, but no more ADA.

He has been involved with *i2u* since 1984, first as editor of the i2u newsletter (UNI*forum*) then (since 1986) as member of the Board. He has also been appointed as i2u representative in the EUUG governing board (probably because other members *think* that he is good at English).

Probably the last time most readers of EUUGN heard about *i2u* was the 1986 EUUG Spring Conference in Florence.[1]

Yes, *i2u* is the name of the "fashion house" whose T-shirts, sweaters and umbrellas were sold out in Florence, and no, "fashion" is not our only interest, at least as far as UNIX is concerned.

Teus Hagen said that Florence was a good conference, David Tilbrook said the same, the *i2u* chairman said that it was a success, the *i2u* secretary said that it had a good financial outcome, and we all began to "dormire sugli allori" (sleep on laurels), as is said in Italy (from Julius Caesar on we have got a long tradition of laurels, and of sleeping, too).

When we woke up we found that half of *i2u* members had gone away, while at the same time the interest in UNIX was still growing. It was 1987, and now, after a whole year, I see a fully renewed *i2u*, with a better support of members' needs by the secretariat, more concerned with "what Italians want to know about UNIX" and "what members ask from an organisation like this".

Furthermore, we are acquiring new, interested and collaborating members, and the membership base is increasing as it should be in a country where UNIX is quite diffused, and that is said to be somewhere around the fifth most industrialised nation. About two thirds of *i2u* members are commercially involved with UNIX, while one third come from Universities. This means we have to be more attentive to *industrial issues*, such as "standardisation", "relational data bases", "UNIX market perspectives"; on the other side there is little interest in advanced technical issues, and there are very few opportunities of technical contributions of Italian members to the EUUGN, or even to our languishing *i2u* newsletter: UNI*forum*.

1. Unfortunately, this is true also for many Italian members who thought it a bargain to subscribe both for the Florence Conference and *i2u* membership, but who never participated in national group activities since then and did not renew the membership...

As the editor of this newsletter I can't blame it too much: it has a good column on Book Reviews, sometimes it is timely in reporting EUUG and overseas (i.e. Uniform and USENIX) Conferences, it has had an interesting "Berkeley Corner" for some issues (Roberto Zicari, where are you now?). The newsletter is definitely bad in regular delivery and it lacks a large base of contributors.[2]

In 1988 *i2u* is beginning to offer new services to its members: every new member will receive an "annotated bibliography" of books and magazines *everybody should have heard of* in the UNIX world; the secretariat is building up a documentation center, where all the periodical publication about UNIX will be tracked and interesting issuers pointed out to members.

A "Who's Who of UNIX in Italy" or "The Portrait of *I2u* as a Young Organisation" (the exact title has not been decided yet) will be delivered later on this year, and it will based on a survey questionnaire distributed to all members. Using it every member of *i2u* will be able to know who has solved his own problem, or who is using the machine he is looking for, or who is selling the software package he is seeking. The idea of a software catalogue has being boiling up for quite a long time, but it is not cooked enough, if I can say so. We think that it had better be a (network-based) archive of software products, but first we need a well established network of UNIX machines.

The Italian part of EUnet is morally and financially supported by *i2u*, and we consider the network one of the more viable means for reaching people and circulating information. Until now, the "E-mail" culture has not been widespread in Italy, and the network backbone is undergoing a major restructure. This may be the subject of a future report "from *i2u*" in the EUUGN.

Finally, conferences. In 1987 we had a two day conference in Milano, 24 – 25 June. It was announced as the "*i2u* annual meeting", stressing that from now on it will be held every year on a regular basis. There was a half day of overviews of "the state of the art" and market perspectives either in Italy or worldwide; then some presentations of interesting UNIX experiences and applications. A round table entitled "What niche for UNIX" concluded the first day. The main events of the second day were an extended introduction to the X/OPEN consortium and all its technical branches, a two-voice debate on "NeWS vs. X-Window System" featuring Dave Rosenthal (SUN) and J. Bettels (DEC). There was no exhibition, but short commercial presentations were scheduled in small, and crowded, rooms during lunch-time. Proceedings of the Conferences, with summaries of all presentations, have been sent to all attendees and *i2u* members. Other copies can be obtained from the *i2u* secretariat.

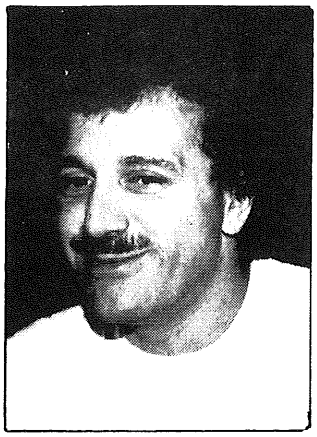
The 1988 "*i2u* annual meeting" is planned for the last week of May (and not for the 1st of May as the EUUG wall calendar says). Later this month the program will be defined; there will only be invited speakers (in order to represent all existing experiences in a fair way). An exhibition is planned and we are trying to have machines from different vendors "that talk to each other" and a few distributed applications running, even if it is rather difficult to find significant applications that can hit the imagination of a casual attendee.

I2u is happy to invite all EUUG members to its Conference; if you are interested, don't hesitate to contact the *i2u* secretariat.

---

2. Some issues have been written and typeset (`troff` and laser-printer, of course) by this one editor. Those who say that EUUGN is too UK oriented have never read UNI*forum*!

UK Activities



Sunil K. Das
sunil@nss.cs.ucl.ac.uk

City University
London
U.K.

Sunil K. Das became UKUUG Chairman in 1984, and was re-elected in July 1987. He first encountered the UNIX system in 1977 whilst employed as a research fellow in the Computer Networks Research Group at University College London. In 1980, Sunil joined the academic staff of City University's Computer Science Department, where his interests have included operating systems design, local area networking, systems programming and software engineering.

Sunil is well known for designing the mathematical model of the switching circuits, and designing and implementing the algorithms for the computer environment which controls the movements of the scenery hoist system in Olivier Auditorium of the National Theatre of Great Britain.

Since the EUUG Autumn Conference in Dublin, we've had a busy time in the UK organising the EUUG Spring 1988 Conference, and the UKUUG's two day Winter Technical Meeting. The Technical Program that that has been put together for the EUUG's event is documented elsewhere in this Newsletter. There will be 2 days of tutorials, 3 days of conference, and 2 days of what has been named the UNIX Showcase. Many more details will appear in the Pre-registration Booklet which should be in your hands by mid February *at the latest*. You will be pleased to know that *Dennis Ritchie* and *Steve Bourne* will be attending the Conference.

In addition to Conference organising, meetings have been held with the Chairman of /usr/group/UK, continuing the discussions about the possibility and feasabilty of merging the two groups. The talks are at an early stage and we'll keep the EUUG membership informed about how they develop. At the top of our requirements however, is the continued affiliation to EUUG, and the holding of a referendum to determine whether the UKUUG membership wants a merger.

The UKUUG Winter Technical Meeting took the form of a Workshop on Networking in the UK. Sunil K Das of the Computer Science Department, City University London and UKUUG Chairman, and Peter Collinson of the University of Kent and Executive member of the EUUG, organised and co-ordinated the workshop which concentrated on all aspects of UKnet, the UK part of the world-wide UNIX computer network.

The highly successful two day meeting, with more than 200 delegates in attendance, was held at City University just before Christmas. Over 100 stayed overnight in the University's Residence Hall, where the Workshop dinner was held on Monday 14th December. (The Hall is the same one which will be offered to participants of the EUUG's Spring Conference as an alternative to expensive hotels.)

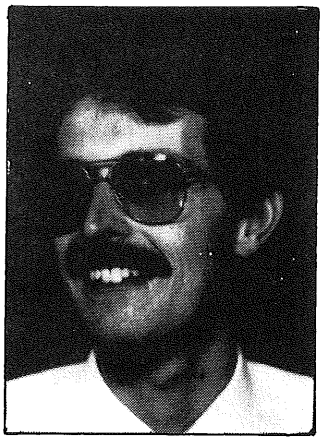
On Monday, the presentations were tutorial flavoured, primarily aimed at people who wished to find out what the network is, how it works, what software is required, and to provide a discussion forum for other general issues. The session covered most of the software packages currently in use on the Network.

The session held on Tuesday was more technically flavoured and mainly of interest to administrators who are currently involved in running the network. Peter Collinson of the University of Kent at Canterbury addressed issues such as charging and general policy, and the topology of UKnet. Other topics of interest were discussed by speakers from Cambridge, London and Newcastle Universities and British Olvetti.

The workshop was only open to EUUG members because it was subsidised by the UKUUG. However, because of the merger discussions being held with /usr/group/UK, their membership was welcomed. A 50 page document reporting the technical presentations and discussions is being produced and sent to all delegates, members of the UKUUG, and the Secretariat of each EUUG National Group. Further copies may be available via Sunil (sunil@nss.cs.ucl.ac.uk).

The delegate numbers attending the Workshop was very encouraging, because this bodes well for the delegate numbers for EUUG's Spring 1988 Conference which will take place in April at the Queen Elizabeth II Conference Centre, Westminster.

UKUUG UKnet Workshop



Richard Murphy
richard@cs.bbk.ac.uk

Birkbeck College
London,
U.K.

Richard Murphy is a programmer in the Department of Computer Science at Birkbeck College, University of London.

This note describes a very brief personal view of the UKnet workshop held at the City University, London on December 14 and 15 1987. A more complete note is being distributed to UKUUG members and participating delegates. A single copy is being sent to each EUUG national group, enquiries about additional copies should be made to Sunil Das (sunil@nss.cs.ucl.ac.uk).

