AUUGN

Australian Unix systems User Group Newsletter

Volume 6 Number 6

Registered by Australia Post Publication No. NBG6524

The Australian UNIX* systems User Group Newsletter

Volume 6 Number 6

August 1986

CONTENTS

Copyright © 1985. AUUGN is the journal of the Australian UNIX systems User Group. Copying without fee is permitted provided that copies are not made or distributed for commercial advantage and credit to the source is given. Abstracting with credit is permitted. No other reproduction is permitted without the prior permission of the Australian UNIX systems User Group.

Editorial

It is with a very large amount of pleasure (and a little regret) that I now tell you that this is the last AUUGN to be edited by me. Our apparently hopeless search over the last few years, for a person silly enough to take on the job of AUUGN editor, has finally born fruit. The fruit comes in the shape of John Carey from Monash University. He may be contacted as

John Carey,
Computer Centre,
Monash University,
Wellington Rd.,
Clayton VIC 3168.

ACSnet: john@monu1.oz

I hope that John meets with at least as much success at the job as I did, hopefully more!

Memberships and Subscriptions

Membership and Subscription forms may be found at the end of this issue and all correspondence should be addressed to

The Honorary Secretary,
Australian Unix systems User Group,
PO Box 366,
Kensington NSW 2033.
Australia.

Next AUUG Meeting

The next meeting will be in Canberra at the Australian National University. Further information appears at the end of this issue.

Contributions

Come on you people out there in UNIX-land, get behind your new editor and PUSH! Send him your views, short stories, reviews or whatevers.

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

Books

We have two reviews in this issue, the first sent by an interested reader of AUUGN, and the second stolen from netnews.

Book Review: Pocket Guide to UNIX

Pocket Guide to UNIX, Lawrence Blackburn and Marcus Taylor, Pitman Publishing, 1984, ISBN 0 273 02106 0, RRP $6.95.

The past year has seen several UNIX books published at last, ranging from the definitively technical Kernighan and Pike for Real UNIX users, through the gentler and more cosmopolitan Kochan and Wood (which describes *vi* and towards the not-yet-attained popularized end of the spectrum

The Pitman Pocket Guide *UNIX* is towards the serious end of the spectrum. In its 62 small pages (100 x 150 mm, or 2/5 of the size of Kernighan and Pike's pages) it aims to be an introduction to UNIX, a training aid, and a quick reference for the new user. It is bound on wire loops through the heads of the pages like a flip top notebook to stand up conveniently beside a terminal.

In 62 pages the Guide is necessarily tightly packed. A concise 7-page introduction touches on kernel vs. utilities, the shell, editors, commands, and arguments, background processes, standard input and output, redirection and pipes, files and directories. This section has the densest text, while the others are organized more openly as introductory paragraphs and pages of command descriptions, examples of their use, and typical output. The eight main sections cover login and logout; the line editor *ed* (in 3 pages); a bunch of system status and control utilities; file and directory management; simple *sh* programming, including variables, *if, test* and and system administrator's utilities such as *su, mount, sync, mknod* and Two single page indexes, by commands and by subjects, complete the book.

The strength of this guide lies in its brevity. As a ready reference and reminder in a novice's first few UNIX sessions it is a clearer and easier introduction than the UNIX Programmer's Manual and briefer than the other books. It makes the best of the unfortunate variety of UNIX flavours by concentrating on concepts and commands common to most descendents of the 7th Edition.

But the book suffers badly from its small size and from the choice of the material squeezed into it. Important topics such as the organization of the system directories and the PRM, and the existence of terminal control characters, are omitted entirely, while system administrator's commands (hardly suitable for a novice) occupy several pages. The number of misprints is a minor annoyance; more serious are the inaccurate nomenclature, inexact command formats, incorrect page numbers in about half the index entries, and lack of care in noting possible differences between flavours of UNIX. A case in point is the description of *cat* with a *-u* option which "unblocks the output to the terminal...removing unused blank lines at the end of a page." This effect is not an attribute of *-u* on any of Edition 7, BSD4.2, or System V; the option exists but acts to unbuffer the output and reduce delays, but filters nothing out. (Filters don't get a mention either.)

The problem of introducing UNIX in ready reference form in 62 pages is an awesome one, which this book comes somewhere near solving. I would recommend this Pocket Guide to the person just starting on UNIX, if that person were not using UNIX more than occasionally. The price, convenience and depth of treatment are suitable. For the tertiary student I would prefer Kernighan and Pike [Brian W. Kernighan and Rob Pike, *The UNIX Programming Environment*, Prentice-Hall 1984], albeit at four times the price, with more than 4 times the life in it and more accuracy; or Kochan and Wood's book [Stephen G. Kochan and Patrick H. Wood, *Exploring the UNIX Environment*, Hayden 1984], for the user (rather than programmer) who is going to spend any time on the system.

Chris Johnson
Dept of Computer Science, Faculty of Military Studies UNSW (Duntroon)

Book Review: **Document Formatting and Typesetting on the UNIX System**

Document Formatting and Typesetting on the UNIX System, Nerain Gehani, Silicon Press, 1986, Summit, New Jersey. ISBN: 0 9615336 0 9 Lib. Congress Catalogue Number: 85-61997

I've just finished reading Narain Gehani's new book. Gehani works for Bell Labs in Murray Hill. He has written two books on Ada, three books on C, and has edited some others.

The book is said to be (according to the blurb on the back)

> "... the first comprehensive book about the UNIX system document formatting facilities ....
>
> "... written especially for readers with little or no experience with the UNIX system formatting facilities...
> experienced readers will also find [it] very useful.
> The novice is gradually introduced to the ... facilities.
> The reader familiar with these facilities will learn about their advanced aspects."

It is neither comprehensive nor suitable for novices. The introduction has all the gradual-ness of a cliff-face. It does not get very advanced, either -- if you have already read the relevant UNIX documentation[1], you are very unlikely to learn much. That's rather strong criticism. After describing the book somewhat less emotionally, I'll explain why I felt that way.

There are 8 chapters and 3 appendixes, plus glossary, bibliography & index. These are:

1 Introduction -- an overview including some basic troff and -mm requests. 13pp

2 Specifying The Document Format -- a detailed description of the -mm package, and some of the troff basic escape sequences (, , distance units (i, c, p etc), □ and so on). There are also more details on preparing the input file. Macro definitions are covered very briefly. 88pp

3 Specifying Tables -- in which tbl is explained in some detail, although not all of the features are covered. The interaction between -mm and tbl is mentioned at a user's level -- including .TH and multi-page tables, although there is no example of this. 42pp

4 Specifying Figures -- pic is covered in some detail. The version dexcribed is rather new, though; there is a summary of recent changes, but mention when the features are introduced would probably be better for most users. 68pp

5 Specifying Formulas -- this chapter treats eqn in some detail. A few techniques not mentioned in the Kernighan-Cherry paper are described, but some important basics are omitted or explained only very briefly. Again, the version described is new, supporting macros with parameters. 43pp

6 troff/nroff - The Formatters -- a brief overview of nroff and troff. This contains mostly excerpts from the original manual by Osanna, but written in less formal English and with only a small subset covered. The format is almost identical.

7 WRITER'S WORKBENCH Software -- an overview of the various utilities provided by wwb, including spellwwb, punct, splitinf, double, diction & style. Gehani also mentions 'grope', which is not part of wwb, but he doesn't say who distributes it. 12pp

[1] The "other papers" I've mentioned, and have called "documentation" or "manuals" are in Volume Two of the Version 7 Unix Programmer's Manual. If you have 4.[123]BSD, they're in the same place, mostly in vol.2A, but also in vol.2C. If you have System V, look in the DWB [or WWB] binders (there seems to be more than one); they may be elsewhere, too. The papers cover eqn (Kernighan & Cherry), ideal (C.J.Van Wyke), troff (Ossanna), pic (Kernighan), refer (M.E.Lesk) (BSD systems come with a clearer document in Vol.2C), and tbl (M.E.Lesk, 1979)

8 Example Document Templates -- Some examples of combinations of the mm macros to give various styles for letters, papers, books and so on. There is a brief overview of how to produce an index using ".tm" and "sort", but this contains bugs. For most people this will probably be the most useful chapter in the book. 22pp

App.A More Document Formatting Tools -- some other UN*X programs, most of which are either Berkely-ware or have been unbundled. These include ideal, grap, the -ms macros (a very brief summary, shorter than "man 7 ms " gives), the -mv macros (in the same sort of non-detail), and refer. No mention is made of bib, or of the BSD refer support programs, although lookbib gets a single sentence. 5pp [sic]

App.B Document Formatting Commands -- excerpts from the manual pages for some of the programs mentioned, somewhat paraphrased. 11pp

App.C Some Font Examples -- ten pages of font samples. As the typesetter used to produce them isn't mentioned, they seem a little pointless. But they're quite pretty. Perhaps Gehani doesn't realise that fonts vary dramatically from machine to machine. 10pp

There are also a very short (3pp) glossary, an annotated bibliography (5pp), and a comprehensive index (18pp).

In each chapter, Gehani seems to have taken the original paper or reference manual section-by-section, and done a quick re-write. There are minor errors and gross omissions. In every case, I was left with the feeling that a beginner would do better with:

- a short introduction which explains how to prepare input for troff and which sets out the basic concepts, and

- the original papers and reference manuals

Although Gehani has attempted to simplify and de-mistify, the incompleteness will only confuse. As an example of what I mean: In the explanation of how to define your own troff macros, [on pp.88-9] the string "\$n (1 <= n <= 9)" is given to be learnt parrot-fashion as a means of accessing arguments, although _no_mention_ is made of the way in which backslashes are stripped. This is like telling someone that in C, "++x" returns a value one greater than "x", and forgetting to tell them about the side-effect it has of incrementing "x"...

There are many other omissions, and there are other occasions when the reader is given a faulty mental model, a wrong way of looking at things, that could cause confusion. There's a bug in his example of index generation, for instance, where his example gives the output of ".tm *xxx*" as "*xxx*". This is perhaps because he wanted to simplify the explanation. I prefer accuracy, or at least a mention that details are omitted.

Perhaps it will be useful to have a lot of information in one volume, and it does seem to have been reasonably well bound. It is certainly rather easier reading than the technical and somewhat dry troff documentation, and may well be suitable as an introduction to them. But the omissions are too important to be ignored, and beginners would be advised that the book is not enough.

Which makes me wonder if it is worth-while at all, especially because there's no real mention of the areas in which the book is lacking.

Overall, give people the manuals with (or instead of) this book.

Russell Quin
Computer Science, Warwick University, UK.

Excerpts from NetNews

Here are some excerpts from recent netnews articles, borrowed without permission. I have removed most header lines, and some included text, signatures, etc, to save space. Occasionally where I was particularly offended by the formatting of an article, I have "fixed" it.

If you look at the selection of newsgroups these articles are drawn from, you will probably easily detect the newsgroups that I read. No apologies for this, I simply don't have time to read every (or even most) newsgroups, even if they weren't mostly utterly boring. So, I am looking for people who read other groups, or even these groups, who come across articles they consider worthy of republication here to send me either the article, or its Message-ID.

The articles I choose are ones that seem to contain useful information, or which I find amusing, and some others just so you cannot try to categorise all of the following as one of those!

I am also looking for feedback, are there too few articles, too many, are they too long, too short, etc..

Robert Elz (kre@munnari)

From: larry@JPL-VLSI.ARPA
Newsgroups: mod.computers.vax
Subject: Ada question
Date: 18 Jan 86 00:08:21 GMT

The DEC Ada compiler is probably the best around for VMS. Verdix may have the best for Unix systems. They both cost about the same--$25,000. Third-party sources, even if they exist, are not going to cost less. Remember that Ada is the property of the richest SW market around: DoD. And considering market realities, if you pay less you get less.

If you want to pay the least money in the absolute sense, if you have an IBM PC/AT or compatible try Alsys (in Massachusetts 617 890-0030). They're about to release a compiler that will run under PC-DOS and create DOS programs. Price is not set but my guess is about $2500. Alsys also has a product for Sun and Apollo, though you may want to get the Verdix system instead. (The Alsys product is supposed to be pretty good, however. I doubt if you'd go far wrong using either.)

Remember that the DOS AT is a single-user system. A VAX can support 10 times the Ada users at any one time, and hundreds each week, so the per-person cost of a multi-user Ada system is less.

There's also an Ada-oid compiler from R&R sytems, but it's not full Ada and some of it's not really Ada. Ada has a lot of subleties, and you can have a lot more trouble unlearning subtle differences from the real thing (which you'd have to do eventually) than learning a completely new language.

For further ARPAnet info contact INFO-ADA at USC-ISIF.

Larry @ jpl-vlsi

From: ronb@dmsperth.OZ (Ron Baxter)
Newsgroups: aus.general
Subject: Notes from the AUUG meeting in Perth (this is LONG)
Date: 12 Feb 86 08:57:51 GMT

NOTES ON THE AUUG MEETING IN PERTH

I have been asked by others in our own Division who did not get the chance to attend the Perth meeting, if I could make some notes and report back to them. Well, if I am going to make notes I may as well post it to news for all to see. So first my disclaimer. What follows is a totally subjective, some what idiosyncratic, possibly inaccurate, account of some things said at the meeting.

Monday 10th

Kirk McKusick, Univ of California Berkeley.
The History and Future of Berkeley UNIX.

This was a very interesting account of the history of UNIX according to Berkeley. Kirk started way back with mention of Alice, then Multics, Ritchie & Thomson hacking on a PDP 7, and the a PDP11/20 purchase in 1971 for their legal department to do word processing on. So UNIX (and also nroff) was created and developed to Version 4 (the first one written in C) by 1973. Berkeley got hold of this system onto a PDP 11/45 that they shared with Maths & Stats in 74 which ran RSX part of each day and UNIX at other times. They moved to an 11/40 and then an 11/70 in 75 when Thomson brought the new Version 6. He spent a year there. Bill Joy's name pops up and there are others, McKusick hacking Pascal and so on. In 78 they got a Vax (serial # 10) and set about hacking version 7 (or is it 32V?) to do virtual memory. By 1980 this had developed to 4BSD, they had defense funding (DARPA), and were receiving contributed software from elsewhere. More things were added and the BBN company was contracted to do part of the networking stuff that was to appear in 4.2. However, 4.2 was released with re-written TCP/IP code after the BBN code was junked. This produced tensions that came to a head last year and delayed the release of 4.3 (now due on Feb 18).

As to the Future, the DARPA funding continues, there will be work on adding remote file system capabilities - maybe similar to (or is it upward compatible with) the NFS of Sun Microsystems. They will add something with the functionality of streams as in SVr2, and intend to rework the virtual memory system to take into account the larger amounts of physical memory now available and the fact that swap space may at times be on remote file-systems, and of course to clean it up.

Questions were about the complexity of the VM code (should be able to simplify it), whether Sys V compatibility is in their minds (maybe similar to what SUN Microsystems are doing, i.e. support the Sys V interface (?with exceptions?)), would there be more contributed software (yes), and what about if DARPA funding ceases (Berkeley is diversifying there sources of funds).

Roger Hicks, PAXUS computer Systems and NZUSUGI.
UNIX in New Zealand

Roger produced figures for the number of UNIX installations worldwide (they are half Xenix), and showed that NZ spends more per head on computers than elsewhere (provided you count human heads and not sheep heads). He caused laughter when he claimed that the high rate of sales through a Commodore outlet showed the sophistication of the NZ market. Then he had more figures to show that only 27% of UNIX system users in NZ thought they would use 4GLs. "What is a 4GL ?" was asked. The answer: don't know, but whatever they are people don't think they want to use them (more laughter).

Chris McDonald, Univ of Western Australia
fsh - A Functional UNIX Command Interpreter.

This was about creating a shell with a language structure like the functional programming language of Backus. I cannot do any justice to this talk as I have no idea what the functional programming language is. I copied his first simple example hopefully correctly:

```
def nextfile (name, version) x
        let filenm x name | '.' | ltoa(version)
        in
        if exists(filenm) then nextfile(name,version+1)
        else filenm;
```

which created the next version of a file. He had plenty more and says it run on 4.2BSD and used 0.25 Mb.

John Lions, Univ NSW
Assessing Interactive Programs via Batch Processing

In a course that John is giving for the 15th time students are given as assignment to write an "inspect" command that recursively moves down a tree looking at the contents of files. It requires the use of system calls such as fork, exec, kill and signal. The process of assessing these programs requires them to be run, to be checked on extreme cases such as short or empty files, special files and so on. He automates this process by having the programs run and some informative output produced. Particular thing to watch for are programs in infinite loops or producing repetitive output, programs that create or delete files, that dump core etc.

There were questions and it Andrew Hume commented that similar techniques were used for testing compilers.

Glenn Huxtable, Computer Science, Univ of WA
UNIX: Fighting Fires - /bin/echo "Help Fire!"

Glenn illustrated his talk with slides of burning houses, fire trucks, historic shots of how fire alarm systems were once monitored - all by way of introduction to the problem - a computerized system for monitoring fire alarms. In fact, his introduction took all his allotted time so that the discussion of software only came after his time was supposedly gone. They use a PDP 11/73 with Informix data-base (supposedly a 4GL although Glenn doesn't know what one is either), and specialized hardware for delivering the alarm messages over serial lines. They have data bases of building statuses, alarm locations, street directorys, etc. They did do a couple of kernel hacks to the Level 7 system they were using to create a /dev/msg device.

Steve Landers, Esso, Sale.
Controlled Applications Environment.

Esso uses Vax 750 with System V and Oracle. He described how they develop applications quickly with a quite standardized user interface by using what he called the Application Development Environment (ADE). This consists of UNIX + Oracle + tools. Mainly he described these tools. There is a visual shell (called vsh) which provides a standard format scrolling menu system. Pick was a program used as a standard module in shell scripts to let users select arguments or files from menus. These tools were designed to handle frequent tasks in a standard manner. Examples of applications developed were a data-base of Esso owned property with details of rates etc, and workcare - with data about workers compensation cases.

Questioners asked about the availability of vsh (maybe).

Jeremy Firth, CSIRONET, Hobart.
Interfacing UNIX to CSIRONET.

As a recent user of UNIX systems, Jeremy started with comments on his experiences in getting into UNIX, and by noting the every UNIX conference seems to have some discussion of numbers. This seemed to be with respect to the strange sequence of numbers like 6, 7, III, 4.2, 8, V, or whatever. He then described CSIRONET and then his particular interest in software for file exchange between small systems and CSIRONET. He has started with UNIX but will then do similar software on PCs, Macintoshes and Sigma Data NGENs. The software consists of "cnsend" to send a file from the small system onto a CSIRONET host to be stored or executed, or to a CSIRONET peripheral; "cnfetch" and "cnpoll" which together can get a file onto the small system; and "cnlogin" which is for logging onto CSIRONET hosts. It is currently available on Level 7 and 4.2BSD and will soon be on System V. It costs a few hundred dollars.

Robert Elz, Computer Science
Interfacing Appletalk to UNIX.

This paper scores the prize for fast delivery complete with complex diagrams of networks and layers that appeared and disappeared at a rapid rate. The aim was to connect a bunch on Macintoshes strung together with Appletalk, to the TCP/IP ethernet system with a bunch of UNIX systems. Stanford has software called SEAGATE that works with hardware based on a SUN board. Melbourne Uni will adapt this to the slightly different hardware they will use (based on a Unison board). At this stage of the afternoon my concentration was no match for this rapid talk, but I did hear him mention the Mac floppy looking like a part of the UNIX file system, and there was talk of wrapping and unwrapping this and that sort of packet.

Tuesday 11th

=============

Roger Hicks, PAXUS computers and NZUSUGI
New Zealand UNIX System Users Group Inc

Roger traced the history which started with a small group organizing a Workshop to be held in May 84 with names like Ken Thompson and Ian Johnstone. They got companies to sponsor them, had T-shirts and wallets printed and 200 people attended. Proceedings were subsequently published. They were incorporated in Aug 84, got a newsletter going that included advertising. Zilog donated a UNIX machine but that started a long saga where it was stuck in customs. The second conference was in May 85 and was called UNIX Workshop Exhibition 2 with a logo made up of the letters UWE2 inside the outline of a kiwi. ("Why not a sheep?" someone interjected) The newsletter which was very demanding is now to be done as a supplement within a NZ glossy computer magazine (Computer Scene), and the next Workshop in May 86 is to be more upmarket. It will be in the Travelodge at Rotorua with an all inclusive pricetag of about $NZ375.

Mark Ellison, Univ of WA
Multiple Redirection using Stream Digraphs.

Unix pipes are linear, so Mark has been working with a syntax that can extend this. Consider the case where the output of a program is to be used as input to 2 other programs:

```
prog1 | tee tmp | prog2; prog3 < tmp
```

or the case where a program works with the input from 2 other programs such as:

```
prog1 > tmp; prog2 | diff tmp -
```

In addition to the usual meanings of < > and | he defined

```
<| {pipe}       read from the (named) pipe
|> {pipe}       write to the pipe
~               force input from stdin
.               force output to stdout
```

so that the above examples become:

```
prog | tee |> a | prog2; prog3 <| a
prog1 |> c ; prog2 | diff <| c
```

Lots more examples were shown that did exciting things like, count, list powers of 2, or Fibonaci numbers. It is implemented in a shell called gsh, and utilities such as tee have been modified.

There was lively discussion about whether this was all a waste of time or a brilliant extension in the UNIX style. Andrew Hume gave an impromptu description of the named file descriptors of version 8.

AUUG Meeting

There was no formal agenda. John Lions thanked the conference organizers and the overseas speakers. Kem McDonnell outlined the tutorials that will be held at Unix-world in May, and mentioned the prices. Also, there will be dry-runs in Sydney and Melbourne in March. Nominations for office-bearers of AUUG should be in by May 1, and a postal vote will be organized (if necessary). There was some inconclusive discussion about expences for committee members attending meetings. I got the impression that if the committee had put a concrete proposal for maybe 2 committee meetings a year with fares (inexpensive ones presumably) paid, the meeting would have agreed. But they didn't even have an initial costing or any financial summary for the group. John then asked for some discussion of the general direction that AUUG should take. Should we become more highly visible like NZUSUGI? What is to be learnt form the gap between Usenix and /usr/group in the US? If anything there was a tendency to agree that AUUG should concentrate on useful technical meetings and not get too involved in the commercial hype. Andrew HUME said "If you look at the people going to these conferences in the US - an typically they are rather large....." (laughter).

