

AUSTRALIAN UNIX USERS GROUP NEWSLETTER

CONTENTS

Editorial	1
Balance sheet for AUUG meeting January 1980	4
Report on World UNIX Meeting, Melbourne, October 1980	5
Melbourne agenda and attendees	8
Melb. Paper - 'Share Scheduling Works!'	11
Melb. Paper - 'SUN - The Sydney UNIX Net'	15
Melb. Paper - 'MX - An Indirect Many-to-One Terminal Driver'	18
Melb. Paper - 'BFI-11 - An Interface for Network Applications'	21
Software P and E - 'UNIX with Satellite Processors'	24
Delaware conference agendas	34
A Delaware conference report	45
Another Delaware report (very poor copy) from CUUGN	51
Letters about the software catalogue	54
More from the CUUGN	60
Letters	66
An interesting job in Western Australia	80

Editors Rave

Well, here it is. The LAST AUUGN for Volume two. And another bumper issue it is too.

Number 6 was due for production in the first few weeks of October, but I took two weeks holiday down on the farm. Good country air and good country people. Much better than producing AUUGNs.

Then on the 16th we all packed up and headed for Melbourne for the conference. I delayed again, hoping for a flood of copies of the papers presented in Melbourne. Well here it is, November, and AUUGN is still looking up at me from my desk. So I have gathered the trickle of presented papers, the flood of letters, odds and ends, mixed well with my own policy statements about future and past AUUGNs, and stuffed it all into the mail bag.

Future AUUGNs

I should like to make the following policy statement.

The Newsletter in future shall be considered a public journal and a Western Electric license will not be required to receive it. The newsletter will publish general descriptions of modifications or extensions to UNIX, and short bug fixes. The Newsletter will not publish lengthy extracts or modifications to original UNIX source code.

Advertising will not be accepted, although new product announcements will be accepted, with the proviso that the editors decision is final as to whether they should be classified as advertisements or not.

Now that I have that off my chest, I should point out that nothing much will change, since the two points above have been observed (most of the time) in volumes one and two.

AUUGN Backissues

At this time I have on hand about ten copies (on A4 bond paper) of ALL backissues. When this stock is exhausted backissues will be available ONLY on micro-fiche.

This has been necessary since I can only produce a 'reasonable number' of extra issues with each edition. Once these have been distributed to new subscribers, it is not cost effective to produce small numbers of backissues on bond paper. The cost of preparing and duplicating micro-fiche is small in comparison and most subscribers would have access to micro-fiche equipment.

Reproduction Of Material From Other Newsletters

Recently, the editor of the UK newsletter (see letter this issue) suggested that we (the editors) should do two things in future.

Always include a detailed contents page for each issue and always reproduce the table of contents from other newsletters. Readers then have some basis to judge whether to obtain the whole of the other newsletters or not. Second, we should acknowledge items from other newsletters.

From this issue forward I shall endeavour to do these things, though I don't know exactly what level of detail I will be able to include in the contents page.

At The World Meeting

The 'World UNIX Meeting' held in Melbourne in mid October was probably one of the pleasantest meetings I have been to. The only trouble was the lack of overseas visitors. Our thanks should go here to Ian Perry (L.E.R.S. France) and John Hine (Victoria University, New Zealand) for at least showing the overseas flags.

I am also grateful to Ian Perry for supplying me with a report on the meeting which I have included in this issue, along with the few papers I have received from the speakers.

I was also contacted by several new UNIX users during the 'wine and cheese' sessions, and I am happy to answer questions on the newsletter, licensing or UNIX in general.

The Software Catalogue

Now that the students exams are over, further progress is being achieved on the catalogue. Some replies have been received from interested subscribers and these are included in this issue.

As I mentioned at the Melbourne meeting, we have been thinking about how to fund the catalogue, and feel the following scheme has merits. Once the database is running, all queries would be charged a set fee. Should the recipients of the output from the query provide a more detailed review of the software obtained a substantial part of the service fee will be refunded.

I don't know whether it is practical to distribute complete copies of the database, and associated software archives due to the sheer bulk of data. This may be possible for certain sites far removed from Sydney but the cost for tapes alone is likely to run into hundreds of dollars.

Anyone For RSX/IAS

Andrew Hume, of the A.G.S.M. soon to be at Bell Labs, drew my attention to a copy of the 'RSX/IAS Fall 1980 Menu', in which appeared a list of things users of these systems would like to see in their systems. A few good items are reproduced below.

Provide a mechanism for incorporating user-defined commands into the system command line interpreter.

Add support to the MCR to automatically attach a task to its terminal when it is invoked and allow multiple copies of a task to be invoked from a terminal

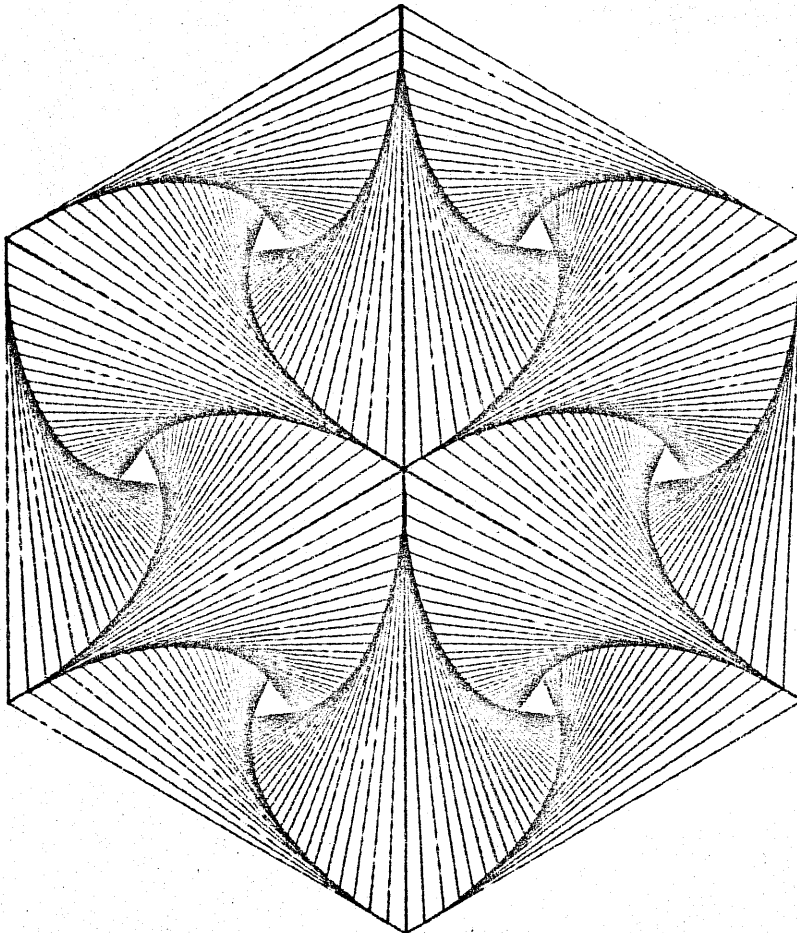
Provide assembly language symbolic debugging tool with support for PDP-11 instruction mnemonics and local and global symbols.

Support overlap disc seeks and disc I/O seek optimisation.

Provide a general purpose tape utility (ASCII/BCD/EBCDIC).

Well, rush out and buy your licenses now.....

Pic Of The Month



Peter Ivanov
Dept. of Computer Science
Electrical Engineering
PO Box 1
Kensington 2033
AUSTRALIA

(02) 662-3781

BALANCE SHEET FOR THE UNIX USER'S GROUP
MEETING OF JANUARY, 1980, IN CANBERRA

<u>INCOME</u>		<u>EXPENDITURE</u>	
1. Conference Registration, 45 delegates @ \$10.00	\$450.00	1. Luncheon and teas, billed for 44 @ \$7.00 (cheque #1)	\$308.00
2. Double payment due to late arrival of Commonwealth cheque.	10.00	2. Cheque book	1.50
3. Interest on account	1.47	3. Stamps and envelopes (cheque #2)	14.85
4. Credit on unused cheques	1.30	4. Charge for closing account	3.00
	\$462.77	5. Refund of overpayment (cheque #3)	10.00
		6. Cheque to Melbourne Unix Group (cheque #4)	125.42
		7. Balance at 30 Sept 80	0.00
	\$462.77		

Signed by *Paul Bently* Convenor

L.E.R.S.

LABORATOIRES D'ETUDES ET DE RECHERCHES SYNTHELABO



Paris, October 27 th, 1980

Correspondance à adresser :

Mr. Peter IVANOV
Dept of Computer Science
University of New South Wales
P.O. Box 1, Kensington 2033
AUSTRALIE

Dear Peter,

Enclosed is a copy of the report I have prepared for the UK Unix User Group Newsletter and my bosses on a meeting I recently attended in Australia ! You may be interested in it ! If you feel it doesn't give a fair representation of the meeting you needn't print it. I enjoyed meeting you all in Melbourne regards to everyone.

Yours sincerely

Ian R. Perry

Report on International Unix meeting Held at Monash University,
Melbourne 18-19/10/80

This meeting followed the second part of the IFIP80 conference and was attended about 50 people. Unfortunately there were only 2 true foreigners present myself and John Hine from New Zealand, however residents of Melbourne regard anyone from Sydney as a foreigner. It was rather disappointing for everybody to see that no Americans participated and in particular that neither Bell nor Western Electric felt able to send a representative. It was good, however, to see that Unix and PWB were frequently mentioned at the IFIP 80 meeting.

Before I arrived in Australia I was lucky enough to visit New Zealand and one Unix site at the University of Canterbury in Christchurch. At least three of the NZ universities hold licences or are very interested in unix. Hopefully with decisions about to be made concerning new computer systems for all the NZ universities some large scale usage of unix may start there soon.

Well what happened at the user group meeting. The meeting lasted two days and was the first Australian User Group meeting to be held in Melbourne. The general flavour of the meeting was rather like a two day version of a UK User Group meeting, i.e. good fun, full of laughs, with interesting presentations and interesting people to meet, the beer and food was not bad either. Unfortunately there were no charming young ladies present (I heard that one presented a ghastly paper at the previous meeting and got rather an unfair hearing). IFIP 80 managed to find plenty of charming young ladies some of whom even programmed computers.

The unix meeting started with a sparkling demonstrations by David Hunt of how to sort out a Hungarian Magic Cube in under 2 minutes. The explanation lasted half an hour and most people still could not sort their cubes out. After lunch on the first day the serious side of the weekend started with a short business meeting. Although Australians in general seem less formal than Europeans and their user group is less formally organised than the UK group Australians are punctual and the meeting ran much more to schedule than the last few UK meetings even if the programme was marked "probable agenda".

I started off the technical sessions by describing our use of unix at L.E.R.S. and was followed by several more interesting presentations!! Dave Horsfall from University of New South Wales (UNSW described their use of SCCS (Source Code Control System), i.e. the most interesting part of PWB (apart from unix itself) and described SCCS itself for the uninitiated. Ken McDonnell (who must be thanked for his efficient organisation of the meeting) gave an insight into the use of paging unix on VAX. David Milway from UNSW described two forms of high speed communications interface which he has designed for networking applications; these appear to be interesting alternatives to DRII type devices and may at some future date be available commercially. Piers Dick-Lauder described Sydney University's approach to networking (this subject was rediscussed inconclusively on the second day, see later). Graham Menhennitt ended the session by describing how they manage students at Melbourne University and then digressed into talking about bridge playing programs.

After beer and more excellent food there was a panel discussion held in a room called "the pit". The panel for this was chaired by John Lions and included Ken McDonnell from Monash University, Peter Ivanov (The Australian Newsletter editor) from UNSW, Richard Miller from Wollongong and guess who from Paris. My participation in this panel was restricted to comments on the European scene and how it might develop in the 80s. The rest of the discussion consisted of a mixture optimism and pessimism about unix and the future. Specific comments were: that the late seventies were the great days for Unix, that distributions were becoming very difficult to handle, that unix would be superseded by something else in the 80s, etc. The highlights of the panel session were John Lion's anecdotes about the early days of Unix at UNSW. For me, however, the highlight of the first day was seeing the kangaroos, wallabies and emus in the zoology department park at Monash when some of us felt a break and walk were essential between cocktails and dinner. Sunday morning started with three further presentations. Ross Nealon from Wollongong described their experiences with multiplexed files and inspired some questions about named pipes and their differences from multiplexed files. Paul Dunn from Melbourne University described a language called ICON which some people might say is a better replacement for SNOBOL4. Adrian Freed from UNSW (but rumoured to be coming to work in Europe soon) ended the presentations by giving his views of how to make miscellaneous intelligent terminals provide new useful facilities but all look alike to your teletype drivers.

After coffee an interesting but rather inconclusive discussion on unix networking in Australia was held. It appears that geographically adjacent sites will continue to develop local networks using various protocols but that access of some sort must be available to distant (overseas even) unix and non unix sites. The meeting ended after Sunday lunch with a poorly attend SIG on graphics and CAD and a visit to the Monash unix site. My conclusions are that it was a very useful meeting even if it did lack the hoped for substantial overseas participation. I learnt a lot, including details of some new sites in Europe, made lots of friends and look forward to my next visit to Australasia.

