

AUSTRALIAN UNIX USERS GROUP NEWSLETTERCONTENTS

Editorial	1
Queensland meeting agenda	3
Queensland meeting attendees	4
Bad block utility - Queensland paper	6
Sanders printer - Queensland paper	8
The Sydney UNIX Network - Paper	12
Intro to the Berkeley Network - Paper	18
SUN netmail	30
Mail	33
I like Chinese!	44
Mine is bigger than yours	47
Other	48



---

In This Issue

---

This issue tidies up some loose ends left in the 'to be published' file. The Sydney UNIX network has at last been summarised by Piers and Bob from Bassar and the Berkeley network introduction is interesting.

---

Last AUUG Meeting

---

The last AUUG meeting went well with about fifty people attending. A list of attendees and an agenda appear in this issue.

I spoke about the 'birds of a feather' meetings at the Austin conference. I have not had time to summarise what I said but plan to for the last issue of this volume, should no better summary appear before then. High School UNIX appears to be an interesting topic and I asked David Woodrow, of St. Peter's Lutheran College, to send me a write up of the system that Darryl Whimp spoke of during the conference. Ross Nealon told us how PE UNIX was faring at Wollongong and I hope to see some words from him too.

Dave Horsfall and Chris Rowles have set the example by summarising their talks on 'A simple bad-block utility' and 'Using the Sanders printer'.

---

Formal AUUG

---

I put forward two proposals to formalise the AUUG at the Queensland conference, and what a 'bun-fight' ensued. The proposals were that the Australian UNIX users form a DECUS SIG (you need five bodies to start it up) and that also we should elect a number of formal AUUG office bearers as the publicly visible representatives of the group.

I won't go into the discussion that followed but we elected a few people with the following jobs.

David Horsfall - Chairman

Computing Services Unit  
Uni of N.S.W.  
P.O. Box 1  
Kensington 2033  
AUSTRALIA

Chris Maltby - Secretary/Treasurer

Bassar Dept of Computer Science  
Madsen Building. F09  
Uni of Sydney  
N.S.W. 2006  
AUSTRALIA

Bob Kummerfeld - Editor in chief

Basser Dept of Computer Science  
Madsen Building. F09  
Uni of Sydney  
N.S.W. 2006  
AUSTRALIA

Chris Rowles - Editor

Dept of Chemical Engineering  
Uni of Sydney  
N.S.W. 2006  
AUSTRALIA

Ron Baxter - Committee member

Elected in his absence

Please note that the primary role of the committee is, initially anyway, to look into the benefits and draw-backs of becoming formal, either under the auspices of DECUS or on our own.

---

### New AUUGN Editors and Subscription Costs

---

Bob Kummerfeld and Chris Rowles, the new AUUGN editors, regret to inform you that the subscription cost for AUUGN volume four will rise as from the 28th of September 1981. The new price is (Australian)\$24-00 and will apply to all subscriptions received after that date.

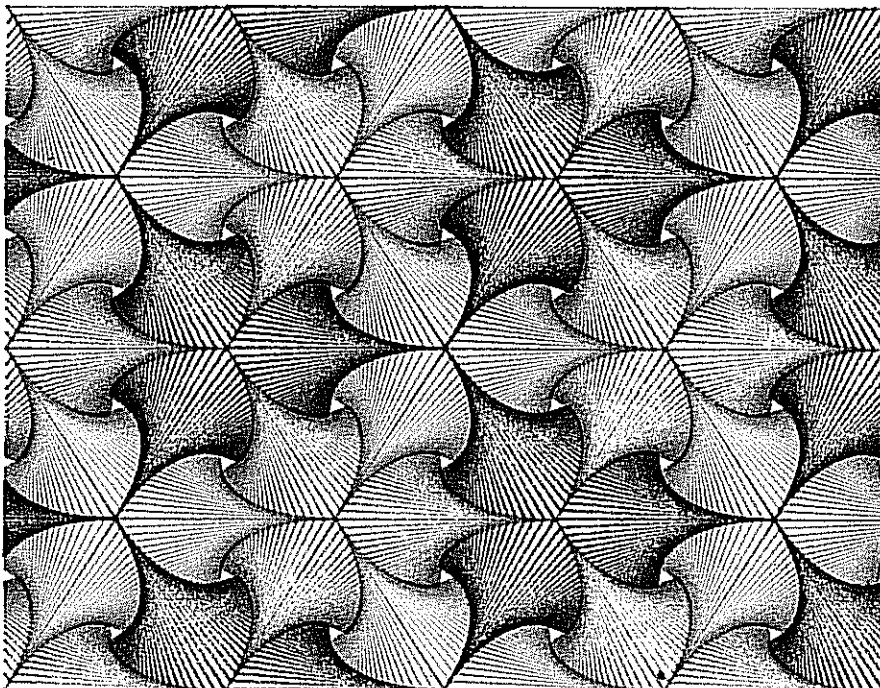
Reasons for the price rise are obvious. The last issue of AUUGN, volume three number four, cost \$2-73/issue to produce and mail. This is even with the very cheap production I am able to obtain, through the School of Electrical Engineering and Computer Science, and the large pool of volunteer labour I have to draw on.

All you people who subscribed early have saved yourselves \$12-00. Send subscription letters and monies to Bob at the address above.

---

Well it's better than Kev's chicken

---



Peter Ivanov  
Dept. of Computer Science  
Electrical Engineering  
University of N.S.W.  
PO Box 1  
Kensington 2033  
AUSTRALIA

(02) 662-3781

9:30 \*\*\*\*\* Registration & Coffee \*\*\*\*\*

10:00 Peter Ivanov  
>>>> U.S. UNIX Meeting odds and ends <<<<<

10:25 Darryl Whimp  
>>>> High school UNIX <<<<<

10:45 Ross Nealon  
>>>> What's on at Wollongong <<<<<

11:10 Dave Horsfall  
>>>> A simple bad-block utility <<<<<

11:35 Chris Rowles  
>>>> Using SANDERS printers under UNIX <<<<<

12:00 >>>> Formalisation of AAUG <<<<<

12:30 \*\*\*\*\* LUNCH & Sticky-Baking \*\*\*\*\*

14:00 Kevin Hill  
>>>> Modifications to V7 <<<<<

14:25 George Gerrity  
>>>> ?????? <<<<<

14:50 >>>> Formalisation of AUUG (again!)  
& selection of new editor  
& selection of next meeting site <<<<<

15:30 \*\*\*\*\* Afternoon Tea \*\*\*\*\*

16:00 Whatever takes your fancy

Aug 29 08:51 1981 Australian UNIX User Group Meeting, Union Co.

Academy Computer Software Ltd.  
Vandenberg, Geoff

Australian Atomic Energy Commission  
Peady, Glynn

Australian National University  
John Curtin School of Medical Research  
Kinns, Howard

C.S.I.R.O.  
Division of Mathematics and Statistics (Melbourne Region)  
Nelson, Graham

Fawnray Ltd.  
O'Brien, John

John Fairfax and Sons Ltd.  
Dunn, James  
Gallagher, Terry

Melbourne University  
Computer Science  
Elz, Robert  
Green, Ross

Queensland State Government  
Department of Forestry  
Chadwick, Phil

St. Peters Lutheran College  
Mackerras, Paul  
Riley, Brett  
Springall, Peter  
Whimp, Darryl

Sydney University  
Chemical Engineering  
Rowles, Chris

Computer Science  
Maltby, Chris

University Computing Centre  
Cole, Geoff

University of New South Wales  
Architecture  
McGregor, Craig

Computer Science  
Hill, Kevin  
Ivanov, Peter

Computing Services Unit  
Horsfall, Dave  
Kable, Greg

R.M.C., Duntroon  
Ellis, Michael  
Gerrity, George  
Smith, Malcolm

University of Queensland  
Chemical Engineering  
Burridge, Mike  
Newell, Bob  
White, Ted

Computer Science  
County, Phil  
Enchelmaier, Bob  
Kratzmann, Kirk  
Roper, Tim  
Schulz, Mark  
Stevenson, Rick  
Thiele, Doug

Electrical Engineering  
Barham, Ron  
Harridge, Clary

Mining and Metallurgical Engineering  
Algie, Steve

Prentice Computer Centre  
Currie, John  
Peterson, Brett  
Robbie, Mark  
Woodland, Allan

Psychology  
Gayler, Ross  
Groves, Mark  
Murdoch, Vince  
Pamment, Peter

University of Tasmania  
Computing Centre  
Bilson, Rod

University of Wollongong  
Computing Science  
Nealon, Ross

A Proposal  
for a  
Simple Bad Block Utility

Dave Horsfall  
Computing Services Unit  
University of NSW

(dave:unswcsu)

1. INTRODUCTION

This is a proposal for a utility to "flaw" out bad blocks on a file system. It was prompted by having to write off an RPO3 disk pack which had suffered from a minor head crash, resulting in a small scratch across the surface. Given the track density of the pack, it was impractical to logically remove the affected cylinders by redefining the file systems on it. Similarly, any genuine bad block support utilizing the sector header information would be uneconomical given the amount of work required to implement it and the benefits gained. It should be mentioned too that it was only slightly more expensive to buy a new pack than to refurbish the old. Some intermediate solution would therefore seem to be attractive.

The proposed utility FLAW offers the following advantages:

- \* No modifications to operating system,
- \* No reserved filenames or i-numbers,
- \* Usable on all disk drives (even floppies!),
- \* Flexible enough for most purposes.

It does have the drawback that if the bad block happens to be the super block, inode etc then nothing can be done about it. However, since these form but a small fraction of the file system this is not expected to be a problem. The accent is on simplicity, not omnipotence.

2. METHOD

The method is to make the bad blocks unavailable to the file system by allocating them to a known inode. The i-number of this inode is kept in the super block. Note that there is no pathname, and there is nothing "special" about the i-number (unlike V7 UNIX). Programs such as DTP and CPIO are therefore unaffected by it, and programs like DUMP can automatically ignore that inode. Of course, image copies will fail. Programs such as ICHECK, DCHECK and NCHECK will need to be modified, since these blocks will show up as "missing" etc.

The superblock entry is initially zero, implying no bad block file present. When a bad block is discovered (by reading the error log, or perhaps exercising the file system), the file system is unmounted and the FLAW utility run to allocate an inode (if required) and chain the specified blocks into the bad file. FLAW will need to check to see that the blocks are in the free list, as it is deemed undesirable to deallocate files automatically.



Of course, when the pack is reformatted all information is lost, and when MKFS initializes the file system the super block entry is cleared automatically. Should one wish to reinitialise the file system without reformatting it then MKFS can be modified to accept a list of bad block numbers.

### 3. SPECIFICATIONS

FLAW would need to have the following capabilities:

- \* Enter a block into the bad block file,
- \* Assign the bad block inode if necessary,
- \* Display bad block numbers,
- \* Only accept those blocks in the free list,
- \* Check all blocks on the file system, and possibly flaw them,
- \* Check individual blocks for possible flaws.

Future capabilities can be added later.

### 4. CONCLUSION

The solution offered here is simple enough to be implemented almost overnight. Its drawbacks of not handling all situations are outweighed by its simplicity and elegance. The concept of the bad block file fits in neatly with the UNIX file system, and no reserved filenames or inodes are used.

TELEPHONE:

or: 692 2455



## The University of Sydney

THE DEPARTMENT OF CHEMICAL ENGINEERING  
N.S.W. 2006

### UNIX AND THE SANDER'S MEDIA 12 TYPOGRAPHIC PRINTER

#### THE PRINTER:

The Sander's Media 12 is a dot matrix multipass printer. It produces letter quality printer output. The printer is microprocessor controlled (based on the Zilog Z-80) and has software generated character sets. There are currently over 40 character sets commercially available. Special purpose character sets can be purchased at roughly double the price for the stock set. These fonts are held in Texas Instruments' TMS 2716 (triple voltage supply) ROMs. Font modifications or new font sets could be made by the user by simply programming a set of ROMs. Physically the Sanders is built up of four printed circuit boards. One contains the printer control program, the Z-80, its communication interface and up to four fonts. A second card is available as a font expansion card. This allows an additional twelve fonts to be loaded. A third card controls the vertical paper motion and the print head mechanism in horizontal motion. Control allows horizontal resolution of 1/1000th inch and vertical motions of 1/243th inch. The last board controls the pin firing mechanism in the print head itself.

A second model has recently arrived on the market. In this model the print head is controlled by the user. It operates in a similar manner to an electrostatic plotter, in that the user must supply the raster information necessary to produce the printing. Very little is known about this model, although it should prove to be a better buy in the long run. All further discussion relates to the Media 12/7 printer with ROM loaded font sets.