Kirk McKusick, Univ of California, Berkeley.
Using GPROF to Improve Program Performance

Kirk had notes for a 2 hour tutorial, and 20 min to summarize. He offered to give the more complete tutorial if he is invited while in Australia for the next couple of weeks. [If anyone finds out that he will be doing this, please post details to the net.] He really deserves some award for the tranparencies (= vu-graphs) he used which had been prepared with the help of a graphic artist. He started with "motherhood and apple pie" stuff about the kernel being a hierarchical program with the individual routine being grouped into modules. The traditional UNIX profiler simply gives times spent in each routine which is not all that useful for something as complex as the kernel. Gprof keeps track of where a routine has been called from and then allocates the time spent in the routine amongst the callers on a proportional basis (the assumption made by this allocation are not always true, but it is hardly practical to try and gather more complete data). An output is generated from the data that assists in identifying the modules where the time is spent. The things you may find are inefficient algorithms, unexpected counts, etc. He illustrated all this by referring to "namei" the routine that translates a path name into an inode number.

Eric Allman, Britton Lee Inc.
UNIX Databases.

Here was another talk being shortened. Eric was involved in the research version of Ingres at Berkeley, and now works for Britton-Lee which makes a "database box", but his talk was a general review of database terminology and the things you need to keep in mind as you identify your own needs and talk to vendors. He started with the main database models - hierarchical, network, and relational and identified the various IBM candidates which would cause you to buy 1 or 2 more CPUs. He covered the basic operations that they all have (retrieve, append, delete, replace, projection, selection and join). Then, at last, here we had someone prepared to tell us what a 4GL is. A 4GL is a marketing term for a non-procedural language. A non-procedural language can be thought of one where you get what you want by filling in forms. They usually have provision for ad-hoc queries, a programming language, query by example, report writing and browsing. Features that various systems offer include arithmetic capabilities, aggregation of data, data structure independence, multi-file capabilities, concurrent access, crash resilience ("commit" and "sync"), transactions (more complex "commit"s), audit trails, backup/recovery facilities, protections, semantic integrity, non traditional data types (graphics etc). Finally, how suitable is UNIX ? The protection facilities are adequate, locking is unacceptable although partly fixed in 4.2, file syncing is unacceptable although fixed in 4.2, file system performance is unacceptable because of the read ahead strategy - but is improved in 4.2.

Andrew Hume, Bell Labs
Folding Regular Polyhedra

Well if there awards for visual aids - this one really gets 1st prize as Andrew had many carefully constructed clear plastic models of the solids he was describing which could be placed on the O/H projector to good effect. He also had a short movie based on impressive color graphics that showed the flat cutouts being folded into polyhedra. The work he was describing is really a hobby since he is interested in building models and is fascinated by the geometry of these polyhedra. He keeps a data base of polyhedra, using a notation to record the angles and coordinates, etc. He uses an HP plotter to create the flat "nets" from which the solid are obtained by folding.

Bob Buckley, Macquarie Univ

Gould UTX/32 in a University Environment

This Gould dual-processor is fast and runs what is claimed to be BSD4.2 "but is obviously different from that distributed by Berkeley". He gave an account of the various problems such as that there is no way of stopping users from reading kernel memory, the assembler code produced by adb is not the same as the code accepted by "as", and the Pascal delivered was definitely unsuitable for use in a student environment. A main aim of getting the machine was to get students doing graphics. They had a GKS implementation callable from C, but decided to use Fortran (since Pascal was not good enough). The combination of 'f77', linking with the large GKS library, and lots of students rally taxed the system. In summary, he feels it is OK, the teething problems were to have been expected, it is good at floating-point work, and good for student graphics, but don't try and use the Pascal system for students.

Roy Rankin, Comuterlink Design, Sydney.

An Introduction to the UNIFY Databse Program

Roy illustrated the use of UNIFY by example. He had transparencies to show the commands required to work with the database, and slides to show the screen layout of the menus etc.

Rober Elz, Computer Science, Melb Univ

ACSnet

Robert started by making it clear he had not come to talk, but only discuss so anyone not wanting to discuss could ....
He then talked for pretty much all of 60 minutes which must deserve some type of award. He described ACSnet by first outlining what the SUN III software was and how you get it (the commercial price is apparently $500). Then said how you join ACSnet once you have SUN III software. He complained that ACSnet was a terrible name for the network and appealed for better names. (Chris Maltby tried Piers' Univ Sydney Network because the acronym appealed to him).

Robert then seemed to foreshadow a trauma for the network as follows:

- munnari is the gateway to the rest of the world
- all long haul transmission of ACSnet is via CSIRONET
- munnari is about to lose its link to CSIRONET

Possible alternative arrangements were vaguely floated.

The mail, news and fileserver services were outlined, as well as the networks we can access (ARPAnet, CSnet, Usenet).

Then to money. Last year contributions were invited but this hasn't worked well. Consequently, a charge has been determined for overseas mail - both outgoing and incoming.
It will be 10 cents flagfall plus 2 cents for each 64 bytes or part thereof. Outgoing mail from sites that don't pay will be bounced back. News is currently only 25% of total traffic so getting payment for mail operational is the first priority.

However, they have several schemes for funding news in mind and will publicize these on the network soon.

Closing Comments.
================

John Lions thanked people and said the next meeting would be in Canberra on Sept 1 and 2 this year. The one for Feb 87 is likely to be in Sydney.

--
Ron Baxter,
CSIRO Div Maths & Stats,
Lindfield, NSW   ronb@natmlab.oz

From: weemba@brahms.BERKELEY.EDU (Matthew P. Wiener)
Newsgroups: net.news.group
Subject: The undead net.bizarre
Date: 2 Apr 86 22:14:06 GMT

I have heard a rumor that the undead newsgroup net.bizarre feeds on net.rumor as a parasite and it is now invading other newsgroups like a cancer in metastasis. The standard practice apparently is to begin the posting with the innocuous phrase "I have heard a rumor ..." and to conclude the posting with the obligatory "Can anyone confirm this?" Can anyone confirm this?

ucbvax!brahms!weemba   Matthew P Wiener/UCB Math Dept/Berkeley CA 94720

From: johnl@ima.UUCP (Compilers moderator)
Newsgroups: mod.compilers
Subject: Re: C Decompiler Query (humour)
Date: 8 Apr 86 23:21:00 GMT

In article <179@ima.UUCP> you write:
>has anyone had any experience in writing a C-decompiler? The
>input is assembler source, and the output is C code.

How about:

```
sed -e 's/^/asm("/' -e 's/$/");/' -e '1i\
main() {' -e '$a\
}' < file.s > file.c
```

For the record: :-) :-) :-) :-) :-) :-) :-) :-) :-)

--
Tim (radzy) Radzykewycz, The Incredible Radical Cabbage.
        ARPA: calma!radzy@ucbvax.berkeley.edu
        UUCP: {ucbvax,sun,csd-gould}!calma!radzy

From: chris@umcp-cs.UUCP (Chris Torek)
Newsgroups: net.unix-wizards
Subject: One Emulex SC41/MS (UDA50 emulator) user's experience

We bought an Emulex SC41/MS controller and two CDC 9771 drives. The SC41/MS is an 'MSCP compatible' device that emulates a DEC UDA50. This article is an anecdotal description of our experiences thus far with the controller and drives.

When we first obtained the hardware several months ago, we ran into a few snags. The University bureacracy had managed to mangle the order into listing the machine for which the controller was purchased as a Vax 11/780; in fact, it was a 750, and we needed the Emulex cassettes to format the drives, but of course because we said '780' they sent us console floppies. As it turns out, you can fix this with 'arff': The following procedure copies a floppy to tape:

```
                                # log in as root on 780,
                                # and insert floppy #1
        cd /tmp; mkdir floppy1
        (cd floppy1; arff x)    # extract
        rcp -r floppy1 750:/tmp # copy to 750
                                # repeat for each floppy

                                # log in as root on 750
        cd /tmp/floppy1         # go to floppy directory
        arff crmf /dev/tu0 *    # put everything on the tape
```

Of course, you still need a boot block on a bootable tape; but I kludged around that by putting the bootable image from the first floppy on our root file system, and making a companion to /boot that loaded it. You could also just copy the boot block from any other DECtape that has it:

```
        dd if=/dev/tu0 of=bootblock     # with good tape
        dd if=bootblock of=/dev/tu0     # with new tape
```

In any case, the formatter worked fine, and after about eight hours, both drives were formatted and verified. (I should mention here that Emulex did indeed send us the proper tapes; I was simply impatient.) Now we had nearly 1.35 gigabytes more space on our machine. Wonderful! Now to put it to use . . . so I created file systems and mounted them, and then the fun began.

After about five minutes, the machine hung. It was clearly a bug in---what else?---the UDA50 driver, as interrupts were still working and CPU bound tasks kept going. But the moment anything tried to touch a drive, CDC or DEC RA81, it was blocked. This was quite repeatable: with the CDC drives mounted and in use, the machine would hang within thirty minutes. 'Well,' thought I, 'time to fix the driver.'

Now at the time, we were running a modified version of the RIACS driver. For those of you who have not heard of it, this is the one with dynamic bad block revectoring, so that when your RA81 begins to bobble bits, you need not reformat the entire drive, with the attendant and painful dump-and-restore sequence. The key words describing this driver are 'useful', 'large', and 'thoroughly unreadable'.

After a few days I gave up the task of fixing the existing driver. It was long overdue for a rewrite anyway; and I decided that I should, instead of just fixing it, try my hand at writing a generic MSCP driver, so that if and when we got a TMSCP tape, it would then be a simple task to talk to it. So of course the next step was the first required when writing any driver: obtain the hardware documentation. 'No problem!' thought I. 'I shall call DECDirect and give them the order number straight from the Emulex manual.' That I did, and this I discovered: DEC does not sell the MSCP documentation. Yes indeed, it does exist; no you cannot get it.

Well, that stumped me for a while. How can you write a driver without knowing what it needs to do? Ah, but wait! We already have a driver---nay, in fact, *three* drivers---that probably do mostly the right things. To make a long story short, I cannibalised parts of the RIACS driver, the original 4.2 driver, and the 4.3 beta driver, to put together a completely redone version of my own. Along the way I found out what all the CPU-dependent code was for, and I changed the Unibus support code to do BDP allocation 'right'. It took several weeks, but at last I had a driver that booted and ran. (It took several more days before it crashed properly---a bug in the dump code---and it was still more later that it handled Unibus resets, but it ran!) I brought the CDC drives on line, and waited for the driver to hang. 5 minutes . . . 15 . . . an hour, more . . . *hooray! It runs!*

Well, at last all our troubles were over. Right? Wrong.

A few nights later I went to dump the new file systems from the CDC drives to tape. We use a special kernel hack to make dump run fast, so there I was loading tapes onto the TU80 and watching them stream at 100 ips. Well, make that about 75 ips average. Performance was not teriffic; but that must be expected with Unibus disk drives, for the fastest transfer rate achievable on a 'real' Unibus is 550K/sec, and of course we had seek delays to deal with as well. (Incidentally, for those to whom seek time is important, the CDC drives list an average seek time of 18 ms., and no head switch delay; compare this with, I think, 31 ms. and a 6 ms. head switch delay on the RA81.) Running iostat showed that the top performance of the CDC drives was actually lower than that of the RA81s: doing large raw disk reads, peak performance on the CDC drives was about 350K/sec, while on RA81s it reaches the 550K/sec maximum. Presumably Emulex has not properly laid out the sectors rotationally; and there is no way to change the sectoring: It is in firmware on the controller. Perhaps Emulex will read this and put in a format parameter in the next version. ---But so what if the performance was worse; we needed the disk space. At least it worked.

Or so I thought. 'DUMP: NEEDS ATTENTION: ...' Time to change tapes again. Ok, tape number 5, go. Watch the reels: ZOOOOOM forward, blip back, ZOOOOOM, blip, ZOOOOM, blip, blap. Blap? Hey, what gives? The tape drive has stopped. Uh, Oh. Wait, no console response; must be hung at interrupt level. Time to get another crash dump. Type control P. I said control P. *Control P*.

Oboy. Look at the console lights. POWER on, ok. RUN on, ok. ERROR off, ... off?

I quote from the DEC hardware handbook:

```
---------------------------------------------------------------
*Error* indicator        Lighted red brightly to indicate that
                         the CPU is stopped because of an
                         unrecoverable, control-store parity
                         error. Because console commands are
                         ignored, the *reset* switch mustbe used
                         to clear the error.

                         Lighted red dimly to indicate that the
                         CPU is functioning normally.
---------------------------------------------------------------
```

Do *you* see any mention of 'off'?

Well, to make another long story short, the machine would hang quite thoroughy as long as the Emulex controller and the TU80 controller were on the same Unibus. We moved the TU80 to another Unibus adapter, so that now the SC41/MS was all by itself on UBA zero, and the hangs stopped. (No software changes, of course.) Also interesting was the fact that with the CPU cabinet open, the performance of the Emulex card changed. It ran faster. With the cabinet closed it would sometimes slow down so much that the TU80 dropped to 25 ips streaming. (This makes an enormous difference in dump times for one CDC drive, from about two hours for a 330-megabytes-used file system up to about six hours.) With the TU80 on the other Unibus, that problem went away too.

Since then (it has been about a week) we have had exactly one crash, this time due to a response packet from the Emulex controller containing the wrong command reference number. It should have said '8009fec8'; but it said '80090000', so all is still not well. Yet it only happened once; it could be a kernel bug; we have installed the kernel RFS from Todd Brunhoff, and we know of at least one bug, so there may well be others.

Summary: The controller seems to work, as long as it is on a Unibus by itself, or at least as long as it does not have to compete greatly with another controller for Unibus resources. But you may want to avoid this particular controller, at least until it has been exercised a bit longer. The drives, on the other hand, are very nice. It is wonderful to run 'df' and see /usr only 64% full, with another 188 megabytes there alone, and more than 300 megabytes free on the other drive. It is too early to guess at reliability, but there were no bad sectors at all on one of the two drives!

--

In-Real-Life: Chris Torek, Univ of MD Comp Sci Dept (+1 301 454 1415)
UUCP: seismo!umcp-cs!chris
CSNet: chris@umcp-cs   ARPA: chris@mimsy.umd.edu

From: benn@sphinx.UChicago.UUCP (Thomas Cox)
Newsgroups: net.news
Subject: A Fable for net.news admins and others
Date: 14 Jan 86 04:04:38 GMT

[]
A Fable.

Once upon a time, there was a network of message carriers that operated among many different small kingdoms. Each kingdom had its local messenger guild. The Guild Council never dictated to the local guilds because these guilds earned their own money and licensed their own messengers. But if most of the guilds agreed upon a thing, the others, having to exchange messages daily with their fellows, usually went along, lest the entire structure cease to function.

One strange day, the kingdoms all experienced Socialist Revolutions and the guilds were Nationalized. This meant that the Government would give money to the guild, and the guild would carry messages for whoever asked, without taking money from that person. Within a week, the system was overburdened with a strange and horrible new phenomenon. People would address their messages to "everyone," and the Socialist Governments required the guilds to make copies of such messages and, in fact, distribute them to every town and hamlet, even the smallest, where they were posted publicly.

This practice caught on quickly. Debates sprang up between widely separated people on widely separated topics. The heads of the local guilds, along with many and diverse Authorities, felt that this new thing was a Good Thing, and held the seeds of Enlightened Anarchy. The cultural exchange was astounding. Even people on far continents were brought into the discussions. The entire setup became a Structure with a life of its own.

The Socialist Governments, seeing the brilliance of the guild masters, granted them some autonomy in matters of spending. And as the cost of the Structure grew, the guild masters lied to those Governments, saying that the cost of Government communication was high, when in fact it was the messages being sent to "everyone" that cost so much. They hid the costs, for they now feared that the Governments would forbid the sending of messages to "everyone" if they learned of the costs, and the guild masters were strangely attached to their new Structure.

Eventually some of the guild masters observed that the messages they were sending to "everyone" were becoming of a poorer quality all the time their volume was increasing. (One scribe, named Rosenberg the Rich, wrote so copiously that he kept an entire town's messengers busy, although it was widely acknowledged that no one read the scribe's long messages to "everyone".)

Finally, one of the more powerful guild masters, Henry Utzoo, had enough and cut down the number of messages he would forward. Many discussions halted because their debaters no longer could reach each other. Other writers decried this, in long, turgid, and expensive letters to "everyone," saying that it was the Right and Privelege of the writers to have their messages, however empty and banal, transmitted to the far corners of the world at no expense to themselves.

Henry said, "pthpthpthpth."

The volume of messages dropped.

Unfortunately, the other guild masters learned nothing from this. Most looked to the drop in volume as a sign that the Structure could continue as it was. Others, who no longer saw their loyalties with their towns or even with their guilds, for they did nothing but work on ways to improve the Structure, saw the volume and expense problems as things to solve with a more efficient Structure, forgetting that it was the pay of the messengers that cost the most.

At last, one day, the Structure collapsed when the Chair of the Guild Council, Ihnp IV, dissolved the Council and refused to forward messages through his city, which was the Capital of the largest Kingdom. All of the writers went back to more profitable tasks and lived happily ever after (except for Rosenberg the Rich, who was heartbroken, and began to write to his hamsters, who chewed up his letters). The guild masters were upset at having nothing else to do, but soon saw how much money they saved, and were comforted.

But across all the kingdoms, many felt the void in their lives, for they were no longer as closely in touch with their fellow humans as they once had been.

The end.

Thomas Cox ...ihnp4!gargoyle!sphinx!benn

From: stein@hope.UUCP (Bruce Stein)
Newsgroups: net.unix-wizards
Subject: unix guru meeting
Date: 16 Mar 86 22:17:12 GMT

The following appeared in the June 84 issue of Dr. Dobb's Journal, and is reprinted here without permission for your enlightenment.

Get GUMMed

The Gurus of Unix Meeting of Minds (GUMM) takes place Wednesday, April 1, 2076 (check THAT in your perpetual calendar program), 14 feet above the ground directly in front of the Milpitas Gumps. Members will grep each other by the hand (after intro), yacc a lot, smoke filtered chroots in pipes, chown with forks, use the wc (unless uuclean), fseek nice zombie processes, strip, and sleep, but not, we hope, od. Three days will be devoted to discussion of the ramifications of whodo. Two seconds have been alloted for a complete rundown of all the user-friendly features of Unix. Seminars include "Everything You Know is Wrong", led by Tom Kempson, "Batman or Cat:man?" led by Richie Dennis "cc C? Si! Si!" led by Kerwin Bernighan, and "Document Unix,Are You Kidding?" led by Jan Yeats. No Reader Service No. is necessary because all GUGUs (Gurus of Unix Group of Users) already know everything we could tell them.

Bruce Stein on the Line

From: kevin@flinders.oz (Kevin Maciunas)
Newsgroups: aus.jokes
Subject: This IS true!
Date: 4 Apr 86 00:56:50 GMT

As hard as it may be to actually believe that this exists, it does!
[From a guarentee card for a Hong-Kong made air compressor]

o This is an excellent equipment with very few noise and excessive reliablilty. Though unfragile, it is also robust, and should not be belted.

o Circuit arrangements ensure environments and input current is best at both temperatures, including snow and hot.

o Very heavy fuses are supplied in plenty.

o Stability is too good on full battery and this should be lowered, but the input may be reduced to danger level if desired.

o The negative will be and the positive will not if supply polarity is incorrect, also, a humming noise will be introduced together with smoke. [My favorite ---KJM]

o When setting up, the best angle has no smoke and slight smell.

o For accessibilty without vandalism use the many entrances but switch them all off afterwards and before.

o When aligning, twiddle for strong current and prevent sparks.

o The motor should be good for ever, but pregnant wear-out may occur after a few summers if heat is applied.

[here comes the second best bit...]

DO NOT DOUBT THE GUARENTEE, IT IS BACKED BY MANY YEARS IN HONG KONG WITHOUT ODOR, PATIENCE OR THREAT.

>>>Reproduced without consent<<<
/Kevin

From: gjm@packard.UUCP (Gary J. Murakami)
Newsgroups: net.mail
Subject: Re: Mail overload and so-called "smart" mailers
Date: 14 Jan 86 22:05:32 GMT

> I am currious to see any statistics anyone may have about the volume
> of trafic that goes through such machines (mainly ihnp4), and how many
> lines such a site may have dedicated to gateway.

> ... Mikel Manitius ...

Sunday's UUCP traffic on ihnp4

```
H0 Id.,[Line],     ...... total total total    in    in    in   out   out   out
H1 Sys,sys!User    ......   num    KB   Bps   num    KB   Bps   num    KB   Bps
H2 -------------- ------ ----- ----- ----- ----- ----- ----- ----- ----- -----
I ihnp4 ................  4944 30925   103  2500  9420   121  2444 21504    97
```

Monday's UUCP traffic on ihnp4, plus some top UUCP neighbors:

```
I ihnp4 ................  7312 21886   100  3713  7892    97  3599 13994   101
S ucbvax ...............   545   719    62   447   639    60    98    79    82
S seismo ...............   464   668    66   300   421    61   164   247    76
S watmath ..............   208   543    94   138   314    85    70   229   108
S pur-ee ...............   199   293    96   105   120    82    94   172   108
S alberta ..............   164  1024   100    68    85    91    96   938   101
```

ihnp4 averages about 25 Mbytes/day via UUCP (plus similar amounts on two other networks). I've seen as many as a dozen simultaneous uucico's.

Networking hardware info follows.

-Gary

```
Processor:
        AT&T 3B20S Mod I

Networks:
        NSC HYPERchannel        IHCC General Purpose NSC network
        (50 Mbps CSMA/CD LAN)                63 hosts.nsc

        Datakit(r) VCS          AT&T Interlocation network (100+ nodes)
        (8 Mbps/node VCS WAN)               307 hosts.dk

        RJE/BLN                 Bell Labs Network (BLN) (ISO-like host PS)
        (RJE + 56Kbps backbone)             448 hosts.asp

        Phone                   AT&T CORNET, AT&T Communications
        (1200/2400 bps modems)             1375 hosts.acu

        - Phone/CORNET          - AT&T CORNET internal network
          (+ATTCOM access)                 1061 hosts.cor

        - Phone/ATTCOM          - AT&T Communications
                                            314 hosts.ext

Network-Interfaces:
        nusend
                HYPERchannel    1 x NSC interface
        UUCP (out)
                Datakit         1 x HS mux interface (pending)
                Datakit         3 x 9600 bps (+ Phone access)
                Datakit         5 x autobaud
                Phone/CORNET    2 x 1200 bps (+ Phone/ATTCOM access)
                Direct          2 x 9600 bps (ihnp1, ihnp3)
        UUCP (in)
                Datakit         11 x 9600 bps
                Phone/CORNET    8 x 1200 bps
                Phone/ATTCOM    8 x 1200 bps
        usend
                RJE/BLN         1 x 19.2 Kbps RJE
```

HYPERchannel is a trademark of Network Systems Corporation
Datakit VCS is a registered trademark of AT&T

From: dmr@dutoit.UUCP
Newsgroups: net.dcom
Subject: Q-bus integral lightning rod
Date: 7 Jan 86 09:48:18 GMT

These two items appeared side-by-side in this newsgroup:

> Can anyone point me to a surge protector for telephone lines. My
> house got hit by lightning last summer, and trashed my modem....