Ian R. Perry 27/10/80

UNIX USER'S MEETING

Melbourne
18, 19 October 1980

Probable Agenda

SATURDAY - October 18:

- 10:00 Registration
Coffee
General Chat
- 12:00 Lunch
- 13:30 Welcome
Introductions
General Business
- 14:15 Dave Horsfall, CSU, UNSW
"SCCS at the CSU"
(We are using it, Why aren't you?)
- Ken McDonell, Comp Sci, Monash Uni
"UNIX within VMS"
(A small step forward for "them", a giant leap
backwards for "us")
- UNIX Licences (??)
- 15:15 Coffee
- 15:45 David Milway, Comp Sci, UNSW
"A High Speed Communication Interface for Network
Applications"
- Piers Dick-Lauder, Basser Comp Sci, Sydney Uni
"The UNIX Net"
- Graham Menhennitt, Comp Sci, Melbourne Uni
"Managing Students at Melbourne"
- 17:00 Beer
Wine
Munchies
- 18:30 Dinner
- 20:00 Panel Discussion
"UNIX into the '80s"
John Lions, Comp Sci, UNSW (chair)
Ian Perry, L.E.R.S., Paris
Ken McDonell, Comp Sci, Monash
Richard Miller, Comp Sci, Wollongong
Peter Ivanov, Comp Sci, UNSW

UNIX is a trade mark of Bell Laboratories

UNIX USER'S MEETING

Melbourne
18, 19 October 1980

Probable Agenda

SUNDAY - October 19:

09:00 Ross Nealon, Comp Sci, Wollongong
"Multiplexed Files - Implementation and Experience"

Paul Dunn, Comp Sci, Melbourne
"A Vax Implementation of ICON"

Adrian Freed, AGSM, UNSW
"A slightly intelligent terminal for UNIX"
(How to make people with their own terminal driver
look silly)

10:15 Coffee

10:45 Discussion
Australian Unix Network
- uucp / Sydney net / Berknet ??

12:00 Lunch

13:30 Site Visit (Monash) or
Special Interest Groups or
More Talks or
.....

Ron Baxter	CSIRO Div. Maths & Statistics, N.S.W.	A19	Y
David Bellair	Comp. Centre, Monash Uni.		Y
Rod Bilson	Uni. of Tasmania	B17-18)	
J.A. Blair	Dept. Comp. Science, Uni. New South Wales		
Geoff Cole	Uni. Comp. Centre, Sydney Uni.	A16-18)	
Paul Dunn	Dept. Computer Science, Uni. of Melbourne		
Michael Ellis	Dept. Maths, RMC Duntroon	B18	Y
Robert Elz	Dept. Computer Science, Uni. of Melbourne		Y
John Field	CSIRO Div. Maths & Stats, Glen Osmond, S.A.		Y
Adrian Freed	Communications Dept., Elec. Eng., Uni. of N.S.W.	A16-18)	
Mike Georseff	Dept. Comp. Science, Monash Uni.		Y
George Gerrity	Dept. Maths, RMC Duntroon	B17-18)	
Ross Green	Dept. Computer Science, Uni. of Melbourne		Y
Richard Grevis	Dept. of Comp. Science, Uni. of New South Wales	A16-18)	
Lindsay Harris	Comp. Services Unit., Uni. of New South Wales		Y
Kevin Hill	Dept. of Comp. Science, Uni. of New South Wales	B16-18)	
John Hinze	Dept. Inf. Science, Victoria Uni. of Wellington (N.Z.)	A17-20)	
Bill Hollier	Dept. Computer Science, Uni. of Melbourne		Y
Steve Horsfall	Comp. Services Unit., Uni. of New South Wales	A17-18)	
Andrew Hume	AGSM, Uni. of New South Wales	A16-18)	
Peter Ivanov	Dept. Comp. Science, Uni. of New South Wales	B16-18)	
Piers Lauder	Dept. Comp. Science, Sydney Uni.	B17-18)	
John Lions	Dept. Comp. Science, Uni. of New South Wales	A17-18)	
Joe Lonso	Dept. Computer Science, Uni. of Melbourne		Y
Chris Maltby	Dept. Comp. Science, Sydney Uni.	A17-18)	
Ken McDonnell	Dept. Comp. Science, Monash Uni.		Y
Tony McGrath	Wollonsongs Institute of Education	A17-18)	
Richard Miller	Computing Science Dept., Uni. of Wollonsongs		Y
David Milway	Dept. Comp. Science, Uni. of New South Wales	B16-18)	
George Mohay	Dept. Maths & Comp. Science, Queensland Inst. Technology		Y
Ross Nealon	Computing Science Dept., Uni. of Wollonsongs		Y
I.R. Perry	LERS-Synthelabo, Paris		Y
Ian Roberts	Sample Survey Centre, Sydney Uni.	A17-18)	
Mark Ross	Dept. of Computing, R.M.I.T.		Y
Chris Rowles	Dept. of Chemical Engineering, Sydney Uni.	A16-18)	
Munro Saunders	Uni. of New South Wales	A13-18)	
Peter Swain	Uni. of New South Wales	A13-18)	
Colin Webb	AGSM, Uni. of New South Wales	B16-18)	
Bill X	Dept. Computer Science, Uni. of Melbourne		Y
Fred X	Dept. Computer Science, Uni. of Melbourne		Y
George X	Dept. Computer Science, Uni. of Melbourne		Y
Ken Yap	Dept. Comp. Science, Sydney Uni.	A17-18)	

. a		V<-+>A
. b		
. c	Accomod. Type	--+
. d	Accomod. Nights	---+
. e	Sunday Lunch	-----
. f	Dinner	-----

2%

Share Scheduling Works!

Piers Lauder

Basser Department of Computer Science
Sydney University

ABSTRACT

The Share Scheduling Algorithm on Unix provides a per-user scheduler on top of the standard per-process scheduler. It acts to provide equitable allocation of the machine between classes of users, and between users of the same class, according to their allocation of "shares" of the machine.

Introduction

At Sydney University, we have implemented a version of the Cambridge Share Scheduling Algorithm as proposed by J. Larmouth of Salford University, and modified by Andrew Hume of AGSM. This scheduler has been very successful in allocating the limited resources of our student teaching machine (overcoming the otherwise unrestricted generosity of Unix) and has also been surprisingly successful as an advertisement for the sophistication of our teaching service amongst people who might otherwise consider Unix to be a toy operating system unsuited for the real world.

Implementation

The Share algorithm allocates a share of the machine to each user based on his history of past usage, his current working rate, and the ratio of his allocated shares to those of other users concurrently using the machine.

The algorithm as implemented was extensively modified by Hume (as described by Hume in his paper "A Share Scheduler for Unix"). This was mainly due to Larmouth's algorithm being batch oriented, whereas an interactive environment requires quicker responses.

The low level scheduler in Unix allocates use of the cpu amongst competing processes based on their priority which decays very quickly. Thus every process competes on an equal basis. It is possible for a user to increase his share of the cpu by running many "asynchronous" processes simultaneously. In order to allocate the cpu between users, regardless of the number of

October 22, 1980

processes, the low level scheduler was changed to use a per-user priority which is added to each process's short term priority. This per-user priority is manipulated by the Share scheduler to affect a user's long term share of the machine.

To calculate the cost of the user's use of the machine, separate charges are made for cpu time, memory occupancy, terminal I/O, backing store I/O, and system services. This cost is added to the accumulating usage, and the accumulating rate. The rate figure decays quite fast, approximately 10% second, whereas the usage decays quite slowly, approximately 10% day. There are thus two separate figures contributing to the calculations for the real share of the machine to be allocated in the next time period, one based on recent working rate, and the other on long term usage. It is therefore possible for high usage users to do a little every now and then. The real share figure is then used to affect the user's overall priority.

The Real Effect

The Share Scheduler has no effect whatever on an under utilised machine, and in practice this means less than 45 users on our VAX 11/780. However, over this, the effects are very satisfactory. To be most useful, there must be, at any one time, two or more classes of user with different priority of access to the resources. For instance, during class hours, students are deemed to have greater rights to a reasonable response than, say, academic staff. This is achieved by an (heuristic) allocation of shares to each class of user based on expected average use of the machine, and bearing in mind the decay rate of the usage. For example, at Basser we allocate 10 shares to first year students (who are expected to use 3 terminal hours per week), 50 shares to academic staff, and 200 shares to staff programmers. This effectively bars academic staff from using the machine during periods of heavy usage as their figure is always fairly high, whereas first year students (whose usage decays substantially between their terminal sessions) can obtain reasonable response even during times of peak loading. On the other hand, the 200 shares allocated to programmers allows them to carry out normal system support programming with reasonable response at all times. All this is in addition to the fair distribution of resources within each class of user, as users who use the machine heavily are not allowed to reduce the response of similarly classed people who are working lightly.

With 45 users experiencing good response, it would appear that each user needs around 2.2% of a VAX 11/780, and it would be reasonable to nominate a share figure, say 0.5%, below which the system should actively discourage users from logging in, although we haven't done this. The consequence of not doing this, is that if a user's share drops below say 0.1% during a session when many more people have logged in, he may get "marooned" with insufficient resources to allow log off (there being no cpu allocation enough to execute the "exit" system call from the

October 22, 1980

shell!)).

A user may discover his share during a terminal session, and watch it change as costs are incurred, and he is of course informed of it at login. However the changes can be quite dramatic if for instance a user is the first of 75 to log in after a system reboot. It might be advantageous to inform a user whose share has dropped below a magic figure during a session, so that he has a reasonable chance to log off and let someone else use the terminal to greater advantage.

Tools

There are various system monitoring programs to display the performance of the scheduling algorithm, and nearly all parameters can be changed dynamically. In particular all charges can be varied to reflect administrative policies about system resource costs. At Basser we charge at a fixed rate between 9am and 5pm of a working day, 50% of that until 10pm, and 10% overnight. Also users using the "nice" command get charged less for cpu time as appropriate.

Implementation

The actual scheduler is one page of C code and uses floating point extensively, although Hume has proposed a model using integer arithmetic. On the Vax, floating point is easier as the code is much cleaner, and the floating point instructions are reliable. The C compiler produces extremely inefficient floating point code, nevertheless, running once every 4 seconds, the scheduler consumes less than 0.5% of the cpu time managing 80 users on the VAX. Recoding the algorithm in assembler would reduce this to around 0.2%.

Charges are levied throughout the system at the various "cost" points. This is done by adding a variable charge for the resource type (obtained from a charges array) into a per-user "cost" variable.

The scheduler is invoked by the clock interrupt routine requesting a software interrupt at a low priority, once every 4 seconds.

The algorithm first scans all the per-user structures building up an idea of total rates and shares, and simultaneously decaying the usage and rate figures after adding in the cost. The rate and usage totals are then biased by (tunable) factors reflecting their relative weights in the priority calculation.

A second scan of the per-user structures then calculates the per-user priority to be used by the low-level scheduler. A user's share of the machine is thus reflected in his priority relative to the priorities of all other users.

October 22, 1980

SUN - The Sydney Unix Net

Piers Lauder

Basser Department of Computer Science
Sydney University

ABSTRACT

The Sydney Unix Net is a simple implementation of a user initiated file transfer facility. SUN is a "host supported" network, and considerations of low overhead have lead to an efficient design. The network is self configuring with an optimised routing algorithm.

Introduction

SUN consists of linked hosts (nodes) with unique names, where links between nodes may be unreliable and quite slow. Using this network, a simple system has been developed to provide reliable host-host file transfer. The system is implemented in two levels. Level one provides an error free path between nodes, and level two implements a host-host protocol, and maintains a network topology file for routing calculations. Files are transmitted through the network until they reach their destination where they are spooled for later collection by the remote user, who is informed by mail of the arrival of files from the network.

User Interface

The user interface is as simple as possible. A network address is defined as a username:host pair, i.e. the name of the user on the remote host is separated from the remote host name by a colon. This form of address is understood by the mail programs, some print commands, and the netsend command.

Two special file transfer types are recognised in addition to user-to-user file transfers, namely mail and print. Mail is automatically delivered on the remote host by invoking mail, and print files are handled by invoking the print spooler, but user files are spooled in a holding directory for later collection by the designated user. The arrival of files is notified by mail to the user. To complete the file transfer (and assume ownership of the file), a user must invoke the netget program to retrieve the file from the holding directory. Uncollected files can be easily

October 22, 1980

discarded from time to time.

Design

The network programs operate on two levels with clearly defined interfaces. The top level (implemented by the program `net`) accepts files for transmission, calculates the next node in the path, and spools the file together with a host/host protocol header in a directory naming the next host. The lower level (implemented by the program `netd`) accepts files for transmission to an immediately connected host, negotiates with the lower level on the remote host for file transfer to start, and then uses a node/node protocol to transfer the file reliably. On arrival at the next node, the file is passed to the upper level for any further routing.

The file transfer mechanism is half-duplex at the lower level, and store-and-forward at the higher level.

Node/Node Interface

The link between hosts provided by the operating system must be a full duplex, byte oriented special file. The name of this special file is that of the node to which it is connected at the remote end. This file is opened for reading and writing by the network daemon responsible for communicating with the next node.

Actual links between hosts may be physical RS232 type lines, either directly connected, or via telecommunications modems, and be handled by the standard Unix `tty` interface. Many hosts multiplex the physical link using the `mx` `tty` interface (mentioned elsewhere) to which the network requires only one port.