THE SOFTWARE REPERTIORE:

The Sanders operates as an intelligent word processing unit. It has its own command language to control the formatting and printing of letter quality documents. The command set consists of the following operations.

Font manipulation: select font, assign logical font.

Print head manipulation: horizontal spacing (left and right), vertical motion (up/down), horizontal and vertical tabbing, indentation from left margin, select leading (vertical line spacing), set/clear tab stops.

Print format: no adjust, centre text, left justify text, right justify text, block justification (with inter-word and/or inter-letter justification); force printing of current line buffer, select repeat of a character string, insert white space to leave the next string right justified.

Page format control: length, left margin, width (ie right margin), top margin and bottom margin, select bold on/off, select underlining on/off.

Miscellaneous options: block mode, printer, reset ribbon usage command, select draft mode, page eject, initialise printer.

The Media 12/7 performs many text processing functions as a by-product of its normal operation. It determines the line length in terms of the number of printable characters that will fill the current line. Lines larger than this are wrapped around to the next line automatically, with the text broken at the nearest white space prior to the line overflow. Thus the first part of the broken line is justified (if required) and printed, while the remainder of the line passes to the front of the next line. Lines terminated by a carriage return cause the justification to be ignored for such a line. Force print causes the justification to be applied (with some disastrous results). Single words longer than the current line length lead to unpredictable results.

THE COMMAND LANGUAGE:

The command language takes the form of escape sequences intermixed with the output text. The arguments of the escape sequence is a variable size numeric count to set the desired option.

```
eg  <esc> 0 nnn  force vertical motion nnn/1000 in
     <esc> 1 nnn  force horizontal motion nnn/1000 in
     <esc> u n    toggle underline n/1000 in
     <esc> b n    set bolding offset n/1000 in
     <esc> a n    select font n
     and so on
```

NOTE that the number of the arguments is dependent upon the command and that n is expressed as a 2's complement binary number (each digit being 6 bits long (0 to 5) and bit 7 clear, bit 6 set). This makes hand calculation of the arguments inconvenient, but gives an ASCM encoded binary value that can be handled by most serial line interfaces.

This command set is very powerful. It does everything that you could desire, but it has some very nasty operational restrictions. These relate to horizontal motions. There is a strictly enforced limit of 20 horizontal spacing (non-printing) commands per line. There are also restrictions on left motion, limiting it to the width of the last printed character unless a vertical motion immediately proceeds the command.

The vertical motion provides a mechanism by which both restrictions can be averted. A vertical motion forces the current line buffer to be printed and resets the horizontal motion counts. Hence a vertical motion of zero length can be used to leave the print head in the next print position following the last character printed, without needing to calculate where it is in the output page.

UNIX SOFTWARE INTERFACE:

A UNIX filter has been written to drive the Sanders Printer. It validates the basic Sanders commands, it initialises the printer to a

known state and it tries to optimise the command strings by concentrating similar command strings into a single command (eg many horizontal motion commands produce a single command to effect the motion). The filter also accepts some psuedo-commands and interprets them. Notable among these are the revert to previous font fine spacing commands, select paper size format commands.

In its simplest form the filter takes input with imbeded Sanders commands and passes them directly to the printer. Thus text in the form of extremely long lines (ie paragraph length) uses the word wrap feature to produce very acceptable justified output. A simple process to calculate the numeric arguments for the Sanders commands, and to strip out new line and carriage returns, does produce very acceptable results.

A driving table for "NROFF" has also been developed. Although this is not completely correct it does produce very nice outputs from both TBL and NEQN as well. NROFF was modified to handle this hardware underlining and bolding commands by placing Sanders commands into the output stream rather than simulate them. NROFF also produces pseudo fine space commands (ie -e option) that are interpreted by the filter. Hence NROFF justifies the text, and the Sanders filter allows the printer to use its own internal operations to get the job looking really nice.

#### OUTPUT QUALITY:

The quality of the output is good. It is very dependent upon the mechanical adjustment of the plotter spacing, the overall cleanliness of the printer and the quality of the paper used. Given a little care with all three above, the output is very presentable.

The main weakness of the device is the print head itself. The firing pins can shear off the driving relays. They can also stick if the head is dirty or worn, leaving either one faint row of dots, or a score mark in the output. The print head costs \$600 (Australian) to replace. It is not covered in the annual maintenance agreement, but is in the two yearly agreement. This is a strange policy of the Australian agents, so be careful if you have a maintenance contract.

# The Sydney UNIX Network

R.J. Kummerfeld and P.R. Lauder\*

A computer network has recently been established within and between the campuses of the University of Sydney and the University of New South Wales. The network provides a virtual circuit facility for remote terminals and a file/mail transfer facility. The machines participating in this network all run the Unix operating system. The network software evolved out of a desire to allow the system maintainers and developers to co-operate more closely.

Keywords: Communication, Networks, PDP-11, UNIX.

CR Categories: 3.81, 4.35

## 1. INTRODUCTION

The Sydney Unix Network is a network of computers within the Universities of Sydney and New South Wales, that operate under the Unix Timesharing System (Thompson and Ritchie, 1974). The development of the network began in 1978 with the installation of a leased line connecting a PDP11/40 computer in the Basser Department of Computer Science at the University of Sydney (known as "basser40") and a PDP11/70 at the Australian Graduate School of Management on the University of New South Wales campus (known as "agsm"). This connection was used initially simply for access to the AGSM machine from a terminal connected to the Basser 11/40. Subsequently within each campus a number of other links were established and used in a similar way, i.e., a terminal on one machine could be used to login to a remote machine. Users wishing to login to a remote machine would invoke a special program that copied characters between their terminal and the outgoing link. At the remote machine the incoming link appeared as a normal terminal connection.

The limitations of this initial system were: (1) only one user could use the link at a time even though the available bandwidth would satisfy many users; and (2) the line was directional. This may be explained as follows: each computer running under Unix associates a simple login program with each incoming line. This program listens for activity on the line and then normally engages in a login dialogue with a remote human user. If a line connecting two computers is regarded as an incoming line by both, then the two login programs may engage in an endless and fruitless dialogue. Worse, if a connection is established by one of the login programs the other may interfere with the subsequent activity on the line. The standard solution is to allow only one machine to treat the link as incoming and hence constrain all new activity to be initiated by the other machine.

In early 1979, wider issues of network development began to be considered. These included file and mail transfer systems and virtual circuit facilities. Conventional ways of providing these services involve building a network of node computers connected together by reasonably fast lines and with host machines connected to the

network node computers. This solution was not seriously considered since in many cases the cost of a node machine would be comparable with the cost of the host it was to handle. Instead a host supported network system was developed that used no specialised network node computers and utilized the fixed and leased communication lines that were previously used as simple terminal connections.

## 2. PROTOCOLS

The protocols used in the networks do not, at present, conform to a strict layering as in other more conventional networks. For example, the lowest level of software protocol used over the physical communication lines is not strictly necessary for the higher levels, although it is used for all present links. This will be made clear in the description of the higher levels of protocol.

### 2.1 Physical Link Layer

The first "layer" of network protocol consists of the physical lines. These are either fixed lines running between machines in the same building or on the same campus or leased from Telecom. The interface to each machine conforms with the RS232 standard. The line speeds range from 1200 baud for the leased lines between campuses to 9600 baud for the lines between machines in the same building. The current topology of the network (February 1981) is shown in Figure 1.

### 2.2 Multiplexing Layer

The second layer is the lowest level of software protocol. It involves a simple multiplexing scheme that allows one physical line to appear to the operating system as several virtual lines. This scheme was in fact implemented very early in the network development as a solution to the problems that arose with the original single user, directional terminal connections. The multiplexing protocol allowed the lines between machines to be replicated by software. Thus, more than one user could be using a physical line at the same time and several virtual lines could be designated as "to" a remote machine and several as "from" the same machine.

The multiplexing line driver and protocol are known simply as "mx". There is very little in the way of error control. However, flow control is provided.

The mx protocol multiplexes "ports" onto real lines and simply separates data for each port on the line. Each block of data has a four byte header:

"Copyright ©1981, Australian Computer Society Inc. General permission to republish, but not for profit, all or part of this material is granted; provided that ACJ's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Australian Computer Society."

\*The authors are with the Basser Department of Computer Science, University of Sydney, Sydney, NSW, 2006. Manuscript received 3 March 1981.

Sydney Unix Network

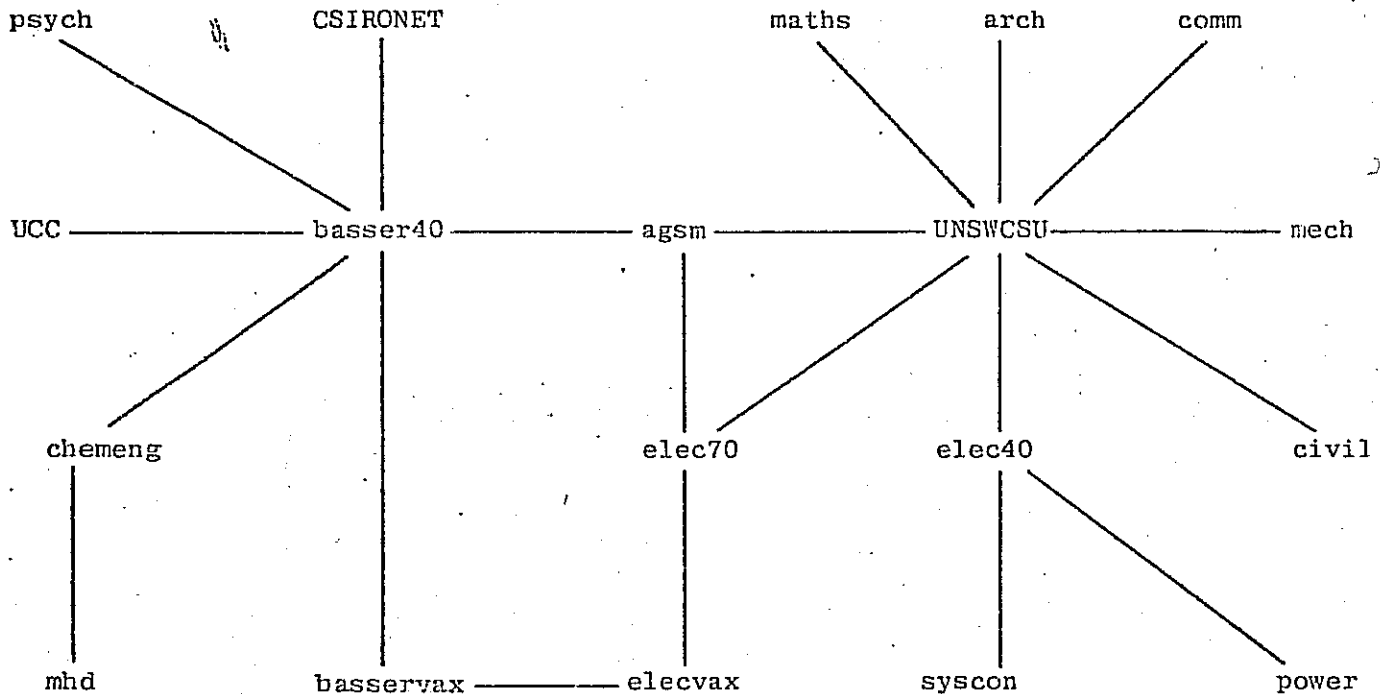


Figure 1. Network topology

SOH (ASCII start of header)  
 port-identifier  
 data-byte-count  
 complement of data-byte-count

This header is easily recognised and checked for consistency. The length of the data block that follows the header is usually limited to 40 bytes. Flow control is implemented by sending blocks without any data, the first byte count zero and the second byte count representing the number of data bytes that may be received. The sender must not send more than the specified number of data bytes before waiting for another flow control block. A simple timeout mechanism is used to recover from lost flow control blocks.

### 2.3 Transport Layer

The next network protocol layer may either be a virtual circuit facility or file transfer. The virtual circuit scheme is simple but effective and grew out of the original machine-machine connections that allowed a user to connect his terminal "through" his local machine to a remote machine. (See Figure 2).

The program now used to make the connection to the remote machine is called con (there have been other similar programs designed for special situations). This program takes as argument the name of the remote machine and from this determines the correct outgoing link to use. The program then performs a system call that would instruct the line driver to "connect" the terminal and the outgoing line. Characters are transferred from one line to the other very efficiently. The outgoing link may in fact be a multiplexed link created by "mx", along with several others, from a single physical line. This is hidden from the user and the con program.