Monday 14 December

- Sunil Das, Chair UKUUG, welcomed everyone to the workshop and showed us how to use an umbrella. He mentioned that UKUUG is (still) having discussions with /usr/group/UK, the commercial UNIX users group in the UK, about joint ventures, merging, etc.
- Pete Collinson, University of Kent, summarised at great length the history of UKnet, how to join it, what it provides, and some detail about how it works.
- Lee McLoughlin, Imperial College London, described the UUCP transport system, in particular UKUUCP, the official UK version (which he "wrote").
- Piete Brooks, University of Cambridge, tried to explain the intricacies of X.25, Yellow Book Transport Service (YBTS), Network Independent FTP (NIFTP), the dreaded Yorkbox, the brilliant UNIX-NIFTP (which he "wrote"), and the forthcoming ISO File Transfer Access Mechanism (FTAM).
- Julian Onions, University of Nottingham, described MMDF (unofficially "My Message Didn't Find you") and how it interacts with various transport services. It has wonderful features such as authorisation, configurability!, nameserver handling, and knows the distinction between domains and channels. Unfortunately only seven people in the world understand it.
- Jim Crammond, Imperial College London, described Sendmail (specifically UK-Sendmail) and how it interacts with various transport services. His notes included useful remedies for a number of common

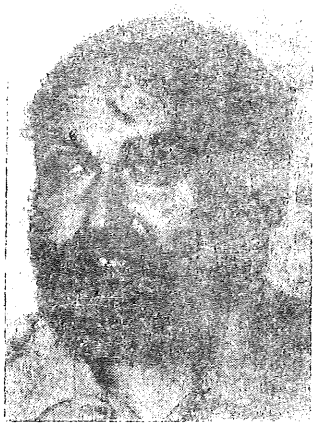
problems experienced by users in the UK.

- Colston Sanger, Olivetti International, described Smail, yet another domain mailer. It contains many equally wonderful features (but no-one seems to use it?).
- Chris Downey, University of Kent, explained what News is and how it interacts with various transport services. He noted the lack of popular support for the recreational equestrian group in the UK. Apparently, if your version of News is 2.10 or less you should throw it away and get version 2.11.
- Peter Houlder, University of Kent, explained how to wade through the bureaucracy of UKnet, especially registration and billing. Apparently we can receive unofficial invoices by electronic mail but official invoices are still printed on bits of paper and sent by snail mail.
- A final panel session (before the bar opened before dinner) fielded a wide variety of niggly questions concerning modems, Yukbox, local area networks, secretaries, mail acknowledgement (ha ha!), the Name Registration Scheme (yawn) and various uncertain legal issues which are probably better off not being printed.

Tuesday 15 December

- Pete Collinson, University of Kent, kicked off (on time) with his view of the current state of UKnet with emphasis on the support role of UKC. Remember not to be rude to Kent, they're even ruder by return mail.
- Bruce Wilford, University College London, diplomatically tip-toed through the current situation at UCL with regard to the DARPA Internet gateways run by them (and the UK Ministry of Defence).
- Pete Houlder, University of Kent, outlined the UKnet topology ably aided by suitable maps and statistics. The situation changes so quickly that all the information was out of date.
- Piete Brooks, University of Cambridge, gave a technical view of News which he apparently obtained from many happy hours of reading the source code.
- Lee McLoughlin, Imperial College London, explained why we should hide a local area network from its own users as well as the rest of the world. Good domain mailers and UKUUCP let you do this.

AFUU governing board changes



P. J. Peake
philip@axis.fr

Axis Digital
PARIS

Following the AGM of the AFUU, we now have a new governing board. This is the first board elected under the revised constitution of the AFUU, and so each person elected is elected for a three year period. The change was made retrospective, so that people who have already served one year will now serve another two, and those having served two years will now serve another one. In practice, this means that about one third of the board will be replaced each year.

Below is a list of the members. The column "responsibility" indicates an executive responsibility, obviously, every member of the board will have other responsibilities also.

NAME	EMAIL	RESPONSIBILITY
Jean-Louis Schneider	jls@afuu.fr	President
Jean-Jacques Rousset	–	Secretary
Philippe Vaudou	–	Treasurer
Christophe Binot	binot@FRCITL71.bitnet	Vice President
Philip Peake	philip@axis.fr	
Jean-Louis Bernard	–	
Veronique Mansart	–	
Pierre-Louis Neumann	neumann@inria.inria.fr	
Nicole Blanie	–	
Michel Wurtz	inria!uparis8!ign!mw	
Anne Francois	–	
Alain Saint-Patrice	–	
M. Sutter	–	
M. Toledano	–	
Pascal Beyls	belys@echbull.fr	

One of the most important immediate tasks for this group will be to contine the preparations for **Convention UNIX 88** which is our annual conference and exhibition, which will be held from the 7th to 10th March. This year the event will be held at a brand new exhibition and conference centre (Espace Champerret) in Paris.

Although this event is aimed primarily at our members, there may well be things of interest to members of other EUUG affiliated groups, for example, not all of the conference proceedings will be in French since we have several speakers from other countries, and the exhibition will be large.

If you have any questions about this event, or about any of the other activities of our group, please feel free to contact us.

The person to contact is:

Anne Garnery (Manager)
AFUU
c/o SUPELEC
Plateau du Moulon
91190 Gif sur Yvette
France

Email: anne@afuu.fr
Tel: (+33) (1) 60 19 10 13

EUUG Spring 1988: The Technical Programme

The EUUG Spring conference will be in London. EUUG members should have already received a pre-registeration booklet.

Technical Programme

Wednesday 13th April

0930 UNIX around the World – Opening Address
Sunil K Das (UK), City University London and UKUUG Chairman
UNIX Past, Present and Future: Changing Roles, Changing Technologies
John Mashey (USA), Mips Corporation

1030 COFFEE

1100 Multilevel Security with Fewer Fetters
Doug McIlroy Jon Reeds (USA), AT Bell Laboratories
Help! I'm Losing my Files !
John Lions (Australia), University of New South Wales
A Tool-based 3-D Modelling and Animation Workstation
Sam Leffler (USA), Pixar

1230 LUNCH

1400 The JUNET Environment
Jun Murai (Japan), University of Tokyo
Measuring File System Activity in the UNIX System
Maury Bach (Israel) Ron Gomes (USA), IBM Haifa
Yacc meets C++
Steve Johnson (USA), Ardent Computer Corporation

1530 TEA

1600 An Overview of the Miranda Functional Programming Language
David Turner (UK), University of Kent at Canterbury
An Overview of the GOTHIX Distributed Operating System
Alain Kermarrec (France), IRISA
A Protocol for the Communication between Objects
Rudolf Schragl & D Lauber (West Germany), UNA EDV-Beratung GmbH

Thursday 14th April

0900 A UNIX Implementation of X25 PLP in ISO 8802 LAN Environments
T Grimstad, A Hussain J Olnes (Norway), Norsk Regnesentral
UNO: USENET News on Optical Disk
A Garibbo, L Regoli G Succi (Italy), University of Genoa
POSIX – A Standard Interface
Jim Oldroyd (UK), The Instruction Set Ltd

1030 COFFEE

1100 Networking for Plan 9 from Bell Laboratories

Dave Presotto (USA), AT Bell Laboratories

Multiprocessor UNIX: Separate Processing of I/O
A J van de Goor et al (Netherlands), University of Delft

Word Manipulation in Online Catalogue Searching: Using the UNIX System for Library Experiments
Michael Lesk (USA), Bell Communications Research

1230 LUNCH

1400 Software Tools for Music or Communications Standard Works!
David Keeffe (UK), Siemens Ltd

UNIX and Arithmetic
Bob Morris (USA), National Computer Security Center

UNIX System V.3 and Beyond
Ian Stewartson (UK), Data Logic

1530 TEA

1600 General Purpose Transaction Support Features for the UNIX OS
Russ Holt, Steve Marcie (USA), NCR Corporation

A Toolkit for Software Configuration Management
A Mahler, A Lampen (W Germany), Technische Universitat Berlin

OFS – An Optical View of a UNIX File System
Paulo Amaral (France), INRIA

FRIDAY 15TH APRIL

0900 Design of and Experience with a Software Documentation Tool
Jose Manas, Tomas de Miguel (Spain), Ciudad Universitaria

Implementation of a UNIX Environment on the GOTHIC Kernel
P Le Certen, B Michel, G Muller (France), Bull/INRIA

Directed Mapped Files is a File Access Method Implemented under V.3
A Meyer (West Germany), Stollmann GmbH

1030 COFFEE

1100 Grep Wars
Andrew Hume (USA), AT Bell Laboratories

Extending Stream I/O to Include Formats
Mark Rafter (UK), University of Warwick

Evolution of the SunOS Programming Environment
Rob Gingell (USA), Sun Microsystems

1230 LUNCH

1400 Software Re-Engineering using C++
Bruce Anderson & Sanjiv Gossain (UK), University of Essex

SunOS Virtual Memory Implementation
J Moran (USA), Sun Microsystems

System V Release 3, Diskless Workstations and NFS
R Cramner-Gordon et al (UK), The Instruction Set Ltd

1530 TEA

1600 The Andrew Toolkit – An Overview
Andrew Palay et al (USA), Carnegie Mellon University

Goodbye
Teus Hagen, Oce-Venlo, Nederland B.V. and EUUG Chairman

10 Years of the EUUG

Peter Collinson
pc@ukc.ac.uk

Secretary, EUUG

Peter Collinson is the Head of the UNIX Support Group at the University of Kent, Canterbury, UK. He has been involved with the UNIX system since 1976, when in the immortal words of Nigel Martin: "UNIX changed me from a Lecturer in Computer Science into a junior Computer Operator".

Peter was responsible for the writing of Cambridge Ring networking software on the VAX, starting with 32V and continuing in the Berkeley tradition ever since. Kent now runs the EUnet backbone for the UK.

Peter has been involved in EUUG since the early days, being the Chairman of the UKUUG for one year before handing this task over to Alan Mason. He has been the Secretary of the EUUG since 1982. He has always tried hard not to become Newsletter editor and was recently saved by the appointment of Alain Williams.

Peter thinks that the word UNIX should be allowed to be a noun but he still cannot work out whether he should use `*+++argv` or `**++argv` or `++**argv`. `getopt()` is for Users.

1. Happy Birthday, EUUG

The EUUG is celebrating 10 years of existence this year and this also marks around 11 years of my personal involvement with UNIX user groups in Europe. I have decided to resign (or more accurately — not stand for re-election) effective from the London Conference in April. The others on the Executive have given me this chance of saying goodbye to you all. Actually, you don't get rid of me as easily as that, I have agreed to be Programme Chair for the Autumn conference in Portugal.

But it will be an end of me going to jet-set meetings of the EUUG Executive. Jetting off to far away places, such as Schipol Airport in Amsterdam on a Sunday morning, sitting in a windowless room discussing EUUG business from 9.30am until 5.30pm, getting on an aeroplane and going home in an exhausted state. Running the EUUG has become very hard work, believe you me.

Anyway, old men are allowed to reflect a little on the past and so I intend to do just that. Sorry folks.