Node/Node Protocol

The network daemons communicate with each other over the node-node interface. They use a half-duplex, multi-buffered, positive acknowledgment data transfer protocol. Before file transfer starts, the daemons negotiate the direction of the next transfer. File transfer proceeds with short data blocks enclosed in a protocol envelope consisting of a header and a trailer. The header contains a sequence number used to provide the multi-buffered message flow, and the trailer implements a simple low-cost error detection capability. Messages must be acknowledged (positively or negatively) with a two byte reply consisting of ACK or NAK and the relevant sequence number. Errors cause the re-transmission of all un-acknowledged blocks. Catastrophic error conditions cause a negotiation for file transfer restart.

Daemon/Spooler Interface

The interface between spooler and daemon is defined by queued files. Each daemon maintains a command directory which it scans for command files. Each command file specifies the path names of

October 22, 1980

files for transmission. The spooler chooses the next host for transmission by the fact that the name of the host is also the name of the command directory for the appropriate daemon. On the other hand, files received by the daemon are passed directly to the spooler program.

Host/Host Protocol

The network routing program net (the "spooler" referred to above) prepends a host-host protocol header to each file before spooling it in a daemon directory for transmission to the next host. This header contains the source and destination network addresses together with routing information and file parameters. Each file arriving from the net is examined for this header and re-routed if the destination is not yet reached. Each host through which the file passes adds a host/time record to the routing information in the header. Amongst other things, this information can be used for network performance analysis.

Topology File

The routing information in each file header is used to maintain the topology file. Each host mentioned in the route is connected to the preceding and succeeding hosts, and these links are maintained in the topology file. It is possible for a topology file to become aware of new hosts in this way, however there are special files whose purpose is to maintain network topology files generally. Thus there are "host-up" messages to inform the network of a new hosts, and "host-down" messages to inform the network of broken links. These messages are broadcast around the network by the net programs who also stop any loops. New hosts coming up receive a special file from any immediately connected hosts containing a copy of their topology files, thus immediately informing a new host of the latest network state.

In order to send a file to a remote host, its name must exist in the topology file so that the routing program can find it, unless it is directly connected.

Conclusions

The initial effort producing the software for a usable network took about 2 man-weeks. Since then many requested enhancements have been implemented involving a further 4 man-weeks of work. As it now exists, SUN has proved very helpful in bringing together many people involved in cooperative work in support of our various teaching systems.

October 22, 1980

**MX - An indirect driver for multiplexing virtual
"tty" lines onto "real" lines.**

Piers Lauder

Basser Department of Computer Science
Sydney University

ABSTRACT

The **mx** driver on Unix implements virtual **tty** lines, multiplexing them onto physical **tty** lines via a simple protocol. The protocol introduces a traffic overhead of approximately 20%, and includes flow control. "Mx" is most often employed in inter-machine networks where the number of physical connections is limited, or is used by concentrators for terminals and line printers.

Introduction

The **mx** driver has spread around the Sydney Unix Network as an easy way of getting many virtual connections out of a few expensive long distance connections. The **mx** protocol is essentially a communications protocol tailored for ease of grafting onto the Unix **tty** driver. It is also used to interface different networks to the Unix net, and to connect slow terminals via concentrators onto higher speed lines. From the point of view of a user, an **mx** port behaves exactly as a standard Unix **tty** interface.

Protocol

The **mx** protocol multiplexes **mx ports** onto **real lines** and essentially just separates data for each port on a line. The protocol blocks data with a 4 byte header:-

```
start-of-header;  
port-id;  
data-byte-count;  
inverse-byte-count;  
data ...
```

This four byte header is easily recognised and checked for consistency with the repeated byte count. The data length is kept small, in the region of 40 bytes. Flow control is implemented by empty blocks with the first byte count zero, and

October 22, 1980

the second byte count representing the flow rate granted before receipt of the next flow control block. It is important to note that flow rates granted are not cumulative, and thus hung lines can be restarted via a simple timeout mechanism.

Interface

The tty interfaces provided by the mx driver appear to the user to be identical to ordinary tty lines as implemented by other Unix tty drivers. In fact the user interface is provided by the common routines implemented in the file tty.c. The device interface uses device driver routines via the cdevsw procedure array and assumes that they manipulate common Unix tty structures. A few changes have been made to the tty.c routines to identify those lines that have indirect drivers such as mx hanging off them. This is used to vector input through a pointer in the tty structure to the input handling routine in the indirect driver. A timeout scan in the mx driver is used to keep up output flow by examining the virtual and real output queues.

Output

When a write is made on an mx line the data is accumulated on the mx output queue until either the maximum block size is reached, or the user write call terminates. At this point output flow rates are checked and used to limit the next stage. Then the data is blocked up with its protocol header and transferred to the real driver's output queue, for later transmission.

Input

Input bytes via the real device are vectored through ttyinput which in turn calls mxinput. Mxinput examines the bytes for protocol headers using a simple state machine. When a flow control block is identified the "flow enabled" count is reset. When data blocks are identified, the bytes are passed via ttyinput to the appropriate mx tty input queue for retrieval by a user read call. A flow control grant block is generated whenever the user input queue is low enough. This is checked as soon as the byte count is encountered, thus providing a certain amount of "read-ahead".

Inefficiencies

For maximum efficiency, user writes should be of the order of the maximum block size. However many Unix programs generate 1 byte writes to tty interfaces, thus causing protocol overheads of 400%, and most line degradation is caused this way. Flow control is granted in small lots equivalent to the maximum size of a data block, doubling the effective overhead. The recursive calls of the tty routines at interrupt time cause a much greater system overhead in indirect drivers implemented this way.

October 22, 1980

Conclusions

As the real traffic rates on high speed lines connected to terminals are much lower than the maximum, the protocol overhead of the `mx` driver is acceptable. Furthermore, output and input data rates for high speed terminals typically differ by a factor of 10 to 1, so two ports operating in opposite directions on the same line have a low impact on each others apparent speed.

Implementation

All the `mx` driver needs to know about each real tty line is the device id (major, minor), the number of ports to be multiplexed onto the line, and the baud rate.

These details are initialised in the configuration structure "`mxdev`" - just alter it to reflect the local conditions. A tty structure is declared by "`mxtty`" for each port, so, on some small systems with kernel memory limitations, the number of ports should be kept small.

`Mxtty` is allocated as a contiguous array, and therefore minor device numbers for the special files must be contiguous. This means that if you want to change the number of ports on the first line, the special files will have to be remade for all the other lines.

The structure "`mxmap`" maps each real tty onto the "ported" tty structures, and contains current information about block transfers.

October 22, 1980

THE BFI-11

An Interface for Network Applications.

David Milway Dept. Computer Science UNSW.

As the use of local networks grows the major limiting factor is the speed of the line being used. The network at UNSW has grown with the use of standard terminal multiplexors with single character transfers, so increasing the speed of the line adds more to the service requirements of the network through greater numbers of interrupts. It is therefore very desirable to reduce the amount of system time devoted to servicing the network. This can be done by the use of DMA interfaces and some of the special network interfaces e.g. DMC-11. Using these devices, service times can be reduced but the dollar cost can be large.

The BFI-11 was designed to replace the standard connections used in the local network at UNSW with a high speed local area network such as ETHERNET¹ using a single line connected to all machines.

The features which I believe to be desirable in a network interface are:-

- Guaranteed transmission. This requires the interface to be able to detect errors in the transmission and retry the transmission (a limited number of times) until successful.
- DMA data transfers.
- High Data Rate (at least 500K bits/second)

Other features which are desirable are:-

- Programmability.
- Small physical size (one board).
- Ability to connect to a multidrop network as well as connect back to back.

The BFI-11 was designed with these features in mind for use in the local network at UNSW. The interface went through a number of different design iterations. An implementation in dedicated logic proved to be too large if many of the microprocessor type peripheral chips were not to be used. It was decided to use a microprocessor to control the interface to take advantage of LSI chips that are available.

A fast bipolar microprocessor proved to require much additional hardware for interfacing to bipolar circuits, and the programming would have been at a very low level. As a result the M6800 was chosen both for its ease of interfacing and its ease of programming. Although there was a loss in raw speed of about a factor of eight, the number of instructions required to perform simple functions was small, so the speed loss was not considered prohibitive.

The final design consisted of

1. Metcalf, Boggs. July 1975 ACM.

- An M6802 microprocessor with 8K bytes of RAM
- An M6854 Advanced Data Link Controller for the communications side using a clock programmable up to 500K bits/second
- DMA and Interrupt controllers consisting of an 18 bit address register, 16 bit DATAIN and DATAOUT buffers and Control and Status registers
- A window into the internal bus of the M6802, being able to read and write every byte in the 64K address space. This window consisted of one 16 bit address register and one 16 bit data register.

Connections to the board include data in and out, clock in and out, handshaking and error lines. These were designed to enable back to back connection for point to point communication or connection to an adaptor for use in multidrop networks.

To use the BFI-11 the control program that runs on the microprocessor must specify the control and status registers, determining what the interface will look like. The program must then carry out the specified tasks as requested by the system.

It is possible, given the interface, to emulate any serial line device or to make up a totally new specification. One interface program written was for a terminal concentrator having 16 channels each consisting of a transmitter control plus an 8 byte buffer and a receiver control with an 8 byte buffer. An experimental driver was implemented and worked successfully using port 0 linked to port 1 in a simulated connection. This type of interface could be used to remove much software from the system kernel (e.g. the MX driver from SYDNEY UNI used to multiplex many terminals on to one line).

An interface program is currently being written to transfer data between two machines using the DMA facilities, but as yet is not complete. It is to be used to connect the VAX 11/780 to the PDP 11/70 at UNSW which is currently connected by a 9600 baud terminal line.

Since the original interface was built in 1979 a new version has been constructed. The design was converted from a special purpose interface to a general purpose programmable DMA controller with the serial line component replaced with a standard microprocessor type bus enabling simple connection to more hardware to produce different controllers. A number of changes and additions were made to the original design. The microprocessor was replaced with an M6809 which has better and faster instructions and can also be programmed with a C compiler presently under construction. A Read-Modify-Write controller was added to enable interlocking between the system software and the microprocessor software. The 8K bytes of RAM was replaced with exchangeable RAM/ROM chips so that the program can be fixed when development is complete. Automatic start up on power on may be used when the program is placed in ROM enabling the BFI-11 to be treated as a non special device. The UNIBUS interface has been expanded to give up to 128 words of registers on the UNIBUS with the address register setting the base for the window into the memory of the microprocessor. When in auto start mode the address register is not accessible from the UNIBUS and the window is fixed by the micro program.

With a small amount of extra hardware the new board can be used for serial line network applications, but the interface can now be used in many more applications. At present a 4th year student² has built a 16 port

terminal multiplexor to replace the DZ-11 or the DH-11 interfaces, fitting all the additional logic on to a single Quad board. This interface is capable of removing the TTY driver from the system kernel (that should solve many arguments).

Other projects under examination are

- ☒ Interface of an Ampex Tape transport to a PDP11
- ☒ Interface of a number of Ampex 30Mbyte disc drives to PDP11s

It is hoped that a pc board will be finished this Christmas which will enable further development of the network hardware during next year.

Pages 24-33 where to have
contained a reprint from

"Software Practice & Experience"

title

"UNIX with Satellite Processes"

by A B Barak & A Shapira

VOL 10 383 392 (1980)

Permission from the publisher was not
obtained in time for this issue.

Maybe next time?

Her I. 22/11/80

REFERENCES

1. S. R. Bourne, 'The UNIX shell', *The Bell System Tech. Journal*, 57, 6, part 2, 1971-1990 (1978).
2. *DEC PDP-11 Processor Handbook* (1975).
3. P. Brinch Hansen, 'Structured multiprogramming', *Comm. ACM*, 15, 7, 574-578 (1972).
4. B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1978.
5. H. Lycklama and C. Christensen, 'A minicomputer satellite processor system', *The Bell System Tech. Journal*, 57, 6, part 2, 2103-2113 (1978).
6. D. M. Ritchie and K. Thompson, 'The UNIX time-sharing system', *Comm. ACM*, 17, 7, 365-375 (1974). Also *The Bell System Tech. Journal*, 57, 6, part 2, 1905-1929 (1978).
7. K. Thompson and D. M. Ritchie, *UNIX Programmer's Manual*, 6th edn., Bell Telephone Lab., Murray Hill, N.J., 1975.