> Technical Magic Inc.... makes a dual Q-bus card ... which emulates
> a DLV11-E and has an integral modem on one port.... [It has] two jacks
> that look like RJ-11's....

Hmm.

Dennis Ritchie

From: phil@amdcad.UUCP (Phil Ngai)
Newsgroups: net.unix-wizards
Subject: Re: Locating a copy of "The Lions Document"
Date: 4 Jan 86 06:08:39 GMT

In article <182@ecrcvax.UUCP> dave@ecrcvax.UUCP (David Morton) writes:
>I thought that Bell or AT & T or whatever they were called at that
>time "persuaded" Lyons (not Lions) to forget the idea of publishing this
>documment which contains (contained) Bell source code ?
>Thus, to my ill informed knowledge, any published copy is illegal. I may
>be wrong on this issue. The document is widely available but not officially
>published.

That's right, you're wrong. In fact, AT&T is a distributor of the document. As part of a Unix source license (for V7 at least) you are allowed to request one copy. I have one. Pretty expensive book, if you don't qualify for an educational Unix license and the book was all you wanted. There are two volumes, a reformatted printout of the V6 source code, and comments on that source code.

Phil Ngai +1 408 749 5720
UUCP: {ucbvax,decwrl,ihnp4,allegra}!amdcad!phil
ARPA: amdcad!phil@decwrl.dec.com

From: ron@ronl.UUCP (Ron Saad)
Newsgroups: net.unix-wizards
Subject: Re: Locating a copy of "The Lions Document"
Date: 9 Jan 86 05:15:17 GMT

According to (800) 828-UNIX, they are no longer distributing the documents. The person I spoke to said that they 'had kept a copy for archival purposes' and totally eliminated distribution as of about 6 months ago. The person said this was due to the document being

Oh well.
Ron.

Ron Saad (4Z4UY)
Sys Adm - Center for Advanced
Technology in Telecommunications
Polytechnic Institute of New York

UUCP: ...{ihnp4,seismo}!{philabs,cmcl2}!ronl!ron
MAIL: 333 Jay St. Brooklyn, N.Y. 11201
PHONE: (718) 643-7303

From: bp@nyit.UUCP (Bruce Perens)
Newsgroups: net.unix-wizards
Subject: XINU book as a replacement for 'The Lions Document'
Date: 11 Jan 86 00:46:22 GMT

I suggest 'Operating System Design, The XINU Approach' as a replacement for 'The Lions Document'. I have both here, and find the XINU book covers most of the concepts needed to understand the UNIX kernel, and covers them much better than the Lyons document.

Here's the Library of Congress Cataloging in Publication Data:

Comer, Douglas
Operating System Design, the Xinu approach.

(Prentice-Hall software series)
Bibliography: p.
Includes index.
1. Xinu (Computer operating system)   2. System design
I. Title   II. Series
QA76.6.C625   1984   011.64'2   83-19270
ISBN 0-13-637539-1

UNIX is a trademark of ATT.
XINU is UNIX spelled backwards.

From: dave@pta.OZ (Dave Horsfall)
Newsgroups: aus.jokes
Subject: Some uses for an LSI-11
Date: 13 Feb 86 02:43:40 GMT

This appeared in my mail, and I thought it was too good to keep to myself:
(Thanks Alex - I hope you don't mind)

> Date: Thu, 6 Feb 86 15:24:26 pst
> From: pyrcorp!pyramid!csg (Carl S. Gutekunst)
> Subject: Curious applications of the LSI-11
>
> This was forwarded to me by a friend at DEC's Massachusetts R&D facility. It was posted on DEC's internal news net.
>
> ---
>
> Folks:
>
> I doubt this will help you market computers, by here goes anyway -- DELETE me if you've heard this one:
>
> My absolute favorite PDP11 application was described by Mike Allen of Lawrence Livermore Labs. As you know, the folks at the National Labs are mostly concerned with things that go *BANG* in the night, even though they usually can't mention those things by name.
>
> Development and testing of these things, however, requires a lot of data collection and reduction. That's where our computers come into the picture. Consider all of this in the context of the Threshold Test Ban Treaty, which limits us to tests of small things buried in the ground.
>
> One LSI-11 system is mounted in a truck over the borehole. The system contains core memory, and is connected via cables to the equipment under test. When the test occurs, the LSI-11 sits there and collects data till the shock wave reaches the surface of the earth. At that time, the LSI-11 automatically shuts itself off (since all of the data is now stored in the core memory). After the truck gets tossed into the air and falls back to the ground, maybe even right-side up, the LSI-11 turns itself back on if it can and transmits out the data that it has collected. If the LSI-11 was broken by the shock, then the experimenters take the core memory back to the shop, put it into a working computer, and read out the data.
>
> That's half the application.
>
> The other half involves another LSI-11 computer, except that this one is DOWN THE HOLE. Its job is to generate the data, up until such time as it is vaporized. After that, its job is complete. As you may have guessed, in these somewhat-beyond DEC Class C conditions, the MTBF of the computer is very short.

> Apparently, LLL considers LSI-11s to be consumables, much as we would consider scratch tapes or Kimwipes, and keeps a whole bunch of these things on the shelf, evaporating them as needed.
>
> Atlant

This article was followed by numerous postings from DEC employees offering their own LSI-11s and MicroVAXen for this application, if they were allowed to throw the switch. :-)

From: oster@ucblapis.berkeley.edu (David Phillip Oster)
Newsgroups: net.micro.mac
Subject: The MacPlus Massacre
Date: 22 Apr 86 22:57:35 GMT

The old macintoshs had 47 signatures embossed into the plastic of the case. The new mac backs are missing 16 and 1/2 of those 47 signatures. Steve Jobs is still on the back, but Jef Raskin, Andy Hertzfeld, and Bill Atkinson are three of the more famous Mac people who have been made non-persons.

Jef Raskin's name and half of Burrell Smith's name are obscured by the plastic of the new frame at the base of the Mac that surrounds the i/o ports.

The other names were on raised areas that have been made level with the rest of the back in the new Mac backs.

MacWorld, p.26, May/June '84 has a picture of the old Mac back, before the conductive RF-shield paint is sprayed on. Those of you with back issues might like to take a momment to look up the names of those who are no longer with us.

```
--- David Phillip Oster          -- "The goal of Computer Science is to
Arpa: oster@lapis.berkeley.edu   -- build something that will last at
Uucp: ucbvax!ucblapis!oster      -- least until we've finished building it."
```

From: asami@kddlab.UUCP (TOHRU ASAMI)
Newsgroups: net.mail,net.news
Subject: Mailing Address for JAPAN -- SYNTAX
Date: 13 Mar 86 02:12:04 GMT

There are many people in the world who do not know the correct mailing address to send a mail to JUNET (Japan Unix NETwork).

They try to find a correct path after sending many mails with incorrect addresses. The current gateway for JUNET is kddlab and almost all the mails for JUNET are delivered via kddlab.

The correct syntax to send a mail to JUNET is as follows:
  (1) IF YOU KNOW THE JUNET ADDRESS, say user@host.domains.JUNET
      kddlab!user@host.domains.JUNET
  (2) IF YOU KNOW THE UUCP ADDRESS, say user@host.UUCP or host!user
      kddlab!host!user

In this way, you can send a mail to Japanese people.

For more details, send a mail to me!
    asami@kddlab.kddlabs.JUNET
        or
    {seismo,mcvax,hplabs,ihnp4}!kddlab!asami
Tohru

P.S. Please read this message before sending an actual mail to
Japan.

From: td@alice.UucP (Tom Duff)
Newsgroups: net.math,net.lang.c
Subject: Re: Integer division
Date: 31 Jan 86 15:43:15 GMT

Pardon my flamage, but what sort of nonsense is this:
[in reference to divide instructions that give -(a/b)=(-a)/b]
>I have NEVER seen an instance where the first one is preferable. Not
>only is it not preferable, it is just incorrect.
Wrong! That's the definition. It can't be incorrect. It might be different
from what a number theorist wants, but by no stretch of the imagination can
it be called incorrect. A mathematician should be able to to handle this
elementary concept.
>Why such a routine
>has been allowed to be 50% inaccurate in every existing language all
>these years is beyond me.
Well, it's that way because that's the way it's defined in the ANSI Fortran
standard, and Fortran is probably a Good Thing for a computer to support --
certainly more important than niggling know-nothing number-theoretic nonsense.
Why does Fortran do it that way?
Probably because the IBM 701 did it that way. Why did the IBM 701
do it that way? Well, at the time people thought that a divide
instruction that satisfied certain identities was more important
than mod function behavior. Certainly in most of the applications
for which Fortran was designed (i.e. engineering numerical calculations)
the behavior of the mod function is of minimal interest.

In any case, why should you be worried that some operation you want to do
isn't primitive. Most programming languages don't provide arithmetic
on multivariate polynomials with arbitrary precision rational coefficients
either (which I want more often than I want a number-theoretic mod function.)
In any case, it's fairly easy to write:

```
        a=b%c
        if(a<0) a+=c
```

I can't believe that you couldn't discover this code sequence yourself.
(Note that it works whether the range of b%c is [0,c) or (-c,c) -- the
C language definition allows either.)

>[Whether CS people should even be *allowed* to make such mathematical
>decisions is another question. In C on UNIX, for example, one has
>log(-x) == log(x), a rather dangerous identity, not based on anything
>comprehensible. Thus, the implementation of general exponentiation,
>a**b = pow(a,b) = exp(b*log(a)) will silently return the wrong value
>if a is negative. (Try taking cube roots this way!)]

This sort of nonsense makes me wonder whether the writer should be allowed to make *any* sort of decision at all. No plausible definition of the log function will let you use it to take cube roots of arbitrary reals in this manner.

On a higher level of discourse, this writer (Matthew P Whiner) seems to think that mathematicians enjoy some sort of moral and intellectual superiority to engineers and computer scientists. Usually, this attitude is a symptom of envy, since mathematicians are so hard to employ, can't get decent salaries when they do find work, and have a much harder time raising grant money. The smart ones embrace computer science rather than denigrating it. The dull ones just say ''Computer Science? Pfui: that's not mathematics,'' thus demonstrating their lack of understanding of the nature of mathematics and of computer science.

In summary:
        It is better to remain silent and be thought a fool than
to speak up and remove all doubt.

From: keith@ssc-vax.UUCP (Keith Nemitz)
Newsgroups: net.micro.mac
Subject: The absolutely positively most REAL mac/amiga/st comparison!
Date: 20 Jan 86 22:18:43 GMT

I have had enough of the wimpy arguments of which 68000 personal computer is best. I decided after seeing what was available, I bought the new Cray-6000 personal computer. Read and weep suckers!

The NEW and terrific/exciting/awesome Cray-6000, designed by that famed bit-brained Johannes Livingston Cray himself will not only make you a believer of the personal computer craze it will change your entire reality frame.

Unlike all 'other' PC's before it the Cray-6000 uses high density marble based solid state electronics. Marble is MUCH denser than silicon. Data has less distance to travel due to the extremely dense nature of the hardware, and subsequently results in the fastest processing available anywhere. This speed is guaranteed to process any precision of pi in under two seconds, and it will automatically spool the digits to your Cray Osmosis (tm) printer at the same time! Why count stones, Whet or Dhry, when you can calculate the circumference of the known universe to 1 million places?

How 'user-friendly' is it? Well, you can toss away your windows and menus folks. The new Cray-6000 personal computer uses Suggestive EXpressions as the most widely common form of communication between human beings for understanding the exact meaning of your command. Not only will your every wish be fulfilled, but any possible legal hassals will be resolved before the command is executed! Just imagine yourself on the road for days, not having made that important vacumn cleaner sale which will put you on the saleman of the week list, when suddenly you grasp the Cray-6000 from your briefcase (its portable too folks) and in response to your frustrated grip the Cray-6000 builds an entire farm, the gulible farmer and his beautiful daughter before you can recite Newton's third law of motion. Creation is far superior to bit-mapped graphics, and you can stop fondling that stupid cigarette box with a tail too.

The styling of the Cray-6000 is superb. Inspired by the craftsman D'Milo, the Cray-6000 is lithsome and smooth. All features of the case are as authentic as the law allows and every Cray-6000 is guaranteed not to cause RF polution or double your money back. Multiple ports are available for third party add ons which can only be described by your imagination, and every conceivable pin-up (make that pin-out folks) is accessible.

All of the power and ease and beauty of the NEW Cray-6000 can be yours for only twice the national debt. (We didn't say is was cheap folks, but our floating credit line is just as amazing as the USA's.) You can't afford not to get one, and one for your loved ones as well.

THE industry has a long way to go folks.

keith
A9F4

From: root@gilbbs.UUCP (The Super User)
Newsgroups: net.news.sa
Subject: phoney addresses, can they be tracked?
Date: 31 Mar 86 01:43:15 GMT

I recently received a mail message which has an obviously phoney address as the return path. I am concerned about this mis-use of the network. Is there any way that we might be able to track such a posting?

The offensive posting header follows:

```
Received: by hplabs.ARPA ; Thu, 27 Mar 86 00:03:55 pst
Received: by hao.NCAR (4.12/4.7)
        id AA08482; Thu, 27 Mar 86 00:26:14 mst
Received: by seismo.ARPA (4.12/4.7)
        id AA03491; Thu, 27 Mar 86 13:45:53 est
Received: by mcvax.UUCP (4.12/4.7)
        id AA09712; Thu, 27 Mar 86 10:48:31 gmt
Received: by moskvax.USSR (4.12/4.7)
        id AA08123; Wed, 26 Mar 86 23:19:33 ust
Received: by kremlin.USSR (4.12/4.7)
        id AA03099; Wed, 26 Mar 86 23:15:03 ust
Date: Thu, 27 Mar 86 00:06:02 mst
Message-Id: <8603270706.AA08411@kremlin.USSR>
From: hplabs!hao!kremlin!andropov (Yuri's Ghost)
Subject: Re: looking for CSUC *TALK* users
Apparently-To: moskvax!mcvax!seismo!hao!hplabs!qantel!gilbbs!mc68020
```

[body of message removed]

Can anyone assist me in determining the origin of this letter?

Thanks in advance.

Sincerely,
Yuri

From: phil@amdcad.UUCP (Phil Ngai)
Newsgroups: net.news.config
Subject: Re: phri-pesnta link closing down
Date: 18 Jan 86 18:55:05 GMT

In article <2136@phri.UUCP> roy@phri.UUCP (Roy Smith) writes:
>Pesnta can be reached through ihnp4, amd, hplabs, etc.

Note that the site "amd" is effectively without an administrator, with all that that means. For example, last time I looked, /dev/tty was an ordinary file and all the dialers were broken. Consider this a suggestion to route around it whenever possible.

Phil Ngai +1 408 749 5720
UUCP: {ucbvax,decwrl,ihnp4,allegra}!amdcad!phil
ARPA: amdcad!phil@decwrl.dec.com

From: sroth@muddcs.UUCP (Steve Roth)
Newsgroups: net.sources.d,net.micro.amiga
Subject: Comparison of Operating Systems
Date: 31 Mar 86 22:26:49 GMT

The recent postings of various versions of microemacs have provided an interesting comparison of operating systems. One version of microemacs that we have been working with has code in it for a number of operating systems including VAX/VMS, UNIX V7, AmigaDos, and MS-DOS. Below is a comparison of the number of lines of C code for certain tasks under each of those operating systems.

	Lines of Code:			
Task	VAX/VMS	UNIX V7	AmigaDos	MS-DOS
Spawn CLI	32	13	7	5
Spawn Command	38	14	10	7
Prepare Term for edit	37	5	1	0
Clean up after edit	10	1	1	0
Write char	3	1	1	1
Flush output	11	1	0	0
Read char, no echo	25	1	3	1

Furthermore, we ran termio.c (the terminal i/o routines, which are the majority of the system-dependent code) through the C preprocessor for each operating system. After deleting references to include files in each one, we found:

```
    Number of #includes removed:
                    5          1          0          1
    Number of characters of preprocessor output:
                    7806       3964       3770       3675
```

Based on past experience, these statistics seem quite indicative of the relative level of difficulty of system-level programming on each of these operating systems. We hope you found this information as amusing as we did. Have a nice day!

Steve Roth
Marc Sugiyama

```
Steve Roth               Harvey Mudd College, Claremont California
ucbvax!sdcsvax!scgvaxd!muddcs!sroth      -or-    STEVE@YMIR.BITNET
```

From: reid@glacier.ARPA (Brian Reid)
Newsgroups: net.news.group
Subject: Re: Looking for newsgroup involved in PARAPSYCHOLOGY
Date: 24 Feb 86 05:40:39 GMT

In article <406@zaphod.UUCP> rond@zaphod.UUCP (Ron Domes) writes:

```
>If anybody out there knows of a newsgroup that deals with parapsychology I
>would like to hear about them.
```

If you had any talent for parapsychology you wouldn't have to ask this question. You would KNOW where the newsgroup was. Forget it.

```
        Brian Reid      decwrl!glacier!reid
        Stanford        reid@SU-Glacier.ARPA
```

From: lrr@princeton.UUCP (Larry Rogers)
Newsgroups: net.unix-wizards
Subject: Re: vax 785 vmstat
Date: 24 Jan 86 19:15:16 GMT

In article <710@dicomed.UUCP> salmi@dicomed.UUCP (salmi) writes:

```
>i have just recently brought up ultrix 1.1 on a vax 785.  after running the
>recently distributed dhrystone benchmarks on the new system, i was appalled at
>the outcome, somethint like 50000 passes in 102 seconds! (480 stones/sec?)
>
>running vmstat on the system gives the following results, with one user on the
>system...
>
># vmstat 2
> procs     memory                       page      disk  faults         cpu
> r b w   avm  fre  re at  pi  po  fr  de  sr r0 x1 x2 x3  in   sy  cs us sy id
> 1 0 0   445 7769  0  0   0   0   0   0   0  3  0  0  04994  18   4  1 82 17
> 1 0 0   445 7769  0  0   0   0   0   0   0  0  0  0  05156 163   5  0 83 17
> 0 0 0   157 7742  0  0   0   0   0   0   0  0  0  0  05192  13   0  0 84 16
> 0 0 0   375 7742  0  0   0   0   0   0   0  0  0  0  05201   8   0  0 82 18
>
>the interupts per second are averaging more the 5000 on an 'idle' system.  our
>loaded up vax 750 running 4.2 gives the following results from vmstat...
>
># vmstat 2
> procs     memory                          page       disk  faults          cpu
> r b w   avm  fre  re at  pi  po  fr  de  sr h0 r0 u0 f0  in  sy  cs us sy id
> 2 0 0  1139 3207  0  1   0   0   0   0   0  3  0  1  0   4  68  20 18 25 57
> 1 0 0   911 3185  0  0   0   0   0   0   0  0  0  0  0   3 239  18  6 19 74
> 1 1 0   868 3176  0  0   0   0   0   0   0  0  0  0  0   6 111  10  4 13 83
> 0 0 0   629 3173  0  0   0   0   0   0   0  0  0  0  0   2  88   9  6 16 77
>
>all of the ports had been shut off on the 785, thinking that a runaway getty
>may have been the culprit, but no change was evident.  another thought was the
>DEC ethernet board may be the bad guy.
>
>anyone got any clues/ideas?  the 785 must be fast, as the response time was
>fairly fast even with the 5000+ interupts/second running rampant.
>
>any/all ideas would be appreciated.  please respond directly to me at:
>
>ihnp4!dicomed!salmi
>
>john salmi
>dicomed corporation
>minneapolis, mn
>612/885-3000
>
>and, as always, thanks in advance :-)
```

I also have Ultrix 1.1 running on a VAX-11/785 with the ports disabled and the Deuna enables and I get:

```
 procs     memory                page                  disk        faults     cpu
 r b w  avm    fre    re at  pi  po  fr  de  sr r0 r1 r2 r3  in  sy  cs us sy id
 2 3 0 66367  58438   0  0   0   0   0   0   0  0  0  1  0   2   7  13 92  2  6
 2 3 0 66367  58438   0  2   1   0   0   0   0  0  0  3  0  19 341  29 96  4  0
 2 3 0 66367  58438   0  1   0   0   0   0   0  0  0  0  0  15 277  26 96  4  0
 2 0 0 66291  58408   0  0   0   0   0   0   0  0  0  0  0   7 113  19 93  7  0
 2 0 0 66344  58394   0  0   0   0   0   0   0  0  0  1  0  10  53  19 95  5  0
```

Only a normal number of interrupts. If you look at the output carefully, you may get the impression that this machine has (66344+58394)K of memory, roughly 124M. This is indeed correct as we have 128M on this baby. We don't page much.

Larry Rogers
Princeton University
Department of Computer Science
Engineering Quadrangle Building, Room C334
Princeton, NJ 08544

UUCP: princeton!lrr
PHONE: 609 452 6483

From: roy@phri.UUCP (Roy Smith)
Newsgroups: net.unix-wizards,net.dcom
Subject: Re: /usr/lib/aliases name format
Date: 18 Jan 86 16:23:33 GMT

> For further quesions, read the "sendmail" documentation.
>
> Guy Harris

Was this meant to be a joke? Yes, the "sendmail" documentation certainly does provide a source for questions; unfortunately it doesn't provide much in the way of answers.