In order to allow a user to connect to a remote

machine that is further away, i.e., that involves more than one intermediate machine, the con program carries out the extra function of routing. As part of the file transfer system, a network topology file is maintained. The details of this file will be described later. However, it does contain a list of all the network hosts and machines they are directly connected to. For example, from Figure 1 the topology file (called "netstate") would contain an entry for the host "basser40" as follows:

```

basser40 + agsm
          + basservax
          + chemeng
  
```

and the entry for "basservax" is:

```

basservax + basser40
          + elecvox
  
```

The con program consults this file before making a connection and determines the route of such a connection. For example, if a user at a terminal that is directly connected to the "basser40" types "con elecvox", then the con program consults the "netstate" file and discovers that the route from basser40 to elecvox is via basservax. It then sets up a connection to basservax and sends a special message to the login process at basservax that indicates that a connection is required to elecvox. The login process at basservax then in turn starts a con program with "elecvox" as argument and the process repeats. When the login process at elecvox receives the special message (now generated by a con program on the basservax) it ignores it and the login proceeds as usual.

When the user has finished his session and logged off the remote machine, he types a special escape character that is interpreted by the con program running at

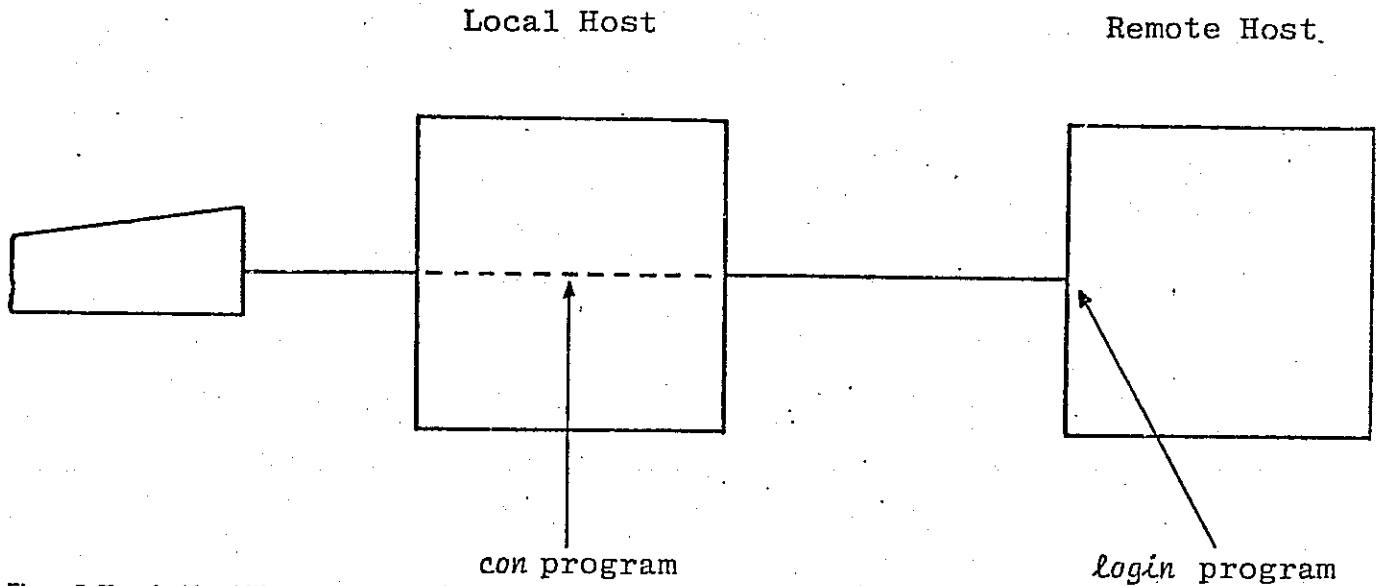


Figure 2. Terminal-local host-remote host

the local machine (A in Figure 3). This machine will in turn pass an escape character on to the next machine in the route (B in Figure 3) and so on until it reaches the remote machine. The con programs running at the remote and intermediate machines terminate, returning control to the login process in each case. The con program at the local machine also terminates and the users command interpreter regains control.

This scheme is very simple and has proved to be an effective solution. Connections are made from one side of the network to the other without problem.

### 3. FILE TRANSFER LAYERS

The other major part of the network is the file transfer system. It is implemented in three levels that may or may not be used above the "mx" level. It is possible for it to operate on dedicated lines between machines but in practice it uses one port of an mx controlled line.

The top level is the user interface layer, the second level implements an end-to-end protocol and maintains

the network topology file, while the lowest level provides an error-free path between directly connected nodes. Figure 4 is a diagram of the protocol levels of the system.

#### 3.1 User Level

At the user interface level, files, electronic mail or printer output may be transferred by a user from his home machine to any other on the net by using a single command. Network addresses are of the form "username:hostname" where "username" is a valid login name on the remote host whose network name is "hostname". This form of address is recognised by the mail program and the mail item is then handled by the "net" layer of the protocol.

The file transfer system is implemented with two user commands. These are "netsend" to send one or more files to a remote host, and "netget" to retrieve a file at a user's local host which has arrived as the result of a netsend on a remote machine. The netsend command has at least a username:hostname argument and may have one or more file name arguments specifying the files to be

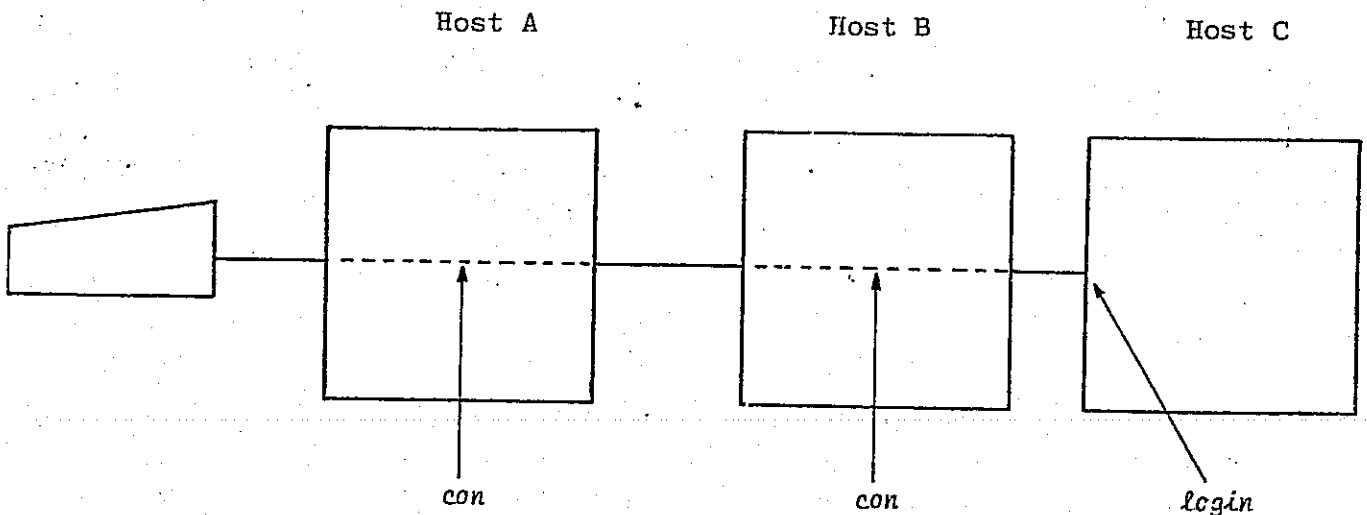


Figure 3. Terminal-host A-host B-host C



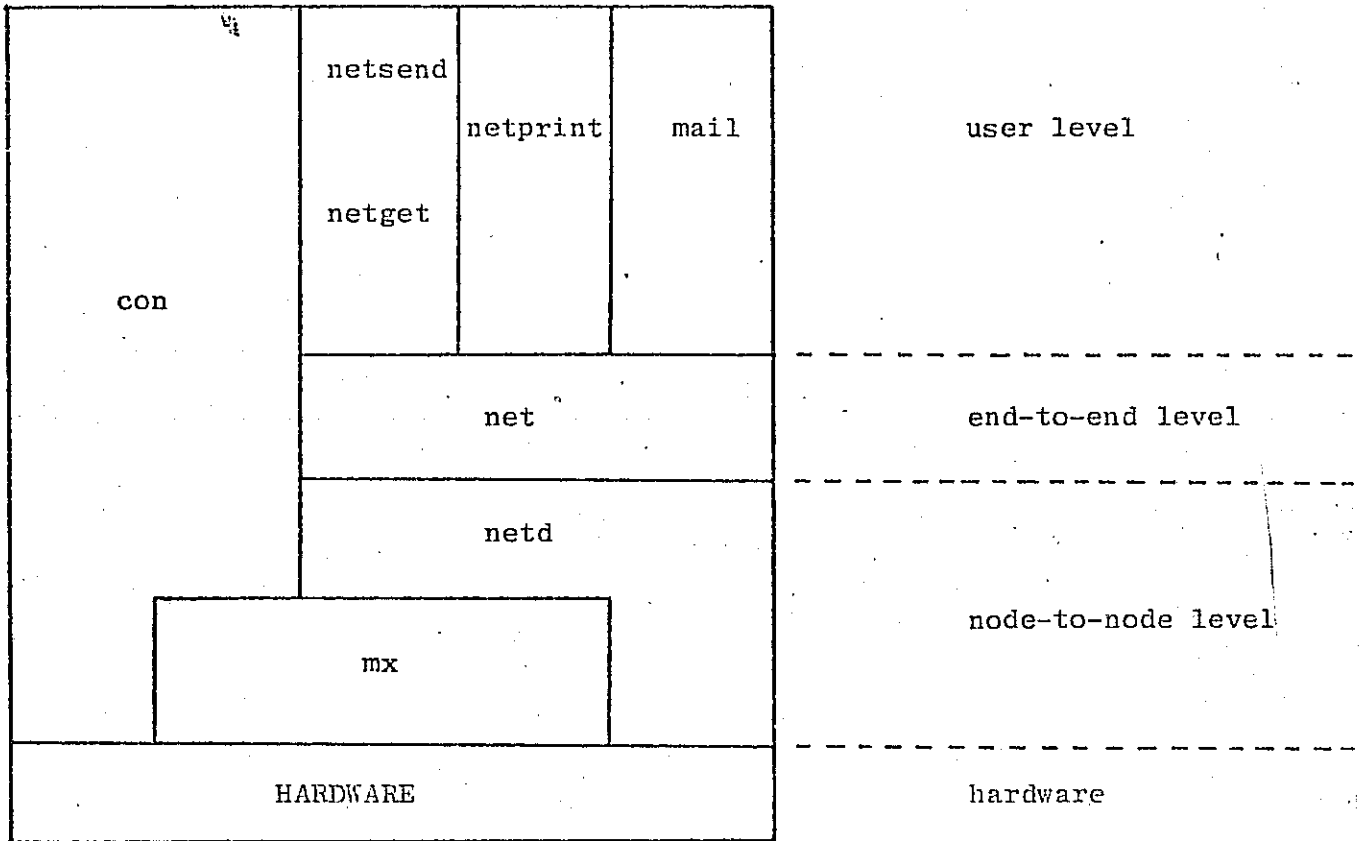


Figure 4. End to End Level.

sent. If no file name is specified, the standard input file is used. This is usually the users' terminal but may be a "pipe" from another program. The netget command takes zero or more option and filename arguments; when used without arguments it retrieves files that have been sent from other users or hosts and placed in a holding directory (or "spool"). It is important to note that netget does not allow a user to reach out from his local machine to some remote host and copy files. It is one of the major design decisions of this network file transfer system that this facility not be provided, the main reasons being security and the desire for simplicity. If the facility were available, a system of network wide user validation would have to be implemented. Since almost all the machines on the net are independently managed, and some have large numbers of potentially hostile student users, it was felt that such a scheme would be difficult to implement in a secure way. With the current system a user wishing to get a file from a remote machine would invariably have a valid user-name on the remote machine and so would be able to use con to the remote machine and "netsend" the file to his local machine.

The other service currently available at this level is a remote printing facility. A user may use the command "netprint remote-host-name file-name ..." to print a file on a printer attached to a remote host. The file is sent using the normal file transfer system but is passed to the line printer handling program (or "daemon") when it arrives at the remote host.

Also available to the user are two ancillary programs, "netq" and "netstate". The "netq" program gives

information about files waiting to be sent to directly connected hosts. The origin, destination, size and position in the relevant queue are given. The "netstate" program lists the network topology file in a readable form.

### 3.2 End-to-End Level

The next lower level, implemented by the program "net" (see Figure 4), accepts files for transmission, determines the next node in the route to the destination host and saves the file together with an end-to-end protocol header in a directory set aside for the next host in the route.