SOFTWARE TOOLS MEETING
 John M. Clayton Hall
 University of Delaware
 Newark, Delaware
 Monday, June 16, 1980

Agenda

- 9:00 AM - 10:30 AM Tools and Primitives Session
 Robert Munn, University of Maryland
 Ratfor/ratmac and Fortran 77
 Skip Egdorf, Los Alamos Scientific Laboratory
 Defining the Software Tools Primitives
 Walter Brown, Moravian College
 Evaluating the Ratfor Preprocessor
 Dave Murray, Analytic Disciplines
 Ratfor-T, for Reduction of Navy Software
 Development Costs
- 10:30 AM - 11:00 AM Coffee Break
- 11:00 AM - 12:30 AM Implementation Issues
 Mike O'Dell, Lawrence Berkeley Laboratory
 Strategies for Implementing the Software
 Tools on IBM Systems
 Wally Wedel, University of Texas
 Implementing the Software Tools on
 CDC Systems
 Neil Groundwater, Analytic Disciplines
 Bringing up the Software Tools when they
 are Several Generations Removed from the
 Parent Operating System
 Greg O'Brien, Digital Equipment Corporation
 DEC and the Software Tools
- 12:30 AM - 2:30 PM Lunch Break
- 2:30 PM - 3:00 PM Discussions
 Haves and Wants
 Standardizing (or not) the ratfor preprocessor
 Defining the primitives
 The text editing and formatting tools
- 3:00 PM - 4:00 PM Special Interest Group Meetings
 Ratfor preprocessor
 Networks
 Primitives
 Text Formatting and Editing
- 4:00 PM - 4:30 PM Future Directions
 Reports from the special interest groups
 Discussion of organization of the group and possible
 merger with USENIX organization

National USENIX Meeting

Tuesday, June 17 through Friday, June 18

John M. Clayton Hall
University of Delaware
Newark, Delaware 19711

Schedule of Sessions

- 9:00am to 10:30am -- Morning Session, Part 1, Room 128
10:30am to 10:45am -- Coffee and Donuts, Lobby
- 10:45am to 12:15am -- Morning Session, Part 2, Room 128

12:15pm to 1:30pm -- Lunch, Pencader Dining Hall

1:30pm to 3:00pm -- Afternoon Session, Part 1, Room 128
3:00pm to 3:15pm -- Coffee and Soda break, Lobby
3:15pm to 4:30pm -- Afternoon Session, Part 2, Room 128

4:30pm to 7:00pm -- Dinner Break

7:00pm to 9:00pm -- Evening Session, if scheduled

Birds of a Feather sessions will be scheduled on
Tuesday evening and Wednesday afternoon.

Scheduled Birds of a Feather Sessions

Tuesday Evening, June 17

Non-DEC Equipment on DEC Systems
Walt Brown, Moravian College
8:00pm, Room 128

Wednesday Evening, June 18

Text Processing
Dennis Mumaugh, Dept. of Defense
8:00pm, Room 128

Thursday Evening, June 19

VAX Issues
Ed Szurkowski, Univ. of Delaware
7:00pm, Room 128

Interactive Systems Customers Meeting
John Donnelly, NCAR
8:00pm, Room 125

Graphics (tentative)
Mike Wahrman, Rand (?)
Room and time will be announced.

Tuesday Morning, June 17

Introduction and What's Happening with Unix
Chair: Dan Grim, Univ. of Delaware, Computing Center

Pete Warter, Univ. of Delaware, Chairman, Dept. of Elec. Eng.

"Welcome to Delaware"

[5 minutes]

Bill Munson, Digital Equipment Corp.

"More Gnus From DEC"

[45 minutes]

Wally Wedel, Univ. of Texas

"DECUS Unix SIG progress report"

[5 minutes]

Peter Ward, U.S. Geological Survey

"Unix Extensions at the U.S. Geological Survey"

[20 minutes]

Gary Fostel, Intermetrics, Inc.

"ISO Pascal Announcement"

[5 minutes]

Clement Cole, Tektronix, Inc.

"What Tektronix is Doing With Unix"

Melvin Lax, City College of New York and Bell Labs

"Tmac.eq, An Equation Macro Package for NROFF/TROFF" (In
this session due to the speaker's schedule)

[15 minutes]

Mike Muuss, U.S. Army, BRL

"BRLNET, Multi-CPU Unix Development: Progress Report"

[15 minutes]

Bill Plauger, Whitesmiths, Ltd. plauger, whitesmiths, ltd. plauger, whitesmiths, ltd.

"Another Flavor of Unix" "Another Flavor of Unix" "Another Flavor of Unix"

Tuesday Afternoon, June 17

Kernel Extensions, Performance, and Guru Panel Discussions

Chair: Ian Johnstone, Bell Labs

David Tilbrook, Bell Northern Software Research

"Extensions to PWB Unix to Support MASCOT kernel"

Jeff Schriebman, Jeff Schriebman Consulting

"Cross Linking Terminals for Instruction"

[10 minutes]

Ed Gould, Univ. of California, Berkeley

"Berkeley Version 7 Performance Improvements"

[10 minutes]

Ron Reisor, Univ. of Delaware
"Delaware Version 7 Improvements"

Guru Panel

Ed Gould, Univ. of California, Berkeley
Neil Groundwater, Analytic Disciplines
Ian Johnstone, Bell Labs
Mike Muuss, U.S. Army, BRL
Ron Reisor, Univ. of Delaware

Tuesday Evening, June 17

Birds of a Feather sessions to be scheduled.

Wednesday Morning, June 18
9:00am, Room 128

Languages and Porting C and Unix
Chair: Joe Yao, Science Applications, Inc.

Al Arms, Western Electric

"News from Western Electric"
[30 minutes]

George Ronkin, Bell Labs, Piscataway

"Unix on the Univac 1100 Computer"
[20 minutes]

Paul Cabbage, The Wollongong Group

"Porting Unix to Interdata/Perkin-Elmer"
[20 minutes]

Craig Forney, Onyx Systems, Inc.

"Unix version 7 on the Z8000"
[15 minutes]

Ed Fourt, Lawrence Berkeley Lab

"Command-Line Argument Cracking"
[5 minutes]

George Rosenberg, Univ. of Pittsburgh

"On Processing Abstract C Programs"
[15 minutes]

Alan Nemeth, Bolt, Beranek, and Newman, Inc.

"The C Machine"
[30 minutes]

Jean Butler, Boeing Aerospace

"Retargetting the Portable C Compiler and F77 to the Z8000"
[10 minutes]

Randall Howard, Mark Williams Co.

"Portable Implementation of C, Pascal and Fortran 77"
[15 minutes]

Bob Kirby, Univ. of Maryland

"ULISP, A Dialect of Lisp"
[10 minutes]

Wednesday Afternoon, June 19
1:30pm, Room 128

Vendor Presentations

Chair: Ed Szurkowski, Univ. of Delaware

Mike Tilson, Human Computing Resources
"HCR Products and Services"
[15 minutes]

Dennis Ferland, Onyx Systems, Inc.
"The Computer Manufacturer of Tomorrow/Life in a Frisbee
Factory"
[30 minutes]

Robert Schwartz, Mark Williams Co.
"The Coherent Operating System"
[15 minutes]

Scott Leisy, System Industries, Inc.
[15 minutes]

Gary Donnelly, Interactive Systems Inc.
[15 minutes]

Wednesday Evening, June 18

Outdoor Barbeque

Refreshments starting at 6:00pm
Food served starting 6:30pm - 7:00pm

Reversi Tournament and E.E. Site Visit
8:00pm, Rooms 358 and 360, P.S. duPont Hall

m/c Tex.

354

Thursday Morning, June 19
9:00am, Room 128

Data Bases
Chair: Ed Adami, NBS

Neil Groundwater, Analytic Disciplines, Inc.
"The Navy's Use of INGRES to Monitor Hardware Reliability
and Maintainability"
[15 minutes]

John Kornatowski, Univ. of Toronto
"MRS, Micro Relational System"
[15 minutes]

Mike Lesk, Bell Labs
"Publications, Indexes, Bibliographies"

Ron Bauer, International Data Base Systems, Inc.
"SEED"

Gary Fostel, Intermetrics, Inc.
"Implementation Independent Data Structures"
[15 minutes]

Rhea Gur, Johns Hopkins Medical School
"Data Entry and Unix"
[20 minutes]

Richard A. Witt, Field Museum of Natural History
"Field Museum's Use of Unix for Data Entry of Artifact
Collection and Recording"
[10 minutes]

Bill Taylor, Logicon, Inc.
"LIS/11, Information Storage and Retrieval System"
[30 minutes]

Thursday Afternoon, June 19
1:30pm, Room 128

USENIX Business and Overflow
Chair: Lou Katz, Columbia Univ.

Lou Katz, Columbia Univ.
"USENIX business and announcements"
[30 minutes]

Mike Muuss, U.S. Army, BRL
"3D Graphics Editor - GED", with movie
[40 minutes]

Arthur Wetzel, Univ. of Pittsburgh
"Using VAX Compatability Mode Under UNIX-32V"
[5 minutes]

David Farber, Univ. of Delaware
"MMDF: An Experiment in Memo Distribution"
[15 minutes]

Brad Cox, Hendrix Electronics
"What's Happening With Unix at Hendrix"
[10 minutes]

Mike Tilson, Human Computing Resources, Inc.
"MAP Array Processor Support"
[20 minutes]

Robert Morris, Univ. of Massachusetts, Boston
"Should Unix Programmers Know Assembly Language?"
[20 minutes]

Bruce Borden, Rand Corp.
"MH, A Unix Approach to Software Design"
[15 minutes]

Kurt Akeley, Univ. of Delaware
"A Xerographic Printer"
[15 minutes]

Thursday Evening

Birds of a Feather Sessions

VAX Issues

Ed Szurkowski, Univ. of Delaware
7:00pm, Room 128

Interactive Systems Customers Meeting

John Donnelly, NCAR
8:00pm, Room 125

Graphics

Mike Wahrman, Rand Corp.
8:00pm, Room 121

Friday Moring, June 20
 8:30am - 12:00am, Room 128

Text Processing and Office Automation
 Chair: Dennis Mumaugh, Dept. of Defense

Bill Malldree, RLG Assoc.
 "OPUS, Office Automation Package"

Stuart Kern, Physical Review
 "Experiences and Future Plans With Typesetting 'Physical Review' Journal"

Dave Buscher, Johns Hopkins, APL
 "CAT Phototypesetter Fundamentals"
 [15 minutes]

Mary Ann Jackson, M.A. Jackson and Assoc.
 "Text Processing Training"
 [30 minutes]

Ben Domenico, NCAR
 "A Complete Document Production and Retrieval System"
 [20 minutes]

David Yost, The Rand Corporation
 "E, The New Rand Editor, and LA, A Line Access File I/O Package"
 [15 minutes]

George Toth, Johns Hopkins Univ.
 "Graphics Inserted Into Text Processing, Terminal Independent Graphics Package, and Terminal Independent Text Processing"
 [15 minutes]

Dennis Mumaugh, Dept. of Defense
 "Classification by Paragraph"
 [5 minutes]

Walt Lazear, Air Force Data Services Center, Pentagon
 "Text Processing Tools at the AFDSC"
 [30 minutes]

John Kornatowski, Univ. of Toronto
 "OFS, Office Forms System"
 [15 minutes]

Friday Afternoon, June 20
1:00pm - 3:00pm, Room 128

Communications and Networking

Chair: Doug Shannon, U.S. Naval Research Lab

Steven Holmgren, MITRE/METREK Corp.

"Unix/Arpanet History"
[30 minutes]

Doug Shannon, U.S. Naval Research Lab

"A Simple Solution to the Arpanet Long Leader Situation"
[10 minutes]

Steven Bellovin, Triangle Universities Computation Center

"Introduction and Overview of UUCP"
[20 minutes]

Stepher Daniel, Duke University, Computer Science Dept.

"USENET, An Informal Network of Unix Systems"
[20 minutes]

Steve Holmgren, MITRE/METREK Corp.

"Unix - 28000 Local Networking"
[20 minutes]

Greg Chesson, Bell Labs

"DATAKIT Network At Bell Labs"
[30 minutes]

Peter Miller, Purdue, and Bill Croft, SRI International

"Load Balancing on the Purdue Local Network"
[10 minutes]

Delaware USENIX Meeting

In June, I attended the USENIX meeting at the University of Delaware. The following is a brief report of that meeting. Unfortunately, my notes are very sketchy (I failed to take detailed notes, as proceedings of the meeting were to have been available at its conclusion, they should be available soon.) My memory is worse, so there is much that happened at the meeting that won't be mentioned, and some of what is here might not have happened at all. I apologise for this in advance, but I have the impression that if I delay sending this till I receive a copy of the proceedings, the editor of this 'journal' might never talk to me again. To be certain I have provided at least some useful information, I have included a copy of the agenda, though it wasn't always followed strictly.

The meeting was held over 4 days, preceded by a one day Software Tools meeting. I have little recollection of the Tools meeting, but do recall that it included several entertaining talks describing Tools implementations on various machines. There was also a discussion of whether the group should merge with USENIX - it was decided not to.

For Software Tools tapes (a temporary LBL version) you should send a 2400' tape and return postage to:

Vicki Vax,
Software Tools,
DFSEC,
US Air Force Academy,
Colorado. 80840

DFSEC,
Colorado 80840

You should specify the format desired:

Portable (specify blocking, density, ...)
RSX-11M
HP1000
HP3000
VAX/VMS

The USENIX meeting began with a session on the general theme of "What's happening with Unix". This included a talk by Bill Plauger (Whitesmiths Ltd.) who described the current state of IDRIS, and upset one member of the audience who thought that the ban on commercial advertising had been breached. It did not seem that way to me.

The second session concentrated on UNIX* kernel extensions. Jeff Schriebman (Jeff Schriebman Consulting)

*UNIX is a Trademark of Bell Laboratories.

described a method of connecting multiple terminals for student instruction, allowing students to observe the instructor's session on their own terminals.

Ed Gould (UCB) described performance enhancements to PDP-11 V7, including the alteration of the buffer size from 512 to 1024 bytes, hashed lookup of buffers and inodes, pseudo dma for terminal output, and more, supporting a 1.5MB 11/70 with 2 300 MB and 2 RS04 drives, and 80 tty lines.