2. The Early Group

When I was writing my science fiction piece for Usenix last year (which was also published in this august journal at some point), I tried to understand why the early groups were important to the people who formed them. In some ways, I came to the conclusion that their main aim was to bring people together. At this stage UNIX people were often isolated in their work environments, having no adequate documentation of the system which they were running, but gaining understanding by reading the code. Meetings were really about exchanging information which you had found from looking at the source; learning how to make the system run faster and fixing those bugs lurking in the code.

UNIX was the "underground", running on a cheap mini which many Computer Science departments had bought as a private machine. It was never clear at that stage quite how many people were running UNIX.

At Kent, we got the first inklings of its popularity when we organised a meeting with Kernighan and Thompson as the guest speakers. We expected 50 people, and got 200. I think that meeting wrecked our RK05 drives but we got some useful code — mostly from the Dutch.

There was no money involved in the group at this stage; when it looked that the newsletter was going to cost money to produce and mail out, the group became a DECUS SIG and DECUS were always helpful in their support. DEC didn't really understand UNIX at a corporate level, the salesmen did though. I wish I had had £100 for every sale I made for DEC at this stage. A salesman would ring me and ask questions like "I hear that you can program in this UNIX on a PDP-11/34, so-and-so is asking about...". UNIX was like BASIC... but it sold machines in certain sectors.

3. The Second Period

I think that the next phase of development started with the first Paris Conference which was the first really "European" event. This was in 1982, five years after the first meetings in Glasgow. UNIX had changed. V7 had happened. UNIX was portable. V7 didn't run on many machines which had been running V6, the first EUUG "product" was probably the EUUG V7 strip down system using overlays in the kernel to run some of the bigger programs which had come along with the new release. By 1982, Kent had had its VAX11/780 for two years and we were thinking about that funny BSD system.

UNIX was already being overhyped by the people selling UNIX boxes ("Uniboxes" as Mike O'Dell calls them) as the solution to the world's computing problems — which it manifestly was not. UNIX has had big problems getting over all the overselling which was done in this period. The cries of "there is no applications software" which was true then still appear in the computing press even though the statement is incorrect today.

By the time of the first Paris conference UNIX was beginning to make an impact on the commercial marketplace. Unfortunately for me, the conference made a big impact on my bottom. This was the first meeting where I started taking notes and producing Conference reports which were designed to fill newsletters. French students are kept awake by making them sit on hard benches and I remember sitting though endless vendor presentations on these self same hard benches wishing I had never agreed to take notes. However, the reports kept EUUG's newsletter going, bulked out ;login: and filled several pages in the organ of the Australian Group. Since they were mostly long lists they were fundamentally boring, so I began to insert tales of drunken excesses until ultimately people only read them to discover what the local alcoholic beverage was like.

In EUUG terms, this second period saw the formation of the national groups as we know them today. EUUG helped this process providing expertise and a model which could be followed by a group at its inception. EUUG often actively promoted the formation of a local group by holding an important conference in the country, this raised interest and brought local people together.

The second Paris conference in 1985 was a financial disaster and showed that we could not run a group unless we had proper financial planning, budgets and all that entails. Paris has become the spectre of failure, when we are discussing conferences in the executive, "this can't be another Paris" is often heard. The group is now run much more tightly and much more professionally.

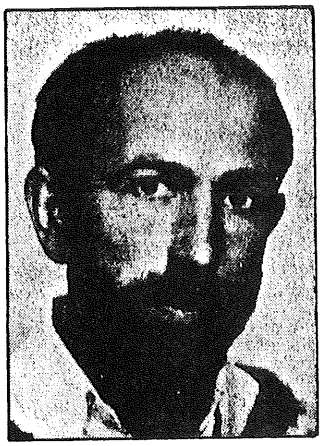
EUUG is still run largely by volunteer labour which means that the executive committee members have to find time to do things. We are lucky that the employers of the existing committee are understanding about the amount of time that people give.

Work done by volunteers must always take second place if there is a choice between the EUUG and doing work to put bread on the table. This means that no-one on the executive can guarantee that they can do a particular task at a given time. In recent times we have got round this by employing people. This newsletter is put together by a part time editor who does the work of chasing people for articles and ensuring that the newsletter comes out on time. I think that we will see this trend continue into the third period.

4. The Third Period

The third period starts now. I feel that change is again in the wind. National groups are becoming bigger and rightly worrying more about dealing with their members in their own language. We are seeing the emergence of large national conferences running with the language of the country as a first choice. I hope that these national groups will learn and profit from some of the mistakes and successes which we have had in EUUG over the years. I wish them and the EUUG well.

EUnet



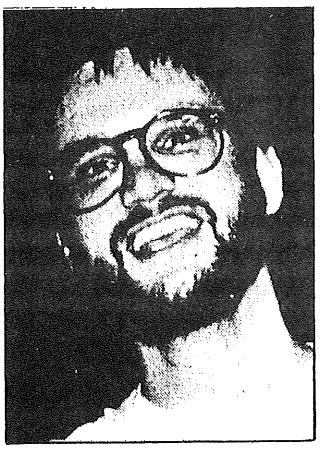
Peter Houlder
uknet@ukc.ac.uk

Computing Laboratory,
University of Kent

1. Introduction

Readers will be much relieved to know that this introduction is my only contribution this time. This article is from Daniel Karrenberg of the Centrum voor Wiskunde en Informatica – I still call it "mcvax". The article gives details of the latest state of the EUnet network. Please keep sending me articles, so we can keep this column going. Over to you Daniel ...

EUnet Update



Daniel Karrenberg
dfk@cwi.nl

Centrum voor Wiskunde en Informatica,
Amsterdam, Netherlands

1. Started in 1982

EUnet is a pan-European cooperative computer network for information exchange comprising almost 1000 sites in 19 countries. Like many other cooperative networks it originated from a limited community of users with similar interests, in EUnet's case the UNIX operating system. It was started in 1982 when some of the few UNIX sites in Europe connected to each other and set up a link to a similar network in the United States called Usenet.

2. For Research and Development

Due to the lack of networking facilities for the research and development community in Europe, the scope of EUnet widened almost immediately as mathematicians and computer science researchers discovered it

as a convenient means of communication with colleagues worldwide. Now researchers in fields not directly related to computer science also are making use of EUnet as the "critical mass" of people with access has been reached.

EUnet was never limited to the academic community alone, and very soon became a vehicle for technology transfer supporting joint projects of academia and industry as well as enabling researchers of both communities to exchange information quickly and informally. Because of its cooperative nature and low initial connection overhead, EUnet has also been available to the small and medium size companies without much capital which are common in the software industry. For them it is especially important to keep up to date with developments in their fast paced industry.

3. Services

EUnet is a vehicle for information exchange rather than for sharing computer resources. It provides two services: Electronic Mail and News. Both of these services are closely linked to the North American UUCP and Usenet networks. No interactive services like remote login are currently provided by EUnet.

4. Loosely Coupled Organisation

EUnet has a loosely coupled, distributed rather than centralised organisational structure. It is run by agreements between the participants that are kept as informal as possible; in particular no one enters a formal obligation to provide a service to others. While this mode of operation may not be adequate for commercial networks, it fits the cooperatively minded research and development community very well since it reduces necessary investments and running costs of the network considerably as well as minimising the economic risks for those actually providing a service to others.

EUnet as a whole is represented by the European UNIX User Group. Formal decisions affecting the whole network are taken during the two annual meetings of that group.

5. The Users Pay!

Each EUnet site bears its own costs of connecting and operating the network connection. Some nodes in the network incur a disproportionate amount of the communications costs by relaying data and supporting the network in general. These costs are distributed to the individual users by a hierarchy of local and international arrangements customised to local conditions and designed to minimise overhead.

Apart from donations of equipment and volunteer work by individuals, EUnet receives no support from third parties. Thus the users themselves pay for the services they use, based on actual usage! And because of EUnet's low overhead structure these prices are affordable.

6. About a Thousand Sites

It is not easy to give the size of EUnet since the measures used to express network sizes differ and some data is purposefully hidden by local entities to make routing easier. To give the number of nodes is misleading because a node can be anything from a single user UNIX machine to a local area network with dozens of machines and hundreds of users. The number of machines is not known because any number of machines can be hidden behind a gateway node. Therefore we let the number of sites refer to geographically and/or organisationally separate entities.

Of all 978 EUnet sites, 132 are currently official subscribers to the News service, receiving a varying scope of newsgroups.

Another measure of network size is the volume of traffic. Because of the high interconnectivity inside EUnet this cannot be measured centrally and is therefore difficult to obtain. The throughput figures for larger national backbones at the moment are well above one Gigabyte per month.

<table>
<tr><td colspan="4">News
(RFC 850)</td><td colspan="4">Mail
(RFC 822)</td></tr>
<tr><td colspan="3">rnews
(RFC 850.4)</td><td>rmail
(RFC 976)</td><td>NNTP
(RFC 977)</td><td colspan="3">SMTP
(RFC 821)</td></tr>
<tr><td colspan="4" rowspan="2">uucp</td><td colspan="4">TCP
(RFC 793)</td></tr>
<tr><td colspan="4">IP
(RFC 791)</td></tr>
<tr><td>g</td><td>f</td><td>x</td><td>t</td><td colspan="4" rowspan="2">Ethernet, Arpanet, serial lines, X.25, 802.x, Hyperchannel,
...</td></tr>
<tr><td>phone lines
unreliable</td><td>reliable
7-bit</td><td>X.25</td><td>TCP</td></tr>
</table>

7. Interconnections

From its inception EUnet has actively sought to make communication with users of other networks and communications services in the research and development community as easy as possible. EUnet is currently operating direct Electronic Mail gateways to the DARPA Internet (only part of which is the ARPANET), CSnet, EARN/BITNET, JANET, ACSNET, UUCP, JUNET and the RARE experimental MHS service. All other major networks used in the research and development community are reachable from EUnet. The News service has gateways to Usenet and the DARPA Internet.

8. Technology

EUnet currently uses the UUCP and TCP/IP protocol families. UUCP is mainly used on wide area connections, TCP/IP is mainly used in local area networks. A ''stacked boxes'' diagram of EUnet protocols can be found on the next page. National wide area connections use both the telephone and the public X.25 networks. International connections use the X.25 networks almost exclusively. EUnet maintains a leased line to the Usenix Association's ''uunet'' node in the United States.

EUnet grows by about 100% a year both in terms of sites and in terms of traffic. The large amount of traffic makes the use of public X.25 networks for international connections very uneconomical. EUnet will therefore restructure its international infrastructure soon. An improved infrastructure is expected to provide additional services to sites which are willing to pay for them.