The end-to-end protocol header contains the following information:

- destination host name
- destination user name
- origin host name
- origin user name
- action
- file name
- action parameters
- statistics

When a file is received by the lower "netd" level it is passed to the net program for processing. If it is destined for the local machine it is placed in a directory to be retrieved eventually by a user via the netget program. Also a message is sent (by system mail) to the user notifying him of the arrival of the file. A time limit is placed on the files in the reception directory after which they are

removed. The user is warned of this time limit in the arrival notification message. If a received file is enroute to another host, it is placed in the appropriate directory and the "netd" level program will send it on to the next node in the route. The next node is calculated by the "net" program from information found in the "netstate" file. This is a "staging" system that involves complete reception of a file at an intermediate machine before sending it on to the next machine in the route. This has the disadvantage of requiring enough file storage at intermediate nodes to contain all the in-transit files, but this has not proved to be a problem. The benefit of the scheme is that it greatly simplifies the end-to-end protocol and allows routing decisions to be distributed across the network.

The net program also maintains the network topology file called "netstate" which contains an entry for each host. Each entry includes the time that the named host was last active and the transmission rate achieved. Following this is a list of hosts that are directly connected to the named host. Example entries have been given earlier. Each time the lower level "netd" program is started it signals "net" to send a message to all directly connected hosts indicating that the local host is now active. The directly connected hosts propagate this "host-up" message through the rest of the network and also send a copy of their "netstate" file back to the new host. When a directly connected host does not respond to some message and is deemed to be down, the netstate file is modified to indicate the new state. Thus, the network topology information is maintained by the "net" program to reflect the current state of the network.

### 3.3 Node-to-Node Level

Below the "net" program level the node-node file transfers are handled by a program known as "netd" or network daemon. In each machine there is a copy of this program executing for each directly connected host. For example, in the basservax host there are two invocations of netd, one for the basser40 and one for the elecvox host. In the elecvox and basser40 there would be a netd program running to handle the link to basservax. The netd programs at each end of a given link co-operate to transfer files from one node to the other.

Netd uses a half-duplex, multi-buffered, positive acknowledgement data transfer protocol. Before each file transfer the netd programs negotiate the direction; file transfer then proceeds with short data blocks enclosed in a protocol envelope.

The header consists of the following 8 bit bytes:

```
SOH (ASCII start of header character)
sequence number
size of block
complement of size of block
```

while the trailer is:

```
longitudinal parity check
sum check
```

The sequence number is used to provide the multi-buffered flow control. Each packet must be responded to with a two byte reply consisting of an ASCII ACK or NAK character and the relevant sequence number. Errors cause

the retransmission of all un-ACKnowledged packets. Catastrophic error conditions cause a negotiation for restart of the file transmission. The error check characters were chosen as the simplest to compute and yet have a reasonable chance of detecting errors.

The netd program executing on behalf of a particular host-host link scans a designated file directory for files to be sent. The files are placed in the directory by the "net" program.

## 4. IMPLEMENTATION EXPERIENCE

The system was implemented by one of the authors (PRDL) over a period of two months with a total time spent on the project of six man weeks. It has now been operating for six months and has fulfilled the original aims of allowing closer co-operation between system maintainers and developers. Other, non-system programmer users are now beginning to use the system for such things as preparation of teaching materials.

The programs are all written in the programming language C (Kernighan and Ritchie, 1978) except for the "netsend" and "netprint" commands which are command interpreter procedures that invoke the "net" program with appropriate parameters. The number of lines of C code is just under 7000. The size of the binary code of "net" is approximately 22,000 bytes and of "netd" is 19,000 bytes. The system has been implemented on a number of different machines and versions of Unix. It is currently operational on Vax 11/780, PDP11/40, PDP11/34, PDP11/60 and PDP11/70 machines under Unix versions 6 and 7.

## 5. FURTHER WORK

While the file/mail transfer system is operational only on systems running the Unix operating system (but on a variety of different machines), there are a number of non-Unix machines that can be reached via the program. These include CDC Cyber machines at both Sydney and NSW University computing facilities. Recently a link has been established between the basser40 and a CSIRONET node machine located on the Sydney University campus. This link uses the "mx" multiplex protocol and allows several users simultaneous access to CSIRONET hosts. It also allows users connected to the CSIRONET to connect to the Sydney Unix Net.

The network will undoubtedly grow with more Unix hosts (at least three more are expected at Sydney University in 1981). The main enhancements to the network software will involve gateways into other networks such as CSIRONET and experiments with trans-network file and mail transfer. This will entail modifying or adding to the existing programs and protocol layers. For example, a trans-network mail system that uses the facilities of CSIRONET and the American Telenet is planned. This will allow mail transfer between local users and colleagues in America.

## 6. CONCLUSIONS

The Sydney Unix Network is a simple but effective system that provides a virtual circuit facility, a terminal connection system and a file/mail transfer system. The simplicity of the design meant that a comparatively small amount of effort was required to implement the system and it has proved to be very easy to maintain. At the user level it provides an effective means of co-operation between users at widely separated sites.

## 7. ACKNOWLEDGEMENTS

The design of the Sydney Unix Network software was influenced by many people. System managers at most of the host machines contributed ideas and suggestions. Especially helpful were Peter Ivanov, Kevin Hill, Andrew Hume, Chris Maltby, Chris Rowles and David Horsfall. Ian Johnston contributed to the early development of the system. The link to the CSIRONET was made possible through the efforts of John Gibbons.

## 8. REFERENCES

- KERNIGHAN, B.W., and RITCHIE, D.M. (1978): *The C Programming Language*, Prentice-Hall, Englewood Cliffs, N.J.
- RITCHIE, D.M., and THOMPSON, K. (1974): "The UNIX Time-Sharing System", *Comm. ACM*, Vol. 17, p. 365.

## BIOGRAPHICAL NOTE

Robert J. Kummerfeld received a BSc with first class honours in Computer Science from the University of Sydney in 1973 and a PhD in Computer Science from the University of Sydney in 1979. Dr Kummerfeld has been a lecturer in the Basser Department of Computer Science at Sydney University since January 1978. His research interests are computer networks, distributed processing and interactive systems.

Piers Lauder was born in Nicosia, Cyprus. He received a BSc in Engineering and Economics from Warwick University, UK in 1969, and a Post-graduate Diploma in Computer Science from Bradford University in 1973. Since 1974 he has been a programmer with the Department of Computer Science at Sydney University. His major interest is in networks.

# An Introduction to the Berkeley Network

*Eric Schmidt*

Computer Science Division  
Department of Electrical Engineering and Computer Science  
University of California, Berkeley  
Berkeley, California 94720

May 1979  
(revised March 1980, March 1981)

## ABSTRACT

This document describes the use of a network between a number of UNIX† machines on the Berkeley campus. This network can execute commands on other machines, including file transfers, sending and receiving mail, remote printing, and shell-scripts.

The network operates in a batch-request mode. Network requests are queued up at the source and sent in shortest-first order to the destination machine. To do this, the requests are forwarded through a network of interconnected machines until they arrive at their destination where they are executed. The time this requires depends on system load, inter-machine transfer speed, and quantity of data being sent.

The network enforces normal UNIX security and demands a remote account with a password for most commands. Information can be returned to the user in files, for later processing, or on the terminal for immediate viewing.

## Introduction

A network between a number of UNIX machines on the Berkeley campus has been implemented. This document is a brief introduction to the use of this network. Information which is specific to the local network has been gathered into Appendix A. The new user should read both this introduction and Appendix A in order to learn to use the network effectively.

This document is subdivided into the following sections:

---

†UNIX is a Trademark of Bell Laboratories.

### Use of the Network

- 1) Copying Files over the Network
- 2) Listing Requests in the Network Queue
- 3) Removing Requests from the Network Queue
- 4) Sending Mail over the Network
- 5) Reading Mail over the Network
- 6) Using the Lineprinter over the Network
- 7) Net Prototype Command

### Setting Defaults

How to Specify Remote Passwords

Appendix A: The Network at Berkeley

Appendix B: Getting Started — An Example

This manual is written in terms of three mythical machines, named X, Y, and Z. Specific names at Berkeley are in Appendix A, along with more local information.

### Use of the Network

The network provides facilities for issuing a command on one machine (the *local* machine) which is to be executed on another (the *remote* machine). Network commands are available to transfer files from one machine to another, to send mail to a user on a remote machine, to retrieve one's mail from a remote account, or to print a file on a remote lineprinter. These commands are described below, as is the more general *net* command which allows users to specify the name of some command or shell script to be executed on a remote machine. Network requests are queued on the local machine and sent to the remote machine, forwarded through intermediate machines if necessary.

Most of the network commands require that you have an account on the remote machine. If a remote account is not needed for a particular command, it will be noted in the following discussion. The first example introduces procedures and responses which are applicable to all network commands.

#### 1. Copying Files over the Network

Suppose that you have accounts on both the X and Y machines and that you are presently logged into the X machine. If you want to copy a file named 'file1' from your current directory on machine X to machine Y (the *remote* machine), use the command:

```
% netcp file1 Y:file1
```

The net will make a copy of 'file1' in your login directory on the Y machine. (The 'Y:' will not be part of the filename on the Y machine.) In order to verify your permission to write into the Y account, the *netcp* command will prompt you with:

```
Name (Y:your-name):
```

You should respond with your login name on the Y machine, followed by a carriage-return. If you have the same login name on both machines, just type a carriage-return. Next a password will be requested:

```
Password (Y:remote-name):
```

Now type in your password followed by a carriage-return (you must type it even if your passwords are the same on both machines). The *netcp* command will make a copy of your 'file1' in a queue destined for the Y machine, and will then return you to the shell.

Likewise if you wanted to transfer a file named 'scan.p' from Y to X,

```
% netcp Y:scan.p scan.p
```

would place that file in your current directory on X.

The network will "write" you when it has executed your request (if you are still logged in), or will "mail" you a message (if you are not). You may use the `-n` option (described later) to disallow the interruption and thus force mail to be sent. A typical message might look like this:

```
Message from Y:your-name at time ...
(command: netcp file1 Y:file1, R: 0, sent April 1 18:03, took 10 min 3 sec)
-----
```

The message includes the current time, the time you sent the command on machine X, and the exit code of the command (zero normally means success).

The network response will tell you if it was unable to execute the remote command successfully by returning an error message some time later. If, for example, you type the wrong password, you will get the response

```
Message from Y:your-name at time ...
(command: netcp file1 Y:file1, sent April 1 18:03, took 10 min 3 sec)
Error: bad login/password your-name
-----
```

The `netcp` command is actually a generalization of the UNIX `cp` command, similar to `uucp`<sup>†</sup>. Its syntax is:

```
netcp [-l login] [-p password] [-n] [-q] [-f] fromfile tofile
```

where *fromfile* and *tofile* can be local or remote files. A filename which is not a full pathname is either from the current directory on the local machine or your login directory on the remote machine. The `-l` and `-p` options may be used to specify your remote login name and password on the command line. If the password contains shell meta-characters, it must be in quotes. (These options are useful in shell scripts, but be sure to make the shell script readable only by yourself if you've got passwords in it!) The `-n` option forces any responses from the remote machine to be mailed rather than written to you. The `-f` option forces prompting for a remote user name and password, even if they are set by other options or are in the ".netrc" file (see "Setting Defaults" below). Finally, the `-q` option prevents any confirmation messages from being sent back to you, if there were no errors, the exit code of the command is zero, and the command had no output.

Transferred files may or may not have the correct file protection mode; use the `chmod` (I) command to reset it. When files are to be brought from a remote machine, they are created zero-length at the time the command is issued; when they arrive, they assume their true length. Unlike `cp`, `netcp` does not allow the *tofile* to be simplified to a directory, if the files have the same name.

Examples:

% netcp file1 Y:file1	copy 'file1' from the current directory to Y
% netcp Y:file1 file1	copy 'file1' from Y to the current directory
% netcp Z:file1 Z:file2	cp command on remote machine
% netcp X:lex.c Y:lex.c	copy from X to Y
% netcp Y:subdir/file1 file1	copy from a sub-directory
% netcp file1 file2	an error— use the <code>cp</code> command

<sup>†</sup> See the UNIX Programmers Manual (Version 7 only).

## 2. Listing Requests in the Network Queue

To see where your command is in the queue, type

```
% netq
```

A typical output of which looks like:

From	To	Len	Code	Time	Command
X:your-name	Y:remote-name	100	b99999	Mar 23 18:05	netcp file1 Y:file1

The format is similar to that of the *lpq* command. The files are sent one at a time, in the order listed. If *netq* tells you the queue is empty, your request has been sent already. The queues for different destinations are totally separate.

```
% netq Y
```

will list just the queue destined for the Y machine. *Netq* summarizes requests from other users. The command

```
% netq -a
```

will print the requests from all users.