Ron Reisor (Univ Delaware) talked on the same subject. He has an inverted disc structure (so the inodes of two logical filesystems are adjacent in the middle of the disc), an altered scheduler that runs every 1/10 of a second (it 'feels better'), and has altered

```
(p = proc; p < &proc[NPROC]; p++)
```

to use a linked list.

The session was concluded with a panel of Gurus, chaired by Ian Johnstone (Bell). Audience questions on a variety of subjects were answered, largely on PDP hardware peculiarities.

The Wednesday morning session started with Al Arms giving his last presentation as Western Electric's representative. Al has moved on to other things, and responsibility for UNIX licences is now in the hands of Mr. Otis Wilson.

Al's talk revealed the following:

The setuid bit patent held by Dennis Ritchie has been dedicated, (ie: the \$400 per CPU licence is no longer required).

All C compilers are now interchangeable, if you are licensed for one, you may obtain any of the others (assuming you can find someone to give it to you).

The equipment test package will probably be included in the next UNIX release.

Small business systems are currently being licensed.

There is a new educational licence form, just 2 pages, for quicker response. Additional CPU requests are also easier now.

Manual sets are available at \$120 per set from Irma Biren.

If you want UNIX (4.0) to appear, write to Al Arms [It would be to Otis Wilson now].

360 UNIX, in two pieces (one owned by Bell, the other by IBM), might be released.

The possibility of a release of the Hardware Design Aids is being reviewed, but it is unlikely to be approved.

C 370 is being licensed, a later version will probably be released soon.

C 6000 (Honywell) is being released.

George Ronkin (Bell) spoke on Univac 1100 UNIX. It sits on top of EXEC8 (which copes with regular users as well). EXEC8 does all swapping and scheduling. The system runs on multiple CPU's, mutual exclusion has been added (in "hundreds of places"). The system maintains a more stable filesystem than is usual, requiring consistency on the disc (though some data could be lost in a crash, no other harm should be done). The system took 4.5 man years to do, there are no release plans.

Paul Cabbage (The Wollongong Group) gave a boring and somewhat misleading talk on Richard Miller's work in moving UNIX to Interdata (Perkin-Elmer) equipment.

Craig Forney (Onyx Systems Inc.) spoke on Z8000 UNIX. The system uses special memory management (not the Zilog chip), and although the CPU is faster than an 11/45, the system runs slower due to slow Winchester discs. Z8000 C code is 120% of the size it would be on a PDP-11, it is to be reduced to 95%. The system supports a maximum of 8 users, and no source is available. The conversion took 1.5 man years.

The afternoon session contained the vendor presentations (that is fancy wording for advertisements). I recall nothing notable.

The Thursday morning session concentrated on data bases and similar systems. John Kornatowski (Univ Toronto) spoke on MRS ("Micro Relational System") a relational data base system claimed to use 1/4 the space and run 4 times as fast as INGRES. However, floating point is not supported, and the result of a query is not in a form suitable for use in another query (ie: is not a relation). To get a copy of this, write to

Chris Robertson,
MRS Distribution Manager,
C.S.R.G.,
University of Toronto,
Toronto,
Ontario,
Canada.
M5S 1A1

and be prepared for lots of legal paper, and to send \$200. MRS was used to handle the conference registrations for this conference.

Mel Ferentz said that he has MRS for version 7, which was an easy conversion. He also has V7 INGRES, which involved man months of effort, and is not ready for release yet.

Rhea Gur (Johns Hopkins Medical School) described a compiler for data entry forms, which was not ready for distribution.

The sixth session (thursday afternoon) included general business, and oddments. USENIX business included a discussion of the tape distributions (which were reported to be about to commence). The main problem now seems to be ensuring that material is not sent to organizations not licensed to receive it. This was to be solved by requiring distributors to specifically state what licences are needed to obtain material on their tapes. The distribution format also faced a licence problem, most of the 'standard' formats are not available to all recipients. Western Electric have agreed that the the binary of 'bcpio' (for PDP-11's) can be distributed to anyone [I think, it may be anyone with some UNIX licence ... kre] so this will be the format used. Bcpio is supposed to be closely related to cpio, but different.

Arthur Wetzel (Univ Pittsburg) described the use of the VAX 11/780 Compatibility mode under UNIX 32V and VMUNIX. Bugs in sendsig()/machdep.c were required to be fixed, and if setuid PDP-11 programs are to be run, new system calls seteuid() and setegid() need to be added. The system does not support separated I/D space (this cannot be done), shared text (or text write protect), or PDP-11 floating point. There are emulators for V6, V7, and RT11; INGRES runs under the V6 emulator. There are also shell mods, so a compatibility mode program can be run without pain. The kernel mods, V6 and V7 emulators and shell mods are on the Delaware distribution tape.

Robert Morris (Univ Massachusetts, Boston) gave a controversial talk in which he proposed that all UNIX programmers should be taught assembly code, so that they can

better exploit the 'features' of C and write more efficient programs. It was not generally well received.

The last day began with a session on text processing and office automation. Michael Wharman (Rand Cpn.) described the latest version of the Rand Editor (to be called 'E'), which is to use a Line Access library. This will permit the editor to do without a temp file, by keeping a file of changes instead. This is not yet ready for release yet, it should be in a 'couple of months'.

Lorinda Cherry (Bell) spoke on "Writer's Work Bench". This is a set of programs to analyse English text. The first (style) produces 4 readability indices, plus information as to sentence length, word usage, sentence openings and much more. It also has options to allow the user to find where the 'bad bits' are. The second (diction) is basically just an extension to fgrep plus a large dictionary of ugly phrases. It is used to identify possible poor language usage. The third (explain, which might also be called suggest) gives alternative usages to those that diction complains of. This work builds upon that described in the paper "Statistical Text Processing" in the UNIX issue of the Bell Systems Technical Journal, (Jul - Aug, 1978). This has been released and is available to UNIX licensees from Irma Biren.

Denris Mumaugh (US Dept of Defence) described a set of macros and a filter to enable US government security requirements to be carried out. He was willing to distribute to this to anyone who needed it.

John Kornatowski (Univ Toronto) described OFS (Office Forms System) which uses UNIX to design forms that can then be used on an LSI-11, MINI-UNIX... This interfaces to MRS (ie: you must have MRS), and like that system is only available from Toronto for V6 systems (though it requires no kernel modifications). To get a copy, write to the same person as for MRS, but call him

OFS Distribution Manager

There is a \$200 charge (independent of that for MRS), and all the same paperwork.

The final session was entitled "Communications and Networking". Stephen Daniel (Duke University) talked on USENET, a network of UUCP sites. Duke are willing to call sites without ACU's if those sites pay the phone bills. Two V7 bugs were also mentioned. The first is that /usr/dict/words is not sorted correctly for look. The other concerns the saving of the floating point registers. It is apparently possible for these registers to fail to be saved until after the process is (at least partially) swapped out.

The fix reported was to explicitly save the registers in `newproc()` and `expand()`, if they are not already saved.

Greg Chesson (Bell) described his work with the DATAKIT network. This is a high speed network, implemented largely by hardware. It was described as running at 7.5 Mbits per second, and was to be increased to 100 Mbits/sec for local networks, and 24Mbits/sec across trunk lines. Data is transferred as 9 bit bytes, with the top bit used to distinguish between control and data bytes. The protocol is intended to be efficient for single character and large block transfers, and to allow `ioctl`'s to be passed through. A global file system (across the network) was also mentioned. File name interpretation is to be done in a dedicated processor.

Peter Miller (Purdue) and Bill Croft (SRI International) described the Purdue network, which permits work to be moved from one machine to another to balance load.

During the meeting, there were several 'Birds of a feather' sessions. Those I attended didn't seem to accomplish much, though I was somewhat surprised to learn that the vast majority of users of non DEC equipment on PDP-11's were very happy. This contrasts with the Australian situation, where horror stories are normal. Perhaps it is really the local distributors who cause almost all of the problems. The VAX meeting produced a rumour of a VAX 11/980, which may be released in 2 to 3 years, and run 3.5 times as fast as the 780. The LSI VAX and the 580 (or is it 750) Unibus only VAX are both almost ready.

Robert Elz
Melbourne University
Computer Science

Henry's Gossip Hotline

or

Highlights of the Delaware USENIX Conference

Henry Spencer
23 June 1980

- [Fact] It actually is possible to define a usable operating-system interface that is a subset of almost every existing operating system, and is rich enough to write most programs. Whitesmiths has done it. Details known, ask me.
- [Solid] The Pascal Validation Suite has bugs.
- [Quote] "VAX/VMS is not a bad operating system, just the parts the user sees..." - Greg O'Brien of DEC.
- [Solid] Ampex 300-megabyte disk drives have a good reputation except that you must be *religious* about filter changing - reliability drops surprisingly sharply as you run past the recommended times.
- [Fact] \$18 000 will get you Onyx's Z8000 system with an IMI 10-meg Winchester disk, a DEI 12-meg tape cartridge drive, and a binary-only V7 UNIX licence.
- [Fact] The rumors that 8in. Winchester disks are knocked out of alignment by even the slightest shock or vibration, and hence cannot be moved around without trouble, are false - at least for the IMI drives that Onyx Inc. uses. They routinely take them on car trips and plane trips with no special precautions apart from locking the heads. IMI disks use voice-coil closed-loop positioners - this may be significant.
- [Solid] Deliveries of the new 11/44 have been delayed because a bad power-supply problem was only found and fixed quite recently.
- [Fact] Contrary to previous rumors, UNIX comes up quite easily on the 11/44, with a total of about 25 lines of code changes.
- [Fact] USGS has done overlaying for UNIX, to run programs much bigger than 64K. Tested and working. One result: V7 Fortran 77 on a non-split-space machine.
- [Fact] A second version of the VU ("Netherlands") Pascal has been released even better than the first. Separate compilation, linking to C. \$50.

- [Fact] Whitesmiths' *Idris* UNIX-lookalike is still not out, because providing the various system utilities turned out to be more important than they thought. Something like 70 utility programs are needed for a good, usable system that will keep users happy.
- [Solid] How to prevent Western Electric trouble when writing a UNIX lookalike: tell Al Arms he can send a man around any time he likes to look at your code, provided he then says, in writing, either "you're ripping us off" or "you aren't ripping us off" - Bill Flainger of Whitesmiths.
- [Fact] Many performance improvements to V7 are possible.
- [Solid] Updates and enhancements to UNIX will continue to be released, but release of really new software (e.g. a Version 8) is unlikely.
- [Fact] The multiplexed-files code distributed on the V7 tape is known to crash the system. The only reason it was released was so that a later, working, version could be classed as an update rather than as new software (see previous item).
- [Fact] Early DH11's may have a hardware flaw that makes them misbehave when the V7 DR driver tries to juggle the sile level.
- [Fact] PDP11 bus-error failures may be due to marginally flakey memory or memory management. DEC diagnostics are not good at spotting this.
- [Fact] DEC DH11's do not work (at all) on the VAX Unibus because they try to defer to other DMA devices. ACT's DH11 lookalike does not do this, and hence works fine on the VAX.
- [Fact] Western Electric has stopped bothering with the royalty on the solid LIT.
- [Fact] Bell has a 370 UNIX running internally. Probability of release uncertain, because IBM has a finger in it somehow. There is also a UNIX on the Univac 1100 series; no release plans yet.
- [Fact] The "parasite system" - something like the older SPS - is in the works for release.
- [Fact] C/6000 (Honeywell machines) is released; release of the new (and improved) C/370 is being considered.
- [Fact] Western Electric has decided that, for licensing purposes, her various UNIX C compilers are considered interchangeable. So everybody (see point 7 licenses) can get the newest C compiler. A letter making this official is being sent to all licensees.
- [Fact] It is possible (not easy) to rease the kernel code dealing with file systems to greatly reduce filesystem damage done by crashes. Some details known, ask me.

23 June 1980

Henry's Gossip Hotline

- [Fact] BBN's C machine is much closer to reality but not ready for release yet.
- [Rumor] You can often make collect calls to your computer by telling the operator that you're calling a machine and that if it answers with a tone, that means it accepts the charges.
- [Fact] UNIX V7 filesystems are incompatible between the 11 and the VAX, because the 16-bit words of a 32-bit integer are not in the same order.
- [Fact] RAND has a new UNIX mail system, vastly superior to existing ones. It does *not* invent a new command language, is *not* a single monolithic program, and does *not* use its own strange file format. It's a collection of small, easily-modified single-function programs, using the UNIX file system as its database. People who can get it are unanimous in its praise; general public release is coming shortly.
- [Fact] The EE Dept. at U. of Delaware *built* a 384-points-per-inch xerographic pseudo-typesetter out of a modified Teletype Model 200 telecopier. Costs, exclusive of the Teletype gadget itself, circa \$2000.
- [Fact] DMC11's at 1 megabaud take a lot of babying on the VAX.
- [Solid] The VAX-on-a-board has been seen; release date unknown.
- [Fact] *Physical Review's* experiment with UNIX has turned out very nicely, and all the APS journals are now switching to UNIX typesetting. They intend to offer a substantial discount on page charges for input submitted as machine-readable *troff*-compatible text.
- [Fact] Xerox has a PDP11 Ethernet interface - \$5000 *if* you can get it.

19.September.1980

Peter:

Greetings once again from north of the equator. Reading the current issue of the AUUGN, I am faced again with a quandary that the AUUGN often presents. Namely, when seeing solicitations for suggestions/surveys, I never am quite sure of whether they are intended for local (i.e. Australian) users or for all readers. That is why, for example, I didn't send in our installation report. (What good would it do somebody in Woolongong to know that we have a blah, blah, blah...?) Nevertheless, this time I have a few suggestions to make regarding the "software cattle-dog", and if it is truly only for locals, then you may feed my suggestions to the dogs.