Roy Smith <allegra!phri!roy>
System Administrator, Public Health Research Institute
455 First Avenue, New York, NY 10016

From: hedrick@topaz.RUTGERS.EDU (Charles Hedrick)
Newsgroups: net.unix-wizards,net.mail
Subject: a brief tutorial on sendmail rules
Date: 16 Jan 86 00:41:38 GMT

A previous message suggested using "sendmail -bt" to see how sendmail is going to process an address. This is indeed a handy command for testing how an address will be processed. However the instructions given were not quite right. To see how sendmail is going to deliver mail to a given address, a reasonable thing to type is

```
        sendmail -bt
        0,4 address
```

Even this isn't quite right, but with "normal" rule sets it should work.

Because there is so much confusion about sendmail rules, the rest of this message contains a brief tutorial. My own opinion of sendmail is that it is quite a good piece of work. Many people have complained about the difficulty of understanding sendmail rule sets. However I have also worked with mailers that code address processing directly into the program. I much prefer sendmail. The real problem is not with sendmail, but with the rules. The rules normally shipped from Berkeley have lots of code that does strange Berkeley-specific things, and they are not commented. Also, typical complex rule sets are trying to handle lots of things, forwarding mail among several different mail systems with incompatible addressing conventions. A rule set to handle just old-style (non-domain) UUCP mail would be very simple and easy to understand. But real rule sets are not doing simple things, so they are not simple.

For those not familiar with sendmail, -bt invokes the rule tester. It lets you type a set of rule numbers and an address, and then shows you what the rules will do to that address. In addition, rule test mode automatically applies rule 3 before whatever rule you ask it to apply. As we will see shortly, this is a reasonable thing to do.

Before describing the rule sets, let me define two terms: "header" and "envelope". Header refers to the lines at the beginning of the message, starting with "from:", "to:", "subject:", etc. Sendmail does process these lines. E.g. with uucp mail it will add its own host name at the beginning of the from line, so that the final recipient stands some change of replying to the message. However sendmail normally does not depend upon the from and to lines to perform its actual delivery. It has more direct knowledge, passed on to it from the program that generated the mail, or if it came from another site, the mailer at that site. This information is referred to as the "envelope", since it is like the addresses on the outside of an envelope. For Arpanet mail, the envelope is passed to the next site by the MAIL FROM: and RCPT TO: commands. For UUCP mail, it is passed on as arguments to the remote rmail command. To see why there have to be separate addresses "on the envelope", consider what happens when you send mail to "john@vax, mary@sun". Two copies of the message will be dispatched, one to vax and the other to sun. The "to: " line in the headers will show both addresses. However the envelope will show only the right address that we want this copy to go to. The copy sent to vax will show "john@vax" and the copy sent to sun will show "mary@sun". If sendmail had to look at the "to: " line, it would never know which of the addresses shown there it was responsible for handling.

Anyway, here is what the rules do:

3: always done first. This turns addresses from their normal textual form into a form that the rest of the rules understand. In most cases, all it does it put < > around the name of the host that is next in line. Thus foo@bar turns into foo<@bar>. However it also does a few transformations. E.g. it turns foo!bar!user into bar!user<@foo.UUCP>. Since sendmail accepts either ! syntax or @....UUCP syntax, rule 3 standardizes on @ syntax. It also does a few other minor things. But you won't be far off if you just think of it as adding < > around the host name.

4: always done last. This turns addresses from internal form back into external form. It removes the < > around the host name, and turns foo@bar.UUCP back into bar!foo. Again, there are one or two other minor things, but you won't be too far off if you think of 4 as just removing the < > around the host name.

0: This is the rule that handles the destination address on the envelope. It is in some sense the primary rule. It returns a triple: protocol, host, user. The protocol is usually one of local, TCP, or UUCP. At the moment, it figures this out syntactically. In our rule set, hosts ending in .UUCP are handled by UUCP, the current host is local, and everything else is TCP. As domains are integrated into UUCP, obviously this rule is going to change. This rule does very little other than simply look at the format of the host name, though as usual a few other details are involved (e.g. it removes the local host. So myhost!foo!bar will be sent directly to foo).

1 and 2 are protocol-independent transformations used for sender and recipient lines in the header (i.e. from: and to: lines). In our rule sets, they don't do anything.

Each protocol has its own rules to use for sender and recipient lines in the header. E.g. UUCP rules might add the local host name to the beginning of the from line and remove it from the to line. In our rule set, the complexities in these rules are primarily caused by forwarding between UUCP and TCP. The line that defines the mailer for a protocol lists the rule to use for source and recipient, in the S= and R=.

Finally, here is the exact sequence in which these rules are used. For example, the first line means that the destination specified in the envelope is processed first by rule 3, then rule 0, then rule 4.

```
envelope recipient:    3,0,4  [actually rule 4 is applied only to the
                                user name portion of what rule 0 returns]
envelope sender:       3,1,4
header recipient:      3,2,xx,4  [xx is the rule number specified in R=]
header sender:         3,1,xx,4  [xx is the rule number specified in S=]
```

I have the impression that the sender from the envelope (the return-path) may actually get processed twice, once by 3,1,4 and the second time by 3,1,xx,4. However I'm not sure about that.

Now for the format of the rules themselves. I'm just going to show some examples, since sendmail comes with a reference manual, which you can refer to. However these examples are probably enough to let you understand any set of rules that makes sense in the first place (which the normal rules do not). This example is from our UUCP definition. It a simplified version of the set of rules used to process the sender specification. As such, the major thing it has to do is to add our host name to the beginning, so that the guy at the end will know that the mail went through us.

```
S13
R$+<@$-.UUCP>          $2!$1               u@host.UUCP => host!u
R$=U!$+                $2                  strip local name
R$+                    $:$U!$1             stick on our host name
```

Briefly, the first rule turns the address from the form foo<@bar.UUCP> back into bar!foo. The second rule removes our local host name, if it happens to be there already, so we don't get it twice. The third rule adds our host name to the beginning.

S13 says that this is the beginning of a new rule set, number 13.

```
R$+<@$-.UUCP>          $2!$1               u@host.UUCP => host!u
```

R says that this is a rule. The thing immediately after it, $+<@$-.UUCP> is a pattern. If this pattern matches the address, then the rule "triggers". If the rule triggers, the address is replaced with the "right hand side", i.e. what is after the tab(s). In this rule, the right hand sie is $2!$1. The thing after the next tab(s) is a comment. This rule is used in processing UUCP addresses. As noted above, by the time we get to it, rule 3 has already been applied. So if we had a UUCP address of the form host1!host2!user, it would now be in the form host2!user<@host1.UUCP>. This does match the pattern:

```
        $+        <@$-    .UUCP>
        host2!user<@host1.UUCP>
```

$+ and $- are "wildcards" that match anything. $- will match exactly one word, while $+ will match any number. (By the way, with the increasing use of domains, this production should probably use $+.UUCP, not $-.UUCP.) Since the pattern matches, we replace this with the "right hand side" of the rule, $2!$1. $ followed by a digit means the Nth thing matched by a wildcard. In this case there were two wildcards, so

```
   $1 = host2!user
   $2 = host1
```

The final result is

```
   host1!host2!user
```

As you can see, we have simply turned UUCP addresses from the format produced by rule 3 back into normal ! format.

The second rule is

```
R$=U!$+                 $2                          strip local name
```

This is needed because there are situations in which our host name ends up on the beginning of the recipient address. Since we are about to add our host name, we don't want it to be there twice. So if it was there before, we remove it. $= is used to see if something is a member of a specified "class". U happens to be a list of our UUCP host name and any nicknames. So $=U!$+ matches any address that begins with our host name or nickname, then !, then anything else. Suppose we had topaz!host1!host2!user. The match would be

```
     $=U  !$+
     topaz!host1!host2!user
```

The result of the match is that

```
     $1 = topaz
     $2 = host1!host2!user
```

Since the right hand side of this rule is simply "$2", the result is

```
     host1!host2!user
```

I.e. we have removed the topaz from the beginning. By the way, the class U used by the rule would have been defined earlier in the file by the statement

```
CUtopaz ru-topaz
```

C defines a class. U is the name of the class. The rest of the line is the list of things that will be in the class.

Finally we have the rule

```
R$+                     $:$U!$1                    stick on our host name
```

The $+ matches anything. In this case the name is host1!host2!user, so the result of the match is

```
        $1 = host1!host2!user
```

The result looks slightly obscure. $: is a tag that says to do this only once. The problem is that this rule always applies, since the pattern matches anything. Normally, rules are applied over and over, as long as they apply. In this case, the result would be an infinite loop. Putting $: at the beginning says to do it only once. $U says to use the value of the macro U. Earlier in the file we defined U as our UUCP host name, with a definition

```
DUtopaz
```

Note that there can be a class and a macro with the same name. $=U tests whether something is in the class U. $U is replaced by the value of the macro U.

So the final value of this rule, $:$U!$1, is

```
     topaz!host1!host2!user
```

So this rule has managed to add our host name to the beginning, as it was supposed to. Since there are no further rules in the set (the next line is the end of file or the beginning of a new rule set), this value is returned.

There are several more magic things that can appear in a pattern. The most important are:

$* - this is another wild card. It is similar to $+, but $+ matches anything, whereas $* matches both anything and nothing. I.e. $+ matches 1 or more tokens and $* matches 0 or more tokens. So here is a list of the wildcards I have mentioned:

$*	0 or more
$+	1 or more
$-	exactly 1
$=x	any member of class x

A typical example of $* is a production where we aren't sure whether the user name is before or after the host name:

```
R$*<@$+.UUCP>$*         $@$1<@$2.UUCP>$3
```

This production would test for the host name ending in .UUCP, and return immediately. $@ is a flag you haven't seen yet. It is simply a return statement. It causes the right hand side of this rule to be returned as the final value of this rule set.

The other magic thing I will mention is $>. This is a subroutine call. Here is an example taken from rule set 24, which is used to process recipients in TCP mail. Its purpose is to handle the situation where we might have an address like topaz!user@red. (Our host name is topaz. Red is a local host that we talk to via TCP.) I.e. someone is asking us to relay mail to red. Rule 3 will have turned this into user@red<@topaz.UUCP>. What we want to do is get rid of the topaz.UUCP and treat red as the host. (Rule set 0 would do this for the recipient on the envelope. This rule is used for the to: field in the header.) Here is the rule.

```
R$+<@$=U.UUCP>          $@$>9$1                         in case local!a@b
```

The pattern matches our example, as follows:

```
   $+      <@$=U  .UUCP>
   user@red<@topaz.UUCP>
```

Recall that $+ matches anything and $=U tests whether something is our UUCP host name or one of our nicknames. The result of the match is

```
   $1 = user@red
   $2 = topaz
```

The right hand side is $@$>9$1. The $@ is the tag saying to stop the rule set here and return this value. $>9 is a subroutine call. It says to take the right hand side, pass it to rule set 9, and then use the value of rule set 9. The actual right hand side is simply $1, which in this case is user@red. Here is rule set 9:

```
S9
R$*<$*>$*               $1$2$3                          defocus
R$+                     $:$>3$1                         make canonical
R$+                     $@$>24$1                        and do 24 again
```

The first rule simply removes < >. It is sort of a quick and dirty version of rule 4. In fact we have no < > left, since we have removed the <@topaz.UUCP>. So this rule does not trigger. (Now that I think about it, I suspect it is probably never going to trigger, and so is not needed.)

The next rule is a simple subroutine call. It matches anything ($* matches any 0 or more tokens). The right hand side is $:$>3$1 The $: says to do it only once. Since the rule matches anything, you need this, or you will have an infinite loop. The $>3 says to call rule 3 as a subroutine. The $1 is the actual right hand side. Since the left hand side matched the whole address, what this rule does is simply call rule set 3 on the whole address. Recall that rule set 3 basically locates the host name and puts < > around it. So in this case the result is user<@red>. As you can see, it was not enough to remove <@topaz.UUCP>. That leaves us with no host name. We have to call rule 3 to find the current host name and put < > around it.

The last rule is really just a goto statement. The pattern is $+, which matches anything, so it always triggers. The right hand side is $@$>24$1. The $@ is the return tag. It says to stop this rule set and return that value. $>24 says to call rule set 24. The actual right hand side is $1, so we call rule set 24 with the whole address. If you recall, this ruleset (9) was called from the middle of 24 when we found user@red<@topaz.UUCP>. So what we have done is to change this into user<@red> and say to start rule set 24 over again.

I hope you have found this exposition useful. As a final convenience, here is a "reference card" for reading rule sets. Note that this contains only operators used by the rules. There are plenty of other facilities used in the configuration section which I am not documenting here. (I'd love to see someone produce a complete

```
reference card.)

wildcards:
   $*   0 or more tokens
   $+   1 or more tokens
   $-   exactly one token
   $=x  member of class x (x must be a letter, lower/upper case distinct)
   $~x  not a member of class x

macro values (usable in pattern or on right hand side)
   $x   value of macro x (x must be a letter, lower/upper case distinct)
        At least on the Pyramid, $x is replaced by the macro's value
        when the sendmail.cf file is being read in.

on the right hand side:
   $n   string matched by the Nth wildcard
   $>n  call rule set N as a subroutine
   $@   return
   $:   only do this rule once

in rule 0, defining the return value
   $#   protocol
   $@   host
   $:   user

Rutgers extensions, usable only on right hand side
   $%n  take the string matched by the Nth wildcard, look it up in
        /etc/hosts, and if found use the primary host name
   $&x  use the current value of macro x.  x must be a letter.
        upper and lower case are treated as distinct.
```

From: chris@umcp-cs.UUCP (Chris Torek)
Newsgroups: net.unix-wizards
Subject: Re: # sign in shell/cshell scripts...
Date: 17 Apr 86 20:30:30 GMT

In article <237@Shasta.ARPA> ALEX@SU-SIERRA.ARPA writes:
>I'm encountering weird problems with sh (and sometimes csh)
>dying on some shell scripts depending on whether or not my first line in
>the script is a comment like:
># This script does this (but not that)

Once upon a time, there was the shell. Since there was only 'the' shell, there was no trouble deciding how to run a script: run it with *the* shell. It worked, and everyone was happy.

Along came progress, and wrote another shell. The people thought this was good, for now they could choose their own shell. So some chose the one, and some the other, and they wrote shell scripts and were happy. But one day someone who used the 'other' shell ran a script by someone who use the 'other other' shell, and alas! it bombed spectacularly. The people wailed and called upon their Guru for help.

'Well,' said the Guru, 'I see the problem. The one shell and the other are not compatible. We need to make sure that the shells know which other shell to use to run each script. And lo! the one shell has a ``comment'' called ``:'', and the other a true comment called ``#''. I hereby decree that henceforth, the one shell will run scripts that start with ``:'', and the other those that start with ``#''.' And it was so, and the people were happy.

But progress was not finished. This time he noticed that only shells ran scripts, and thought that if the kernel too could run scripts, that this would be good, and the people would be happy. So he wrote more code, and now the kernel could run scripts, but only if they began with the magic incantation: '#!', and told the kernel which shell ran the script. And it was so, and the people were confused.

For the '#!' looked like a 'comment'. Though the kernel could see the '#!' and run a shell, it would not do so unless certain magic bits were set. And if the incantation were mispronounced, that too could stop the kernel, which after all was not omniscient. And so the people wailed, but alas!, the Guru did not respond. And so it was, and still it is today.

* * * * *

Anyway, you will get best results from a 4BSD machine by using

```
#! /bin/sh
```

or

```
#! /bin/csh
```

as the first line of your script. '#! /bin/csh -f' is also helpful on occasion.

--
In-Real-Life: Chris Torek, Univ of MD Comp Sci Dept (+1 301 454 1415)
UUCP: seismo!umcp-cs!chris
CSNet: chris@umcp-cs ARPA: chris@mimsy.umd.edu

From: mash@mips.UUCP (John Mashey)
Newsgroups: net.unix-wizards
Subject: Re: Shell history, true facts, but long
Date: 18 Mar 86 09:05:12 GMT

At Todd Williams' request:

1) In the Beginning, there was the shell, /bin/sh, by Ken Thompson. I can't remember what version this appeared in, but it was well before my time, i.e., it sure looked and acted like a shell before late 73. For more info on the early days, see DMR's paper in the 2nd UNIX BLTJ. Almost nothing was builtin (on purpose), but there were separate if/goto commands. Goto acted by seeking the file pointer it shared with its parent shell. For a single-stream command-line interpreter, most, if not all of the fundamental (in my opinion) ideas were already present. See reference [1].

2) During the period 1973-1976, lots of people hacked on that shell in various and sundry ways. USG (UNIX Support Group at Murray Hill) added some things, and so did we:

3) The "PWB Shell" first appeared in mid-1975. It derived from a set of requirements and suggestions from me in early 1975 in trying to do serious shell programming. In mid-1975, the shell acquired variables, including 3 that were derived from per-process data. This is where the idea of more generalized path-searching came in. Some of the existing support commands were beefed up, including if/else/endif.
Most of this work was done by Alan Glasser or Dick Haight. I ended up taking this over, doing performance analysis, moving most of the control structure commands into the shell, adding switch,, while, etc, etc.
By this time there was serious shell programming in massive amounts.
This had mostly stabilized by early/mid 1976, although a few more features were added in early 1977. For various reasons this came to be called the "Mashey shell", a term I universally abhorred, since it was fundamentally still a Thompson shell with selected additions and tunings by various of us, constrained to be mostly upward compatible, leading to some things that I found less than esthetic. Even less than esthetic was the title of a later internal Bell Labs course: "Bourne Shell Programming for Mashey Shell Programmers".
See [2] and [3].

4) The Bourne shell work started either in early 1976, or maybe late 1975. The first version was VERY different; for example, it used $ as the escape character instead of \. During 1976 there was a long sequence of modification, dialog, discussion, with various players in shifting alliances [like the game Diplomacy]. Besides the Computing Research crew, SRB himself, and me, this is when (as I recall) Dave Korn started getting involved (as heavy earlier user and extender of Bourne sh).
To summarize, some of the Bourne shell's fundamentals came from the Thompson shell, but some were new. Some of the added semantics, and a few pieces of syntax came either from the PWB shell, or by agreement on what it ought to be, given that everything was changing anyway. [For example, the environment idea was a much-needed generalization of the restricted PWB variables; this came from DMR, SRB, and I.] This did become the standard shell, although it took PWBers a while to convert, since they had huge investments in existing shell procedures. [Not that many had been written in the standard V6 or earlier shells.] See [4].
As a minor tidbit, how many 68K programmers out there have cursed the shell's trapping of memory fault to grow it's storage? I admit to causing this, having goaded SRB into it by moaning at the speed of the then-current Bourne shell vs the faster PWB shell.

5) I haven't tracked the exact influences on the C-shell. However, I believe that certain pieces were at least derived from the PWB shell's documentation (implementation, if not always exact syntax, of control cmds): if-the-else-endif, break, breaksw, onintr, while-end.
Ask Bill where he got everything. Finally, note that there have been a number of other interesting and important shells around, but I'm tired, and I think more people understand the later work.

----

[1] K. Thompson, "The UNIX Command Language", in Structured Programming, Infotech State of the Art Report, Nicholson House, Maidenhead, Berkshire, England: Infotech International Ltd (March 1975), 375-384.

Classic Thompson-style paper of maximal insights per word, incl: "Many familiar computing 'concepts' are missing from UNIX. Files have no records. There are no access methods. User programs contain no system buffers. There are no file types. These concepts fill a much needed gap..."

[2] J. R. Mashey, "Using a Command Language as a High-Level Programming Language", Proc. 2nd Int. Conf on Software Engineering, IEEE (Oct 13-15, 1976), 169-176.
Includes some usage data; state of the world in PWBland in mid-1976.

[3] T. A. Dolotta, J. R. Mashey, "Using a Command Language as the Primary Programming Tool", in Command Language Directions: Proc. IFIP Working Conference on Command Languauges, Sept 10-14, 1979, D. Beech, Ed, North-Holland, AMsterdam, 1980.
3 years later; more data; Bourne shell(called here nsh) included.

[4] S. R. Bourne, "An Introduction to the UNIX Shell", BSTJ 57, No 6, Part 2 (Jul-Aug 1978), 2797-2822.

-john mashey
UUCP: {decvax,ucbvax,ihnp4}!decwrl!mips!mash
DDD: 408-720-1700
USPS: MIPS Computer Systems, 930 E. Arques, Sunnyvale, CA 94086

From: piers@basser.oz (Piers Lauder)
Newsgroups: aus.acsnet
Subject: SYSTEM V ioctl bug
Date: 20 Mar 86 04:34:58 GMT

It seems that some versions of SYSTEM V have an obscure bug that makes the success of system calls doing i/o to stack space problematic.

In particular, this seems to affect those parts of the ACSnet source that declare "termio" structures on the stack which are then used in "ioctls".

The fix is to prepend the word "static" to all those declarations.

PS: I might point out that anyone thinking of comenting on this "feature" should be aware of the very strict libel laws in this country.

From: mike@hcradm.UUCP (Mike Tilson)
Newsgroups: net.news,net.usenix
Subject: Request for UUCP and/or USENET proposals
Date: 7 Apr 86 21:25:57 GMT

The combination UUCP mail service and USENET news service has proved to be very attractive and useful to our community. However, there are a number of problems with the current set of services, due to technical, organizational, and financial problems. Some problems include:

- Service is erratic and unreliable.
- Security is virtually non-existent.
- Efficiency is low. Routing of mail and news does not minimize cost or maximize reliability.
- Overall cost is extremely high, in terms of telecommunications charges, CPU utilization, and disk utilization, especially when the total over all sites is considered. Many of the costs are currently "hidden".
- Costs are not allocated fairly. Continued service depends on the continued goodwill of voluntary "backbone" sites who absorb many of the costs. USENET (the primary cost generator) is in fact very vulnerable to a chain reaction -- if a few key sites drop out, the load and cost at the remaining sites increases dramatically, leading to more dropouts, and so on.
- Traffic is growing without bound, with no sign of abating.
- There are few standards or too many (conflicting) standards.
- There is no established uniform way of addressing messages.
- The "useful information" content of netnews appears low.

As the network grows these problems seem to be getting worse, and many fear the ultimate demise of a useful tool.

The Usenix Association has funded some efforts in aid of UUCP/USENET. These include the network mapping project and the Stargate experiment.

The Association would now like to receive proposals by which we could fund or aid projects that would solve some or most of the UUCP/USENET network problems. At this point, we do not have any set agenda or pre-conceived notions; we are open to any reasonable proposal.