## 3. Removing Requests from the Network Queue

If you want to cancel your net request, and "b99999" (see the *netq* example above) is your "Code," use the command

```
% netrm b99999
```

which will remove the request (if it hasn't already been sent). Furthermore,

```
% netrm -
```

will remove all your net requests in the queues on the local machine (you must have made the request in order to remove it).

## 4. Sending Mail over the Network

To send mail to remote machines, use the *mail* command with the remote account prefixed by the destination machine's name and a ":". "Y:schmidt", for example, refers to an account "schmidt" on the Y machine. The full sequence is illustrated below:

```
% mail Y:schmidt
{your message to user "schmidt" }
{control-d}
```

This will send to user "schmidt" on the Y machine the text you type in. As with intra-machine mail, the message is terminated by a control-d.

You do not need an account on a remote machine to send mail to a user there.

## 5. Reading Mail over the Network

It is also possible to read your mail on remote machines. From the X machine, the command

```
% netmail Y
```

sends a command to the Y machine to take any mail you may have and mail it back to you. As a precaution, the mail on the remote machine (Y in this example) is appended to the file

"mbox". Netmail has `-l`, `-p`, `-n` and `-f` options just like `netcp`. If a machine is not specified, the default machine† is used. If the `-q` option is specified (like `netcp`) no message is sent back if there is no mail.

Netmail also takes a `-c` option:

```
% netmail -c Y:username
```

which turns the command into a "mail check" command. A message is sent back telling the user whether the specified username has mail. No password is required. As above, the `-q` option suppresses the message if there is no mail. This command was designed to be put in C shell ".login" files.

## 6. Using the Lineprinter over the Network

Remote lineprinters can be used with the `netlpr` command:

```
netlpr [-m machine] [-c command] file1 file2 ... fileN
```

which sends the files its arguments represent to the lineprinter on *machine*. It will prompt you for an account and password. The `-l`, `-p`, `-n` and `-f` options may be supplied, as in the `netcp` command. If the `-c` option is specified, a different printing *command* (default is "lpr") can be specified; see Appendix A for the list of printers allowed. Copies of the files are not made in the remote account.

## 7. Net Prototype Command

The above commands all use internally one more general command—the `net` command which has the following form:

```
net [-m machine] [-l login] [-p password] [-r file] [-] [-n] [-q] [-f] command
```

`Net` sends the given command to a remote machine. The machine may be specified either with the `-m` option or in the ".netrc" file (for the specific names, see Appendix A). If not specified, a default is used. `-l`, `-p`, `-n`, `-q` and `-f` are as explained above for the `netcp` command. The `-r` option indicates the local *file* which will receive the output (the standard output and standard error files) of *command* when it is executed on the remote machine. By default this output is written or mailed to you. Thus, for example, to find out who is on the Y machine when you are logged in on the X machine, execute the following command:

```
% net -rn Y "who"
```

which will run the `who` command on the Y machine; the response will be written or mailed to you. Similarly,

```
% net --m Y -r resp "who"
```

will take the output (result) and return it to you in file 'resp' on the local machine. If instead you want the result of the `who` command to remain on the Y machine the command

```
% net -m Y "who >resp"
```

will create a file 'resp' in your login directory on the Y machine. It is a good idea to put the command in quotes, and it *must* be in quotes if I/O redirection (<, >, or other syntax special to the shell) is used.

If you do not specify the remote machine explicitly (or in the ".netrc" file, explained below), the default machine will be used (see Appendix A).

† (see "Setting Defaults" below)



The `--` option indicates that standard input from the local machine is to be supplied to the command executing remotely as standard input, thus if defaults for the login name and password are set up correctly as described below,

```
% net -m Y - "mail ripper"  
    { message to ripper }  
    {control-d}
```

is equivalent to

```
% mail Y:ripper  
    { message to ripper }  
    {control-d}
```

The `net` command also has other options not documented here. See the UNIX Programmer's Manual sections for more details.

### Setting Defaults

Instead of repeatedly typing frequently-needed options for every invocation of the various network commands, the user may supply in his login directory a file `".netrc"`, which contains the repeated information. The `".netrc"` file is typically used to specify login names on remote machines, as well as other options. An example of such a file is given below:

```
default Y  
machine Y, login dracula  
machine Z login dracula, quiet yes
```

This example sets the default machine to `Y` so that for `net` commands where a remote machine is not explicitly specified, the command will then be executed on the `Y` machine. The second and third lines indicate for the `Y` and `Z` machines a login name of `"dracula"` should be used to network commands, and to assume the `"quiet"` option on all commands destined for the `Z` machine. The complete list of options that may follow the machine indication is:

.netrc options for each machine			
Option	Parameter	Default	Comment
<b>login</b>	name	localname	login name for remote machine
<b>password</b>	password	(none)	password for remote login name
<b>command</b>	command	(none)	default command to be executed
<b>write</b>	yes/no	yes	if possible, write to user
<b>force</b>	yes/no	no	always prompt for name and password
<b>quiet</b>	yes/no	no	like the <code>-q</code> option

In setting up the `".netrc"` file, if the `"default"` option is present, it must be the first line of the file. The information for each machine starts with the word `"machine"` and the machine name and continues one or more lines up to another machine indication (or the end of the file). Input is free-format. Multiple spaces, tabs, newlines, and commas serve as separators between words. Double quotes (`"`) must surround passwords with blanks or special characters in them.

### How to Specify Remote Passwords

For the commands which require the password for the account on the remote machine, there are a number of ways to specify the password:

- 1) letting the command ask you, as in the *netcp* example in Section 1,
- 2) specifying it with an alias (if using the C shell),
- 3) putting it into the current environment if the local machine is running UNIX Version 7,
- 4) specifying it on the command line with the `-p` option,
- 5) storing it in the `“.netrc”` file, described in the previous section.

These can be ranked in order of security, from 1 = greatest security to 5 = lowest security, from the point of view of security of passwords from unauthorized use by other users and possibly an illicit super-user. Each is described in turn:

- 1) If you make no effort to specify the remote password elsewhere, the network commands will prompt you with:

Password(mach:username):

Type your password, followed by a carriage return. This is the most secure mode of specifying passwords. If the net command is executed in the background (i.e. with `“&”`) then the command can't read the password from your terminal and one of options 2-5 below must be used.

- 2) The alias feature of the C shell can be used to specify the remote password. The command

`% alias netcp netcp -l godzilla -p $pass`

in the `“.cshrc”` file, followed by

`% set path=your-passwd`

right before using the network will set for subsequent *netcp* commands the login name `“godzilla”` and password `“passwd”`. This alias command must be given everytime you login (see the UNIX Programmers Manual section for the C shell (csh (1)) for more information about *alias*. Do *not* put this alias command in your `“.login”` file.

- 3) If running on a Version 7 UNIX system, the password can be put in the current environment. The command (to the C shell)

`% setenv MACHmch `netlogin -m mch``

or (to the default Version 7 `“Bourne”` shell)

`% MACHmch=`netlogin -m mch`  
% export MACHmch`

will prompt you for a login name and password for the remote machine *mch* and put an encrypted copy of the password in your environment. (Note the back-quotes to the shell.) Subsequent network commands will find it in your environment and not prompt you for it. These encrypted passwords are invalidated after the user logs out. Type `“man netlogin”` for more information on the *netlogin* command.

- 4) Each net command takes a `-p` option on the command line to specify the password. These are usually put in shell command scripts. These shell script files should have file mode 0600 — use the `chmod(1)` command to set the mode.
- 5) The remote password can be specified in the user's `“.netrc”` file. If passwords are present, the `“.netrc”` file must have mode 0600 (as in #4 above).

The system managers recommend options 1-3 and warn against 4 and 5. Should someone break into your account on one machine, and you use option 4 or 5, you will have to change your passwords on all net machines for which your passwords have been stored in shell script files or in the `“.netrc”` file.

### Log File

The file `"/usr/spool/berknet/logfile"` has a record of the most recent requests and responses, each line of which is dated. Lines indicating "sent" show the file name sent; lines indicating "rcv" show commands executed on the local machine (C: ), their return code (R: ), and their originator. For example, on the Y machine, the logfile:

```
Feb 28 10:29: rcv X: neil (neil) R: 0 C: netcp design Y:design
Feb 28 10:43: sent tuck to Z (z00466, 136 bytes, wait 2 min 3 sec)
Feb 28 11:05: rcv X: bill (bill) R: 0 C: netcp structures Y:structures
```

shows three entries. In this example, there are two *netcp* commands sending files from the X machine to Y, each from a different user. The second command sent was originated here by "tuck" and is 136 bytes long; the command that was sent is not shown. The command

```
% netlog
```

will print the last few lines of this file. Its prototype is

```
netlog -num
```

where *num* is an integer will print the last *num* lines from the file.

### Acknowledgements

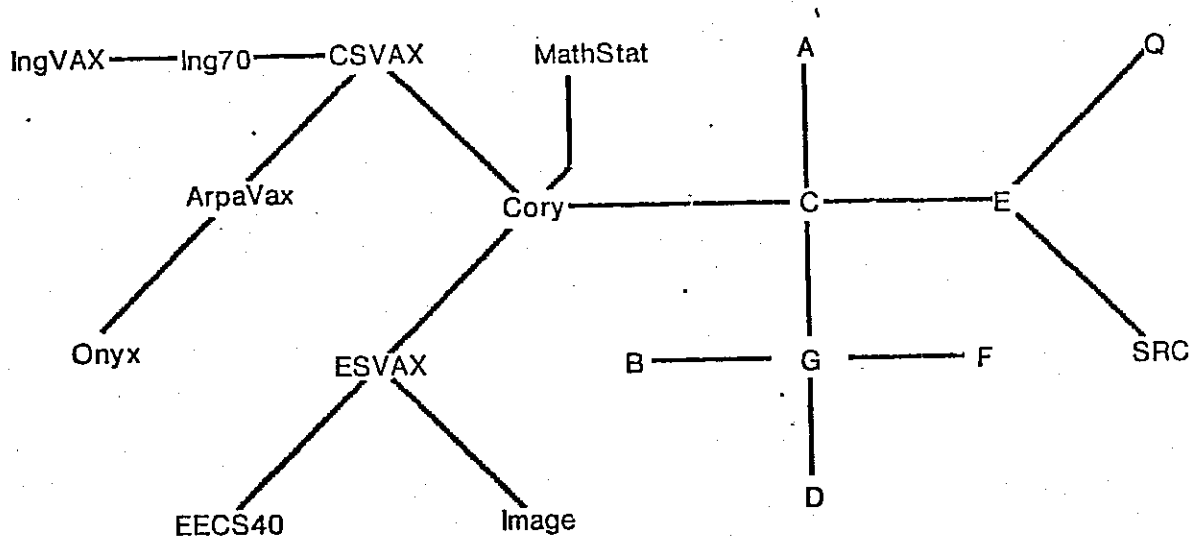
Special thanks go to Bob Fabry, Bill Joy, Vance Vaughan, Ed Gould, Robyn Allsman, Bob Kridle, Jeff Schriebman, Kirk Thege and Ricki Blau of Berkeley, and Dave Boggs of XEROX PARC for their help in making this network possible.

Appendix A

The Network at Berkeley

1. The Configuration (March 1, 1980)

The Current State of the Berkeley UNIX Network			
Machine Name	Internal Name	Run By	Default Machine
A	A	Computer Center	C
B	B	Computer Center	D
C	C	Computer Center	A
D	D	Computer Center	C
E	E	Computer Center	C
F	F	Computer Center	E
G	G	Computer Center	E
Ing70	I	INGRES Group	IngVAX
IngVAX	J	INGRES Group	Ing70
Image	M	Signal Processing	ESVAX
ESVAX	O	EE-CE Research	CSVAX
Q	Q	Computer Center	E
ArpaVax	R	Fabry's Arpa Project	CSVAX
SRC	S	Survey Res. Cent.	D
MathStat	T	Math-Stat Dept.	CSVAX
CSVAX	V	CS Research	Cory
Onyx	X	CS Research	ArpaVax
Cory	Y	EECS Dept.	CSVAX
EECS40	Z	EECS Dept.	ESVAX



If a path exists from the local machine to the requested remote machine, the network will forward the request to the correct machine. Thus Cory users may communicate with all the other machines on the network as well as C and CSVAX (with a degradation in speed because of the intermediate machine(s)). The links between Ing70-IngVAX, CSVAX-ArpaVAX, A-C, C-D, C-E, B-D, E-F, and E-G run at 9600 Baud, the other links run at 1200 Baud.

## 2. Documentation

The network commands (*net*, *netq*, *netrm*, *netlog*, *netcp*, *netmail*, *netlpr*, *netlogin*) are all documented in the UNIX Programmers Manual. For example,

```
% man netq
```

will print the *netq* manual section.