To wit: trying to decide what I would want from a software catalogue, two ideas were most prevalent. I would want the catalogue to

1) help me with kernel problems

2) save me time by preventing re-inventions of the wheel

Under the first point, I include such things as drivers, bug fixes, performance enhancements, tips, contacts for various items, etc. The second point probably encompasses the first as well, but I intend it to be more non-kernel oriented. (Don't ask why. That's just how it occurs to me.) What I mean is best illustrated by an example. Say that I need a utility to keep track of tape allocations on my site. Rather than immediately sit down and design the program, run off and implement it, test/debug it, and then document the whole works, I would like to be able first to see if somebody else may not have already faced the same problem. If so, their experience may save me much time and effort; in fact, their exact program may be applicable to my situation. In other words, I needn't re-invent the wheel.

Now, certainly none of this is new to you. However, I believe that this is the essence of what anyone would like from such a catalogue. (Anyone who thinks like myself, I suppose.) Therefore, though the "list of possibilities is endless", as you point out, I think the functionality is rather bounded. In fact, if you categorize, delineate, and comment the thing to death, you may just defeat your own purpose. I would suggest that rather than approaching the "endless" in your thinking, restrict the organization to a sort of bibliographical

document subdivided into categories. These may include all the ones you suggest, though I wouldn't separate utilities under large and small. I would segregate them according to broad categorical function (e.g. accounting, performance aids, development tools, text manipulation, system maintenance, languages, etc.). Each entry could perhaps be patterned after the following template:

Category, brief description, source, contact, latest update, catalogue location; commentary

Thus, a typical entry might be:

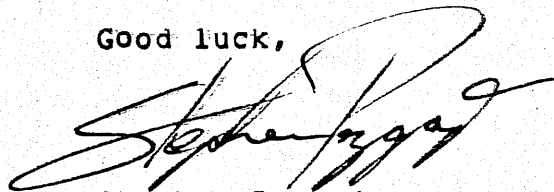
DRIVER, PCL11-A/B communications link driver, Northern Telecom, Steve Pozgaj, Dec79, /NT/sys/dmr;

This driver is a protocol-independent coupler to the PCL11-A/B, a networking device capable of linking up to 31 processors in a local network. It has two interface routines to a user-defined switching process which defines the network protocol. Also allows user-designed input spooler and networking system calls. Works on standard v6 UNIX.

These entries would only need be permuted on the description field, as they would already be categorically grouped. Finally, you would include a list of addresses for contacts.

~~Well, that's it. I hope that you get enough submissions to help~~
point you in the right direction eventually.

Good luck,



Stephen Pozgaj
Software Engineer
Department 1751
Northern Telecom Canada Ltd.
P.O. Box 3000
Brampton, Ontario, Canada
L6V 2M6
(416) 451-9150, X-5996

DIVISION OF MATHEMATICS AND STATISTICS

P.O. BOX 218, LINDFIELD, NSW, AUSTRALIA 2070. TELEPHONE (02) 467 6211. TELEX AA26296.
RIB:JS

25th September, 1980.

Peter Ivanov,
Dept. of Computer Science,
University of NSW,
P.O. Box 1,
KENSINGTON, NSW 2033

Dear Peter,

re: Cattle-dogs

One of our major interests in the software catalogues will be to discover application software in the area of statistics and mathematics which may be of less direct interest to other Australian Unix Users (i.e. software we would not hear about if the catalogue didn't exist).

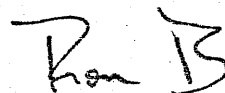
Much of this software would come under the general heading - compilers and large utilities although some may not be "large". Consequently, sub-classifications within this group indicating general area of application will probably be needed. I wonder also whether the distinction between "small" and "large" utilities will be useful or is it better to just depend on the subject classification.

Other information that may assist in determining whether to bother with some offering could include:

- (i) the machine and level of UNIX it was developed on (i.e. did they have separate I & D space etc.),
- (ii) the machines and levels of UNIX it is known to have been moved to (i.e. is it fairly portable),
- (iii) documentation for implementors - do the READ_ME, Makefiles, run files etc. look to be comprehensive and comprehensible,
- (iv) documentation for users - do manual entries or complete separate manuals exist. Is it necessary to get published manuals.
- (v) demonstration programs - do they exist and are they light-weight or realistic problems.

I hope these items may be useful - the main aim is to think of things that can be seen from a quick browse through the distributed directories, that may give hints about the value of the software.

Best wishes to Ford P et al.



R.I. Baxter

P.S. Specific request: any newer whippy 'ratfor' would be appreciated.



Monash University

CLAYTON VICTORIA AUSTRALIA 3168

TELEPHONE: 03 541 0811 TELEGRAMS: Monashuni Melbourne
TELEX: MONASH 32691

DEPT. OF COMPUTER SCIENCE

KMcD:law

Software Catalogue,
c/- Peter Ivanov,
Dept. of Computer Science,
University of N.S.W.,
KENSINGTON, N.S.W.

3rd September, 1980.

Dear Peter et al,

re: training cattle-dog trainers in the ancient art of
training cattle-dogs to do what you want them to do

As someone who supposedly knows something about database management - at least that's what I tell my students - I would be more than willing to co-operate in the initial db design. However this is likely to pose some communication problems given our geographic separation, so I'll offer the following suggestions, with perhaps the option of reviewing the ideas/designs/progress at the forthcoming users meeting in Melbourne.

1. Software to build/maintain the database

PLEASE let's avoid the UNIX devotees reflex reaction to head straight for a terminal to hack code on the assumption that "in C its simple and should take no time" - it certainly is neither simple nor quickly developed. Rather let me suggest that the following be investigated as possibilities before any serious or detailed design is attempted:-

- (a) the relational dbms MRS from CSRG at Toronto (we're likely to have a copy at Monash before too long, I've seen the documentation and I think it would provide more than adequate flexibility and facilities and at \$200 beats hacking code).
- (b) ORACLE - a commercial relational dbms which is starting to be marketed in Australia. I have no firm guarantees, but I believe that we could negotiate to obtain a free copy (it reputedly runs on UNIX - it's written in C) specifically for non-profit use in such a PR-cum-cattle-dog-taming exercise.
- (c) Andrew Hume has mumbled about various db software packages (via some hospital or other?) at various times in the past.
- (d) I have a suite of fortran (ugh, but portable) programs for maintaining bibliographic databases which could be used with only minor modifications.

2. The Cattle-dog

I think the basic information should be stored only once per item, but all access would probably be via a set of indices. Division of the catalogue proper into disjoint areas leads to problems with initial classification and subsequent cross-referencing between the areas.

The suggested areas (system kernel, system extras, small utilities, etc.) probably form the basis for a hierarchic classification of descriptors or index terms which could be used to search the catalogue proper. This arrangement would permit an item to be classified in more than one area without explicit cross-referencing.

3. What should a cattle-dog entry look like?

The following format is proposed as an initial rough-cut.

- (1) item number ; unique valued
- (2) description (brief)
- (3) original source
- (4) licence fee and/or requirements
- (5) minimal hardware configuration
- (6) minimal software configuration (including s/w upon which current item is dependent)
- (7) related software - serving a similar function or purpose (item numbers)
- (8) Australian sites using this item
- (9) abstract - textual stuff, e.g. longer description, bugs, evaluation.
- (10) descriptors (e.g. source language, general categories, specific categories).

Data items 2 - 6, 8 and 9 are free format, inherently variable length and would not be used to search the catalogue. Data item 7 would contain a variable number of fixed format item numbers and would not be used to search the catalogue.

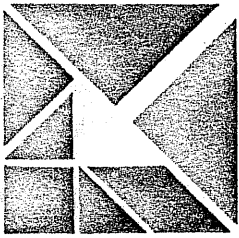
Data items 1 and 10 would constitute the fields over which searches could be conducted.

I trust these comments will prove to be of some use, and look forward to further discussions with the principal "trainers".

Regards,

Ken

KEN J. McDONELL.



the Time-Machine Ltd.

ת"מ.ל. פ'מ"מ-א"ו

Haifa, 14-9-1980

The Software Catalogue
c/o Peter Ivanov
Dept. of Computer Science
University of NSW
Kensington
Australia

Dear Mr. Ivanov,

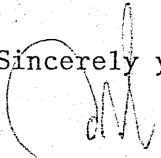
I am glad to see the software catalogue coming into existence. The items which are most relevant to us (at the Time-Machine) are:

- A) Operating system proper, especially 32-bit issues
- B) "Cross" software (assemblers, symulators for micros etc.)
- C) Compilers
- D) Data bases and communication networks

About the Israel Unix users' group, I hope we'll have one soon, as the number of active installations grows.

By the way, we know here pretty well what's going on among the 3-4 Unix'es and we do exchange ideas.

Sincerely yours,


Gershon Shamay

Eder St. 49a, P.O.B. 72, Haifa, Israel. Phone: 04-246033.
Telex 46400 BXHA IL, For No. 8351

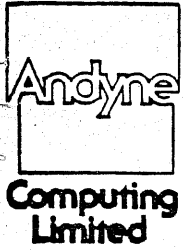
TECHNICAL DISCUSSION

The technical portion of the meeting was then begun, leading off with Doug Ross of Andyne Computing Limited, of Kingston, Ontario. Doug explained a project in which they are studying and implementing additions to a timesharing system, to support realtime activities. In particular, UNIX is the first target of this investigation. Doug explained some of the background information they have collected on realtime processing, and outlined the approach they are taking in this project. He emphasized that the target system will support timesharing and variable-schedule realtime processing; it will likely be driven by a family of scheduling techniques, including deadline-driven scheduling for 'hard realtime' processes.

The most important novel features of the system will be the provision of "hard" and "soft" realtime scheduling as well as timesharing. These are provided by a system of mutual guarantees between processes and system reservations. The process reserves services and guarantees not to overuse them.

Doug then went on to show how these features would be provided at the application level, by showing source level statements for realtime activation. He pointed out that a number of features of PDP-11 based UNIX need to be analysed before a firm understanding can be reached as to the amount of realtime processing possible. These features include interrupt-driven I/O and memory cycle stealing by DMA devices; these and other features reduce the time which the scheduling system can guarantee to user processes. There may be hardware configurations on which pathological cases can arise, thus prohibiting any true guarantee of realtime response.

The second speaker of the afternoon was David Tilbrook, of E-N Software Research, Toronto. David described MASCOT (Modular Approach to Software Construction Operation and Testing). This is a set of design and programming primitives which allow a parallel realtime system to be built in a well-structured, modular manner. The MASCOT system was developed by the Royal Signal Corps in Britain, and is now the standard method of teaching OS principles in Royal Military Colleges in Britain. Although the MASCOT primitives are similar to the UNIX system calls, they handle Interprocess Communication more effectively. MASCOT can be supported on standalone systems, or under existing OS's. David has added the MASCOT primitives to a PWB-UNIX system, thus allowing software to be developed under timesharing, and then ported to target systems. He also hopes to start with a standalone MASCOT system, and build a UNIX using the MASCOT primitives in that system.



P.O. Box 1496, Kingston, Ontario

Software Engineering Services

K7L 5C7 (613) 548-4355

(Presented at UNIX SIG meeting
by D. Ross. --ed.)

Your File No.

Our File No.

REAL-TIME ADDITIONS TO A TIMESHARING SYSTEM

In laboratory settings it is common to have application programs which contain tasks which have time constraints on their execution. A timesharing scheduler does not provide a mechanism for a process to control its execution.

We are taking the approach of modifying a timesharing system to provide support for real-time activities. This will involve modifications in three main areas.

1. Scheduler - The timesharing scheduler will be augmented with a scheduler which provides dynamic priority assignment to processes such that they will receive their required execution time prior to their established deadline. Processes wishing to avail themselves of this service will have to provide their execution time and deadline requirements to the scheduler. Normal timesharing scheduling will be provided to all processes not requesting the special service.

2. Program structure - Programs will be written as phases. Each phase will indicate the service category, execution time, and deadline required. Service categories will be:
-- guaranteed response;
-- requested response;
-- unspecified response.

Unspecified response phases are not explicitly identified but are any sections of the program code which are not guaranteed or requested response phases.

3. Reservations system - Prior to execution of a process containing guaranteed and/or requested response phases, it must have been authorized by the reservation system. The reservation system

.../2

examines, for each set of concurrently running processes, the real-time load they represent. Before being authorized the system ensures that the set of service requests can be handled.

The reservation system performs a worst-case analysis of guaranteed response phase activations. Once a set of processes is authorized the system has guaranteed that it can schedule them all. A similar, but independent, analysis of requested response phases is also performed. Although this appears to be overcommitment of resources, actual usage is expected lower than the worst case analysis allows. In the event that the system does overload during execution of phases, it is the guaranteed phases which receive their required service.

If a guaranteed phase or a requested phase exceeds the demand on system resources specified in its reservation, it will be "fused", i.e. refused service of the quality reserved. It may even be aborted, under some circumstances. The combination of reservations and fusing permits the combination of time-sharing with real time that is the novel feature of this implementation.

(The basic ideas for these developments were initially presented by R. Walton during the development of the PDP-9T real-time timesharing system in 1968-9.)