We would like proposals that answer the following questions:

- What user needs are met? What user problems are solved? How was this determined?
- Who is going to take action, and how are they organized? What person(s) or group(s) are going to take the lead in making things happen? If needed, who will take care of any proposed on-going operations.
- How much does it cost? What funding is required from Usenix? Will Usenix recover any of its expenditures? If needed, where will on-going operating funds come from?
- What technical solutions are proposed? What technology exists, and what must be developed?

The order of the above is significant; we are most interested in a good analysis of user needs and problems, along with an understanding of who exactly will meet these needs. We want to know: "What functions are to accomplished, and who will lead the charge?"

Next we want to know how much money is wanted, and whether any solutions seem technically feasible. However, we believe these issues are much easier to deal with if the first questions have good answers.

This request for proposals is not a formal bid. The Usenix Association has not yet allocated any funds for this purpose, nor has it made a commitment that it will indeed accept any proposal. However, we do feel that modest funds could be made available for one or a few deserving projects. The Association feels it could fund projects on the order of $10,000 if justified. Larger numbers would almost certainly require some cost recovery. As an upper bound, we would consider $50,000 to be financially out of the question without of an assurance of some cost recovery in the near future. The Association realizes that the amount of money it can make available is small relative to the millions of dollars being spent on UUCP/USENET. However, we feel we can play a role in planting the seeds of a solution.

The Association is willing to consider a wide variety of projects, including research, seed capital for an operating organization, studies intended to attract interest from telecommunications carriers, or small projects that directly solve at least a few problems.

We would like to be able to discuss proposals at the next Usenix Board meeting in June. If a proposal can not be prepared by then, we would still be interested in a one page outline. Proposals should be sent to:

UUCP/USENET Proposals
Usenix Association
P.O. Box 7
El Cerrito, CA 94530

Proposals can also be sent electronically to "usenix!jim".

Please direct followup discussion of this proposal to net.usenix.

Michael Tilson
on behalf of the Board of Directors of the Usenix Association.

{decvax,utzoo,usenix...}!hcr!hcradm!mike

From: jrw@hropus.UUCP (Jim Webb)
Newsgroups: net.bugs.usg
Subject: Re: Bug in who(1)?
Date: 24 Jan 86 18:20:09 GMT

```
> I discovered something MIGHTY INTERESTING the other day while logged-in as
> root on a 3B2/300 running Sys5r2.  I tried and repeated it on a 3B2/400
> running the same.
>
> # pwd
> /usr/foobar
> # ls
> file1
> file2
> file3
> # who -b .
> # ls
> . not found
> # cd ..
> ..: bad directory
> # cd /usr
> # ls | grep foobar
> #
```

```
>
> I cannot repeat this as an ordinary user even on a directory which I own
> and which I own the parent of and have write permission on both.  It looks
> much like an unlink() is somehow getting done on the argument of the
> "who -b" but since I don't have source for the 3B2 "who," I don't know
> for sure.  Could someone check this out for me?  When this happened,
> even fsck could not retrieve the lost files (arrrrgh!!!!), it just complained
> about "bad free list".  The '.' was a typo (shaky hands) but it sure caused
> me a lot of headaches from a command that SURELY must be nondestructive :-).
>
> Thanks of course, well in advance.
>
> dan levy
> Path: ..!ihnp4!ttrdc!levy
```

HMM, intriguing little bug you found. I tried this on all of my SVR2 machines, not only 3B2's, and had SIMILAR findings. The problem lies in the utmp library routines. This is what is going on:

1) who calls utmpname with argument "." to change it from /etc/utmp
2) getutent does a check on this "." and sees that it is NOT
   a file filled with utmp-structures, SO...
3) it unlinks it and then links it again, thus creating a new one.

!!! NOT GOOD IF YOU ARE ROOT !!!

Anyway, all is not lost, and you CAN get all of your files back, so sorry about the following long narrative, but here we go...

```
# cd /tmp
# mkdir foobar
# cd foobar
# >file1
# >file2
# >file3
# ls -lai
total 10
   57 drwxr-xr-x   2 root     sys           80 Jan 24 12:54 .
    2 drwxrwxrwx   7 root     root        4480 Jan 24 12:54 ..
   83 -rw-r--r--   1 root     sys            0 Jan 24 12:54 file1
   12 -rw-r--r--   1 root     sys            0 Jan 24 12:54 file2
   69 -rw-r--r--   1 root     sys            0 Jan 24 12:54 file3
# cd ..
# ls -lai foobar
total 10
   57 drwxr-xr-x   2 root     sys           80 Jan 24 12:54 .
    2 drwxrwxrwx   7 root     root        4480 Jan 24 12:54 ..
   83 -rw-r--r--   1 root     sys            0 Jan 24 12:54 file1
   12 -rw-r--r--   1 root     sys            0 Jan 24 12:54 file2
   69 -rw-r--r--   1 root     sys            0 Jan 24 12:54 file3
# od -c foobar
0000000   9  \0   .
0000020 002  \0   .   .
0000040   S  \0   f   i   l   e   1
0000060  \f       f   i   l   e   2
0000100   E  \0   f   i   l   e   3
```

```
0000120
# cd foobar

****  THE FUN STARTS HERE ****

# who -b .

****  who -b . just created a file with the name . in the current directory ****

# ls -la
-rw-r--r--    1 root     sys             0 Jan 24 12:55 .

# od -c .
0000000
# cd ..
# od -c foobar
0000000   ^  \0   .                                          ** YUK ! **
0000020 002  \0   .   .
0000040   S  \0   f   i   l   e   1
0000060  \f       f   i   l   e   2
0000100   E  \0   f   i   l   e   3
0000120
# ls -l foobar
total 0
-rw-r--r--    1 root     sys             0 Jan 24 12:54 file1
-rw-r--r--    1 root     sys             0 Jan 24 12:54 file2
-rw-r--r--    1 root     sys             0 Jan 24 12:54 file3
# cd foobar

**** WHEN NOT GIVEN AN ARGUMENT, ls LISTS . WHICH IS NORMALLY A DIRECTORY ****

# ls -la
-rw-r--r--    1 root     sys             0 Jan 24 12:55 .
# cd ..

**** link and unlink are in /etc.  ONLY root can run them, as ALL error checking
**** is removed ...

# unlink foobar/.
# link foobar foobar/.
# ls -lai foobar
total 10
   57 drwxr-xr-x   2 root     sys            80 Jan 24 12:56 .
    2 drwxrwxrwx   7 root     root         4480 Jan 24 12:56 ..
   83 -rw-r--r--   1 root     sys             0 Jan 24 12:54 file1
   12 -rw-r--r--   1 root     sys             0 Jan 24 12:54 file2
   69 -rw-r--r--   1 root     sys             0 Jan 24 12:54 file3
# cd foobar
# ls -lai
total 10
   57 drwxr-xr-x   2 root     sys            80 Jan 24 12:56 .
    2 drwxrwxrwx   7 root     root         4480 Jan 24 12:56 ..
   83 -rw-r--r--   1 root     sys             0 Jan 24 12:54 file1
   12 -rw-r--r--   1 root     sys             0 Jan 24 12:54 file2
   69 -rw-r--r--   1 root     sys             0 Jan 24 12:54 file3
# od -c .
0000000   9  \0   .
0000020 002  \0   .   .
```

```
0000040   S  \0   f   i   l   e   1
0000060  \f     f   i   l   e   2
0000100   E  \0   f   i   l   e   3
0000120
```

The reason fsck came back with normal results was because the files were in fact still there. The reason I was able to cd about was because I was running ksh, which does not look at . and .. too often (it looks at $PWD). If you just do the links and unlinks as above, you will be able to get to your files. In fact, you can still get to them now if you want, but then you won't be able to do much with that pesky .-less directory.

Jim Webb   ihnp4!houxm!hropus!jrw

From: mark@cbosgd.UUCP (Mark Horton)
Newsgroups: net.mail
Subject: does anybody use upper case in mail names on UUCP?
Date: 21 Jan 86 04:51:26 GMT

We're running into a problem with a BITNET/UUCP gateway. It seems that many people on BITNET have only upper case terminals, and send mail to addresses like USER@HOST.UUCP. It's fair to turn HOST.UUCP into host.uucp for UUCP, but what about the USER part?

The ARPA convention is that the case on USER is supposed to be left alone. This was done because Multics used to be case sensitive (although I understand it no longer is.) 4.2BSD is also case insensitive, so mail to host!host!USER is the same as mail to host!host!user. But I don't think System V is.

This leaves a person with an upper case BITNET terminal, who wants to send mail to a System V user, no way to get the mail through. Unless the gateway does some magic and turns USER@HOST.UUCP into host!host!user.

Are there any systems out there on the UUCP side of the gateway that would break if this change were to be made? Any upper case only names? Any mixed names? Any upper case only mailing lists?

The proposed change is somewhat of a hack, but it just might be the only way to make the gateway work properly.

Mark

From: reid@glacier.ARPA (Brian Reid)
Newsgroups: net.mail
Subject: Re: does anybody use upper case in mail names on UUCP?
Date: 22 Jan 86 07:46:32 GMT
Summary: don't do it. Let them rot, uh, er, LET THEM ROT (UPPER CASE)

I've used upper-case names in uucp for years, just to be pathological and cause trouble. I recently converted "Glacier" to "glacier" when it became a backbone site, out of a sense of civic duty.

>This leaves a person with an upper case BITNET terminal, who
>wants to send mail to a System V user, no way to get the
>mail through. Unless the gateway does some magic and turns
>USER@HOST.UUCP into host!host!user.

The problem with your solution is that it aids and abets the receipt of all-uppercase mail by helpless System V users. My attitude towards this kind of thing is that you should do nothing at all to help people with such radically inferior hardware; maybe the pressure to be able to send mail will motivate them to upgrade to 1970's technology.

```
        Brian Reid      decwrl!glacier!reid
        Stanford        reid@SU-Glacier.ARPA
```

From: glenn@wacsvax.OZ (Glenn Huxtable)
Newsgroups: aus.jokes
Subject: The Further adventures of Luke Vaxhacker
Date: 1 Apr 86 08:19:56 GMT

I picked this up from a tape from the US. It seems to have been seen on USENET, but I haven't seen it out here, so I thought others might like it also..
glenn

---------------------------&<--

The Further Adventures of Luke Vaxhacker   Episode n+1

The story thus far: Luke, PDP-1 and their 'droids RS232 and 3CPU have made good their escape from the Imperial Bus Signals with the aid of Con Solo and the bookie, Two Bacco. The Milliamp Falcon hurtles onward through system space. Meanwhile, on a distant page in user space...

Princess _LPA0: was ushered into the conference room, followed closely by Dec Vadic. "Governor Tarchive," she spat, "I should have expected to find you holding Vadics lead. I recognized your unique pattern when I was first brought aboard." She eyed the 0177545 tatooed on his header coldly.

"Charming to the last," Tarchive declared menacingly. "Vadic, have you retrieved any information?"

"Her resistance to the logic probe is considerable," Vadic rasped. "Perhaps we would get faster results if we increased the supply voltage..."

"You've had your chance, Vadic. Now I would like the princess to witness the test that will make this workstation fully operational. Today we enable the -r beam option, and we've chosen the princess' $HOME of /usr/alderaan as the primary target."

"No! You can't! /usr/alderaan is a public account, with no restricted permissions. We have no backup tapes! You can't..."

"Then name the rebel inode!" Tarchive snapped.

A voice announced over a hidden speaker that they had arrived in /usr.

"1248," she whispered, "They're on /dev/rm3. Inode 1248." She turned away.

Tarchive sighed with satisfaction. "There, you see, Lord Vadic? She can be reasonable. Proceed with the operation."

It took several clock ticks for the words to penetrate. "What!" _LPA0: gasped.

"/dev/rm3 is not a mounted filesystem," Tarchive explained. "We require a more visible subject to demonstrate the power of the RM STAR workstation. We will mount an attack on /mnt/dantooine as soon as possible."

As the princess watched, Tarchive reached over and typed "ls" on a nearby terminal. There was a brief pause, there being only one processor on board, and the viewscreen showed, ".: not found." The princess suddenly double-spaced and went off-line.

The Even Further Adventures of Luke Vaxhacker Episode n+2

The Milliamp Falcon hurtles on through system space...

Con Solo finished checking the various control and status registers, finally convinced himself that they had lost the Bus Signals as they passed the terminator. As he returned from the I/O page, he smelled smoke.
Solo wasn't concerned--the Bookie always got a little hot under the collar when he was losing at chess. In fact, RS232 had just executed a particularly clever MOV that had blocked the Bookie's data paths. The Bookie, who had been setting the odds on the game, was caught holding all the cards. A little strange for a chess game...

Across the room, Luke was too busy practicing bit-slice technique to notice the commotion.

"On a word boundary, Luke," said PDP-1. "Don't just hack at it. Remember, the Bytesaber is the weapon of the Red-eye Night. It is used to trim offensive lines of code. Excess handwaving won't get you anywhere. Listen for the Carrier."

Luke turned back to the drone, which was humming quietly in the air next to him. This time Luke's actions complemented the drone's attacks perfectly.

Con Solo, being an unimaginative hacker, was not impressed. "Forget this bit-slicing stuff. Give me a good ROM blaster any day."

"~~j~~hhji~~," said Kenobie, with no clear inflection. He fell silent for a few seconds, and reasserted his control.

"What happened?" asked Luke.

"Strange," said PDP-1. "I felt a momentary glitch in the Carrier. It's equalized now."

"We're coming up on user space," called Solo from the CSR. As they cruised safely through stack frames, the emerged in the new context only to be bombarded by freeblocks.

"What the..." gasped Solo. The screen showed clearly:

```
                /usr/alderaan: not found
```

"It's the right inode, but it's been cleared! Twoie, where's the nearest file?"

"3 to 5 there's one..." the Bookie started to say, but was interrupted by a bright flash off to the left.

"Imperial TTY fighters!" shouted Solo. "A whole DZ of them! Where are they coming from?"

"Can't be far from the host system," said Kenobie. "They all have direct EIA connections."

As Solo began to give chase, the ship lurched suddenly. Luke noticed the link count was at 3 and climbing rapidly.

"This is no regular file," murmured Kenobie. "Look at the ODS directory structure ahead! They seem to have us in a tractor beam."

"There's no way we'll unlink in time," said Solo. "We're going in."

From: Bakin@HI-MULTICS.ARPA (Jerry Bakin)
Newsgroups: mod.computers.vax
Subject: Re: Problem with rooted directories
Date: 9 Apr 86 02:18:00 GMT

This message may have one or two flames expressed herin:

Whether the problem you have described is a bug or not depends upon your point of view.

I think it is a bug.

- It seems counter intuitive -- counter to Multics, Unix, and other "relative" pathname conventions.
- It causes directories to be "hidden":
  If your rooted directory is "base" for "dma0:[a.]" then if you are in base:[whiz] and blithely type:

  ```
  $ set def [-]
  $ create/dir [.bang]
  $ dir bang.dir
  FILE not found!
  ```

  You don't see any directory! But as you noted, it IS there, it is under the newly created 000000.dir.
- It means the the "transparently" "concealed" rooted directory is neither transparent nor concealed since it has a different (and detectably so) behavior from top level directories.

- It means that commands are pathname dependent. Whether a command string works within a directory depends upon how you got to that directory. Typing

  ```
  $ create/dir [.bang]
  ```

  with the directory whiz may or may not win depending upon how you got to whiz. Did you type

  ```
  $ set def dma0:[a.whiz]   ! or,
  $ set def base:[whiz] !?????!
  ```

  This has bizarre side effects for other commands as well, I have seen other pathname defaulting resulting in print commands fail and not fail under these circumstances.

  So, DEC has a new idea of command languages, one that is not "conservative", that is, applying a command in any location is dependent upon how you got to that location.

DEC does not think it is a bug.

I and others have submitted several SPRS about this behavior, and evidently:

- DEC feels that it behaves as documented, hence NO BUG!

- For some never completely understood by me reason, DEC feels that if a problem were to exist, and they aren't saying that it does, but if it did, then they feel what is needed is a smarter "set def" command, one that would not allow you to change to the [-.bang] directory in the first place. Now, why they feel this is the problem I dunno, because it doesn't fix the problem which occurs when I type:

  ```
  $ create/dir [-.bang] ! at all!!!!
  ```

  At any rate, if they thought there was a problem here, and they don't, they told me they wouldn't implement a better set def program -- let's call it "cwd" -- because it would change the way command files work. Really! Their rational for this is as follows:

  - A certain subset of commands like "set def" can be run in between ^C or ^Y and you can still type $ cont and get back to the first program, I guess image rundown hasn't been performed or something. The claim they made to me was that a better set def couldn't retain this feature and so it would be non compatible and anyway we don't think we have a problem so there.

  - If we made the set def program even complain with a warning about moving to a non existent directory, can you imagine, I mean actually conceive of the incredible number of command files which would break because they use this feature that you can specify a default directory which has no validity. Even a warning would trap all those command files with $on warning enabled. It would be non compatible and anyway we don't think we have a problem so there.

When someone mentioned this to info-vax about six months ago, the general info-vax response was a DEC-CAN-DO-NO-WRONG-apologizer one saying (like DEC) that the behavior is as documented, hence no bug.

I think the bug is in the incredibly ancient and inconvenient pathname conventions under VMS. Seven or more years down the road and they should come up with a more "modern" scheme. Learn from Multics, UNIX, etc., and have fully relative pathname conventions instead of a "default directory specification". By the way, it is this specification that is at the heart of your problem. In VMS, you don't really have directories, you don't really have a "working directory", instead you have a "default directory SPECIFICATION" which seems to be a string and some rules which are applied to every file access if appropriate. It is the rules which contain the inconsistencies.

However, having a "set def" or "cwd" command which -- what a thought -- made sure the directory you specified existed or not might be a bonus.

Anyway, I am still interested in hearing what someone at DEC with even half a brain (no requirements on functionality even!) has to say on these matters. Clearly the SPR folks are all sharing the same ganglial knot passed down by Ken many years ago, and even though it was a good ganglial knot with at least a half-dozen synapses, old-age and stress from too many SPRs overloading the system has trimmed those half-dozen synapses down to one or two.

Jerry Bakin <Bakin at HI-Multics>

From: bzs@bu-cs.UUCP (Barry Shein)
Newsgroups: net.lang.f77,net.unix-wizards
Subject: Re: Any decent Fortrans under Unix ? Which machine ?
Date: 24 Feb 86 16:18:37 GMT

Guy Harris makes some very good points, I would like to add a few, I have dealt with this issue and folklore here, my response has become very simple:

Those here that went with a vendor's proprietary systems because they believed the fortran to be better remain on those systems to this day.

Those that went with UNIX (and a few that were willing to change over having realized their mistake) have doubled and trebled (and, in one case 12x) their performance by replacing their hardware with newer hardware that is coming out mostly running UNIX. Some of these purchases were fully funded by sale of the used equipment, all cost 1/3 or less price of the equipment they were replacing (typically, a $250,000 mini-computer for a $65,000 super-micro which was much faster.)

Two groups have specifically said that they have not taken advantage of this because their fortran code has become too heavily dependant on the vendor's proprietary hardware/software environment and they do not have the personnel resources to switch over.

Don't make the same mistake, project your upgrade options for both cases in the current context (as if you made this decision two years ago.) I think the facts speak for themselves.

What good is it to have a great optimizer on a slow machine?

        -Barry Shein, Boston University

From: jel@portal.uucp (John Little)
Newsgroups: net.bugs.uucp
Subject: Re: stack overflow with Usenet under Xenix/286
Date: 3 Apr 86 08:12:30 GMT

I had similar problems with IBM XENIX on the AT with large model programs. The linker tried to glue the data and stack segments together, and frequently ran out of room. I worked around the problem by making many of my data structures very large (close to 64 K) so that there was no room to put the stack in with the data. For example, 1K arrays became 62K arrays. When the linker merges enough items to fill a 64K segment, it starts a new one. By making as many objects as large as possible, you increase the chance of having enough space for the stack.
Of course, this only works if you have lots of memory.
Looking at the map is slightly useful, but expect lots of trial and error until you link everything together and it all works.

John Little
sun!portal!jel

EUUG
European UNIX† Systems User Group

Newsletter Vol 6 No 1

Spring 1986

† UNIX is a Trademark of Bell Laboratories.

Copyright (c) 1985. This document may contain information covered by one or more licences, copyrights and non-disclosure agreements. Copying without fee is permitted provided that copies are not made or distributed for commercial advantage and credit to the source is given; abstracting with credit is permitted. All other circulation or reproduction is prohibited without the prior permission of the EUUG.

European UNIX System User Group Meeting
Bella Center, Copenhagen — 10th - 13th September 1985

Kim F. Storm & Arne Facius,
University of Copenhagen, Denmark.
edited by Peter Collinson
Secretary

Introduction

I didn't manage to get to Copenhagen and so two willing volunteers were found to take some notes on what went on. The report below contains the abstracts of the conference talks plus some comments from Kim and Arne. Many thanks to them for the work which they have done.

Tutorial sessions, Tuesday 10 September

UNIX System V administration and overview
Jim Joyce
International Technical Seminars, USA

Abstract

This course is designed to introduce essential tools that make the UNIX system administration for System V more efficient and secure. Among the topics are: shell scripts for effective system maintenance, monitoring files that grow automatically, customised system management, correct selection of system permissions, device configuration.

Fast Prototyping Techniques
Rich Morin
Canta Forda Computer Lab, USA

Abstract

This course learns to combine tools with proven techniques to do fast prototyping under UNIX, saving month in production time. The powerful toolkit of utilities link together in a pipeline or in a shell script, to create applications in a fraction of the time it would take to code the equivalent functionality in C. Among the topics: packaging; from prototype to production model; application design tactics in the UNIX environment; idioms; shortcuts to results; channeling data flow effectively; debugging tips and techniques.

LEX and YACC
The Instruction Set Ltd, GB

Abstract

The tools *Lex*, a lexical analyser generator, and *YACC*, Yet Another Compiler Compiler are invariably ignored by all but compiler writers; indeed their names do not cause immediate interest on the part of the ordinary UNIX user.

In this tutorial we examine the tools and their application to everyday activities.

Writing Device Drivers
The Instruction Set Ltd, GB

Abstract

In this tutorial we examine the principles of operation of UNIX device drivers. Their overall structure is studied and their interaction with the rest of the system explained.

It is assumed that attendees are aware of the basic characteristics of hardware and the way it is typically driven by software. It is further assumed that a thorough understanding of the UNIX system at the user level exists.