† The Norwegian part of the network is currently under reorganisation. The Portugese backbone site is being established.

Number of EUnet sites per Country
October 1987

23	Austria
16	Belgium
29	Switzerland
140	Germany
46	Denmark
1	Spain
53	Finland
96	France
244	United Kingdom
7	Greece
6	Rep. Ireland
11	Iceland
28	Italy
2	Luxemburg
110	Netherlands
20	Norway†
1	Portugal†
144	Sweden
1	Yugoslavia
978	EUnet

9. Towards International Standards

In the long term, EUnet plans to move to the use of internationally standardised protocols in order to achieve even higher connectivity and better services such as non-textual Mail. This move, however, will be made gradually and with all due care to preserve the current level of service to the users. It is still unclear which set of ISO protocols will eventually gain acceptance in the research community since protocol suites currently being proposed can't achieve EUnet's current level of functionality and implementations are lacking.

These developments present a lot of problems and need careful planning. The EUUG has financed the first part of a study about migration strategies which is just being completed and it is hoped that this work can be continued with help from the European Commission.

10. Cooperation

The most important issue that faces all European networking efforts is cooperation, because it is very important to present a homogeneous view of the European networking infrastructure to the R&D community inside but especially outside of Europe. One important example of this is electronic mail: although many reliable gateways exist between the various European networks the end user finds it usually very difficult to address messages to correspondents on other networks correctly because of different address syntaxes. On the initiative of EUnet, agreements about uniform mail addressing have been made in some European countries.

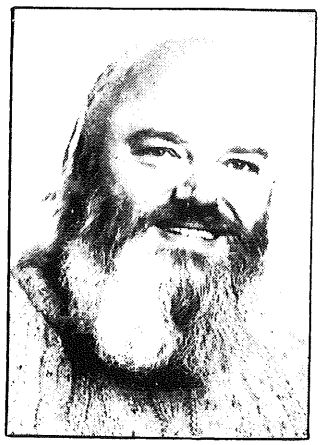
EUnet will actively continue to pursue cooperation in order to improve the networking infrastructure in Europe. It has always cooperated flexibly and pragmatically with other networks; this has resulted in an outstanding amount of connectivity that could be exploited by EUnet users and frequently by users of third networks as well.

Furthermore, the two largest European R&D networks, EARN and EUnet, have recently started talks about even closer cooperation between themselves – not only concerning mutual interworking, but also the establishment and common use of intercontinental gateways and shared infrastructures.

11. Outlook

In the forseeable future EUnet will continue to grow rapidly. As various networking efforts for academic research become operational it will put more emphasis on its users in the corporate R&D community and their integration into the European R&D networks in general. We hope that developments like this will be made possible by a growing amount of cooperation between the various networking efforts in Europe and worldwide.

The Santa Fe Trail



John Carolan
john@puschi

Glockenspiel Ltd
30 Iona Crescent
Dublin 9
Eire

John Carolan is the current chairperson of the Irish UUG. He is also managing director of Glockenspiel Ltd. of Dublin. Glockenspiel has been using C++ since 1985, and John has presented several technical papers on C++. His present work includes the development of C++ class libraries common between OS/2 and X-Windows on UNIX.

C++ heads migrating towards Santa Fe had but one choice of route:

..!Denver!Albuquerque!Santa_Fe

Flying from Denver to Albuquerque by day feels like fast forward through the opening sequences of the film, "Kayaanisquatsi". The drive from Albuquerque to Santa Fe takes you further up into the mountains on a traffic-free, 100+ KPH highway, perhaps the only one of this kind in the US. Santa Fe itself consists of a Mexican-style town centre, complete with adobe buildings, surrounded at 1 Km. radius by the anonymous sprawl of gas stations and fast food common to most American cities. Pretty antique shops populate the Mexican town centre, selling everything from the obviously kitch to presumably authentic Navajo art. Into this quaint, picturesque, objet d'art-oriented setting, the world's first C++ conference burst like a jalapeno pepper on the unsuspecting palate.

The organisers had drawn together such an intensive programme for the conference that no-one had any time to talk to anyone else during the proceedings. We compensated by staying up 'till four in the morning, swapping philosophy and code fragments. The buzz from the conference got to everyone as we competed desperately for time-slices to air our deepest C++ convictions.

The conference opened with an hour-long manifesto from the Perpetrator himself. Bjarne gave the most succinct account that I have ever heard of the history, objectives and future directions of the language. Relatively new information in this talk included the content of release 2.0 of the AT&T C++ translator:

— multiple inheritance

— better implementation of class assignments

— overloading the arrow operator

— class-specific overloading of new and delete

Questioned about the release date, Bjarne referred us to "the guy in the suit at the back". This turned out to be one Paul Fillinich, AT&T's recently appointed Product Manager for C++. Paul, in the classic manner of AT&T people in suits, answered the query with a wry evasiveness. Meanwhile, rumour has it that release 2.0 may ship during Q2 of 1988. (Out of keeping with the classic manner of AT&T people in suits, Paul is doing a lot to help the cause both within AT&T and in the real world.)

Bjarne was followed in by Steve Dewhurst, who – along with Kathy Stark and Laura Eaves – has done much of the work on AT&T's C++ compiler. Steve gave a tongue-in-cheek talk about the design objectives of the AT&T compiler. These include catering for pre-compiled header files and emitting code in the form of abstract representations which can adapt easily to a variety of code generators. AT&T uses the compiler fairly widely internally. The compiler will eventually (mid-1989?) be released as an AT&T supported product. AT&T are quite reasonably concerned about the difficulty of supporting a C++ compiler. The big discrepancy between internal use and external sale of the compiler arises from the need to evolve a verification procedure for would-be C++ compilers before committing to support.

Michael Ball talked about the Oregon Software C++ compiler. It is based on their existing Pascal compiler and will ship initially on Sun work-stations. It may take some time to complete testing of this compiler, since it was developed without inspection of the AT&T translator.

Your flying columnist gave a talk on C++ on OS/2. This talk set out to offend the organisers – USENIX – by digressing into other operating systems than UNIX. My main point, which I laboured, is that C++ is to OS/2 as C is to UNIX.

Ken Friedenbach from Apple described the interfacing of C++ to MacApps. Apple intend to launch a C++ product in Q2 1988, which may supplant the Pascal influence on the MAC.

Roy Campbell from University of Illinois took us through the planning stages of their CHOICES distributed operating system. Roy's scheme for representing class hierarchies was interesting in itself and gave a very clear picture of the use of C++ in the development of CHOICES.

Day 2 opened with another talk from Bjarne – this time on Object-oriented programming. The talk completely avoided theological issues and concentrated on the practical ingredients required for supporting the useful aspects of this most nebulous of concepts. To many people, OOP means slow graphics. To C++ it means

— data abstraction

— strong types

— data hiding

— inheritance

— dynamic binding

all implemented consistently and efficiently.

Tsvi Bar-David, from AT&T training, echoed this view when he described how his way of presenting C++ had changed over a few years. He now does it: concepts first, syntax second.

Keith Gorlen, NIH and Ken Fuhrman from Ampex outlined two fairly different OOPS libraries. Both follow the Smalltalk line of investing supernormal power in a cosmic object and deriving everything else from it, so that all objects may partake in its godhead. Keith no longer ships his library himself, you get it from USENIX.

Mark Linton of Stanford illustrated the design of the InterViews class library for X windows. The design seems rather eccentric, but the library is available in the public domain, I think on the X 11 release tape.

Ragu Raghavan from Mentor showed us a C++ class browser. It is similar to the kind of tool one expects with Smalltalk or Actor. I could only see two drawbacks: Mentor don't currently plan to make it public and its understanding of C++ syntax is limited.

Tom Cargill from the Labs was set up to talk about Pi again. His presentation became highly interactive and was – for my money – the most interesting session of the conference. He opined that the biggest lack in C++ environments was the lack of an intelligent *make* utility which could detect when a change to a class declaration would affect a particular source file. Keith Gorlen suggested that Tom only felt this way because Tom already had a debugger for C++ ! Tom went on in a bantering, Bjarne-baiting way to suggest things that could be dropped from the language to make room for a smart make. Among them he numbered references and operator overloading. My hero !

The conference ended with a bunch of very short talks on miscellanea. The one that raised the most interest was a talk by Michael Tiemann on the GNU ''C++'' project. GNU have a free C compiler which is evolving into a free sort-of C++ compiler. It will run on VAX and Sun. I don't think anyone in Europe has a copy of this – please mail me if you do. The impression I got was that it was some months away from being available, you would need to put considerable effort into getting it to run and it was in some ways gratuitously incompatible with Bjarne's C++. Hello out there – I would like to publish a review of any C++ products you get hold of !

210 people attended the conference. They flew over a million people-kilometers to get there and back. They listened to 8000 hours of C++ per ear.

The conference had a very powerful impact on everyone I spoke to.

Once C++ was a research project at the Labs – now it's a mainstream successor to C.

Complete proceedings of the conference are available from

USENIX Association
PO Box 2299
Berkeley, CA 94710
USA

From the US and Canada, you must send $15 with your order. From Europe, $30. Please mark your order ''C++ Proceedings''.

Hot gossip

I wasn't at the Sun User Group conference in San Jose in December, but normally reliable sources quote Bill Joy as having said something to the effect that...

''AT&T and Sun have founded a joint UNIX Technology Centre in Menlo Park, CA. One of the functions of the UTC is to produce a UNIX out of merging System V and Sun OS (which incorporates BSD). The merged UNIX product will be written in C++.''

Asked ''why C++ ?'', Bill said that the information hiding capabilities of C++ made it a much better choice than C.

Technical tip

Sometimes you can find it convenient to provide an interface to a service by means of a single class. Different implementations of the service can be plugged in later by implementing the service in derived classes....

```
class Service
{
     virtual void Implementation();  // private virtual function
public:
     void Interface();               // public interface
};

void Service::Interface()
{
     Implementation();
}

class Service1 : public Service     // inherits Interface()
{
     void Implementation();         // private implementation
};
```

News from dt+@andrew.cmu.edu

David Tilbrook
dt+@andrew.cmu.edu

Associate Director
Information Technology Center
Carnegie Mellon University
Pittsburgh, PA 15213
U.S.A.

David Tilbrook has been an associate director of the Information Technology Center (ITC) at Carnegie Mellon University (CMU) since June, 1987. His primary responsibility is the evolution of a software managment approach for the exportation of CMU software products.