Biosciences Data Centre
The University of British Columbia
2204 Main Mall
Vancouver, B.C., Canada V6T 1W5

(604) 228-6527

June 2, 1980.

The Editor
Canadian UNIX Users Group
Human Computing Resources Corporation
10 St. Mary Street
Toronto, Ontario

Dear Group:

I thought that I would mention some of the things that I have been doing since I last had a letter published in LOGIN (obviously quite a while since LOGIN hasn't published anything of interest except meeting information for several years).

System changes: I have changed our system in the following ways that might be of interest to others:

In preparation for getting a large (300 MB) disk drive I changed the disk allocation free mechanism to maintain the disk block free count in the superbloc so as to speed up "df". It turns out that most of the time spent in "df" is taken up in the SYNC calls to update the superbloc. I made these calls optional and "df" now runs much faster.

I changed "trap.c" so that "trap" instructions may be intercepted. There is a new system call "systrap" that accepts an address in user D space. If this address is non-zero during a system call, it tests that word and if it in turn is non-zero, treats the TRAP instruction as a bad system call. This allows one to write a program that traces what system calls another program makes, as well as being useful for running DOS, RT-11 or RSX programs under UNIX.

I changed the swap space allocation routines so that if we run out of swap space (we often do so on 2 RK05's with 9 or 10 users), it will remove any of the unused STICKY programs and try again. This makes the penalty for STICKY programs less severe.

I changed the line-printer driver so that it picks up duplicate series of characters (usually blanks) and feeds PUTC/GETC with a character plus a (negative) count. This speeded up the driver by about 50 percent when in plot mode.

Programs Available:

Fortran 77... My Fortran 77 compiler is available to any UNIX installation for a \$50 distribution charge.

Basic ... I have an ANSI Basic (written in C), available for a \$25 distribution charge.

NTP... A version of TP written in C (from the UNIX library tape), modified to use floppy disks, automatically create directories, and protect files on extracts.

PP... A plot filter for Printronix 300 line printer. Would have to be modified to run with non-UBC plot routines.

LISP... A LISP interpreter written in C.

FAKERT11 ... An program that runs RT-11 programs under UNIX.

RTUNIX ... A subroutine library that lets you run most UNIX programs under RT-11 (but only use it for your own programs, not Bell's).

AR ... An interface to AR that allows both new and old AR formats.

COST ... Two routines that let you figure out system utilization and charge for it (requires a few system changes in EXIT to work).

FS ... File save program for Mag-tape. Allows access to multiple versions of files. Requires modified MT driver.

CCC ... Interface to CC that only recompiles those files that have been changed since the last compile. (like MAKE, but simpler).

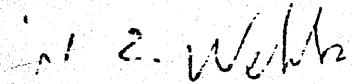
ASA ... Converts ASA (Fortran) carriage control into normal ASCII control characters.

UNARCV ... Converts V7 archives back into original V6 format.

FILECP ... Reads a V6 file system under RT-11 or UNIX V7 or whatever.

I am planning on putting together a UBC distribution tape if there are enough requests for it. Cost will be about \$100 and it will include Fortran77 and Basic and just about anything else of interest done at UBC.

Sincerely,



W. E. Webb.
Systems Analyst

WEW:wew

Craig McGregor
Computer Graphics Laboratory
Faculty of Architecture
University of New South Wales
P.O. Box 1
Kensington 2033
NSW
Australia

Dear Peter,

These are some comments that I hope you will see fit to publish in the next AUUGN. They were prompted by Adrian Freed's letter in a previous edition.

To start with, I shall describe the UNIX terminal that I would like to see. A number of the features I shall describe have already been implemented in various teletype drivers. The features are in two categories, operator control functions, and output command functions. The operator control functions would be a single keystroke. Many of these would have some visual response on the display. I shall not go into the possible responses as there are any number, most of personal appeal, some appropriate for VDUs and some for printing terminals. The most important operator commands are:

delete - delete the last character entered.

erase - delete every character entered in the current line.

repeat - repeat every character in the current line. This is only needed if proper screen editing is not available, as on a printing terminal.

stall - stop outputting any more characters until resumed. No characters are lost, when restarted the next character sent is the one expected after the last one sent.

flush - lose all characters sent to the display. There will need to be some function to start outputting characters again.

I would also like to see some control functions available within the character stream:

delay - it should be possible to delay for some number of milliseconds or clock ticks. This is desired to handle the delays for various types of terminals which require delays at all sorts of, strange places in plot commands.

conversion - it should be possible to toggle the LCASE bit from the character stream. This is required because some single-case terminals require the full 7 bits for plotting and hence require a lot of 'ioctl' system calls which can be slow and also fouls up the buffering.

Other features that are of interest:

tandem - similar to stall mode, only the other way around. That is the computer sends the stall character to the terminal, normally

another computer, when it wants to stop inputting.

break - means to transmitting and receiving break signals.

true raw - reading a string of exactly "n" characters in raw mode. At the moment, Bell's raw mode usually returns after reading only one character.

case conversion - instead of a single LCASE bit, it may be better to have a conversion mode number. This would allow selection of other case conversion schemes even EBCDIC.

The terminal control functions should also be available to a user program: screen clear, cursor addressing etc. The implementation of these functions may lead to some problems; the "teralib" implementation is alright for some purposes but will not solve the problem of doing the "line clear" required for the most friendly form of "line delete" on a VDU.

Next, I shall attempt to describe my personal preferences for C program formatting. Whilst a lot of my conventions are purely subjective, I feel that some of them have real advantages over some other formats, such as that produced by 'para'.

I find that the positioning of braces on lines by themselves can easily cause mis-reading of code, particularly when a page-feed occurs between the brace and the associated lines. ~~When this occurs the following can happen:~~ When this occurs the following can happen:

- 1 The 'while' part of a 'do' statement can appear to be a 'while' statement.
- 2 The complex statement part of an 'if' statement can appear to be a block.
- 3 An 'if' statement can have a non-apparent 'else' part.

Some other rules that I tend to use are:

- 1 When possible don't use variable lists in declarations. Attempt to place all variables on separate lines. I have even gone so far as to line up the variable names in columns.
- 2 I try to place comments in only two places: on the end of a line containing a single line statement; in a block by itself using the same left margin as the surrounding code. I never place a comment on the same line as a brace, excepting the following example.

```
char fred[] = { "mary" }; /* ... */
```

In the case of a multi-line comment, I format it so:

```
/*  
 * ....  
 * ...  
 */
```

The following is my version of Adrian's example. As well as reformatting the code, I have made some slight changes which I believe improve the readability of the code.

```

unsigned int      argi;
extern int        ldivr;

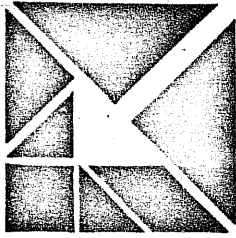
main(argc, argv)
unsigned int      argc;
char              *argv[];
{
    register char *cp;
    register int  wd;

    argc--;
    for (argi = 1; argi <= argc; argi++) {
        for (cp = argv[argi]; *cp != '\0'; cp++) {
            if (*cp == '\n') {
                if (*++cp == '\n') {
                    putchar('\0');
                    continue;
                } else if (*cp == 'c')
                    exit(0);
                else if (*cp == ' ') {
                    putchar(' ');
                    continue;
                } else if (*cp == '0') {
                    wd = 0;
                    while (*++cp >= '0' && *cp <= '7') {
                        wd <<= 3;
                        wd |= *cp - '0';
                    }
                    putchar(wd);
                    cp--;
                    continue;
                } else
                    cp--;
            }
            putchar(*cp);
        }
        putchar(argi == argc ? '\0' : ' ');
    }
    exit(0);
}

putchar(c)
char      c;
{
    write(1, &c, sizeof(char));
}

```

Craig McGregor



the Time-Machine Ltd.

ת"מ-מכשיר

Haifa, 26-8-1980

Peter Ivanov

Dept. of computer Science

Electrical Eng.

POB 1

Kensington 2033

Australia

Dear Mr. Ivanov,

We currently have UNIX V7 running on our 11/34. It took about 1 man-month to upgrade from V6. We squeezed out the accounting and the multiplexed files from the kernel, but even so, the size of V7 is causing some problems (looks like it was written only for separate I/D machines). Utilities like the portable C, F77, AWK etc, cannot be run at all.

Our main problem now seems to be the number of inodes and I/O buffers we can allocate. During development work, each terminal needs about 10 inodes (for pipes, standard I/O's etc.). Since on 11/34 the maximum physical memory which can be allocated to kernel text and data is 48K bytes, the machine hardly supports more than 3 online users (absurd, isn't it?).

Do you know of someone who solved this problem? There must be some way to get the kernel smaller, so we could add drivers for more peripherals (currently only the RL01 and DZ11 drivers can accomodate).

I would also like to know if someone is running the portable C compiler on an 11/34.

Best regards,

Gershon Shamay

The Time-Machine

Eder St. 49a, P.O.B. 72, Haifa, Israel. Phone: 04-246033.
Telex 46400 BXHA IL, For No. 8351

TELEPHONE
345 1844
TELEGRAMS
UNIMELB PARKVILLE



University of Melbourne

DEPARTMENT OF COMPUTER SCIENCE

Parkville, Victoria 3052
19th September, 1980.

W.N. Joy,
Department of Electrical Engineering and Computer Science,
University of California at Berkeley,
Berkeley,
CALIFORNIA.

Dear Bill,

I thought I would let you know what I have been doing to VMUNIX since you gave me the tape with your latest system at the end of June.

Needless to say, the system ran without any difficulty, though one strange thing was that the file `/usr/include/wait.h` wasn't in the `/usr/include` directory (nor in `/usr/src/sys/h`). This was no problem, but it would be nice to see a copy of your file so I can make ours the same (ie. to get the 'correct' values for the option flags, so the systems remain binary compatible).

I have made the following changes, many of which were in our version of the original VMUNIX release. These are (more or less) in order, some mods depend on earlier ones.

- 1) Buffers are allocated at boot, rather than at `ld` time. There is also a fix for the bug that caused `bss` to fail to be cleared properly at startup (a stray sign bit not being reset in an address). Some dead code in `mba.c` was also removed (it would have been wrong if ever executed after this change).
- 2) Two new system calls have been added. `Tlimit(n)` sets a CPU time limit of `n` seconds (for student `f77` programmers). `Uname (& utsname)` is from Bell's later versions (3.0 or 4.0 or something). Incidentally, I don't really know what the `release` and `version` fields should contain, do you? I guess the `sysname` should be "VM/UNIX", and the `nodename` is easy.
- 3) Function `uprintf()` was added to `prf.c`. It is just the same as `printf()`, but prints on the controlling `tty` for the current process, and guarantees not to block (though there is no need for that at the minute). This function is called to warn users who encounter no space/inodes on device conditions. The system also delays (uninterruptibly) for 5 seconds any process that causes an 'out of space' message (so the system doesn't bog down with `printf()`'s to the console).
- 4) Disc quotas (without change to the way I described them) were implemented. These are still raw.
- 5) `Hp.c` has been fixed (finally). ECC correction is modelled (very closely) upon that in `up.c`, but handles `RMO3`'s correctly. Struct `mbaregs` now includes `mba_map`, so the hideous `#defines` for the map addresses can be removed.

6) Device error logging has been added (in `hp.c` & `machdep.c` (for mem) anyway). This is just the same as is on the Delaware tape.

7) Bugs in `closef()` and `umount()` have been fixed (I think). Block device close routines get called at the correct times (via `closef()` only if not mounted, and via `umount` only if not open). In core blocks associated with a closed block device are invalidated (though this is not well done yet, the code needs to be rewritten). Also, the close routine for a special file is not called if the device is open via an alias inode (i.e. non-link that refers to the same device).

8) A structure, and an instance of it, containing pointers to the ends of the system tables, and their sizes, has been added. References of the form `&proc[NPROC]` have been altered to `v.v.proc`. (I think this originated in Bell Unix). Of itself this is of no benefit (it should even make the code run slightly slower) but it forms the basis of a later mod, and of future mods to come.

9) The `bfreelist` scheme has been extensively modified. There are now 4 freelists, `brelse` chooses which to add a buffer to depending on how quickly we want it reused (as indicated by `b_flags`). It would be easy to add more queues if some hierarchy of re-use times were developed. In this case, `buf` structs should probably include a `b_queue` field to indicate which queue the buffer belongs on (replacing `B_AGE` and a couple of new flags I have added). The 4 queues are: (a) Never to be re-used, (b) Delay re-use as long as possible, (c) Re-use if necessary, and (d) Re-use as soon as desired. The criteria for placing a block on the queues are:

(a) None (this queue isn't used yet - it will be in the next mod)

(b) None of `B_AGE`, `B_ERROR`, and `B_INVAL` is set.

(c) `B_AGE` is set.

(d) `B_INVAL` or `B_ERROR` is set.

`B_INVAL` is a new flag set by `geteblk()` to indicate that the buffer doesn't contain valid data (it is easier to test than `b_dev == NODEV`). It is also used to mark a block invalid after its device is closed (in which case `b_dev != NODEV` anyway). Naturally `getblk()` and `geteblk()` make sensible choices when a buffer is needed. All 4 queues are FIFO, there appears to be no reason (other than expediency) for having any LIFO 'queues' as there used to be.