Functions in the System V Shell
The Instruction Set Ltd, GB

Abstract

The Bourne shell had not altered significantly since its introduction in Version 7; not that is until functions were introduced into the System V.2 shell.

In this seminar we explain the basic principles of operation of shell functions and examine some functions that are in regular use.

The distinction between shell functions and shell scripts will be explained.

Technical Sessions, Wednesday 11 September

AWK as a General-Purpose Programming Language
Brian W. Kernighan
AT&T Bell Laboratories, USA

Abstract

Awk is a pattern-action language with convenient facilities for processing strings and numbers. It was originally intended for simple data validation, report generation and data transformation tasks; *awk* programs for such tasks can often be expressed in only a few lines of code.

Awk has often been pressed into service in ways far beyond what was originally intended. Recent changes to *awk* make it more suitable as a general-purpose programming language. This paper will discuss new features — new built-in functions, multi-file input, dynamic regular expressions and text substitution, and user-definable functions — and illustrate some interesting applications that they allow.

Comment

The original purpose of *awk* was to solve simple data manipulation problems for which people normally would not write a special program. The programs written in *awk* were intended to be small. But larger and larger programs have been written because of the strength of the language. Programs of 1000 lines of code are not unusual. The new version of *awk* has several new features which will ease the task of writing large programs.

Some of the new features are:

- One can close an open file.
- A 'system' function.
- Trigonometric functions like 'sin' and 'cos'.
- The command line arguments can be accessed through *ARGC* and *ARGV* variables.
- A 'getline' function which can read ahead in the input, or read from a file or a pipe.
- Substitution functions 'sub' and 'gsub' (equivalent to s/../../ and s/../../g).
- The field separator (FS) can now be a regular expression.
- New functions may be defined. Parameter passing is call by value for scalars and call by reference for arrays, as in C. To avoid declarations arguments are local while the rest of the names are global. †

Awk has been used for several unrelated purposes, e.g.

- Fast prototyping (but with rather slow execution).

† is this really a language for large programs? KS

- Report generation, data transformation, data validation, information retrieval.
- Compiler implementation language.
- Object language for compilers.
- As peoples first programming language, because of easy transition from idea to program.

Some questions resulted in the following answers:

- Only one pipe open at a time, because of limitations in **popen**.
- There is a manual.
- The new awk is NOT available (but ask UEL).
- Better error messages (at last).
- Array elements may be deleted.

The Streams facility in UNIX System V rel. 3
Bob Duncanson
Unix Europe Ltd., GB

Abstract

Streams is a set of mechanisms within the UNIX System Kernel that allow character I/O to be implemented in a modular way. As a result, the drivers for character-type devices have a structured interface to the UNIX system kernel. The *streams* mechanisms also provide a flexible framework within which different networking architectures can easily be designed and implemented.

Streams now includes a conceptual model of character I/O, a well defined interface with the rest of the UNIX system, and a library of utility programs, other utilities (such as an administration driver) and several new (or modified) UNIX system calls.

Comment

Historically the optimization of character input and output did not draw much attention, because in those days the TTY-33 was state of the art. Nowadays, we do not only have fast terminals. But terminals may be connected through networks, and one terminal may talk to several machines. Streams are based on message passing, and they are used in System V (R3) to split the kernel into modules, giving a clean interface between drivers and the rest of the kernel.

One can add functionality to a driver by inserting modules between the driver and the kernel. Such modules may for instance implement a specific protocol, for UUCP say, and just by replacing this module a new procotol can be used, which leads to network independent user-level routines. Future versions of UUCP will be independent of the specific protocol used. Streams are incompatible with sockets, but work is going on to implement sockets on top of streams.

Object Formats and Memory Management techniques in UNIX
Martijn de Lange & Hans van Someren
Associated Computer Exp

Abstract

The talk will give an overview of memory management techniques encountered in modern (UNIX) computer systems, and of the relation between object formats and memory management. After an extensive introduction on these general topics, the ACE tailored System V kernel memory management will be described.

The ACE UNIX System V kernel for MC680X0 based systems is portable over a wide range of Memory Management units. Internally, the system uses a flexible layered database for hiding the MMU dependent parts and for the segmentation housekeeping details. On top of a small interface to the MMU dependent parts, and together with this flexible database, a powerful memory management system has been implemented. It improved the internal structure of the system, the performance and the user's interface to the segmentation.

Fundamental to the user's interface is the System V COFF format (Common Object File Format). The S-V COFF format is accepted as a standard load format, but COFF extensions to record the segment attributes made the multi-segmentation management system much more powerful.

The management system allows swap on demand and highly efficient process forking using copy on write segments. The user is able to allocate segments to any extent, to allocate segments dynamically, and to use shared run time library segments. With the extended COFF format it is possible to specify by means of indirect sections that the segment description is to be taken from another COFF file. A process uses a set of segments, which is described by it's private view with as many as 16 attributes for each individual segment.

The advantages are obvious: reduced memory usage, swap on demand, memory sharing, sharing of libraries, and an improved system performance and maintainability.

The extended COFF is implemented for all MMU's supported by the ACE tailored System V kernels for single and multiprocessor Motorola MC680X0 based systems. In the future, a multi-segmentation system with paging might be expected.

Comment

The abstract says it all. We can only add that they report disk space savings upto 30% because of the shared libraries.

However Martijn de Lange gave a new interpretation of the evolution of UNIX, of which we can only give a defective summary: UNIX was a kid born in a wealthy family (Bell Labs) with a happy childhood through the ages of 5, 6, and 7. After that the kid went to the university (UCB), got a middle age crisis between 41 and 42, went into therapy and found back to the age of 3 (III), and now it has a split personality, one at 43 and one at 5.

An improved UNIX Kermit file transfer program
Barbro Malm & Johan Helsingius

Abstract

While local area networks and packet switching long-haul networks are getting increasingly popular, a surprising amount of file transfer work is still being done over slow and unreliable asynchronous lines.

In the UNIX environment, this is usually done by the UUCP system, but often only "transient" file transfer capability is needed, and the configuration of UUCP links can get too complicated, especially for casual users.

When attempting to move data between computers using incompatible operating systems, the situation is much more complicated. Things like incompatible character sets, buffering techniques, half and full duplex transmission, and interpretation of special characters all make communication a definitely non-trivial task.

One solution is a public-domain file transfer protocol, called KERMIT (KL10 Error-free Reciprocal Micro Interconnect over TTY lines). Kermit was originally developed at the University of Columbia, but its development work has since spread to sites all over the world.

Implementations currently exist for a remarkably wide range of systems, from Commodore and Apple home computers to CDC and IBM mainframes. University of Columbia, as well as many other sites, maintains and distributes the latest releases of the various Kermit versions.

Kermit file transfers are done using cooperating programs running at both ends of the communication line, passing small, checksummed packets over regular TTY lines. Facilities exist for transmitting 8-bit data over 7-bit lines, as well as terminal emulation capability.

Our recent work has been aimed at extending the current Kermit protocol to incorporate some more advanced transmission protocol techniques, such as 'floating windows', and simultaneous full-duplex bi-way file transfer. A prototype 'advanced' UNIX kermit has been implemented using, among other tools, LEX and YACC.

One important possibility offered by this development is the capability of providing UUCP-like network services to non-UNIX machines, such as PC's. Some of the implications of this will also be covered in this talk.

The UNIX Kermit program (as well as Kermits for other operating systems) is available for free distribution to all potential users, except for explicitly commercial purposes.

Comment

This was the first speaker after lunch, and we were all likely to go to sleep; Johan assured us that we would. For the sake of the readers, we kept ourself awake, revealing that the abstract covers the contents of the lecture perfectly. We can only add that the extended Kermit protocol has a 90-95% efficiency compared to the 50-70% efficiency of the standard protocol. He claimed that his participation at the conference was sponsored by the Finnish tourist agency, and that he had to do some advertising to get EUUG to hold the next conference in Finland. He showed us some very fine slides, which at least convinced me, that Finland would be a good place to hold a conference.

Networking with ISO and UNIX
Sylvain Langlois & Alain Gauteron
Project Rose, Bull, F

Abstract

During the last 5 years, operating systems have been evolving towards new concepts and design in order to integrate inter-system communications. The work done by standardization bodies within ISO has been conceptualized into the Basic Reference Model for Open Systems Interconnection. Protocols based on this Model have now reached a level where they are quite well specified (state of International Standards), and thus can be incorporated into operating systems.

This communication presents an application of the OSI Model to the case of a Local Area Network environment. It mainly covers integration of the 4 lower layers of the Model into the UNIX kernel. A brief overview of the gateway to a public network, which is currently in progress, is also given.

The LAN architecture is based on an ETHERNET, using the ISO Internet Protocol (DIS 8473), and the Class 4 Transport Protocol (IS 8073), which has been slightly simplified to operate on top of a connectionless Network Protocol. Nevertheless, it is important to note that the resulting Transport implementation enforces ISO conformance rules; the simplifications can rather be seen as implementation choices for optimization in a local area networking context. The gateway is a Transport level relay to ISO Class 3 Transport sitting on top of an X.25 network.

After a short presentation of this architecture, and especially modifications which have been made to Transport Protocol specifications to adapt it to a connectionless LAN, we describe the implementation in more detail.

The ease of adding communication facilities to the UNIX System strongly depends on which version of UNIX is chosen. IPC and memory management mechanisms developed by Berkeley, and work done for TCP/IP implementation, made the 4.2BSD version more attractive for such a development. We believed that it was possible to use this particular system architecture for what we wanted to do, without too many changes in the existing facilities offered by 4.2BSD. Even if we have been obliged to make some small changes for implementing the Transport Service, we would like to consider this presentation as a kind of example of adding new communication domains in a 4.2BSD kernel.

Finally, we would like to address some problems we are now encountering when trying to incorporate a connection oriented network protocol, such as X.25, in the 4.2BSD kernel.

Comment

The ROSE project funded by the EEC ESPRIT. Its overall purpose is to establish electronic mail, file transfer, remote login, and teleconferencing for the other ESPRIT projects. The aim is to implement ISO/OSI under UNIX. The project schedule says that an operational connection between UNIX (4.2BSD) machines should be ready in 1986.

They use the standard socket interface of 4.2, but with some modifications to set/getsockopt, and a new confirm system call. They have a new address family called AF_ISO, which is almost compatible with the standard 4.2 code.

The system they use is a V7 UNIX with 4.2 network enhancements; TCP/IP is still present in the system, but it is not working. They do not have any performance figures yet.

To the question why they did not use System V, since some of the participants in the project were also members of X/OPEN, he answered that it might be the next step, but that it would be done by another group.

Sendmail now, and its next generation
Miriam Amos
DEC, Ultrix Engineering Group, Berkeley Division, USA

Abstract

Berkeley's Sendmail has become the comparative yardstick for UNIX mail systems. It provides a reliable mail delivery system. However, its extensive size and flexibility may be considered overkill. This is a discussion of Berkeley's current mail system and the next generation of the mail system to

come.

Comment

Miriam would like to hear what Sendmail should do from everybody. She talked about the improvements that are planned for future Sendmail systems.

The design goals for a new Sendmail are: compatibility with existing mail facilities, reliability (no lost mail), expandable (possible to write code for something that might exist), no configuration information should be compiled into the programs, individual forwarding and accessing of mailing lists, and finally optimisation of network traffic.

The current Sendmail has aliasing and forwarding, it does its own queueing, the SMTP protocol is built into the system, it has a flexible configuration, and can do automatic routing to network gateways. However, Sendmail is not 'UNIX like', because it does too much in a single program, trying to gateway between all sorts of formats and systems. The 'rewriting rules' used in the configuration are simple in themselves, but their connection is cryptic. Maintainance is also a headache, it is 50000 lines of code, and changing the configuration demands the presence on an expert. The plans are to break Sendmail into 'one problem — one program' thunks; a 'mail crossbar' which does gatewaying between dedicated mailers, a generalized queue management like lpr, separate programs for maintaining the aliasing files, and a separate SMTP server. 4.3 will still have the old Sendmail, but it is faster, better, and perhaps with better documentation.

Remote File Systems on UNIX

Douglas P. Kingston III
Centrum voor Wiskunde en Informatica, NL

Abstract

In the past few years, the nature of computing has changed dramatically as the technology has made it possible to provide computers for small groups or individual users, while sharing more expensive resources via networking. Unfortunately this has also created problems since it is still desirable to easily access data belonging to others that may now reside on another system. When this capability is provided across a network in such a way that the remote files retain the basic attributes of a local file, it is referred to a Remote File System or *RFS*.

While not in widespread use, there have been a large number of attempts at developing a RFS capability for the UNIX operating system. The scope of such work has varied greatly between different attempts. This reflects changes in the design criteria, the underlying capabilities expected in the UNIX system, and the extent to which people were willing to alter UNIX itself. The various implementations have varied greatly in their transparency and efficiency, the two most important qualities of a RFS. In addition, the proprietary nature of many of the RFS implementations has greatly hampered their widespread acceptance by the UNIX community.

The ideal RFS is totally transparent to all UNIX user processes, has no noticable effect on operating system performance, and is available to be implemented on a variety of systems.

The paper will survey many of the RFS on UNIX, explain their basic designs, and comment on how well they approximate the ideal RFS, at what cost and with what disadvantages.

Based on the preceding review, and the implementations the CWI has been able to obtain, we have undertaken to implement a RFS for 4.3BSD that will be generally available to the UNIX community. We will detail the design and implementation choices and report on current progress and current or anticipated performance. The CWI RFS is implemented in the kernel for transparency and efficiency. Implementation has begun and a test system was completed two months ago.

Comment

There are zillions of RFS for UNIX, but none of the publicly available are really "good"; either they are primitive or their performance is too bad for practical use. The public domain CWI RFS will be based on the idea of having a **/net/hosts** directory where the remote file systems are mounted, which is similar to the systems from Lucas Films and Harvard. The plan is to implement a canonical file system interface in 4.3, and then implement RFS on top of this interface.

Q. What is wrong with '/..' as net prefix?

A. Nothing, but in this way '/..' is no longer equal to '/'.

Q. Shouldn't a really portable system be implemented in user space?

A. It is not reasonable not to touch the kernel.

Q. But not everyone has a source licence?

A. That is true, but even the Newcastle Connection may run in the kernel for efficiency.

EUUG General Membership Meeting

Teus Hagen started things off by making a short statement.

There are now national user groups in 11 countries. There are associated groups in Switzerland, Ireland, and Austria. The DECUS UNIX group also listen in on the EUUG activities. The governing board is composed of representatives from the national groups. Individual members have no influence on the Governing board.

The Newsletter is now edited by Jean Wood, who has taken over from Jim·McKie.

The EUUG treasurer is now Nigel Martin, who has taken over from Mike Banahan.

EUUG actvities are:-

> The EUUG Newsletter, with 4 issues per year.
>
> The EUUG Conferences. They are planned two years ahead. The plans are to put more weight on the technical part of the conferences, and have smaller (or no) exhibitions, and make the industrial sessions larger.
>
> There are talks with AT&T, and frequent contacts to UEL. EUUG participate in standardisation (X/OPEN) and internationalisation. EUUG also interchange information with its sister organisation USENIX.
>
> EUUG helps members in licencing questions.
>
> EUUG is running the international EUNET network. It administrates the backbone structure, software maintainance and distribution, accounting, and establish gateways to other network, e.g. EARN, CSNET, USENET, and ROSE.
>
> EUUG has published two catalogues on UNIX system micros and UNIX products in Europe.
>
> A major part of the EUUG work is the tape distributions, with more than 100 tapes per year. There are tapes with UUCP, GNU EMACS, and USENIX software.

A future plan is to cooperate with Apt Data Services in the production of a weekly newsletter 'UNIGRAM'. Members will obtain a discounted subcription to the newsletter.

EUUG try to stimulate UNIX activities, and UEL offers awards for student papers, e.g. free travel and participation to conferences. The project results must be free for the UNIX community.

The next EUUG conference will be held in Florence, April 21-24, 1986.

The Autumn 1986 conference will be held in London, September 9-12, 1986. There will be no exhibition, but more workshops.

The venue for future conferences has not been decided, and one idea is to alter the conference frequency to one per year.

There were then some questions and answers.

Q. Can EUUG make the ISO transport layers, e.g. from ROSE, available?

A. Are working on it, but no success yet.

Q. Who are members of the EUUG?

A. All national members, including students at the universities, are automatically members of the EUUG.

Q. Can we enforce deadlines for abstracts and papers to the conferences? it would be nice to have the proceedings before the conference.

A. There is always a conflict; to get interesting talks at the conferences, you often have to beg people to give them. The current programme was not ready until 14 days before the conference.

Q. The proceedings should be made available to non-participants and non-EUUG members sooner.

A. One thought is to put the proceedings in the newsletter.

Here, a vote showed that nobody was interrested in letting a professional publisher print the proceedings, while the majority was in favor of publishing them in the Newsletter.

By a show of hands it was clear that most of the participants only attended the technical sessions.

The impact of an exhibition is that the conference must be held in a place that is large enough to hold the exhibition, and conference centres are quite expensive. EUUG made a lot of money in Nijmegen, lost it all in Paris, and will make a small profit on the Copenhagen conference.

It was suggested that the exhibitors should be allowed to send technical people only, that is — no sales people, but exhibitors like to sell computers.

Some attendees cannot come if there is no exhibition to wave in front of their management's noses as a reason for participation.

The final remark was to ask the exhibitors what they think.

Technical Sessions, Thursday 12 September

Overview of 4.3 BSD

Kevin J. Dunlap
DEC, Ultrix Engineering Group, Berkeley Division, USA

Abstract

4.2BSD provided new functionality, but due to the lack of time was not tuned to the level the developers would have liked. 4.3BSD is the next release of Berkeley's Unix offering. This release includes the system tuning that time restraints prohibited on 4.2BSD as well as additional functionality.

Performance improvements were provided by use of cacheing, optimization of existing algorithms, selection of more efficient search algorithms, and utilizing the more efficient facilities provided by 4.2BSD. Some of the new functionality added were expansion of the network capabilities to handle subnets and gateways, support for windows and system logging. As well as extending the libraries and utilities to handle the new Internet name server, new system management tools, and Pascal support for dbx.

Comment

An overview of the optimizations performed in 4.3BSD was presented. Of specific points to mention is:

Kernel optimization:

- Process management & scheduling: break list into 3 parts: active,zombie,unused.
- Filesystem:
  find big block instead of many small 4/8 K blocks.

Improvements to libraries & utilities:

- hashed databases
- buffered i/o
- mail system
- C run time library
- network servers
- csh

Kernel extensions:

- increased number of file descriptors per process (from 20 to 64)
- increased kernel limits
- memory management
- signals
- system logging
- windows

Functional extensions to libraries & utilities:

- name servers
- system management
- routing

- compilers

Q. What about string copying?

A. VAX specific.

Q. Will it run out of process descriptors?

A. No.

Q. Running out of swap space?

A. Believe it's improved.

Q. In Ultrix?

A. At some point, probably.

Q. UDA driver?

A. Has been fixed in 4.3.

The 'cat -v is dangerous' attitude is itself dangerous
David M. Tilbrook
Imperial Software Technology, GB

Abstract

'This feels like a Republican victory party ...'

Vic Vyssotsky in keynote address at a USENIX conference

The so-called UNIX-philosophy has been preached from the pulpits of UNIX conferences by the high-priests of orthodoxy at many a conference. Does this zealous fervour have any connection with the failure of UNIX to make any significant advances in the recent years?

Why are there large areas of computer science that seem to be ignored by the UNIX world or why is that when some areas are attempted on UNIX they prove to be as unworkable or cumbersome as they were on the more traditional environments?

This presentation is a highly personal view by one who has been dismaid by the failure of the UNIX community (himself included) to make any significant advances in real-time systems (what ever they may be), software engineering (something palatable at least) and a variety of other problems.

Comment

David was sorry that Peter Collinson wasn't present, because his notes were so good. We were sorry too! To keep up with the speed of this speech and the number of foreign words wasn't easy, but here's the best we could do.

David started to point out that he really didn't have any content to motivate people to yell at him. He quoted an article in the Guardian about 'group thinking' comparing this to the UNIX community. As Tom Duff said at Toronto, 1975: "...the first clue that something is wrong with APL is that when two APL programmers come together they always start discussing extensions" — this is no argument against the talk — but a lot of effort is expended on minor things, the major problems are not solved by West Coast. A usual way today are to build a tool, use it for some time, then throw it away and begin again, what about doing this with UNIX? Anyone against? This is of course threatening to a lot of people, like those in the room here. But the next generation is not reached by expanding UNIX. In doing so, we must preserve some kind of Phoenix, rescue the good and throw away the bad.

A few personal views:

In the beginning, Kenneth said: "let there be a hierarchical file system" and it was good. A small system, not necessarily new ideas, real-time system and IPC left out at will, and then the problems began! The two man project was now beginning to evolve towards a Mega-project. The nature of the community meant that new things were included, but who knew what to avoid? Large areas are not developed, and if they are it's done by Unix novices, e.g. SCCS.

Eleven years ago he made a large system, a system with signals, pipes and file system, and I'd better be careful with those pipes! Today we have a large system with signals, pipes and file system, and I'd better be careful with those pipes still.

/usr/man was a remarkable database of documentation eleven years ago. At least when you were coming from a system with either doc's in libraries or non existant. It was so good, that we are still using it today, but it hasn't changed in eleven years. In the next iteration, I want some improvements. But, of course, *make*, a small set of SCCS, phone_numbers etc. was good. UNIX has grown a lot, but not gracefully and not in compatibility with the original system. STONEMAN — the name showing the intellect of the man having made it — has borrowed a lot of ideas from Unix. The UNIX community are not moving forward any more, there is a need for radical change, a change of direction, a need to look for a successor. I put it out, that this forum should find out how to make it!

Q. (Mike O'Dell) You are right for all the wrong reasons. The facts are right, but one need to separate implementations of UNIX from UNIX itself. People adapted UNIX, but they didn't know what was behind it. If 4.2 and SUN etc. was put together...

A. Isn't this just another iteration?

Q. (Stroustrup) Supersetting is a bad idea! UNIX has the problem that none will be touching a new system because expectations have grown enormously.

A. Not clear.

Q. Happy that UNIX doesn't do any real-time handling. It wasn't designed to.

A. I didn't mean real-time should be put in, shoe-horned in as one could call it.

Q. Would it be a good idea to have a new project like redoing V7 and put it in the public domain, without AT&T behind and controlling it.