David came to England in 1981. In 1983 he joined Imperial Software Technology in London and started his PhD at Imperial College (still in progress) on software engineering systems.

David is an honourary life member of the EUUG and has served as the chairperson of numerous conferences in Europe and North America.

Greetings.

This is the first of what may prove to be a series of columns that I will try to submit, time permitting, to the EUUGN. Alain and Sally suggested some sort of informal chat about what is happening here and as many of you probably do not know much about Carnegie Mellon University (home of MACH and Andrew) I will initially concentrate on CMU and our major projects.

But first the burning question ... ''Just what is that '+' behind my userid?''. Here is the official explanation as printed on the back of the Andrew Message System staff members' business cards.

> The plus sign (''+'') in Andrew mail addresses permits a variety of uncommon services. A trailing ''+'' denotes a user by login id (''smith+@andrew''), while ''smith@andrew'' ambiguously matches many users. Thus, any unambiguous name (''Zachariah.Smith@andrew'') is a valid address, and the ''userid+'' form may be thought of as the result of a name lookup.
>
> Additionally, ''+'' is used for automatic classification (''smith+encryption@andrew''), and for special forms (''+dist+<filename>@andrew'', a distribution list). ''+'' is permitted in local addresses by all relevant standards, and causes relatively few problems in non-conforming mailers.

Aren't you glad you asked? By the way, we have a single campus-wide `/etc/passwd` file with almost 8,000 entries and a large annual turn-over.

Assuming that some of you will have attended (ah the pluperfect subjunctive) the USENIX conference in Dallas, you will have seen a demonstration of or can read about the scheme described above.

Since the conference is in two weeks as of this writing and I assume the normal EUUGN delay in publication, I won't tell you about it, other than to say the Information Technology Center (it's their name so we spell it their way) is giving five papers on the Andrew system. The second paper is on the Andrew toolkit by Andy Palay and will be presented at the London EUUG conference.

Some of you may have another question about a rumour regarding a future hire at CMU. Yes indeed ... Jaap Akkerhuis has taken leave of his senses (assuming he had any to begin with) and will be joining me here May 1st. Note that this is after the London conference, but he assures me that that was not the reason for his delay. We are looking forward to his arrival and believe that he will be a tremendous addition to our already star-studded staff.

Now the serious stuff.

The Information Technology Center (the ITC) is a joint IBM/CMU research and development organization. There are currently six full time IBM employees on our 34 member staff. For your information, our relationship is similar to that of IBM and the University of Karlsruhe, who are jointly developing Hector.

There are four major aspects to the ITC development: the network that hooks together the 1000 odd PCs and MACs, and 500 Sun, IBM, and DEC workstations; the Andrew File System (AFS); the Andrew Toolkit; and the Andrew Message System, which includes work on the Office Document Architecture (ODA) toolkit.

The network development group is no longer part of the ITC so I won't try to explain their work in this article.

The Andrew File System

The goal of the Andrew File System is to provide users, application programs, and system administrators with the amenities of a shared file system in a distributed environment with potential growth to thousands of workstations. It shares information between workstations by copying and saving entire UNIX files. Once a workstation has cached a file it can use it independently of the central file system, which dramatically reduces network traffic and file server loads as compared to record-based distributed file systems. Implementation is with many relatively small servers rather than a single large machine. The key fact about the AFS from an end user's viewpoint is that it closely resembles a standard UNIX file system, yet allows the user to sit down at any workstation attached to the local area network and get at the same set of files.

The Andrew Toolkit

The IBM Andrew Toolkit is an extensible object-oriented system for the development, display, and manipulation of graphical objects in a workstation environment running either the X Window System or our own window system. It provides an application starter set, and tools and interface needed to construct additional applications.

The Toolkit Application Set constitutes the core of the Andrew Toolkit. It provides the user with a windowed menu driven environment for document preparation and manipulation, integrated with facilities for executing and managing other Andrew and applications. The Application Set consists of a multimedia editor and several supporting programs along with the text inset for creating documents.

The Toolkit Extension Set supplements the basic Application Set by providing the user and programmer with new sets of facilities and tools for working in extended multi-media environments.

Perhaps the best demonstration of the power of this system is the ability to insert animations, line drawings, rasters, tables and equations into mail messages. But see the Dallas papers or visit Chalmers Tekniska Högokola in Göteborg to play with their Andrew system, because a word is only worth one one thousandth of a picture.

The toolkit and AFS are both described in the Dallas proceedings. The toolkit itself will be available on the X-tape due for release by MIT in the near (well nearly near) future. The AFS is available under certain conditions from IBM ACIS.

Well that's it for now. The next column will introduce some of the other ongoing projects at CMU such as CMUTutor (son of plato), MACH, Camelot, and interactive video disk computer aided learning systems.

Cheers.

David M. Tilbrook (a.k.a., dt)
Associate Director
Information Technology Center
Carnegie Mellon University
4910 Forbes Ave.,
Pittsburgh, PA 15213
USA

Tel: (+1 412) 268-6726

P.S.: The local Public Broadcasting System radio station is called WQED.

Hot news flash

When I wrote the column I was not permitted to give you the following news, but this restriction was lifted as of February 2nd, 1988 when IBM announced five new products. One was their new bitmap display workstation, the 6152 – a very clippy machine with a 40 Meg disk and 4 megabyte of memory for approximately $6k (but don't quote me). The operating system is IBM Academic Operating System 4.3 (IBM/4.3) which contains the *IBM Andrew Toolkit*. Furthermore there is another product of interst to us: the *IBM Andrew File System*, PRPQ 5799-CRH.

They are available to colleges and universities eligible for an educational allowance who have an AT&T and 4.3BSD source licenses. I am not quite sure what that means and I am not going to type in all the license terms. I think that the educational allowance clause may imply that it is restricted to the U.S., but then again I have always though that Göteborg was in Sweden.

The significant thing is that IBM has made a long term commitment to UNIX – not to mention Andrew. Remember the impact they had on the personal computer world when they got into it. Should be interesting.

Draft Proposed ANSI/ISO C Standard and POSIX Standards Developments

Cornelia Boldyreff
corn@ee.surrey.ac.uk
...!uoseev!corn

UK POSIX Panel Convenor
UK C Panel Convenor
Gould Fellow in Software Engineering
Department of Electronic and Electrical Engineering
University of Surrey
Guildford Surrey GU2 5XH

Cornelia Boldyreff is a member of the British Standards Institution technical committee on Application Systems, Environments and Programming Languages. She acts as Convenor and Chairman of the BSI C Language Panel; and is one of the UK Principal Experts on the ISO Working Group on C. She is also Convenor and Chairman of the BSI POSIX Panel; and is one of the UK Principal Experts on the ISO Working Group on POSIX.

1. ANSI/ISO C Standard

1.1 Recent Meetings

1.1.1 The ISO Working Group Meeting

The last ISO/TC97/SC22/WG14 meeting was held in Amsterdam, 16 – 17 November 1987. This meeting was the third meeting of WG14 held at the Free University of Amsterdam hosted by Ed Keizer, the Dutch NNI representative. The meeting was attended by representatives of Denmark, Finland, France, the Netherlands, the United Kingdom (myself) and the United States. Bill Plauger of the USA, Convenor of WG14, chaired the meeting.

With the exception of the Netherlands and the USA, all other countries presented papers of comments on the draft. Many of these had been separately submitted to X3J11 for consideration at their December meeting. A paper summarising outstanding issues in the current draft on which there was consensus that these must be addressed before the draft can become an ISO standard was prepared by the WG members during the meeting. It was presented by Bill Plauger at the December meeting of X3J11.

The main outstanding issues identified by WG 14 are as follows:

— Grouping Parenthesis/Unary Plus

— Multibyte Characters

— Alternative to Trigraphs

— Preprocessor Tokenisation and Semantics

— Lines in Files

— Truncated Files

— Equality and Relational Operators and (`void*`)

There was strong sentiment for another European venue for the next meeting of WG14; and the next meeting of WG14 will be in London on 13 – 14 June 1988.

1.2 Future Meetings and Projected Targets

As anticipated the December meeting of X3J11 voted out a draft of the standard for a second formal public review of two months. This will take place in the first quarter of 1988. Copies of the draft standard will be available for the public from Global Engineering Documents, Inc, by calling (800) 854-7179 or (714) 540-9870. Expected Single Copy Price US$65.00 (draft standard and rationale).

Global Engineering is located in Santa Ana, California, USA, which is in the Pacific Standard Time (PST) zone.

Any comments on this draft will be processed at X3J11's Spring meeting; and the resulting draft will be reviewed by the ISO WG14 meeting in June. If it is acceptable, WG14 will put it forward for registration as a DIS; that is assuming the outstanding ballot on approval of an earlier draft as a DP is successful.

If following the second public review, no substantive changes have been made to the draft by X3J11, it will go forward for administrative processing by ANSI and emerge as an ANSI C Standard in the latter part of 1988. An ISO C Standard is likely to follow either at the end of 1988 or early in 1989. Already the BSI Quality Assurance Services in the UK have in hand the development of a C Compiler Validation Service in anticipation of an approved ISO C Standard.

Recent and Future Meetings:

ANSI X3J11	7 – 11 December 1987	Austin, Texas
BSI IST/5/14 C Panel	9th February 1988	London, England
ANSI X3J11	18 – 22 April 1988	Nashua, New Hampshire
BSI IST/5/14 C Panel	10th May 1988	London, England
ISO TC97/SC22/WG14	13 – 14 June 1988	London, England
ANSI X3J11	15 – 19 August 1988	Cupertino, California
BSI IST/5/14 C Panel	9th August 1988	London, England
BSI IST/5/14 C Panel	8th November 1988	London, England
ANSI X3J11	12 – 16 December 1988	Seattle, Washington
ANSI X3J11	10 – 11 April 1989	Phoenix, Arizona

2. POSIX Standards

2.1 ISO Past and Present Action

The New Work Item on POSIX received official approval by ISO/TC97 in July 1987; and as expected Member Bodies at the ISO/TC97/SC22 Plenary Meeting in September 1987 gave their support to the establishment of SC22/WG15-POSIX with Jim Isaak as the Convenor and the USA providing secretarial support and project editor. To expedite progress on this standard, it was agreed also that IEEE P1003.1 POSIX Draft 12 be registered as a Draft Proposal and forwarded to all SC22 Member Bodies for comments; this resolution was unanimously adopted.