10) `Mount()`, `umount()`, and `update()` all avoid the `bcopy()` of the super block. This is done by arranging that the super block go on `bfreelist` queue (a) so it can not be re-used (this would seem to be the reason for getting an extra block and not `brelse`'ing it). This also permits I/O directly to the super block (without any disc I/O for input). `Df` runs faster (without `-l` anyway), and `fsck/icheck` can re-build the super block of the root without requiring a re-boot.

11) A dynamic upper limit of used slots in the `proc` table is maintained. This makes searching it quicker in all those searches which are still linear. The same ought to be done for files, and possibly texts (though a direct pointer from the inode would stop most text searches anyway). Inodes should have some form of LRU scheme (which means that they are always all in use (like `bufs`)). I haven't re-installed this as I am not sure how it relates to your hashing scheme, also I believe I have a better method than the way it was originally done at Toronto under V6).

12) The `fentl(fd, cmd, arg)` sys call (from Bell Unix) has been added. `Open()` has been modified to implement all the fancy flags (`O_APPEND`, `O_CREAT`, `O_TRUNC`) except `O_NDELAY` which I don't know how to do. I don't even really know what it means. (Does it imply non blocking I/O for all I/O requests, only special files, or only some special files? Does the I/O proceed anyway, in parallel with the process? Or is it just that if data is not ready the sys call returns? Do you know?) I have added `O_WRTFULL` however. It causes the size of the inode to be restored if a write fails for any reason. Its purpose is to stop `/usr/adm/wtmp` from becoming corrupted whenever `/usr` runs out of space half way through writing a record (it is better for all the data to be lost than half of it). I have added (or more correctly Sydney Univ. has) `fentl` commands to lock and unlock files (deadlock is prevented by only permitting one locked file per process). The implementation is weak if a locked file is subject to `dup()`'s or `fork()`'s, but survives the simple, usual cases. There is also a new file type `IFLOK` which locks automatically on open, but is identical to `IFREG` in all other respects (that is untested so far, I haven't modified `mknod.c` nor `fsck.c` yet).

So much for what I have done, what I want to do in the future (which will begin sometime in November when the students go away) is:

- 1) I want the system tables (or most of them) to be allocated at boot. This is useless by itself, but ...
- 2) Then `NPROC`, `NBUF`, ... should become variables, so they can be altered via `adb` on a `/VMUNIX` file (there are already no `&proc[NPROC]` references to become inefficient by this change). Again no great saving (1 or 2 less re-compilations a year), but ...
- 3) Finally, do dynamic allocation/deallocation of tables as required. This will be easy on the VAX, but is probably not doable on `pdp 11`'s or `Interdatas`. (The limited address space of a `pdp 11` would probably make it useless anyway).
- 4) I want to add a `F_TRUNC` cmd to `fentl`, so files can be truncated without knowing their names. It would be ideal if it were possible to delete only the last section of a file (i.e. shorten it) but the `arg` to `fentl` is only an `int`, and this is just not good enough. I don't think it worth a new sys call.
- 5) I still have to remove the limitation to 64K of buffers.
- 6) Now that there exists the means to lock a block in core, I intend to use the sticky bit on a directory inode to cause the blocks of the directory to be locked in core once read, then apply it to `'/'`, `'/usr'`, `'/bin'`, `'/usr/bin'`, and `'/usr/ucb'` - 9 blocks total on our system. However this really needs more than the 60 buffers we currently allocate (i.e. the previous mod), especially since 10 of those could be in use by our 5 printers. I am also unsure what sort of limit to apply to blocks retained this way. An absolute limit would be easy, but not really what is wanted, a limit per directory wouldn't be so easy, but is closer to being satisfactory. Possibly the system should just reset the sticky bit whenever a new block is added to a directory, and leave it to some SU to decide?

One strange thing that has occurred lately, is that the system has been executing the `printf("issig")` in `sig.c`, which is supposed to be impossible. Does this occur on your system, or is it something I have done? I haven't looked for the cause yet (it doesn't seem to do any harm, so there is no urgency).

Another bug that I have fixed (and forgot above) is that two lines in `exit()` were in the wrong order. The code used to be

```
noproc=1;  
p→p-stat=SZOMB;
```

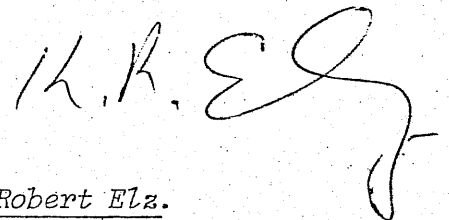
If a clock interrupt occurs between the two lines, `clock()` may try to alter the priority of the exiting process, causing "panic: remrq" when it can't find the process on its assigned queue. Reversing the lines should cure it.

I would appreciate hearing of any recent changes you have made. Did you get any explanation from anyone at Bell for the strange way `disksort()` behaved? If so I would love to know it. If you want a tape with any or all of these mods on it, just let me know and I will send it (I still have the tape you gave me, which must be returned sometime). You should be able to get a short message to me by mail to `ianj` (hope you don't mind Ian) containing a request that he forward it to me via Sydney, but it probably won't be much quicker than the ordinary postal service, (I don't log on to Sydney often). I also intend to send a copy of this letter to the local UNIX newsletter (AUUGN) so the rest of the country can learn what I am doing. If you reply, and don't object, I will send a copy of your reply too.

Finally, you may recall that I asked you about 'scald' when I visited you, but whoever was responsible for it wasn't around. Could you pass on the following request to that person (I don't think that you mentioned a name).

We would like to obtain a copy of 'scald'. Can we get it from you, or do we have to approach the original authors? (Stanford)? If the latter, can we get a 'pi' running version from you if we get the other first? If you can give us a copy, do you want me to send a tape, or would you prefer us to send the value of a tape and let you supply one. Naturally we will pay any postage/handling charges. Thanks,

Yours sincerely,



Robert Elz.

cc: Peter Ivanov, AVVON Editor

University of Essex

Department of Electrical
Engineering Science
Wivenhoe Park
Colchester CO4 3SQ
862286
Tel: Colchester 223214 (STD Code 020 6)
Telegraphic address: University Colchester
Telex: 98440 (UNILIB COLCHSTR)

DBA/MS

22nd September, 1980

Peter Ivanov,
Computer Science,
University of New South Wales,
P.O. Box 1,
Kensington,
New South Wales
AUSTRALIA 2033

Dear Peter,

Thanks for the tape. Now for a trip to the nearest Unix with tape and RK05 - not too far and a good excuse to travel! Our disks are being installed today.

Your request for V7 strip-down has been passed on to Alan Mason - I'm sure he'll send you a tape.

I notice that most of AUUGN. 2-5 is from the UK newsletter. Perhaps we ought to do two things from now on: (i) have a detailed contents page for each issue, and always reproduce that in each other's newsletters; and (ii) acknowledge items from other newsletters. This means that readers can judge whether to obtain the whole of other newsletters, and also won't go chasing after things they've already seen.

All the best,

Yours sincerely,

Bruce

D.B. Anderson.

*Bruce,
I agree. I was also pleased to see in the last UK newsletter
that you left the "AUUGN" page on the section from
us. It helps me find out whether I have already printed
something or not. I shall endeavour to do both things above
from now on.*

TELEPHONE
345 1844

TELEGRAMS
UNIMELB PARKVILLE



University of Melbourne

DEPARTMENT OF COMPUTER SCIENCE

Parkville, Victoria 3052

4th October, 1980

Peter Ivanov,
Dept. of Computer Science,
Electrical Engineering,
University of New South Wales,
Kensington.

Dear Peter,

Please find enclosed my subscription for AUJGN Vol III. I meant to send it with the other junk I sent a couple of days ago, but it got forgotten.

The delay might have served a useful purpose, since I have found an `stdio` bug that might be worth mentioning.

In `fclose()/fdbuf.c` `iop->_base` is set to `NULL` (which incidentally is a stupid thing to set it to in any case, as `NULL` is a `FILE` pointer) only if one of the flags `_IOMYBUF` or `_IONBF` is set. One of these will be set in all cases, except where `iop == stdin`, or a buffer was allocated by the user via `setbuf()` (or possibly if `iop->_base` is already `NULL`, but that is irrelevant).

If `setbuf()` was called, it is essential that `iop->_base` be cleared in `fclose()`, as otherwise the same buffer will be user next time this particular `_iobuf` is opened. If the buffer used just happened to be in a stack frame that has since vanished, or in some other random location, nasty side-effects can occur (as you can imagine I am sure).

If `iop` was `stdin`, the only effect that clearing `iop->_base` could have, would be to waste `_sibuf` next time `stdin` was opened. (And how many programs ever `fclose(stdin)` anyway).

So I suggest that the test be removed entirely. If there is a desire to avoid wasting `_sibuf`, the following code can be placed in `_filbuf()/filbuf.c` and `_flsbuf()/flsbuf.c`:

```
if (iop == stdin) {
    extern char _sibuf[];
    iop->_base = _sibuf;
    goto tryagain;
}
```

In both cases it goes just before the call to `malloc()`.

I hope that this might save some of your readers the problems that it gave me (after all, it is not normal to suspect `stdio` is at fault if your stack is being destroyed). The bug probably hasn't been noticed before, as it will only appear in fairly unusual circumstances.

Yours

Robert Elz.

Peter Stevens
C/O TIE Communications
5 Research Dr.
Shelton, Connecticut
USA 06484

24 September 1980

Mr. Peter Ivinov
Dept. of Computer Science
Electrical Engineering
University of New South Wales
P.O. Box 1
Kensington 2033
New South Wales
AUSTRALIA

Dear Mr. Ivinov,

Charles Barrata, of the University of Waterloo, tells me that you may be able to send me some information about the Australian Unix Users Group. TIE has a commercial Unix version 7 license and runs Unix on a PDP 11/44.

Charles also said that AUUG publishes a regular newsletter that is informative and carries articles that contain programmes. He quoted a figure of \$10 as the AUUG membership dues. Would it be acceptable for me to send a cheque or bank draft for the equivalent amount in US Dollars? Could you please quote a price for packing the newsletter by airmail? Thank you for your help.

Yours sincerely,

Peter Stevens

Peter Stevens
Software Development Group
TIE Engineering



CMG

DEPARTMENT OF FORESTRY



DIVISION OF TECHNICAL SERVICES

Ryan House 366 Upper Roma Street Brisbane
 PO Box 5 Brisbane - Roma Street 4000 Tel. (07) 229 6500
 Telegrams and Cables "Forestry Brisbane"

Enquiries to Mr P. Chadwick
 Extension 10

Your Ref

Our Ref 267/2 (TS13)

Date 21 October 1980

Peter Ivanoff
 Department of Computer Science
 University of N.S.W.
 P.O. Box 1
KENSINGTON 2033

Dear Sir

Further to my verbal request to publicise software needs in the UNIX newsletter.

Forest Research Branch is looking for software to run under version 7 of UNIX which will communicate with a UNIVAC 1100 series mainframe using any of the UNIVAC protocols listed below:-

U100
 U200
 1004
 UDLC

Any help you could give would be greatly appreciated.

Yours faithfully

for Director of Technical Services

Department of Psychology



University of Queensland

St. Lucia
Queensland
Australia 4067
Telephone: (07) 377 1111
Telex: UNIVQLD AA40315
Cables: Brisbane University

RG:dm

21 October 1980

Peter Ivanov
Editor AUUGN
Dept. of Computer Science
University of New South Wales
P.O. Box 1,
KENSINGTON. N.S.W. 2033

Dear Peter,

Please find enclosed a cheque for that most magnificent of publications, AUUGN. I will need a receipt to recover my \$12 from the Department. Has Volume II Number VI come out yet, as I don't seem to have received a copy?

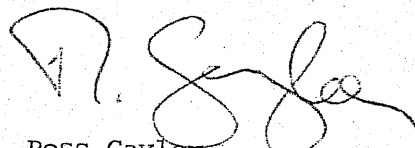
Possibly the most interesting piece of software news from here is that Rick Stevenson of Computer Science has been putting the UNSW mods and a few of his own into Version 7. The result is a BIG-UNIX and MAPPED-BUFFERS Version 7 on an 11/34. There is also a driver for the TS11 tape drive in the pipeline.

While on the topic of Version 7, could you ask John Lions what is happening about the V7 manuals that sundry U.Q. people ordered quite a while back?

I would also like to ask a final favour. The University is buying a VAX for use as a central computer. The Computer Centre wants to run VMS but there is a group lobbying for UNIX. To help us in this could you send us a copy of the AUUGN site configuration/ mailing list? We would like to be able to indicate the size of the Australian UNIX user population and the range of applications covered. We also wish to send a short questionnaire to all the Australian sites.

Thank you very much for your help.

Yours faithfully,



Ross Gayler

Enc.



Human Computing Resources Corporation 10 St. Mary Street, Toronto, Ontario, Canada M4Y 1P9
416-922-1937

24 October 1980

Dr. Peter Ivanov
Department of Computer Science
Electrical Engineering
University of New South Wales
P.O. Box 1
Kensington 2033
Australia

Dear Dr. Ivanov:

Enclosed is the ad we discussed over the phone, also sent to you by Telex today. I would appreciate your giving this the widest possible circulation within Australia.

Should you happen upon an individual who appears to fit the bill, please call me collect or have him or her call me collect right away.

I sincerely appreciate your assistance.

Yours very truly,

Ronald Baecker
President

/ab
ENCL.