A. Standardization for those who want to keep with it — rethink for the ones to go on!

Q. What are the motivation behind your speech. Are you practising what you are preaching?

A. Can't do it myself. I'm an application person, but will support it. Where we were and why it went that way is the philosophy.

Q. There is a time constraint. The longer they dig, the bigger grave they dig.

A. Military dependent, but don't care about them digging their own grave — sorry! — but we can't keep the two things together.

Q. But the money is coming from the industry?

A. *Research* are not being commercially directed. If Ken was asked 15 years ago, he wouldn't have been able to.

Q. (de Ridder) Some researchers are looking into making the successor of UNIX. It's not a technical problem to develop it, but it will take some time.

Recent work in UNIX Document Preparation Tools

Brian W. Kernighan
AT&T Bell Laboratories, USA

Abstract

Document preparation based on *troff* continues to be an active area of research. This paper describes a new tool, *grap*, a program for typesetting graphs. *Grap* is a preprocessor for *pic*, rather than the usual *troff* preprocessor. Although originally intended only for document preparation, it has also been used for algorithm animation and exploratory data analysis, and has served as an "assembly language" for several compilers for specialized graphs.

The paper also describes enhancements to *pic*, particularly built-in functions and control-flow primitives, that permit the creation of figures of some complexity.

Comment

Kernighan introduced the speech by noting that we all owe something to D.Knuth (TEX and METAFONT). He then went on to talk about the possibilities in the UNIX environmenht.

Tbl can be used to create large tables, which contain lots of information, which cannot be recognized easily. Graphs are much easier to comprehend quickly. This was introduced with a number of unsuccessful tries:

```
            options
numbers        v
------->     GRAPH     ------>    plot    ------>   PLTROFF   ---->   TROFF   -->
            language

            options
numbers        v
------->    GRAPHIC    ------>    pic     ------>     PIC     ---->   TROFF   -->
            language

           macro defs.
macro          v
------->      PIC      ------>   TROFF     -->
 calls
```

Should have written a new language — done now:

```
grap
------->   GRAP   ------>   PIC   ------>   TROFF   -->
input
```

Basic idea:Should give a **minimum** specification of a graph — given just numbers, it produces a graph, with suitable axis, scaling etc.
Can control axis labels, ticks on axis, line types, and may even escape to *pic* commands to draw arrows etc. Axis may be logarithmic.

In fact it is a small programming language, not rich but useful. Still another preprocessor to *grap*, *scatmat*, which is are used to plot scatter matrices.

grap and *pic* shares a lot of code, and Brian has now integrated *grap* into *pic*, which has been modified also, e.g. more typesetter independent, no internal array limits and other enrichments.

Examples:

```
.G1
1911  255.4
1915  252.5
 .     .
 .     .

grap [file] | pic | troff

.G1
label bottom "World ...."
label left ....
ticks left from 220 to 260 by 10
ticks bottom at 1920,1940,1960
draw solid
copy "mile.d"
```

Pic changes:

- features from *grap*
- output in inches, not in units
  ("…. could easily be changed to centimeters for more advanced countries", as Brian stated).

TROFF changes:

- 30% faster
  (width cache, hashed name tables)
-

\s+(dd	for new eqn
if line-number filename	like #line in cpp
\X'whatever'	passed through to output uninterpreted
\C'longname'	longer character names
\N'nn'	absolute number

.fp n XX long-font-name synonym for long font names

Nroff changes:

Revised to use ASCII terminal table instead of reading a.out file format: ".... done because they were going to set up a committee to discuss ways to do this" — Brian did it in an afternoon.

Q. Anywhere in the future where we can see what we get on the screen?

A. There are such systems, but not sure the WYSIWYG is useful in all cases, e.g. complicated graphs and linguistics.

Q. What about other languages, e.g. in *troff?*

A. Yes, it would be a great improvement, the notion of a preprocessor has been tried. But, it may break many things.

Q. What about improvements on *eqn?*

A. Yes, several — in danger of being shot by the session leader (Helen), he showed another slide.

Q. Availability?

A. In Writers Workbench II — perhaps in the end of the year.

Q. Changes to intermediate language output by *troff?*

A. Only trivial extensions (superset).

Q. X/OPEN — want more fonts!

Principles of Font Design for Personal Workstations
Charles Bigelow
Stanford University, USA

Abstract

The personal workstation offers powerful tools for the computer literate, but these tools depend on typography: legible fonts in meaningful compositions. Digital typography is constrained by the limited resolutions of current screen displays and printers, which cannot faithfully reproduce traditional analog letterforms. To optimize the legibility of digital text, new fonts must be designed for screens and printers. These designs will be most effective if they consider the mechanisms of the human visual system, the evolutionary principles that shaped our modern alphabets, the technology of the digital image, and the conceptual structures underlying typographic variations and compositions. The workstation requires a digital typography that preserves the fundamental features of literacy while expressing them with new clarity in a new medium.

Comment

David Tilbrook was very enthusiastic about him coming, especially for his gaily coloured tie!

There are three levels to consider:

- readability
- legibility
- deciferability

In history, development has gone from iconic to symbolic signs or abstract entities. Today we have gone 3000 years back using icons to 'name' entities, in fact the opposite way of our ancestors.

Letters have a very long history, not very many tech. stuff has survived that long. They are very archaic and we still are able to read roman stone carved letters. To begin with, it was only big letters, but handwritten text became lowercase letters due to physiognomy. Later both upper and lowercase letters were introduced. With printing, a development of letters to very fine sculptured (art) letters began.

Nowadays, what we want is a regular rhythm of smooth shapes, black on white, formed by nice sculptured letters — what we get are "computer printout" and screens!

Font design must be based on **how the eye works**. An example of a dot-matrix printed word with a big distance between the dots was shown compared to the same word in ordinary print, where only part of the letters were seen. It was very clear, that the latter was much easier recognizable. The eye recognizes the smooth curved edges to a high degree.

Furthermore, there is an optimal spacing between types, which is not linear over sizes and parts of the letter which traditional typesetting did just right, but which is lost in current electronic typesetting.

A list of what we need:

- better fonts for workstations
- should not remove curves and diagonals
- differentiation of size and of style
- device-independent outlines or device-dependent bit-maps
- point variations
- how prevent degradation on laserprinters
- METAFONT — Knuth designed it for his own purposes "....mathematician who is still a programmer!"
- designed by hand — and then digitized, a system of curved splines
- split letters in fundamental parts

Q. (Mike O'Dell) When can I get the fonts you showed for my laserprinter system?

A. They are available for other devices.

Q. What about terminals, is black on white the best?

A. White on black gives often flicker. For people doing text processing it's very much better, because they refer to papers with black on white.

Q. Any advice for people doing low-resolution fonts for terminals?

A. Don't over simplify them. Get an even spacing. Use proportional or carefully tune monospaced. Build them by hand.

Q. Any experience with grey levels?

A. There is a lot of research. But problems, how to go from high-resolution to grey levels? Some research indicates that it may be more fatiguing than pure black & whited, the eye is looking for sharp edges.

Smalltalk on UNIX Systems
Georg Heeg
Universitt Dortmund, D

Abstract

The talk will describe experiences with the implementation and use of Smalltalk interpreters on MC68000-based Unix systems. It will try to relate some basic ideas of the Smalltalk programming environment in the process.

Comment

Heeg refered to the speech of Kernighan. In Smalltalk, you would have three windows, one with graph, one with data and one with commands. Changes should affect the graph immediately!
Needs one VAX per user, bitmap screen and a mouse.

Smalltalk-80:

Goals:

- support for nonspecialized computer users
- object-oriented paradigm
- class descriptions

User interface (the most famous thing):

- window and mouse — ("all later systems bears this inheritance") the meaning of the mouse buttons are uniform throughout the system.

Objects (basic concept):

- everything is an object
- belongs to classes — classes defines the actions to be done. Classes are also objects.

Provides a very advanced programming environment. Everything in the Smalltalk environnment is in **source**.

Smalltalk -> UNIX access:

1) file system

standard: directory and files can be accessed
advanced: structured filesystem editor with graphical representation

2) program execution (batch)
Smalltalk communicates with string to/from a UNIX filter

3) terminal emulation on certain advanced systems

4) thoughts about representing UNIX parts as Smalltalk objects.

Q. Can one use normal UNIX programs in the Smalltalk environment?

A. As filter, batch via files, or if it exists — via a terminal emulator.

Q. Any important applications?

A. Very advanced database/spreadsheet programs.

Simula and C, a Comparison
Georg P. Philippot
NCR Education Nordic Area, N

Abstract

A newcomer in the UNIX world, *Simula* is an object-oriented language of the same family as that of C. They are both block structured, machine independent, general purpose high level languages. Though in principle you may use any of the two for any given task, there are individual areas for which one is better that the other. This paper describes the similarities and differences between two great languages, assuming some basic knowledge of C. It is hoped that this may give the audience some appetite for learning more about *Simula* and how it fits into a UNIX environment.

Comment

Algol(father), Simula, C, Pascal and Ada are all related. They look a little bit different, but development and compilation of programs are the same.

An overview of common features, C advantages and Simula advantages was given.

Conclusion:

- Simula domain:
  safe development, complex programs
  fast development, simple programs
- C domain:
  when speed can really be increased by an order of magnitude
  binary file and bit manipulation

Q. Availability?

A. For the VAX and the SUN — through Simprog in Stockholm.

Q. Why not choose ADA?

A. ADA is far too complex (overkill).

Q. Are strings in Simula null-terminated?

A. No.

A C++ Tutorial
Bjarne Stroustrup
AT&T Bell Laboratories, USA

Abstract

This is a tutorial introduction to the C++ programming language. With few exceptions C++ is a superset of the C programming language. After the introduction, about a third of the text presents the more conventional features of C++: basic types, declarations, expressions, statements, and functions. The remainder concentrates on C++'s facilities for data abstraction: user-defined types, data-hiding, user-defined operators, and hierarchies of user-defined types. Finally there are a few

comments on program structure, compatibility with C, efficiency and a caveat.

Comment

To correct some mistakes, there are no languages called 'C' or 'Old C'. The majority of C-programmers at Bell Labs do not use C++, many of them are involved in maintaining old programs! But most new programs use C++.

C + Simula67 classes + type checking + operator overloading -> C++.

Basic idea is Simula class. In Smalltalk they took away any idea of efficiency and gave you user-interface! C++ goes the opposite way. C++ allows the programmer to express ideas clearly and precisely.

- programs should contain more info in the same space —
  i.e. shorter programs
- modularity
- fast — both in compilation, it finds errors faster — presently 30-100% slower in code generation — and execution.

For all practical purposes C++ is a superset of ANSI-C. More compatible than most will believe. A lot of work has been done to do this right.

A class declaration contain a private and a public part. It may contain function declarations, which contain argument types and they may be overloaded — determined on the argument types at call time.
Operators are declared in the same way as functions.
A new datatype called *references* is introduced. It may be used to implement *call by reference*. Stroustrup doesn't like call by reference, but examples were given showing that you have to use it sometimes.

New operators defined: *new* similiar to *malloc* but able to determine the size from the type itself. *delete* to free it again.
You can also have inline functions.
Generic types must be done by macros, which is unpleasant but doesn't cost at execution time.
Classes may be expanded to avoid the traditional 'copy and modify source' problem. They are called derived classes.

C++ is expressive, easy to learn and is based on proven concepts.

Q. Any protection against interrupts (signals)?

A. Not taken care of.

Q. Major changes since release A?

A. Default scope of a global name is the whole program.

Q. Availability?

A. Available to Universities from UEL.

Nottingham's experience of X.25 under 4.2BSD

William Armitage
University of Nottingham, GB

Abstract

When looking for an X.25 implementation, we came across the University of British Columbia's EAN X.400 mail system. Tucked away in a corner, hardly mentioned, was an implementation of the CCITT domain for the Berkeley 4.2 kernel. We have successfully adapted this code to the UK academic environment.

This talk first examines UBC's implementation of X.25, and goes on to describe our experience in adapting the code, covering layering protocols over X.25 (Yellow Book Transport Service and RFC877 IP Encapsulation) and performance issues.

Comment

A very technical description of the protocols and the interfaces was given. For a full understanding we must refer to the papers from the speaker.

Technical Sessions, Friday 13 September

Error recovery for Yacc parsers

Julia Dain
University of Warwick, GB

Abstract

We aim to improve error recovery in parsers generated by the LALR parser-generator *Yacc.* We describe an error recovery scheme which a new version of *Yacc* automatically builds into its parsers. The scheme uses state information to attempt to repair input which is syntactically incorrect. Repair by alteration of a single token is attempted first, followed by replacement of a phrase of the input. A parser for the C language is generated from existing specifications and tested on a collection of student programs. The quality of error recovery and diagnostic messages is found to be higher than that of the existing portable C compiler. The new version of *Yacc* may be used by any current user of *Yacc*, with minor modifications to their existing specifications, to produce systems with enhanced syntax error recovery.

A Prolog description of a symmetric solution for automatic error-recovery in LL(1) and LALR(1) parser generators

Abstract

It is shown that the LL(1) and LALR(1) parsing schemes are too close to each other to justify the quality gap between their respective error-handling capabilities. Even within the constraints of a one-symbol-lookahead strategy a reasonable error-recovery with clear and precise syntax-derived messages is possible without adding any explicit error information.

A considerable improvement is obtained by synchronizing on so-called fiducial symbols. Prolog was suitable to find a formal specification of the set of fiducials for any grammar. The fiducials optimize error-handling without disturbing the efficiency and robustness of the underlying parsing scheme. They are implemented by context parameters in LL(1) procedures and by automatic insertion of error tokens in LALR(1) grammars.

Available syntax descriptions in YACC of ADA and AWK are used to demonstrate the effectiveness of the method.

Screen based History Substitution for the Shell

Mike Burrows
Churchill College, Cambridge, GB

Abstract

Several UNIX command interpreters now incorporate a history mechanism to assist interactive users in repeating or correcting commands. Many of these are similar in style to Bill Joy's C Shell, which provides a simple line oriented interface and a numbered history list. More advanced shells, such as the Korn Shell, allow screen editor facilities, but still reference history items by an event number or an explicit pattern matching syntax. This paper describes an interface that allows history substitutions to be performed as a command is typed and with minimal user effort. Entire lines or single words may be substituted with equal facility. The technique has been fully integrated with more normal editing features and Tenex-style filename completion.

The interface is extremely simple to use even with large history lists, without requiring the user to repeat "event numbers" or to review the history list periodically. Versions of the interface have been added to various existing shells and have been in use for several months, proving popular with both novices and experienced users. The lastest version has been implemented as one of a number of enhancements to the System V.2 shell.

Comment

To use Mike's own words, he is one of those people who likes to add new features to the shell when he has nothing else to do.

The M-shell, as he calls it, has a built-in editor (uses termcap/terminfo), a history mechanism based on command completion, job control, tilde (~) expansion, and functions.

The editor is not based on either *vi* (no modes) or *emacs* (no windows), still it is very easy to use, because it will adapt itself to the keyboard of the terminal, using the special keys (e.g. insert char) if they are available.

This talk focused on the history mechanism, which is completely different from the event number based mechanisms used in C-shell and Korn shell. The command completion is very simple to use: type in the first few characters of the command, hit the escape key, and the rest of the command is filled in from the history list. If the command from the history list was not the correct one, hit escape again, and the next command that matches the prefix is shown. Commands are shown from the newest to the oldest, with duplicates removed.

The same mechanism can be used to complete a command line argument: hitting escape brings forward all matching arguments from the commands in the history list one at a time. When all the argument lists are exhausted, matching file names from the current directory will be substituted.

Experience has shown that in most cases, one hit on escape is enough, and more than three is very unusual.

Q. Doesn't the automatic adaption to various keyboards make people confused?

A. If they become confused, they can always use a standard control sequence.

Q. Can the entire history list be edited?

A. It is possible.

Q. Start-up time?

A. Interactive start-up time is less than one second.

Q. Availability?

A. Needs a source licence, but mail me (mb@cl.cam.ac.uk)

European Languages in UNIX

Conor Sexton
Motorola International Software Development Center, IRL

Abstract

This document describes the approach adopted by Motorola to the internationalization of a UNIX System V derived operating system — Convergent Technologies CTIX 3.0. The initial goal was the provision within CTIX of character sets enabling easy use and interchangeability of U.S. English, French-Canadian and six European languages. The problem is broken down into its constituent parts, and the solution, as well as the scheme of character-set representations employed, is outlined. The many difficulties encountered during the development and testing are described. Finally, an insight is given into the direction of future Motorola development in the area of international UNIX.

Comment

The current implementation uses an 8 bit internal representation of the character set, which is not sufficient in general, but to go further, Motorola waits for AT&T to put 16 bit characters into System V. Even the limitation to 8 bit characters gave lots of problems with both the standard utilities, and application programs. For example none of the editors worked with 8 bit characters, which gave some bootstrapping problems. The hardest job was to test out all the utilities and applications to see whether the job was done.

Their system uses a special terminal with down-load facilities for keyboard mapping, fonts, and dead key sequences (e.g. ’e). The terminal uses three byte codes which represents a 16 bit character in a Xerox standard character set; these codes are converted to and from a single 8 bit character in the device driver.

Q. Does the sort utility work?

A. Not at all!

The COSAC X.400 Message based Network

Claude Kintzig
CNET, F

Abstract

COSAC is a message based network, that uses the principles as described in the CCITT X.400 documents. The talk will include a short introduction into the principles of X.400.

1) MHS-X.400
   - presentation of the model
   - services
   - Pl and P2 protocols
2) COSAC Version 5
   - COSAC under UNIX
   - functions: message transfer, mail, file transfer
   - COSAC port to IBM, VAX, MULTICS
   - network growth
3) The future
   - Additional functions: job transfer, distributed bulletin board, directories.

Comment

The current system is written in Pascal with a clean adapable interface to the operating system. There are mail interconnections between UNIX, MULTICS, EUROKOM, MISSIVE, VAX/VMS, and IBM. Present developments aim at interconnections between COSAC and EUNET, and between COSAC and CSNET.

Q. What network do you use?

A. X.25

Q. Have you interconnected to EAN, the X.400 system from University of British Columbia?

A. We are planning to do that.

Q. What UNIX version?

A. Version 7.

Communications Solutions for Mainframe UNIX

Jim Hughes
Summit Operations, UNIX Systems Development Lab, Amda

Abstract

Bringing UNIX to the world of high speed mainframe computers presents major challenges to communications. UNIX was developed around full duplex asynchronous terminals, which are not commonly used in large data centers.

In creating the UTS port of UNIX to the 370 architecture class of processors, special interfaces and protocols were established to allow the use of full duplex, asynchronous terminals as well as the more standard bisynchronous terminals.

With this full duplex support software users can interface with UNIX applications in exactly the same fashion as would be used on mini- or microprocessor based implementations of UNIX. Thus an editor such as *vi* is available.

This paper discusses the technical aspects of full duplex support software, as well as current interfaces for X.25, Ethernet, and other networks. Through these communications solutions and the power of UNIX on a mainframe computer, the end-user has an effective tool for creating and accessing corporate applications.

Comment

A long (and rather dull) presentation of full duplex/half duplex/echoplex (and he kept on and on with it!), with special attention to the problems on the mainframe side, was given. Amdahl had to make special code for the communication processor to make it work. Problems with ASCII contra EBCDIC on the mainframes was discussed too.

Q. Other comm.processor than 4705?

A. Yes, 7171!

Q. SNA?

A. Investigation going on, but X.25 is seen as a better solution.

Q. Ethernet solution?

A. Some users are looking into it, but it's new technology whereas 4705 is tested equipment.

The ANSI Draft Standard for the C Programming Language

Mike Banahan
The Instruction Set, GB

Abstract

The ANSI draft standard for the C programming language (ANSI X3J11) has been published. This talk will discuss the work of the committee, the major new features or changes that the standard includes, and the likely effect on current and future programs written in C.

Details will be given on how to obtain a copy of the draft, and where to send comments for consideration before publication of the 'final' proposed standard.

N.B. This is a talk, not a technical paper. No serious technical issues will be addressed, no Nobel-prize winning concepts introduced.

Comment

ANSI X3J11 C standard.
Who are they? ("....travelling for the dinner?")

- approx. 50 committee members
  - industry
    — users (some)
    — suppliers (lots)
  - educators
  - observers

Almost anybody may join in, there are presently 2-3 European members.

What is the standard about?

- define syntax of language
- standardize library routines
- consider external factors
  - character set
  - file handling

Aims:

- remain true to the spirit of C ('....but cleaned up')
- improve portability ('....hard to do')
- break fewest existing programs
- introduce limited enhancements
- remove ambiguities
  e.g extern-declaration in inner scope has been handled in 3 different ways in the past.

Conformance:
A conforming implementation will:

- guarantee mandatory semantics
- define certain others
  — right shift extension
- leave some undefined
  — a <<= BITS_PER_LONG+1;
- some things are not specified
  — f(getchar(), getchar());

Classes of programs:

- erroneous — use defined behaviour
- 'non-portable' — use implementation-definition behaviour
- conforming — use only defined behaviour

Language:

- complete rewrite of reference manual.
- Rules on **extern** and **static** declarations disambiguitized.

Identifiers:

- up to 31 characters
- different objects must have first 31 characters unique
- names of the same object should be equal in more then 31 characters
- external identifiers — 6 chars, monocase only — 'blaim brain-damaged archaic linkers'

Types:
Some new types

- signed char
- unsigned char
- char
- [signed/unsigned] [long/short] int
- [long] double

Storage classes:
New storage classes

- volatile — may be subject to external changes
- const — "I promise not to write to this"

Void
The type with no type (eh!). Already in modern C, but (void *) is the new universal pointer.

Arithmetic on float expressions, may now be done on float, not double.
enum IS int
structure members have different space.

Functions prototypes:
If used once, they must always be in scope of the calling function.
Arguments are converted — sqrt(4) is ok.
Scope of function argument names extended into first block:

```
f(x) { int x;   —   now gives an error
```

New function def.:

```
f(float f_arg, register int i_arg){
```

Libraries:
Two file types recognized

- text
- binary

binary — 1:1 correspondence to file contents
text — map \n to \r\n

Effects?
How much code does this break? — New keywords always breaks code!