There are two more documents available:

```
Network System Manual
Berkeley Network Retrospective
```

The Manual is intended for the systems staff who will maintain the network. The Retrospective is my Master's report and details the history of the project, discusses the design, and lists future plans.

There is an up-to-date news file:

```
% news net
```

or

```
% help net
```

or

```
% cat /usr/net/news {if those fail}
```

which prints news about the network, dated and with the most recent news first.

The UNIX Programmer's Manual, section I, has information on the *chmod*, *cp*, *mail*, *who*, and *write* commands mentioned in the text. Also, the *help* command has information about file protections:

```
% news access {on the Cory machine}
```

or

```
% help permissions {on the CC machines}
```

## 3. Features at Berkeley

- a) There is a built-in character limit of 500,000 characters per single transmission, which cannot be overridden. Files larger than 200,000 bytes will be queued for transmission between Midnight and 6AM. Longer files must be split into smaller ones in order to be sent.
- b) The 1200 Baud links between machines seldom transmit any faster than 80 characters per second (for 9600 Baud links, 600 characters a second), and can slow to a fraction of that in peak system loading periods. This is due to an expansion of the data packets to accommodate a seven-bit data path, wakeup time on the machines, and the packet sent in acknowledgement. Heavy file transfer is faster by magnetic tape.
- c) On the VAX systems the net commands are all in '/usr/ucb'. Your search path on these VAX's should be set to include the directory '/usr/ucb'; otherwise you will have to prefix all net commands by '/usr/ucb', as in '/usr/ucb/netcp'.
- d) Limited Free Commands

Users who do not have accounts on remote machines may still execute certain commands by giving a remote login name of "network", and no remote password. The commands currently allowed are:

bpq	netlog	rsc	vpq	whom
epq	netq	rsclog	w	write
finger	ps	rscq	where	yank
lpq	pstat	trq	who	

The *lpr* command is allowed on the INGRES machine. Also, *mail* to remote machines and *netlpr* between Computer Center machines do not require a remote account. The EECS40 machine allows no free commands (but allows the sending of mail).

For example, to execute an *lpq* command on the A machine, the user would type:

```
% net -l network -m A "who"
```

- e) If no machine name specification is in the front of a full path name, the first four characters are checked and the machine is inferred from that if possible. In the command

```
% netcp file1 /ca/schmidt/file1
```

the second file name is equivalent to "C:file1", if you are "schmidt" on the C machine.

- f) The network can only send files in one direction at a time. Thus confirmations can slow down heavy file transfer. If you regularly use a shell script to transfer a set of files, the *-q* option to *netcp* will improve transfer time.
- g) The network creates a heavy load on the system and thus is expensive to run. If general user throughput is adversely affected, a charge will be implemented on the Computer Center machines.
- h) When transferring files, quota overflow will result in a partial copy, so you should check the space requirements of the file being sent.
- i) The Computer Center "A" machine's phototypesetter is usable from other network machines. If on one of the B-E machines, you do not need an account on the A machine. You simply type

```
% troff -Q other-options file(s)
```

instead of the normal

```
% troff other-options file(s)
```

The *troff* command is executed on the local machine and the phototypesetter instructions are sent to the A machine. You will be sent mail both when the file is queued and when it is finally typeset. To see your place in the *troff* queue, type:

```
% trq
```

on any Computer Center machine. There is a command

```
% trrm code
```

(where *code* is the code from the *trq* command) to remove queue files before they have been typeset. *Trrm* must be executed on the same machine from which the job was submitted.

If you are on a non-Computer Center machine, you may use the *nettroff* command:

```
% nettroff options file(s)
```

which is similar to the "troff -Q" command earlier. You will need an account on the A machine and the *trrm* command doesn't work from a non-Computer Center machine.

If using *nettroff*, no more than 15 pages may be sent to the typesetter. If using *troff* more than 15 pages may be sent only if the *-s* option is specified (see *troff(1)* for more information). The network will not transfer any file longer than 500,000 characters to the A machine. (It is best to aim for files of 25,000 characters or less)†. For more

† Characters from *troff's* output, not the user's source files. It is our general experience that *troff* outputs roughly twice as many characters as are in the source file (before any *eqn* or *tbl* preprocessing.)

information, type

man troff (on the Computer Center machines)

or

man nettroff (on the other machines)

The *nettroff* command is not supported by the Computer Center:

- j) The *netlpr* command allows "epr", "bpr", and "vpr" as alternate lineprinters (using the *-c* option).

#### 4. Bugs in systems at Berkeley (As of March 1, 1980)

- a) If you set the variable *time* in the C shell, extraneous time stamps may appear in response messages. The correct way to set the variable *time* in the C shell is

```
if ($?prompt) then
    set time = num
endif
```

where *num* is the time interval in seconds.

- b) The file mode should be preserved by *netcp* and it should be possible to default the second file name to a directory as in *cp(1)*.
- c) Various response messages are lost. This includes "fetching" files when the file being retrieved never arrives. I suspect this has something to do with unreliable delivery of error messages, but this is not reliably reproducible.
- d) The network makes no provision for errors in transit on intermediate machines, such as "No more processes" or "File System Overflow". While these occur only rarely, when they do, no message or notification is sent to anyone.
- e) The network commands are too slow on heavily-loaded instructional machines. The *net* command has to read the password file, ".netrc" file and the "/etc/utmp" file.
- f) The queue files are normally sent shortest-job first. Unfortunately, under heavy loading the queue-search becomes too expensive and the network will choose the next file to send from the first 35 queue entries it finds in the queue directory, so the user should not depend on the requests being sent shortest-first.
- g) Comments and bug discoveries are encouraged and can be sent by local or remote mail to "csvax:schmidt".

From kev Tue Mar 31 21:26:40 1981 netmail from elec70

Pete: I have just found out that cpio is MUCH worse than even I had feared! Forgetting for a moment its lack of a tape directory that means it must read the entire tape to find something, and forgetting that it goes at a speed approaching absolute zero, you have only to create a zero-length file called "TRAILER!!!" to completely stuff it. It uses this to indicate the end of the dump you see, and cannot differentiate between a normal cretin's file called this, and the real McCoy that it sticks on itself. Are these idiots allowed to breed over there? Does Bell double-up as the local mental asylum?

kev

From mhtsa!ianj Thu Jul 9 16:32:44 1981 remote from usa netmail from basser40  
this arrived for you

I pass it on without comment

you can respond via

mhtsa!ihuxa!pur-ee!mike

ian

>From uucp Thu Jul 9 01:05:17 1981

>From \*\*RJE\*\* Wed Jul 8 20:35:55 1981 remote from ihuxa.

>From mike Wed Jul 8 14:19:36 1981 remote from pur-ee

To: ihuxa!mhtsa!ianj

For Peter Ivanov

To enable an 11/44 to work as a unibus terminator on the vax there are 2 things which must be done.

1. Modify the m7094 board so that addresses from 64k to 128k appear to be in the I/O page. To do this, see the last page of the m7094 prints. Insert a 74S05 in the spare socket location E124. Connect pin 1 of the 74S05 to pin 9 of the 74S373 at E93. Connect pin 2 of the 74S05 to pin 8 of the 74S22 at E83. Without these modifications the 11/44 cannot talk to the VAX memory.

2. You can only put 64k of memory on the 11/44, as memory on the 11/44 ties up UBA addressing space for the buffered data paths. To do this, take the 128kw memory which comes with the 11/44 and make it look like it has 64kw by inserting jumper W2 on the M8722 memory board. If you somehow managed to get a 64kw (128kbyte) board from DEC you won't have to do this.

Plug a unibus cable into the last slot in the 11/44 and connect to the UBA where the arbitrator normally resides (M9044 card). The 11/44 can be halted and the vax unibus will still work.

From lindsay Tue Aug 18 09:44:32 1981 netmail from agsm

Wanted:

A part time programmer to convert a Fortran program, developed on an IBM 370 system, to use f77. Initial work will be on AGSM's 11/70, but completion will be done on a VAX (with UNIX). The program concerned solves linear programming & integer programming problems. Estimated conversion time is about 2 weeks.

Contact: Lindsay Harris, rm423 at AGSM - Ext 273.



From andrew Sun Aug 23 14:44:21 1981 remote from usa netmail from basser40  
DSW(1) UNIX Programmer's Manual DSW(1)

NAME

dsw - delete from switches

SYNOPSIS

(put number in console switches)  
dsw  
core

DESCRIPTION

dsw reads the console switches to obtain a number n, prints the name of the n-th file in the current directory, and exits, leaving a core image file named core. If this core file is executed, the file whose name was last printed is unlinked (see unlink(2)).

The command is useful for deleting files whose names are difficult to type.

SEE ALSO

rm(1), unlink(2)

BUGS

This command was written in 2 minutes to delete a particular file that managed to get an 0200 bit in its name. It should work by printing the name of each file in a specified directory and requesting a 'y' or 'n' answer. Better, it should be an option of rm(1).

The name is mnemonic, but likely to cause trouble in the future.

Printed 8/11/81

PDP-7 local

1

From kev Sun Sep 20 20:02:27 1981  
Subject: panic!

You may be interested in a method proposed by richardg for forcing the vax to panic (and successfully tried by myself). The old odd-address-in-pc won't work of course, because the vax considers instructions on an odd boundary a real challenge, and leaps in there with all boilers going. The method is to halt the cpu, deposit a non-existent virtual address in pc (such as 60000000) and continue it:

H  
D PC 60000000  
C

and whammo! one dead (but nicely panic'd) vax.

kev

From kev Sat Sep 19 13:41:50 1981 netmail from elec70

Pete: some light reading:

From 8037185 Mon Sep 14 13:22:49 1981

Subject: There is an unwanted file of mine named 'mastermind' sitting

in the ^/ , (is this root?), directory. I don't know how it got there, maybe some misshap while I was transferring it to another of my own directories, but I don't want it. I tried removing it but that didn't work. Can you refer me to some type of wizzard.

/user1/6.641/6.641-1/8037185

At this point I removed the file and mailed this chap for more information. His reply was:

From 8037185 Tue Sep 15 09:18:03 1981

Subject: About that file in /.

It was originally mailed to me by another user on the 70, (thus it was in mail). I wanted to transfer it to my directory "stuff" direct from the mail file. I think i typed "s stuff/mastermind", which did place the mastermind file in "stuff".

That was last Friday. I didn't suspect that there was a duplicate file in / until i spotted it while i was jumping around directories yesterday, (Monday).

Although i could not remove the file, or change its name, i could edit it, so i reduced it size from about 2000 to 2.

Thats about it.

user1/6.641/6.641-1/8037185  
j.mitchell.

My reply to his reply was:

Subject: That file in /

Thank you for your assistance in this matter. Plainly it represents a gross breach of security that you were able to create a file in /, so we will now devote considerable energy to putting an ice-pick through "mail". In the meantime, we would appreciate it if you would remain silent on the matter.

kev

My opinion is that I am fed up to the back teeth with these half-hearted ALL-POWERFUL programs that are plainly written by incompetents and half-wits. I think Michael should spend next week looking very carefully at mail, and clean the rotten pile of worm-infestation up. I can understand now why other places leave mail non-blessed, and simply demand that you leave a writable-by-others mail file around. The point is, if it wasn't for all the useless bloody gimmicks, it COULD be done properly (as Steven Fraser proved under level 6). With a little bit of care and attention, and one-tenth of a brain, it should be possible again.

PS: I won't even bother to voice an opinion on Mail.

PPS: you can feel free to publish this!

PPPS: no reply from bobk yet on his mail.

PPPPS: today, for example, I only had to remove one (1) root-owned mbox, and one (1) root-owned .mail.lock. Wonderful! Things are really improving!

# EUUG

EUROPEAN UNIX USER GROUP

## COMMITTEE

Chairman : Alan Mason, Heriot-Watt University  
Editor : Bruce Anderson, University of Essex  
Member(s) : Peter Collinson, University of Kent

R.A. Mason  
Computer Engineering  
Heriot-Watt University  
Mountbatten Building  
31-35 Grassmarket  
Edinburgh EH1 2HT  
(Tel. 031-225-8432 x 155)

12th June, 1981

Peter Ivanov,  
Editor AUUGN,  
Dept. of Computer Science,  
P.O. Box 1,  
Kensington 2033,  
Australia.

Dear Peter,

I just received a copy of AUUGN where you announced the 'UNIX V7-small machine' distribution that I sent you. I felt that I should point out that this is not a 'Heriot-Watt' product, but truly an EUUG product. Although a number of individuals from my installation were involved (notably Jim McKie), so were many from other sites. Heriot-Watt is simply now acting as a distribution centre for the package and for collecting fixes as they come in (23 pages now, most of which are required in the original Bell version).