POSIX Draft 12 (ISO DP9945) has been available for comment since the end of November 1987, and is the subject of an ISO ballot closing on the 23rd February 1988. It is anticipated that ISO Member Bodies will

vote giving approval of the Draft Proposal.

The inaugural meeting of ISO WG15 will on the 2nd – 4th March 1988; the first day of this meeting will be specifically set aside for a joint meeting with the OSCRL Working Group. The venue for this meeting will be Birkbeck College, London, England.

2.2 The Future

There are three main hurdles to be cleared in the immediate future:

— the IEEE balloting and approval from the IEEE Standards Board (this began in November 1987 and approval is anticipated in March 1988);

— a 60 day public review period prior to ANSI approval (May 1988 at the earliest);

— progression from an approved ISO DP (dependent on the result of present ISO ballot closing in February 1988) to registration of a Draft International Standard for POSIX (possible in Summer 1988).

The long-term goal is still parallel progression towards an IEEE Full-Use Standard for POSIX and an ISO PO SIX Standard.

Recent and Future Meetings:

IEEE P1003	7 – 11 December 1987	San Diego, Califomia
BSI IST/5/14	26th January 1988	London, England
Joint OSCRL/POSIX	2 March 1988	London, England
ISO TC97/SC22/WG15	3 – 4 March 1988	London, England
IEEE P1003	October 1988	Japan (targeted)
ISO TC97/SC22/WG15	October 1988	Japan (possible 2nd meeting)

C Compiler Validation

Paul D Neale

Senior Certification Officer
British Standards Institute

Paul Neale is an Honours Degree graduate in Computer Science. He has worked as a programmer, technical support specialist and as a manager of software and hardware engineers. In his present position at BSI, he is responsible for the Pascal Compiler Validation Service and establishment of new services such as C compiler validation.

1. Introduction

Since the 1960s when the US Government introduced the concept of compiler testing as a procurement requirement, validation of compilers has been improved and refined and the number of computer languages covered has increased. As standards are published for computer languages, so services are being established to validate the relevant compilers that are available on the open market.

2. So What is Validation?

Basically validation is carried out by submitting a series of test programs to a given compiler and the results collated and reported (witnessed by a representative of the validation authority). The suite of programs are chosen to demonstrate the degree of compliance of the product against the relevant standard. Naturally this basic approach varies from language to language and service to service.

3. Why Validate?

Given the existence of a standard (increasingly international ones) it is logical for a manufacturer to want to publicly demonstrate that their product(s) conforms to the standard. The only meaningful way is to have the product tested by an independent authority such as BSI. The British Standards Institute (BSI) is renowned, world-wide, for its quality assurance and testing services, which cover a vast number of disciplines, not the least of which is the information technology area.

The advantage to the consumer is obvious; an objective report which outlines the conformance of a would-be purchase gives the potential purchaser further information on which to base his decision. He also knows that any programs developed using the validated compiler are more likely to be acceptable to other validated compilers and this improves the chances of portability and successful upgrading.

4. Compiler Validation at BSI

BSI has been operating the Pascal Compiler Validation Service for many years and will be introducing a new service for C compilers. The draft C standard is making good progress towards publication by ANSI in 1988. With this in mind BSI has been involved with reviewing the existing C test suites which are currently available to find one which is suitable as a basis for a C compiler validation suite. BSI has now selected the most suitable candidate, the Plum Hall C test suite from Plum Hall Inc. With this test suite as a basis, BSI intends to develop a C Compiler Validation Service (CCVS).

Naturally the CCVS cannot operate until the ANSI C Standard is published, but in the meantime BSI will be distributing the C test suite along with revisions as the draft C standard emerges. This will eventually lead to release one of the C Validation Suite which all existing customers (for the C test suite) will receive as part of their maintenance agreement.

BSI will also be receiving assistance for the establishment of the CCVS from the EEC as part of their conformance testing services programme. BSI had to submit a tender in collaboration with other standards bodies (in this case AFNOR of France and IMQ of Italy). The BSI consortium has been awarded the contract and so BSI and its international partners will be developing a world-wide service, from the outset.

In addition to Pascal Compiler Validation (and C of course!) BSI also offers a range of test suites and services unique within the IT industry. BSI currently offers a Modula-2 test suite (over 4Mb of code!), with a validation service to follow on from the publication of the standard. BSI also offers the Ada Evaluation Suite and services (under contract to the UK MoD), along with the Pascal Evaluation Suite and Pascal Evaluation Service.

5. *Interested?*

For further details please contact:

Mr P D Neale
Senior Certification Officer
Information Technology
Certification and Assessment Service
BSI Quality Assurance
PO Box 375
Milton Keynes
MK14 6LL
United Kingdom

Telephone: +44 908 (Milton Keynes) 220908
Telex: 827682 (BSIQAS G)
Fax: +44 908 320856
EMAIL: 84:MNU 174 (Telecom Gold)
BSI1@UK.ac.rl.gm (JANET)
bsi1%gm.rl.ac.uk@ukc.uucp
Teletex: UK 944-908320041=bsi

UNIX CLINIC

Nigel Horne
njh@root.co.uk

UniSoft Ltd.

Musical Director
Pangbourne & District Silver Band

Nigel took over the position of Musical Director of the Pangbourne & District Silver Band in May 1986. Since then he has brought the band back into contesting and lead them on a successful trip to Bitz in Southern Germany.

1. Laying The Ground

I have received some comments asking for clarification of which flavours of UNIX will be covered in this column. Clarification? Oh dear, that presumes I had some sort of stance before hand. So I've decided to just relate to the systems I have here, at least that way I can be blamed fair and square if a bloomer appears in this column, and it so happens that what is available here is a fair cross-section and anything not represented will at least bear some close resemblance to one of them.

Based on this I will give examples based on 4.3BSD (Mt. Xinu with NFS actually) on a VAX and System V Release 2. Where there are known differences with 4.2BSD I shall try to highlight them.

2. Cron

`Cron` is a subsystem running on a UNIX system which arranges for set tasks to be run at a previously set time, or set of times. An example of this is a system back up which you may want to be run automatically every weekday during the night, when CPU cycles are cheap. Let us assume that you decide to back up the system using the `dump` program, and that you wish to perform a level 9 `dump` every weekday evening at 10PM.

To tell the system to do this you need to edit `cron`'s database file of jobs. This file is called `/usr/lib/crontab` on 4.3BSD. You will need to be the superuser first as the file is only writable by root (or at least it should be – check on your system!). Add this at the end of the file:

```
0 22 * * 1-5            root            /etc/dump 9uf /dev/rmt0 /usr
```

The first five fields tell `cron` when to run the job. They are: minute of the hour (zero as we want the job to be run on the hour), hour of the day (22 for 10P.M. or 2200 hours), day of the month (an asterisk * means "all"), month of the year, and day of the week (1 for Monday). The "1-5" in the day of the week field states that the job should be run on Mondays to Fridays inclusive. The next field (which I have

separated by a tab character to aid readability) tell the system which user code the job should be run as. As a dump script needs to access privileged files such as tapes and discs this will need be run as "root" – the superuser. After this (separated again by a tab) appears the command and its arguments. So the above example says:

Run the command `/etc/dump 9uf /dev/rmt0 /usr` every weekday evening at 10:00.

The username field does not appear on 4.2BSD systems: on those systems everything runs as root. You will have to play around with the command `su` to have jobs run as a different user on 4.2BSD. Say, for example, you wish to run a special command on a file which doesn't need superuser status to access it, you could have:

`0 4 * * * su` **your_username** `<` **command_file**

Where **command_file** is the filename of a set of commands to be run at 4.00 A.M. every day as userid **your_username**.

On System V.2 the `cron` subsystem is rather different. The system allows users to set up and maintain their own `crontab` file, the system's `crontab` file being that which is owned by root. Instead of editing the `crontab` files directly, a program called `crontab` is provided as a user interface. Only some users are allowed to have `crontabs`, the names of these users is listed in the file `/usr/lib/cron/cron.allow`. The superuser will be expected to vet this list.

Let's use the above example about dumps again. Set yourself up to superuser status. Check that root is allowed to have a `crontab` by examining the `cron.allow` file. If it hasn't, just add it to the list. See whether root currently has a `crontab` file. You can list it by the command

```
crontab -l
```

which will print the `crontab` on the terminal. If you have no `crontab` file `crontab` will tell you. You will see that the format of each entry is the same as on 4.2BSD. To change the `crontab` file all you need is to save the output of the above command, and then edit it with your favourite editor, for example

```
crontab -l > foo
vi foo
```

If there is no current `crontab` file, just simply create one in your current directory, called for example `foo`, using the 4.2BSD format outlines above. Then to overwrite your current `crontab` (or to create one if you don't already have one) all you need to do is to run `crontab` without any arguments, with input redirected, thus

```
crontab < foo
rm foo
```

3. EUnet

I thought I'd briefly mention a few lines about the European (and in fact Worldwide) news and mail facilities available to any UNIX site which has a modem (or X.25 connexion if you're extravagant). Please see also the EUnet column on page 36.

There are several reasons for mentioning it here. Firstly, you may have seen or heard mention of a mystical UNIX network, and maybe even seen a few references to it in organs such as this one, but not understood it's implications or wanted to know more. Secondly, it is a good forum for debate and asking questions (so it is a natural progression from this very column). And thirdly, it gives you an electronic mail contact to many other computer systems worldwide, many of which are running UNIX.

In Europe each country has a main site known as a backbone. This site has various responsibilities, one of which is the administration of the network. You will need to contact this backbone about being added to the net (as it is colloquially called). They will organise "feeds" for the network mail and news. The reason you must contact them first is to prevent anarchy as they try to maintain up to date maps and lists, and because (well there's always a bad side to these things) it'll cost you a small annual fee. The net is not

run to make a profit for the backbone sites, the cost is your share of the backbone's 'phone bill, otherwise you could be repeatedly sending mail to the US at a small percentage of the telephone costs, and the backbone sites have to pick it up. You will also have to pay your share of the cost of getting the network news from the US. For "historical" reasons you will have to pay for all mail you both send *and* receive from the US. The backbone will also be able to arrange for a distribution of software allowing you to receive and read the news items. The site which has offered to be your feed will be willing to help you get started.

Readers of this column may find the newsgroup `comp.unix.questions` a good start.