The people actually involved with the package were:

Hugh Conner - Heriot-Watt University, Elect. Eng.  
Alan Mason - Heriot-Watt University, Computer Eng.  
Jim McKie - Edinburgh Regional Computing Centre.  
Zdravko Podolski - Glasgow University, Computing Science.  
Colin Prosser - Science Research Council, Computing Division.  
Dave Rosenthal - Edinburgh University, Dept. of Architecture.

and many have since carried on to develop it further. To be announced in the near future (July/August) is a second tape with lots of other goodies:

User & Kernel Overlays (i.e. F77 & Lint will work)

Terminal Description Libraries

Graphics Package

New & Reworked Utilities

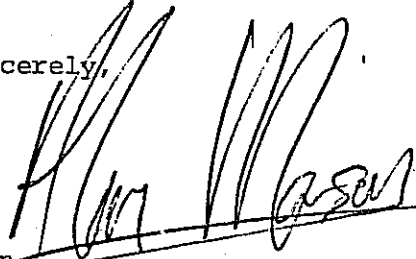
Contiguous File Handling

Ongoing and planned for distribution next year is a system which is so removed from the original that we might call it V8, but more likely V7+.

The point I am making is that no one individual or site is responsible for these developments, and the credit (or otherwise) should be given to the Group as a whole.

In all fairness, it is really my fault for misleading you. I should never have had our site named on the packaging documents, but that was where I wanted the BUG reports to come.

Yours sincerely,

A handwritten signature in black ink, appearing to read 'R.A. Mason', written over a horizontal line.

R.A. Mason

# EUUG

EUROPEAN UNIX USER GROUP

R.A.Mason  
Computer Engineering  
Heriot-Watt University  
Mountbatten Building  
31-35 Grassmarket  
Edinburgh EH1 2HT  
(Tel. 031-225-8432 x 155)

20th July 1981.

Dear *Peter,*

The EUUG have recently had an election for the period 1981-1982 and formed the committee:

Chairman	:	Alan Mason
Secretary	:	Peter Collinson
Treasurer	:	Emrys Jones
Membership Officer	:	Hugh Connor
Executive Editor	:	Jim McKie
Editors	:	Bruce Anderson Cornelia Boldyreff
Member	:	Nigel Martin Mike Banahan

As you will see from this list, we now have three editors! This is an attempt to push up the frequency of our newsletter. Two of these individuals (newsletter editors) are in point of fact responsible for the physical collation, printing, publishing and distribution of the newsletter. The first (executive editor) is responsible for the gathering of material and re-distribution to newsletter editors and the remaining committee. It is for this reason that I now write. Could you in future please send our exchange copies of your newsletters, meeting announcements, meeting reports etc to our Executive Editor:

Jim McKie,  
Executive Editor, EUUGN,  
Unix Support Officer,  
Edinburgh Regional Computing Centre,  
c/o EdCAAD,  
20 Chambers Street,  
EDINBURGH,  
Scotland.

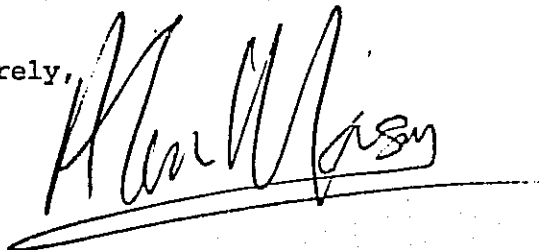
I would further appreciate it if you could ensure that members of your committee and meeting/conference chairpersons/hosts are made aware of this policy change so that we can be assured that most publishable material is coming through the one channel.

As in the past I myself will continue to receive your software offerings and to carry out local re-distribution of them, to those of you that handle software distribution on a per-meeting basis, I would appreciate it if the meeting organisers could be kept informed of our exchange agreement.

As a final request, if any of your members are intending European visits, I would suggest that they make contact with me giving a short note of their interests. This done, I can suggest to them a number of sites that they may wish to visit, and if the timing is right, invite them to attend local and/or European meetings.

Yours sincerely,

R.A. Mason

A handwritten signature in dark ink, appearing to read 'R.A. Mason', is written over a horizontal line. The signature is cursive and somewhat stylized.

cc: USENIX  
USR/GROUP  
UNI-OPS  
AUG  
CUUG

# uni-ops

Walter Zintz/ Uni-Ops

Post Office Box 5182, Walnut Creek, California 94596 USA

260 Marshall Drive, Walnut Creek, California 94598 USA

Telephone (415) 933-8564 mornings

Peter Ivanov / Computer Science  
Electrical Engineering  
University of New South Wales  
Post Office Box 1  
Kensington 2033, Australia

number: 3007

date: 16 JUL 81

I hope you have received at least the first issue of the Uni-Ops newsletter, Pipes & Filters, which has been airmailed to you long since. If not, let me introduce Uni-Ops as a users' group for Unix and C in business and personal computing.

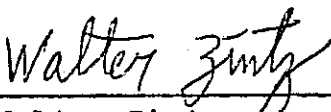
The first major Uni-Ops project is to compile the first worldwide mailing list of Unix/C people of all sorts. This is intended to benefit both those who want to mail to such a list and those who want to receive information by mail on the latest products and services in the field. My compilation method is simply to proclaim that anyone who believes his name should be on this list should send Uni-Ops his name and address. I'm asking for a mention in your newsletter to spread the message in your part of the world.

To help the chary reader make up his mind, Uni-Ops has adopted two firm policies. First, NO NAMES WILL BE GIVEN OUT, EVER. All mailers must ship their materials to Uni-Ops to be addressed and mailed, so even list users will never actually see the list. Second, use of the list (to the extent provided above) will be open equally to everyone whose material concerns C or Unix. There will even be frequent co-op mailings for those who must hold their mailing costs to rock-bottom levels. For overseas mailers' benefit, I can supply contacts with local printers whose work I can vouch for.

To simplify our clerical work, each organization, or department within a large organization, should send a single list of all its Unix/C personnel if possible—Organization name and mail address at the top of the sheet, followed by the names (and mail codes, if necessary) of the individuals at that address. Typewritten lists are preferred; print-out or hand-lettering must be effortlessly legible. All list contributions should be mailed to:

Uni-Ops List  
Post Office Box 5182  
Walnut Creek, California 94596 USA

Inquiries regarding list rental, co-op mailings or local printers should be sent to Uni-Ops List Manager at the same address.

  
Walter Zintz

# uni-ops

Post Office Box 5182  
Walnut Creek, California 94596 USA  
(415) 933-8564 mornings

---

Dear Computer Professional:

Bell Labs' Unix<sup>®</sup>, Software Tools, other equivalent operating systems: do you use them or sell them for commercial applications? Then let me tell you about Uni-Ops, the active new user/vendor group for people involved with Unitory operating systems on business's computers.

Uni-Ops is people talking to people—in print and face to face—about the opportunities and obstacles that arise as the Unitory family becomes the first de facto universal operating system. Talking about Unitory itself, the hardware it runs on, the business programming that runs on it, C and the other languages used with it. Talking about technical matters, business aspects and personal involvement.

Our primary communications channels are a monthly journal, a semi-annual directory of Unitory products, and national conferences starting this Fall—all described below.

Every member receives 12 issues a year of Pipes & Filters, the Uni-Ops journal, featuring news of the Unitory field and the people in it, bulletins and reviews of new or revised products, articles (some technical, all practical), notes on Uni-Ops activities, and pithy letters from members. Classified ads are available under Software, Hardware, Employment, Services and Education headings, at \$7 for up to 100 characters and spaces combined, 5¢ for each extra character or space, payment with order. Preprinted advertising material can be mailed in the envelopes with Pipes & Filters for \$70 per 8½x11 page plus \$35 per piece. The first issue of Pipes & Filters will be mailed to members on June 4th. All editorial and advertising material intended for that issue must reach Uni-Ops not later than May 22nd.



The Uni-Ops Directory is a cataloging of all known Unitory-based software that could be useful in commercial systems, whether offered by vendors or co-op groups. Data tables are arranged by software type and cross-indexed by supplier. There is no charge for being listed, and listing forms will be mailed June 11th to everyone who has asked for them. Deadline for return of completed forms is July 17th, because the first edition of the Directory will be sent to all members on August 6th.

Face to face interaction comes at Uni-Ops first national meeting, being planned for the week beginning October 11th, in San Francisco. (But not at a high-rise hotel in the downtown ratrace—our site is just beyond, in exquisite Cow Hollow.) The technical and professional sessions will emphasize participation rather than chairwarming, and vendor activities are being structured to provide more information and less exhaustion for vendors and visitors alike. Careful simultaneous-activities programming should provide everyone with something of interest most of the time. Uni-Ops members will be the first to learn the specifics of our conferences, through Pipes & Filters, and are entitled to substantial reductions in registration fees.

Of course we're already receiving requests for Uni-Ops involvement in other Unitory matters. Some of the more general topics of interest seem to be:

- standardization in Unitory and C
- renegotiation of license terms
- regional meetings and newsletters
- special-interest subgroups
- programmer training and certification
- promoting Unitory to the outside world

Uni-Ops will be active on these things, co-operating with Usenix and STUG where logical, as member energy permits.

Uni-Ops membership is open to any interested person at \$24 per year; there are no corporate or institutional categories. If all this sounds like your kind of group, then welcome aboard!

*Walter Zintz*

Walter Zintz  
Executive Director

UNI-OPS MEMBERSHIP APPLICATION

Please enroll me as a charter member of Uni-Ops for one year, ending the 31st of May 1982. My dues payment of US\$24.00 is enclosed.

YOUR  
NAME  
AND  
COMPLETE  
MAILING  
ADDRESS

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

If you will jot down your specific Unitory-related interests on any or all of the topics below, it will help Uni-Ops plan its activities.

Which class of Unitory? ( ) Bell Labs' Unix ( ) Software Tools ( ) other  
Commercial Applications Areas \_\_\_\_\_

Applications Languages \_\_\_\_\_

Computer Size/Type \_\_\_\_\_

Specific Computers or CPUs \_\_\_\_\_

Peripherals \_\_\_\_\_

Database Systems \_\_\_\_\_

Networking \_\_\_\_\_

Graphics \_\_\_\_\_

Standardization \_\_\_\_\_

Licensing \_\_\_\_\_

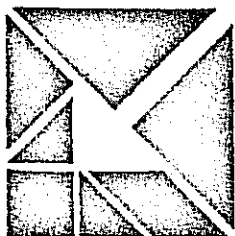
Personnel \_\_\_\_\_

Market Development \_\_\_\_\_

Career Advancement \_\_\_\_\_

other topics \_\_\_\_\_

.....  
UNI-OPS, Post Office Box 5182, Walnut Creek, California 94596 USA



the Time-Machine Ltd.

ת"מל ת"מל-0"0

Peter Ivanov  
Dept. of Computer Science  
University of NSW  
POB 1  
Kensington  
New South Wales 2033  
Australia

Dear Mr. Ivanov,

Thanks for the reply to my questions about the U200 emulator.  
It solved my problem quicker than I thought.

We are considering here a few ideas about UNIX-based systems  
and products. Since you are setting up the software catalogue,  
you probably are the best source of references to the relevant  
people. Three items are now on our "hot list".

1) Real-time UNIX for process control, fast data acquisition  
and number crunching, etc.

2) An ADA compiler / interpreter / whatever under UNIX (hope-  
fully a portable one).

3) A "C" compiler (or even an entire UNIX system) for a Data  
General computer (Eclipse...).

If you happen to know of anyone working on such items, please  
let me know. I also welcome any direct contact from readers of  
this letter.

Sincerely yours,

Gershon Shamay  
Manager, Software Development

April 29, 1981

Peter Ivanov  
Dept. of Comp. Sci.  
University of New South Wales  
P.O. box 1  
Kensington NSW 2033

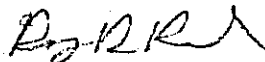
Dear Peter,

Could you please enter my subscription to the Unix User Group Newsletter. I am including a copy of my license and the subscription fee.

I am the guru for an 11/34 system running level 6 UNIX in the School of Electrical Engineering at Sydney University. This system, which is known on the UNSW-SU network as "mhd", is used primarily for numerical modeling of problems related to MagnetHydroDynamic(MHD) power generation. My "mail" address on the network is roy:mhd.

I also maintain an 11/03 system which runs MINIUNIX and is normally linked to the 11/34 via a serial line. The 11/03 system is a transportable data acquisition system which includes a dual AED 6400 floppy disc, a Matrox graphical display, and a High Common-mode Voltage measurement system of our own design. For experiments, the 11/03 is physically transported to our experimental site at White Bay Power Station and is returned at the end of the day.

Yours truly,



Roy R. Rankin  
School of Electrical Engineering  
University of Sydney  
Sydney, 2006



*The University of Western Australia*

Faculty of Architecture  
Nedlands, Western Australia 6009  
Telegrams Uniwest Perth, Telex AA9299z  
Telephone (09) 380 2582

Ref.

25th June 1981

Mr. P. Ivanov,  
Department of Computer Science,  
Electrical Engineering,  
University of New South Wales,  
P.O. Box 1,  
KENSINGTON N.S.W. 2030

Dear Sir,

I have been given your name as contact for the UNIX Users Group. I would be grateful if you could give me any information on the availability of UNIX for a PDP11/24 system with 256 KB memory and dual RLO2's. In particular I wish to discover any hardware requirements of UNIX version 6 and/or 7. My main interest is to establish what are the size limits for a FORTRAN program, i.e., whether it is possible to utilize memory outside a single 64K byte page from a single program.

I look forward to your early reply.

Yours faithfully,

(DR.) G.G. ROY

*Answered by phone  
16817th 7/81*

# UNIX 操作系统综述

丁茂顺

(中国科学院计算技术研究所)

## 一、引言

UNIX是由美国贝尔实验室研制出来的一个通用的、交互型的分时操作系统。其设计者是K-Thompson和D-M-Ritchie, 他们两人于1969—1970年花一年的时间就搞出来了。开始是用汇编语言写的, 1973年又用C语言重新写一遍, 总共约10000行代码(其中约1000行是汇编代码)。事实上, 这部分程序只是系统的内核。以这个内核为基础进行扩充, 陆续实现了许多不同用途的各种UNIX系统版本。利用C语言的可移植性, 实现了UNIX系统的移植。目前, UNIX系统已经在PDP-11系列、Interdata8/32、Honeywell6000等许多种机器上运行。

UNIX系统以其自身独特的优越性取得了巨大的、出人意料的成功! 截至1978年底为止, 已经有1000个UNIX系统散布于世界各地。UNIX系统所提供的分时系统软件比所有其它系统所提供的相应软件的总和还要多。

总效果和其中每条命令(工具)的目标差距甚远。为了说明这一点, 再举一例:

```
who|grep tom|wc
```

该管道线的功能是断定名字叫tom的用户在系统里到底登记多少次——这是一个组合效果(筛选), 初看起来, 这一效果和行中每一命令各自的功能真是大不相同(参看3.2)。

### 3.4 使用 shell 过程进行程序设计

任何目录所链接(统计数=0), 则释放该索引节点, 从而对应的文件也就被删除掉了。

#### 4.4 shell 的实现

shell 解释程序绝大部分时间都在等待着用户打入命令。当一个命令行打好时, shell就分析这个命令行, 同时把诸参数以适合于execut<sub>e</sub>原语的格式准备好。然后执行fork原语。这时候, 父子进程分叉, 新建的子进程要去完成一个execut<sub>e</sub>原语, 使用父进程为其准备好的程序和参数, 这就使得该命令的要求得以实

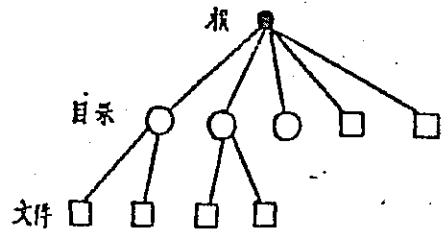


图1 文件系统的层次结构

现的重要部分。从用户的观点看, UNIX文件分为三大类: 普通文件、目录文件和特殊文件。

普通文件就是通常意义下的盘文件, 用于存放用户的程序和数据等信息。

目录文件即指文件系统各个目录。这种文件具有两重性: 它在形式上和普通文件一样, 但系统将其内容解释成目录。既然目录文件是文件, 那么用户就可以象读普通文件那样去读目录文件了, 例如用列表打印程序把指定的目录打印出来! 另一方面, 系统又为自己保留着可以改变目录文件内容的权利。在目前流行的其它操作系统里, 常常把印出目录列表等

统里属于系统内核(即UNIX操作系统)。Shell虽然在级别的划分上不属于系统程序, 但是实际上它亦是UNIX系统当中不可缺少的一部分, 因为如果没有shell就无法使用UNIX系统。以下我们就按这四个方面简略地描述一下UNIX系统的实现思想。

### 4.1 进程控制

#### 4.1.1 数据结构

内核。系统内核小而简单, 只有10000行左右的代码, 但对系统资源的管理能力并不比别的系统弱。许多在别的系统里属于“系统”一级的程序, 在UNIX系统中都属于用户一级, 但是它们在功能上仍然是系统的一部分。这样搞, 为系统功能的修改和扩充, 为系统的可移植, 提供了有利的条件。

文件系统。采用树形层次结构, 并且把目录也当成文件来处理的做法, 实现了对文件的系统化的管理。设立特殊文件, 使得外部设备

# A Summary of the UNIX Operating System

Din Mao-shun

Research Institute of Computing Technique  
Chinese Academy of Sciences

## Introduction

UNIX is a general-purpose, interactive, time-sharing operating system developed by Bell Laboratories in the U.S.A. The designers, K. Thompson and D.M. Ritchie, implemented the system in 1969-70. Initially UNIX was written in assembler language, but was rewritten in the C language in 1973. About 10,000 lines of code (1,000 still in assembler) form the kernel of the system.

A variety of editions of UNIX for different applications have been brought out with the kernel as the base for extensions. Because of the portability of the C language the UNIX system has been transferred to several machines. The UNIX system is now running on the PDP-11 series, Interdata 8/32, Honeywell 6000, etc.

The UNIX system has achieved great and unexpected success for its superiority over other systems. By the end of 1978 there were 1,000 UNIX systems all over the world. The time-sharing system software provided by UNIX is even more than the sum of the relevant software provided by all other systems. (direct translation - ed)

Fig. 1. illustrates the levels of the file system in UNIX from 'root' to 'directory' to 'file'. UNIX files consist of three categories: common files, directories and special files.

.  
. .  
. . .  
. . . .  
. . . . .

pipe line: 'who | grep tom | wc' is a pipe line used to find how many times the user 'tom' is logged-in to the system. The command 'grep' selects lines from the output of 'who' containing the name 'tom' and these lines are counted by 'wc'.

## 3.4 Programming Using SHELL

### 4.1 Process Control

#### 4.1.1 Data Structure

.  
. .  
. . .  
. . . .  
. . . . .

## 4.4 Implementation of the SHELL

The command interpreter, or SHELL, is waiting for user commands at most times. On receiving a command the SHELL analyses it and makes ready all the

parameters suitable for the format of the primitive "execute", then executes the "fork" primitive. At this time father and son procedures go different ways: the new-built son-procedure implements the "execute" primitive using the code and parameters provided by the father-procedure.

The kernel, although it is small and simple, only about 10,000 lines of code altogether, has no less ability to manage system resources than other systems.

Many programs which are at system level in other systems are at the user level in UNIX, though they still are part of the system from a functional point of view. This is to the advantage of revising and extending functions of the system, as well as for portability of the system.

The above was translated from an article I saw in the "CSIRO DMS UNIX News Sheet" about the spread of UNIX to China. The DMS people had not translated it so I asked one of our Chinese exchange staff to have a go.

peteri



A very large loosely-coupled computer system(HITAC M-200H)  
at the University of Tokyo Computer Centre

1. The largest computer system in the world  
being shared by many universities

Computing facilities at the Computer Centre are being used on a shared basis by some 5,000 researchers at many universities mainly in Tokyo area. To fill these users' needs, the Computer Centre has installing what is believed to be the largest computer system in the world. The system is built around one of the fastest general-purpose processors HITAC M-200H and consists of the following components.

- (1) 8 (M-200H) processors and 64 megabytes (MB) of main storage
- (2) 6 IAP (Integrated Array Processors) for all but global processors.
- (3) 8 drums (each 15 MB, total 120 MB) for the TSS system. There are about 60 terminals installed in the centre building.
- (4) 96 spindles of 300 MB discs and 32 spindles of 200 MB discs. Thus the total nominal disc capacity is about 36,880 MB.
- (5) Minimally configured MSS (Mass Storage System) with 706 magnetic tape cartridges (each 50 MB, total 35,000 MB) and 12 spindles of 200 MB staging discs.
- (6) 5 data communications processors and 3 network processors for 500 simultaneous terminal (TSS and remote-job entry) connections.
- (7) For batch processing, the following I/O equipments are provided. Most of these are used directly by users on do-it-yourself basis. The scheme is called "advanced open-batch processing". Many color CRT displays are deployed for this purpose.
  - 7 Card readers
  - 13 Line printers, each with an automatic paper cutter and some with paper-box feeders
  - 12 Magnetic tape transports
  - 1 Kanji laser-beam printer/plotter (720 lines/min, cut sheets)
  - 1 Floppy disc input/output
  - 6 XY plotters
- (8) Graphic displays
  - 1 High-resolution display
  - 3 Storage-scope displays
  - 1 Color graphic display
- (9) Kanji I/O terminals
  - 5 Kanji display with alphanumeric/Kanji keyboards
  - 3 Ink-jet Kanji printers

The features of the M200H(VOS3) system

- (1) Loosely-coupled multi-processor system  
All user files can be shared by all tightly-coupled systems (each with 2 CPU + 16 MB storage)

within DG. AZ-Text, a word processing system that runs under AOS and is developed inhouse at DG, is available to other OEMs, but few have taken advantage of it. Although few OEMs hold a high opinion of AZ-Text, DG continues to push the product because it has spent millions of dollars developing it, according to sources close to DG.

A source inside DG admitted there are battles going on between supporters and detractors of AZ-Text, but added: "There's always controversy within corporations on any big issue."

Another proposed solution to the word processing gap was Wordcraft, reportedly a standalone word processor in an MPT format. But it is believed Wordcraft's fate, if still alive within the company, is being by a thread.

Apple, rumors had been circulating in the from the company.

# Zilog development system runs under enhanced Unix

CUPERTINO, California. — Z8000-based multiuser development system designed to run under the Bell Laboratories Unix operating system has been introduced by Zilog Inc.

The Z-Lab 8000 programmer's develop-

ment system can support up to 16 users and runs under the Zeus operating system, the vendor's enhanced version of the Unix system.

It can be used to develop code for all Zilog MPUs and supports up to 1.5 megabytes of error-correcting memory using 24 megabyte 8in Winchester disk drives.

The Z-Lab concept separates hardware and software development tools into specifically tailored devices that can operate alone, with each other or with devices made by other manufacturers, thus protecting the user's investment in both the hardware and software areas, a company spokesman said.

The large user base and software base of development-related applications in the Unix system offers advanced documentation tools, the vendor said.

The more than 60 utilities include programs that teach the user about Zeus, send messages to users, remind users of tasks to be completed and search files for a particular character pattern.

The Z-Lab 8000 programmer's development system is available in two versions. Model 20, priced at \$US27,000, includes CPU board, two intelligent controllers, 256K-bytes of ECC memory, one 24 megabyte Winchester disk drive and a cartridge tape drive. Model 30, priced at \$US33,950, offers 512K-bytes of ECC memory and two 24 megabyte Winchester drives. Both models feature miscellaneous storage compartments.

Zilog was represented in Australia by Zap Systems Pty Ltd, St Leonards, NSW, which has been acquired by Hanimec. A new distributor for the development system has yet to be appointed.

“...and I know you're from Liprogopolis.”

When Dr. Who said this to the aliens he had inadvertently told them the name of the only computer in the galaxy that is an incredible communicating system, understands all industry standard languages, has a fantastic range of software solutions and can increase its power in seconds.

Prime Computer.

## OFFICE AUTOMATION SYSTEM

Prime Computer's Office Automation System is totally integrated, combining functions previously only available individually, like word processing, communications and data processing. A single work-station handles text manipulation and processing for clerical staff and communications and tracking functions for management, all with secured password-controlled access. All these features are available on the same system that also handles traditional data processing applications.

## APPLICATION SOFTWARE

Prime supplies and supports a wide range of applications software through a division called "Prime Solutions". The large virtual address space, processing power and complete compatibility across the range make Prime the ideal computer for software developers. This has led to a proliferation of available application software packages running on Prime computers

(The aliens, incidentally, turned out to be just some over-enthusiastic Prime buyers.)

## Intel and HP reveal 32-bits

NEW YORK — In recent announcements, both Intel Corp and Hewlett-Packard Co unveiled details of their own 32-bit microprocessor chips, with Intel ready now to market its first two-chip system.

Using Ada as a programming language, Intel's iAPX432 system is comprised of three 64-pin chips containing a total of 200,000 transistors, a company spokesman said. The system includes two general data processors and an I/O processor that can link attached peripherals and memory circuits.

The system, the result of a five-year, \$US25 million research and development effort, will "lead to create its own ALIGN