4. How To Contact Me

You can send questions to me either via EUUG, by direct mail or even using electronic mail if your machine is connected to EUnet either directly or via another machine. If you want to try sending mail electronically try both of the following commands. If neither of them work, it is unlikely that your machine is connected to EUnet.

```
mail mcvax!ukc!root44!njh
```

or

```
mail njh@root.co.uk
```

I'm sorry that I can't enter into any discussions about advice given in this column, and any material sent to me by any of the means above will be deemed to be acceptable for publication.

UNIX User Groups and Publications

John S. Quarterman
jsq@longway.tic.com
uunet!longway!jsq

Austin, TX 78701-3243
U.S.A.

This was taken from the newsgroup `comp.std.unix` of which John is the moderator.

This is the latest in a series of similar `comp.std.unix` articles, intended to give summary information about UNIX User groups and publications; to be accurate, but not exhaustive.

Corrections and additions to this article are solicited and should be directed to the above address.

Access information is given in this article for the following:

user groups:	USENIX, /usr/group, EUUG, AUUG, NZUSUGI, JUS, KUUG, DECUS
journal:	Computing Systems
newsletters:	;login:, CommUNIXations, /usr/digest, EUUGN, AUUGN, NUZ
magazines:	UNIX REVIEW, UNIX/WORLD, IX Magazine, UNIX Systems, UNIX Magazine

Telephone numbers are given in international format, i.e., +n at the beginning for the country code, e.g., +44 is England, +81 Japan, +82 Korea, +61 Australia, +64 New Zealand, and +1 is U.S.A. or Canada.

USENIX is "The Professional and Technical UNIX Association."

USENIX Association
P.O. Box 2299
Berkeley, CA 94710
U.S.A.
+1-415-528-8649
{uunet,ucbvax,decvax}!usenix!office
office@usenix.org

USENIX sponsors two USENIX Conferences a year, featuring technical papers, as well as tutorials, and with vendor exhibits at the summer conferences:

February 9 – 12 1988	Grand Kempinski Hotel, Dallas, TX, concurrent with UniForum
June 20 – 24 1988	Hilton Hotel, San Francisco, CA
January 31 – February 3 1989	Town & Country Inn, San Diego, CA
June 12 – 16 1989	Hyatt Regency, Baltimore, MD
January 23 – 26 1990	Washington, DC
June 11 – 15 1990	Marriott Hotel, Anaheim, CA
January 22 – 25 1991	Dallas
June 10 – 14 1991	Opryland, Nashville

They also sponsor workshops, such as

May 12 – 13 1988	Omni Shoreham Hotel, Washington, DC Fifth Workshop on Real-Time Software and Operating Systems IEEE Computer Society and USENIX Association
August 29 – 30 1988	UNIX Security, Portland, OR
September 26 – 27 1988	UNIX & Supercomputing, Pittsburgh, PA
October 17 – 20 1988	C++ Conference (tentative), Denver, CO
November 17 – 18 1988	Large Installation System Administration II, Monterey,CA

Proceedings for all conferences and workshops are available at the door and by mail later.

USENIX publishes ";login: The USENIX Association Newsletter" bimonthly. It is sent free of charge to all their members and includes technical papers. There is a USENET newsgroup, `comp.org.usenix`, for discussion of USENIX-related matters.

In 1988, USENIX will start publishing a new refereed quarterly technical journal, "Computing Systems: The Journal of the USENIX Association", in cooperation with University of California Press.

They also publish an edition of the 4.3BSD manuals, and they occasionally sponsor experiments, such as methods of improving the USENET and UUCP networks (e.g., `uunet`), that are of interest and use to the membership. They distribute tapes of contributed software and are pursuing expanding that activity.

There is a USENIX Institutional Representative on the IEEE P1003 Portable Operating System Interface for Computer Environments Committee. That representative also moderates the USENET newsgroup `comp.std.unix`, which is for discussion of UNIX-related standards, especially P1003. For more details, see the posting in `comp.std.unix` about Access to UNIX-Related Standards.

/usr/group is a non-profit trade association dedicated to the promotion of products and services based on the UNIX operating system.

/usr/group
4655 Old Ironsides Drive, Suite 200
Santa Clara, California 95054
U.S.A.
tel: +1-408-986-8840
fax: +1-408-986-1645

The annual UniForum Conference and Trade Show is sponsored by /usr/group and features vendor exhibits, as well as tutorials and technical sessions.

February 8 – 11 1988	Infomart, Dallas, TX, concurrent with USENIX
February 28 – March 3	1989 Moscone Center, San Francisco, CA
January 23 – 26 1990	Washington Hilton, Washington, DC
January 22 – 25 1991	Infomart, Dallas, TX
January 21 – 24 1992	Moscone Center, San Francisco CA (tentative)

They also sponsor a regional show, UniForum D.C.:

Aug 2-4 1988	Washington Hilton Hotel, Washington, D.C.

Proceedings for all conferences are available at the shows and later by mail.

/usr/group publishes "CommUNIXations", a member magazine that features articles by industry leaders and observers, technical issues, standards coverage, and new product announcements.

/usr/group also publishes the "UNIX Products Directory", which lists products and services developed specifically for the UNIX operating system. "/usr/digest" is also published by /usr/group. This newsletter covers product announcements and industry projections, and is sent to members biweekly.

/usr/group has long been deeply involved in UNIX standardisation, having sponsored the "/usr/group 1984 Standard", providing an Institutional Representative to the IEEE P1003 Portable Operating System for Computer Environments Committee, and sponsoring the /usr/group Technical Committee on areas that

P1003 has not yet addressed. They have recently produced an executive summary, "Your Guide to POSIX", and a technical overview "POSIX Explored", and funded production of a draft of a "Rationale and Notes" appendix for IEEE 1003.1.

EUUG is the European UNIX systems Users Group.

EUUG Secretariat
Owles Hall
Buntingford
Herts SG9 9PL
England
Telephone +44 763 73039
Telefax +44 763 73260
uunet!mcvax!inset!euug
euug@inset.co.uk

They have a newsletter, EUUGN, and hold two conferences a year:

11 – 15 April 1988, London, England
3 – 7 October 1988, Lisbon, Portugal

AUUG is the Australian UNIX systems Users Group.

AUUG
P.O. Box 366
Kensington
N.S.W. 2033
Australia
uunet!munnari!auug
auug@munnari.oz.au

Phone contact can occasionally be made at +61 3 344 5225.

AUUG holds at least one conference a year, usually in the Spring (August or September). The next one will be in Melbourne on 13-15 September 1988, will be the first three day meeting, will have a larger equipment exhibition than any before, and will be professionally organised for the first time.

They publish a newsletter (AUUGN) at a frequency defined to be every 2 months.

The New Zealand UNIX Systems User Group, Inc. (NZUSUGI) has an annual meeting (in June this year), and publishes a newsletter, "NUZ".

New Zealand UNIX Systems User Group
P.O. Box 585
Hamilton
New Zealand
+64-9-454000

The Korean UNIX User Group has a software distribution service and a newsletter.

Korean UNIX User Group
ETRI
P.O. Box 8
Daedug Science Town
Chungnam 300-32
Republic of Korea
+82-042-822-4455

The Japan UNIX Society has two meetings a year, and a newsletter.

Japan UNIX Society
#505 Towa-Hanzomon Corp. Bldg.
2-12 Hayabusa-cho
Chiyoda-ku, Tokyo 102
Japan
+81-03-234-2611

There are similar groups in other parts of the world. If such a group wishes to be included in later versions of this access list, they should please send me information.

There is a partial list of national organisations in the November/December 1987 CommUNIXations.

Also, DECUS, the Digital Equipment Computer Users Society, has a UNIX SIG (Special Interest Group) which participates in its meetings, which are held twice a year.

DECUS U.S. Chapter
219 Boston Post Road, BP02
Marlboro, Massachusetts 01752-1850
U.S.A.
+1-617-480-3418

See also the USENET newsgroup `comp.org.decus`.

The Sun User Group (SUG) is an international organisation that promotes communication among Sun users, OEMs, third party vendors, and Sun Microsystems, Inc. SUG sponsors conferences, collects and distributes software, produces the README newsletter and T-shirts, sponsors local user groups, and communicates members' problems to Sun employees and management.

Sun Microsystems User Group, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.
+1 415 960 1300
users@sun.com
sun!users

They have not set a date/location for the 1988 conference yet, but are actively looking for a hotel (with good pricing and lots of room). They've narrowed it down to several locations – Miami/Tampa Florida, Houston/Dallas Texas, and New Orleans LA. The date will probably be very early December, 1988.

The main general circulation (more than 10,000 copies per issue) magazines about the UNIX system are

UNIX REVIEW
Miller Freeman Publications Co.
500 Howard Street
San Francisco, CA 94105
U.S.A.
+1-415-397-1881

IX Magazine
Storyplace Ltd.
137-139 Euston Road
London NW1 2AU
England
+44-48-6227661

UNIX/WORLD
Tech Valley Publishing
444 Castro St.
Mountain View, CA 94041
U.S.A.
+1-415-940-1500

UNIX Systems
Eaglehead Publishing Ltd.
Maybury Road
Woking, Surrey GU21 5HX
England

UNIX Magazine
Jouji Ohkubo
c/o ASCII Corp.
jou-o@ascii.junet
+81-3-486-4523
fax: +81-3-486-4520
telex: 242-6875 ASCIIJ

In addition:

Computing Systems
USENIX Association
P.O. Box 2299
Berkeley, CA 94710
U.S.A.
+1-415-528-8649

CommUNIXations
/usr/group
4655 Old Ironsides Drive, Suite 200
Santa Clara, California 95054
U.S.A.
+1-408-986-8840

Some of the above information about magazines was taken from the November/December 1987 issue of CommUNIXations, which also lists some smaller-circulation magazines and newsletters. The following information about bookstores was taken from the same issue. In the interests of space, I have arbitrarily limited the selection listed here to those bookstores or suppliers specifically dedicated to computer books, and not part of other organisations.

Computer Literacy Bookshop
2590 No. First St.
San Jose, CA 95131
U.S.A.
+1-408-4350-1118

Cucumber Bookshop
5611 Kraft Ave.
Rockville, MD 20852
U.S.A.
+1-301-881-2722

UNIX Book Service
35 Bermuda Terrace
Cambridge, CB4 3LD
England
+44-223-313273

Jim Joyce's UNIX Book Store
47 Potomac St.
San Francisco, CA 94117
U.S.A.
+1-415-626-7581

AT&T and Sun Microsystems Announce New Computer Platform

Janet Davis
ukc!uel!janet

AT&T Unix Europe
London
U.K.

Janet Davis is Market Communications Manager for AT&T Unix Europe. She has been with AT&T since September 1987 and is responsible for the promotion of UNIX System V and related products and services throughout Europe. Janet was previously Marketing Executive for THORN EMI Computer Software.

AT&T and Sun Microsystems, Inc. announced in January this year an agreement whereby AT&T may acquire up to a 20 percent interest in Sun over the next three years.

As part of this agreement, AT&T has agreed to purchase, at Sun's option over a three-year period, newly-issued common shares amounting to up to 15 percent of Sun's outstanding common stock. Sun can sell the shares to AT&T in installments at a set premium. AT&T can purchase the remaining five percent of the shares in the open market. At recent trading levels for Sun, AT&T's investment in the Mountain View, California-based computer company would be about $300 million.

The main aim of this investment is to cement the previous business agreements AT&T have with Sun Microsystems.

Back in October AT&T and Sun had announced that they would work together to develop a new computer platform that will use a unified version of AT&T's UNIX System V computer operating system, as well as Sun's recently announced Scaleable Processor Architecture (SPARC), a flexible microprocessor design for chips that use reduced instruction-set computing (RISC) technology.

UNIX System V for the new platform will incorporate popular features of Berkeley 4.2 system and SunOS. These features will include networking and graphics features such as the Network File System (NFS) and X.11/NeWS, a graphic user interface. These features are already included in AT&T's Application Operating Environment (AOE). The AOE defines a set of interfaces aimed at providing a vendor independent environment that allows end-users to have a software environment that is independent of the underlying hardware. NFS and X/NeWS have been included as components of the networking and user interface parts of the AOE.

This co-operation between AT&T and Sun centres on their agreed common objective; to produce a computer platform that will be unsurpassed in its ability to protect customers' software investments, while allowing them to take full advantage of technological innovation.

The new platform will be created in phases. By mid-1988, Sun will make available a version of SunOS that will conform to AT&T's System V Interface Definition. In 1989, AT&T will offer UNIX System V incorporating key Berkeley 4.2 system and SunOS features.

AT&T's investment in Sun Microsystems will ensure that Sun has the financial resources to fulfill its role in the relationship and maintain Sun's independence.

The agreement guarantees premium financing for Sun's continued growth while the structure of the agreement will enable Sun to remain independent and aggressive in the market.

"Today's move will assure our customers and the industry at large that the endeavour undertaken by AT&T and Sun to establish a RISC/UNIX-based standard computing platform represents a strategic commitment on the part of both companies, and that the financial resources necessary to sustain it are in place", said Vittorio Cassoni, president of AT&T's Data Systems Group.

AT&T's investment in Sun is purely a financial transaction meant to strengthen the alliance between the two companies. "It is not an expansion of our previous agreements and will have no direct effect on either company's sales or product programs", Cassoni said. "Through this investment, AT&T will benefit from the value creation of our joint efforts and from the expected future growth of Sun."

Added Scott McNealy, president and chief executive officer of Sun, "At Sun's current rate of growth and cash usage, we foresee the need to raise additional capital during the next 18 months. Among the options we considered, we view this to be by far the most attractive option with the greatest potential benefit to Sun and its stockholders."

As part of the agreement, AT&T will receive a seat on Sun's Board of Directors, and plans to nominate Vittorio Cassoni. His nomination will be voted upon by the members of Sun's board at its next meeting.

Book Review: The X/OPEN Portability Guide

Alain Williams
addw@phcomp.uucp

Parliament Hill Computers
London, U.K.

X/OPEN Portability Guide X/OPEN, Elsevier Science Publishers B.V., January 1987, 5 Volumes, ISBN 0-444-70179-6 also individually – see below. Price 350 Guilders, Soft Back.

Reviewed by Alain Williams of Parliament Hill Computers.

The 5 volumes are called:

1. XVS Commands and Utilities (0-444-70174-5),
2. XVS System Calls and Libraries (0-444-70175-3),
3. XVS Supplementary Definitions (0-444-70176-1),
4. Programming Languages (0-444-70177-X),
5. Data Management (0-444-70178-8).

At first sight I took this to be an A5 spiral bound set of UNIX reference manuals, the first two corresponding to section 1, and sections 2 to 7 of the familiar Bells labs layout – the main thing that it lacked was the permuted index. Each volume is divided into several parts each of which has its own contents page, this made navigation difficult – finger tabs would have helped.

But the Guide is more than that. It is a definition of a CAE (Common Applications Environment). What this means is that it is a list of services (programs) and interfaces (system calls, file formats) that the applications writer can expect a conforming environment to supply. Because some systems do not/cannot supply all of the above, it is clearly marked which are the optional entries. It, unfortunately, does not say how (e.g. at compile time) the programmer is able to configure his software to cope with variation.

Standards to help with physical porting are given, i.e. what are the preferred tape and floppy formats, I noted that no QIC tapes were mentioned, and in the section on `uucp` I read *"The versions of uucp must be compatible"* – not very useful.

This is a standard for UNIX Systems suppliers to aim at, unfortunately the software writer has to take a much more pragmatic approach in that his software *has* to port onto the machine that the customer supplies. This standard does supply him with something to aim at, a starting point from which variations can be made and, hopefully, as more systems conform to the XVS the porting job will become easier.

Ideally this should be the only UNIX manual that is needed by a software house, unfortunately to do so will mean that they won't know about the extra features (or bugs) supplied with a particular system. These features are inserted with the aim of giving a product an edge on the competition and can't be ignored. It is a standard that programmers should be aware of and these volumes would make a valuable addition to the reference shelf.

XVS, by the way, stands for X/OPEN System V specification.

AUUG

Membership Categories

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

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

The membership categories are:

Institutional Member
Ordinary Member
Student Member
Honorary Life Member

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

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

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

Are you an AUUG member?

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

Honorary Life Memberships are a category that isn't relevant yet. This membership you can't apply for, you must be elected to it. What's more, you must have been a member for at least 5 years before being elected. Since AUUG is only just approaching 3 years old, there is no-one eligible for this membership category yet.

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

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

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

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

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

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

Robert Elz

AUUG Secretary.

AUUG

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

*UNIX is a registered trademark of AT&T in the USA and other countries

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

AUUG Membership Secretary
P O Box 366
Kensington NSW 2033
Australia

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

I, ... do hereby apply for

☐ Renewal/New* Membership of the AUUG	$55.00	
☐ Renewal/New* Student Membership	$30.00	(note certification on other side)
☐ International Surface Mail	$10.00	
☐ International Air Mail	$50.00	

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

* Delete one.

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

Date: __/__/__ Signed: _____

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

For our mailing database - please type or print clearly:

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

Address: .... (ah)

..... Net Address: .....

.....

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

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

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

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

Name on card: _____ Signed: _____

Office use only:

Chq: bank _____ *bsb* ____-____ *a/c* _____ *#* _____

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

Who: _____ *Member#* _____

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

I, ... certify that

..... *(name)*

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

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

Title: _____ Signature: _____

AUUG

Application for Institutional Membership
Australian UNIX* systems Users' Group.

*UNIX is a registered trademark of AT&T in the USA and other countries.

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

AUUG Membership Secretary
P O Box 366
Kensington NSW 2033
Australia

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

..... does hereby apply for

☐ New/Renewal* Institutional Membership of AUUG $250.00

☐ International Surface Mail $ 20.00

☐ International Air Mail $100.00

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

* Delete one.

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

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

Date: __/__/__ Signed: _____

Title: _____

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

For our mailing database - please type or print clearly:

Administrative contact, and formal representative:

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

Address: .... (ah)

..... Net Address: .....

.....

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

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

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

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

Name on card: _____ Signed: _____

Office use only: **Please complete the other side.**

Chq: bank _____ *bsb* ____-____ *a/c* _____ *#* _____

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

Who: _____ *Member#* _____

Please send newsletters to the following addresses:

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

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

Write "unchanged" if this is a renewal, and details are not to be altered.

Please indicate which Unix licences you hold, and include copies of the title and signature pages of each, if these have not been sent previously.

Note: Recent licences usally revoke earlier ones, please indicate only licences which are current, and indicate any which have been revoked since your last membership form was submitted.

Note: Most binary licensees will have a System III or System V (of one variant or another) binary licence, even if the system supplied by your vendor is based upon V7 or 4BSD. There is no such thing as a BSD binary licence, and V7 binary licences were very rare, and expensive.

- ☐ System V.3 source
- ☐ System V.2 source
- ☐ System V source
- ☐ System III source
- ☐ 4.2 or 4.3 BSD source
- ☐ 4.1 BSD source
- ☐ V7 source
- ☐ Other *(Indicate which)* .....

- ☐ System V.3 binary
- ☐ System V.2 binary
- ☐ System V binary
- ☐ System III binary

AUUG

Application for Newsletter Subscription
Australian UNIX* systems Users' Group.

*UNIX is a registered trademark of AT&T in the USA and other countries

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

AUUG Membership Secretary
P O Box 366
Kensington NSW 2033
Australia

- Please don't send purchase orders — perhaps your purchasing department will consider this form to be an invoice.
- Foreign applicants please send a bank draft drawn on an Australian bank, or credit card authorisation, and remember to select either surface or air mail.
- Use multiple copies of this form if copies of AUUGN are to be dispatched to differing addresses.

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

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

Address: .... (ah)

..... Net Address: .....

.....

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

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

For each copy requested, I enclose:

☐ Subscription to AUUGN	$ 55.00
☐ International Surface Mail	$ 10.00
☐ International Air Mail	$ 50.00

Copies requested (to above address) _____

Total remitted AUD$_____

(cheque, money order, credit card)

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

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

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

Name on card: _____ Signed: _____

Office use only:

Chq: bank _____ *bsb* ____-____ *a/c* _____ *#* _____

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

Who: _____ *Subscr#* _____

AUUG

Notification of Change of Address
Australian UNIX* systems Users' Group.

*UNIX is a registered trademark of AT&T in the USA and other countries.

If you have changed your mailing address, please complete this form, and return it to:

AUUG Membership Secretary
P O Box 366
Kensington NSW 2033
Australia

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

Old address (or attach a mailing label)

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

Address: .... (ah)

..... Net Address: .....

.....

.....

.....

New address (leave unaltered details blank)

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

Address: .... (ah)

..... Net Address: .....

.....

.....

.....

Office use only:

Date: ___/___/___

Who: _____ *Memb#* _____