Q. Is the manual readable by users?

A. The reference manual is for compiler writers.

Q. Does the function prototype names have to correspond to the formal names?

A. No, they are ignored in prototypes.

Q. Sizeof?

A. The size of *sizeof* is implementation dependent (defined in header file).

Q. Is *entry* not defined?

A. No, dropped as a reserved word.

Q. Union initialization?

A. Was not included.

Standardised Art as a Vehicle for Enhancing Market Penetration
Mike O'Dell
Group L Corp., USA

Abstract

This talk will examine the curious notion of "standardized software" with specific concern for the implications of this notion for groups interested in advancing the state-of-the-art. The approach will be to take a surprisingly valid analogy, and then push it well beyond the breaking point.

Comment

Marketing resembles system programming! More people do it than are capable of doing it. Any ad claims to be "standard" —

'From Now on, Consider it Standard'

looks nice, but what does this mean? What does standard mean? Let's look at some definitions from the encyclopedia:

1. A flag, banner, or ensign.
2. An acknowledged measure of comparison.
3. A pedestal, a base or stand.

(a list of over 10 definitions were given)

And as an adjective

- of average, but acceptable quality

"isn't English wonderful?"

Standards should be minimums not maximums for a system!

Let's look at Software:

- erroneous perception of portability
  - CP/M -> all ran on "the same" Z80 systems ("....simple, nothing to do!")
  - MS/DOS -> **all** PCclones
- vast array of software for sale everywhere
  - some programs are wonderful
  - many programs are of questionable quality, if not down-right worthless
  - cannot see their worth before you buy it
    - due to both number and marketing practice
    - why are there 150 word processors for IBM/PC — is this quality?
  - the captive software market has been incredible lucrative for some
    - if the customers still pay they must like what they get (?)

How does this apply to Art?
In some funny ways. How does it apply to UNIX?

- there are limits to compatibility
  *68000 binaries will not run on *86 machines!
- unmet needs — people needs networks
- there is a difference between utility and dogma
  "....you can't just look at a program to see if you look at it, like you can do with art"

Good things about the UNIX and C standards

- standards shouldn't innovate
- statement of minimum functionality
- stability

Bad things

- deify and encourage the assification .... antique and seriously inadequate technology.
- perpetuate widely held notion

Many people hope it will be lucrative (when UNIX becomes standardized), may it R.I.P. (Rest In Peace).

Q. Problem with standard! (?)

Q. Usenix lots of arguing — already 90% moveable — establish min. standard, not max.

A. Yes, we can easily move! Version doesn't matter this much.

Q. Standards are good in some ways and dangerous in others. The worst risk is that it may lock people to old technology.

The X/OPEN standard

Jacques Febvre
Bull Sems, F

Abstract

The X/OPEN group has been created by major suppliers of computer systems. Their objective is to establish a common applications environment providing portability of applications at the source code level on a wide range of machines.

This common application environment is described in the X/OPEN portability guide presented here. It contains:

- The UNIX System V interface, as described in the AT&T "System V Interface Definition"
- The C Language.
- The FORTRAN 77 Language.
- The ANSI 74 COBOL, extended with accept and display verbs as defined in Microfocus Level II COBOL.
- A definition for ISAM which is a major subset of the C-ISAM product published by Relational Database System Inc.
- A definition of 5.25" floppy format for source transfers

Benefits of this Common Application Environment for independent software vendors, end users and computer suppliers are also presented.

Comment

One of the marketing issues of the X/OPEN work is to increase the perceived credibility of UNIX, because the technical community is talking it down. For example, the existence of many versions make people complain about portability problems, whilst in reality, porting code between UNIX systems is magnitudes easier than porting between almost any other type of system. In Europe, the big manufacturers have not been able to cooperate by tradition, so the X/OPEN common technical infrastructure work is untraditional.

The X/OPEN standard is a minimum standard; so there is still room for competition. The kernel extensions specified in the SVID will be individual options, except that shared memory, semaphores, and messages may be omitted.

X/OPEN also participates in standardisation and internationalisation work, because the spirit of the cooperation is to use existing standards. The X/OPEN group is open for new members to join it.

Q. Why 5.25" floppies?

A. They are already there, and we need them now!

Q. Why have you ignored Berkeley?

A. We haven't. They are important input, but our aim is at the commercial market.

Q. What influence have AT&T had on the X/OPEN standard?

A. The X/OPEN portability guide was made in close connection with the AT&T SVID team.

Q. Why didn't you use the *de-facto* standard, Xenix?

A. The portability guide does consider Xenix.

Q. When will the X/OPEN members use the new standards?

A. They will move towards it in the future.

Q. You have to do this right now, or it will be too late!

A. We agree, so we are interrested in cooperation with the user communities.

Q. Why do you use *cpio*?

A. It is in SV, but we will support tar as well.

Q. (Jim McKie) Why is the X/OPEN book so expensive. To get people to comment on it, it should be given away for free; remember, we are here today because somebody gave something away for free...

A. The book was published by a professional publisher, which determined the high price.

End of the Conference

On behalf of the Conference organisers, Nigel Martin summed up the lessons from this conference:

EUUG will try to book one hotel in Florence.

The tape distribution was a success (but as was shown later, not without problems).

He thanked all that had participated in making this conference work, especially AT&T who flew over certain speakers from the US.

The proceedings from the conference will be sent out to all participants.

The conference ended by giving Jim McKie a present for his long and loyal service for the EUUG, now that he is leaving for the US. The present was, of course, the X/OPEN portability guide, so he got it for free after all.

Impressions from the Industrial Sessions given at the EUUG Autumn 1985 Conference in Copenhagen September 10-11, 1985

Bo Svarre Nielsen, Hewlett-Packard Danmark A/S

September 10, 1985

The first day started with some tutorials given by *The Instruction Set Ltd. (GB)* , - the first about Lex and Yacc presented by *Mr. Jim Olroyd.*

The tasks of a scanner and a parser were well explained. The idea of Lex and Yacc was presented on a conceptual level with some examples including the well known calculator example.

Mr. Bill Fraser-Campbell gave a tutorial primarily on device drivers and secondly on functions in 5.2 shell.

The tutorial on device drivers was quite technical and was mainly intended for people having a source license although other people (like me for example) got a better understanding of the way specially the tty driver works.

The explanation of functions in 5.2 shell was fine with a lot of good examples. There were lots of good ideas in the section on functions to implement user environments.

After a small lunch (very small, because all the resturants vere closed leaving most people to buy bisquits, apples etc.), the President of *Yates Ventures Inc. (USA), Mr. John Kiefer,* gave an overview of the UNIX market.

His main point was a theory that the UNIX market is in good growth although not as fast as some UNIX people believe. Single user systems like MS-DOS will have the highest growth rate, and UNIX is not very well recognised in the large operating systems world. However, UNIX will be the de facto standard for medium size multi-user operating systems. Standardization is extremely important in order to maintain the growth of UNIX. Presently there are more than 35 versions of UNIX.

The next session was an overview of the products from *Relational Database Systems Inc. (USA)* given by the Vice President, *Mr. Skip Hawk.*

The goal of this company is to have full compatibility from micro systems to mainframe for their data management software. Their product line consists of several data mangement packages, e.g. the well known C-ISAM file access method, the relational database system INFORMIX, the query language INFORMIX-SQL, a screen builder and a report generator.

Mr. Neil Urquart, Spinx Ltd. (GB) gave a personal view on the problems in distributing software. The main problem in distributing software was not really the number of different vendors, but the number of different software packages. These two multiplied together gave a fair number of combinations, but with a third dimension, which includes all different versions of hard- and software, the number of combinations became unmanageable.

The last speaker on that day was *Mr. Daniel Flickinger, CCL Datentechnik AG (BRD)* who talked about the problems of developing and maintaining a program system of 230.000 lines of C-code for several UNIX systems.

It turns out that in practice the UNIX programs are less portable than one might think. In order to keep programs compatible, it was necessary to use only a part of UNIX library functions and even rewrite some of them, e.g. the string handling routines.

SEPTEMBER 11, 1985.

In the morning session on September 11, *Mr. P. J. Cameron, Plessey Microsystems Ltd. (GB)* talked about "Development Methods for High Performance Commercial UNIX Systems".

He stressed the idea that high performance systems can only be made by a team of hard-, soft- and firmware specialists so that together they can work out the optimum design. He gave a couple of examples of I/O cards designed for UNIX systems by such a team.

Mr. Peter Frenning, Altos Computer Systems A/S (DK) gave a presentation of the Altos computers concentrating on network types and how the UNIX filesystem could be extended to run in a distributed system.

Just before lunch - and on that day we could get lunch - *Mr. Tony Heywood, Redwood International Ltd. (GB)* talked about integrated office software.

He pointed out the traditional differences between dedicated systems with emphasize on capability, but with little possibility for integration and integrated systems with less capability and how the product UNIPLEX tried to achieve both.

After lunch, *Mr. Peter Grøndahl Nielsen, Danware Data Aps (DK)* explained how DAM+, a database tool for UNIX, worked. Its most important characteristics were the duplication of data imbedding the usual transaction log in the database which assured that the database always was consistent.

Construction of user interfaces was the subject for the next two speakers, namely *Mr Henrik Lindberg, Carl Lamm Systems AB (Sweden)* and *Mr. Paul Clarke, Data Logic Ltd (GB).*

Using different approaches, both gave possibilities for construction of general purpose user interfaces.

Mr Steve Bonniwell, Apollo Computer A/S (DK) continued after a break to present the graphics product structure used in the Apollo computers.

Following this *Mr Ivan Costarov, ICCC Aps (DK)* presented a very technical talk about the problems in implementing UTS 4000 communications systems under UNIX.

The very long day ended with two speakers - *Mr. Mike Southon, The Instruction Set Ltd (GB)* about the criteria for selection of UNIX training and *Mr Basil Cousins, ICL (GB)* about the commercial relevance of X/OPEN to discuss standardization and the reasons for this.

YET ANOTHER GOOD CONFERENCE

Sebastian (Snoopy) Schmitz

The Purpose of this Report

My goals are to amuse and inform (in that order, but in nearly equal proportions). It is also intended that it everybody who did not go will become insanely jealous and will make sure they attend the next conference.

All views expressed are my own and not my employers. All usual trademarks and footnotes are acknowledged. Shame that one always has to say this, but I am no friend of hot water...

Day 0 (Tuesday)

This was travelling day. Had my first frustration because I arrived at work, having forgotten my shampoo. This was terrible. A mad rush to the coiffeur at Munich airport was necessary. He demanded DM 7.50 for a shampoo that normally costs DM 3.00 !! I tried for a discount (i.e. batted my eyelids at him in a VERY obvious manner - but no avail. . .)

Then I rushed into the departure lounge, where I batted my eyelids at the stewardess at the checking in desk. This was rather more useful, as I found out that Duesseldorf was congested. I seized the opportunity and hopped on an earlier plane, as my margin for my connecting flight to Copenhagen was pretty small. Ah, yes. The stuff executives are made of: quick decisions ....

My first ever SAS flight was a very pleasurable one. The stewardesses are actually *nice* and they smile, and they joke around with the passengers. I just could not visualise the efficiency-ticking Lufthansa lass doing same. Any person, who is tall, blond and handsome (like moi) is addressed in 9600 Bd Danish and the first question is not: "Do you want something to drink ?" but rather "Do you want Whisky or Gin ?". The guy next to me had three beers and two whiskies. That would have killed me.

Arrival Copenhagen. Steady drizzle, friendly people.

Day 1 (Wednesday)

The day started off with another frustration: Danish hardware does not support style. That is to say that the plug of my hair-blower did not fit the wall receptacle in the hotel. So much for the trendy/new wavy hair cut. Sigh.

Arrived at the Bella Conference Center with wet hair, where I spotted lots of familiar faces - Achim Brede (DEC), Daniel Karrenberg (the guy who always bounces back our mail from unido and sends us all those bills.). Also met Jean Wood from DEC, who immediately clasped my neck firmly in her hands and started strangling me right there & then. (Achim & Daniel were too stunned to react, not that they would have). She shrieked: "So I finally got you, you who never says anything nice on the net about the MicroVAX2 !!! AAAAArrrggghhh...". I only managed a choked reply: the emotionality of the scene was getting to me. Jean is a really nice person, we got along well. Soft hands, too.

So then it started: I was actually there from the beginning...

Welcome by Keld Simonsen/DKUUG

This time there were about 350 attendees; the EUUG membership has reached 1700... The GUUG (the German chapter) is pronounced Gaagh, Keld politely refused to pronounce the name of the FUUG (Finnish UNIX users group).

He asked why people come to Copenhagen, and responded to his own question with:

- Food
- Cakes

- Ice cream
- Beer
- The Tivoli
- Jens Olsens World clock
- David Tilbrook

Of course, he got it wrong. I came for the SINGING for he sang a very spirited version of "Welcome to Copenhagen", a derivative of his latest hit-single "Welcome to Copenhagen". It was very nice.

AWK by Brian Kernighan

Brian Kernighan is one of the original superstars, who manages to fill the largest venues during his world tours... He gave a talk on the latest developments with AWK. AWK, as we all know, is something we cannot do without. Its most famous error message is "Bailing out near line 1". Actually he made the point that AWK was only intended for one line programs; now he has the problem that people come in with 1000 line programs. He enhanced AWK with the following :-

- A close statement for files
- A system call (like C)
- Trig functions
- Functions (most of those 1000 liners were repeated code due to lack of these).
- Argc & Argv
- Better input functions
- Dynamic regular expressions with variable field separators

Lots of the new features are really necessary and it sounds as if AWK is really getting its act together. Wish I could have it. Unfortunately for us 4.2 sites it is not yet available.

The more interesting point (for me) was that AWK stands for Aho, Weinberger and Kernighan (the three initiators). I seriously always believed it stood for awkward. Oh well - you never stop learning.

Coffee and Tea and Kringeler

Kringeler is Danish for goodies...

Object Formats by Martijn de Lange

Martijn is from ACE, a company that buys their coffee from money made by selling SysV kernels as binaries and support, consulting etc.

These people have extended the COFF (Common Object File Format). Object does not refer to modern art but rather binaries and relocatable files (a.outs and .o's). They have kept the SysV semantics (for all the good it will do them). He highlighted the differences between the two systems (SysV and 4.2) re. memory management. They have extended SysV with page attributes which make for easier dynamic loading of modules etc.

This was a good talk (i.e. Martijn knew what he was talking about), and he gave a witty UNIX history. He claims that Unix is having its midlife crisis "after it was full of Joy, but then it was in the SUN too long". Groan. But we all laughed.

Lunch at the exciting Cafeteria

Achim complained that the meals are too small, We tried to goad him into getting himself a second one - but no use.

KERMIT by Johan Helsingius

Johan is from the unspeakable FUUG (Finnish Unix Users Group). He gave an introduction to the KERMIT file transfer utility, which now runs (would you believe) on CRAYs. He claimed that KERMIT utilizes 90-95 % of the available serial line bandwidth, whereas uucp only uses 60-70%.

The highlight of his talk was the slides he showed us from Finland. The FUUG is trying to convince everyone that one of the next conferences should be held in Finland, under a midnight

SUN, no doubt. He showed very inspired pictures of DEC field service engineers (on sleds) on their way to customer sites (little green tents on vast sheets of ice) where the customers (moose) accumulate.

ISO Networking by Sylvain Langlois

"Panic Trap" Sylvain (of eunet.general fame), did a lot of work putting the ISO protocols into the UNIX kernel. His talk gave an introduction of what ISO is all about and the status of his work.

Sendmail by Miriam Amos

Miriam is from the Berkeley Division of DEC. This means she is employed by DEC, but is on loan to Berserkeley. Actually the idea of an "employee on loan" seems to be a mind-staggering concept for most Americans, as she told me later. Miri helps Berkeley getting 4.3 out the door and onto our VAXen.

She related to us, how happy she was to go to Berkeley, because she does not have to take care of sendmail at DEC anymore. She tap-danced all the way to Berkeley, because she got rid of it.

One of the problems with sendmail is that it can (and does) lose mail, if you kill the process at the right time (for example). This is bad: THOU SHALT NEVER LOSE MAIL !!!

The gist of her speech was: sendmail is too huge, too complex and it just does too much. In short its a real bitch of a program to maintain (mega-applause from Daniel et al!). Its configuration file is really mean too. She reckoned that the sendmail of the future would go back to the good old "KISS" rule. It should really be simpler. More cheers.

Remote UNIX filesystems by Douglas Kingston

Douglas is from mcvax (I say this because nobody knows what the Centrum for Wiskunde en Informatica is.).

He gave a very good talk, describing remote file systems and some of their approaches (i.e. a super root, like Newcastle connection, or remotely mountable file systems, like SUN's NFS). He discussed some of the problems on a higher plane like, software transparency (ideally you don't want to recompile anything), user validation, where do you do a core-dump ? What about binaries etc ?? All interesting, non-trivial questions...

EUUG General Meeting

A lot of controversy erupted over the idea of exhibitions. These cost us (the attendees) a lot of money because the conferences have to be held in sleek and expensive conference centers because of space and power requirements; nearby hotels are sleek and expensive as well.

The general opinion was that we should have no more exhibitions. They are not really worth it. I zipped around the Copenhagen one in about 30 seconds. Nothing really interesting to see.

Evening Entertainment

After the conference, Jean (the Copenhagen Strangler) gave Achim and me a lift to our hotels. Jean's registration materials arrived late and DIS Congress Service booked her into the SAS Hotel (most expensive one in Copenhagen), where you can buy a Bang & Olufsen Stereo and have it charged to your room !!!

Both of them had a massive go at me, because of my feelings towards the MicroVAX2. I managed to get the point across that I really did like the MicroVAX2, but that I did not like the DEC sales staff, because some of them were not very well informed.

In the resulting 'discussion' Jean somewhat took her eyes off the road and managed to nearly stab my eye out with the bumper of a huge SAAB truck, while she was busy chasing two very scared cyclists off the road...

Using the lure of vegetarian food, I managed to rope in several charming ladies for dinner. So we all set off, a right likely bunch: Jean, Achim, me, Ulrike (Weng-Beckmann from Siemens), Ruth Moulton (from Whitechapel, who make workstations) plus a few people (male) from Siemens, whose exact names I cannot remember.

The food place was called "De Gro/ne Ko/kken"; for all you non-compo's this means "The Green Kitchen". The food was very good: Ulrike and I had an "eat-as-you-please" buffet (very good), everyone else had Moussaka. Achim again complained that the portions were too small, but this time he could not get more, because the kitchen closed soon after we were served.

After the desserts we found ourselves on the tiles again. Achim, Jean and I went to the Tivoli. Jean wanted to go on all these rides, but Achim and I politely refused. She accused us of being 'boring Germans'. This is worse than being called a 'boring old fart'. Achim and I sulked.

We then sat down in a Tivoli beer bar, where we were joined by Daniel and others... We entertained each other for a while, and we talked about Unix security. Jean immediately came up with this great idea of a newsletter issue on Unix security, and conned all of us into contributing. Got home at midnight, and worked on my Security paper until about 2 am.

Day 2 (Thursday)

4.3 - An Overview by Kevin Dunlap

Started the day with Kevin Dunlap, who is also in DEC's Berkeley Division (with Miriam), giving an overview of the speed enhancements.

- namei caching etc. have been improved by about 85%.
- Process management is now hashed and in three queues.
- Improved real time clock handling
- Exec is 28% faster now because it uses an improved algorithm, to copy binaries.
- Setjmp and Lngjmp have 13 % less overhad per call now.
- Lots of improvements on the libraries and utulities.
- csh was improved. From 20 sys calls per prompt to only 10.
- Inet daemon instead of one for every network.

I was really happy. Our 785 is already very swift, and even if only some of the claims work out as said, then it should be a really nifty little system. I think we will all love it....

Document Preparation by Brian Kernighan

Brian's second paper was about yatp (yet another troff preprocessor). The GRAP language is used to define graphs etc. Its actually a Pic preprocessor. It was implemented using awk. You can tell Brian works for AT&T: he never throws anything away... Troff was made 30% faster, due to diverse improvements. Nroff as gotten better ascii terminal tables.

I liked this talk, even if I don't use pic/troff/eqn.

Font Design by Art Bigelow

Art looks a bit like John-boy Walton. He wore a really stunning tie (flourescent colours etc) so that we will remember him. He gave one of the best talks at the conference.

He said that 10^12 - 10^13 letter forms (i.e. chars) get produced in the US every day. He classifies them into three categories:

- readable - the chars are so nice, they actually 'pull' your eye along the page, keep up your interest.
- legible - they are fairly easily readable, but since they are more optimised for clarity, you have to concentrate on the reading. Used mainly in technical stuff.
- decipherable - the worst case (He was obviously not thinking of my handwriting, because that isn't. You really have to strain yourself to get the information.

So he claims that the right font can actually make text more interesting. He had lots of very good slides. One good one was how the hieroglyphic symbols became transformed into handwriting and ultimately print.

He also said that the use of icons for computers is basically a step backward some 3000 years. I bet he won't say that on net.micro.mac !

Letters are a lot better for communication: we can still read texts in capital letters, typeset 2000 years ago. No other form of communication has this endurance.

He introduced the concept of spatial frequency, which is the frequency of alteration of black and white marks on the paper. The optimised frequency for humans is roughly that of a 10 - 12 point font. The old typesetters from ages ago already "knew" this - it iS the size of what most of us call "normal" print (from newspapers etc.). Font scaling is really 'tuning' text to optimise the humans perceptiveness. (My phrase - pardon me if I got it wrong, Art). Being constrained to a grid (i.e. like on a 5*7 matrix for terminals), ruins this 'fine tuning'.

As far as most terminals are concerned: he thinks that most screens are unbearable. He is right, too. He sells fonts for workstations etc. (including the Mac). Although, he is new to all this, he is successful and enthusiastic. He was very convincing. And he knows a *lot* about typography.

Simula & C comparison by Georg Phillppott

Georg is from the FUUG. Quote of the day: "Simula does it with class". He compared the two languages - first syntactically (boringly similar - or simular ?) and then semantically, specifically in the way that Simula has processes, objects, semaphores etc.

I learned (from his vintage slides) that 'Kildecode' means source code (obvious, isn't it...).

C++ Tutorial by Bjarne Stroustrup

This was a quick introduction to C++. I don't really like the idea of type checking in C - I like to write uncorrupted core-corrupting C, like on the day it was born. Bjarne (and AT&T) obviously feel otherwise.

C++ changes a lot of things. It also uses streams so printf("Hello world\n") (probably the most famous computer science quote) comes out as:

> cout << "Hello World\n"
>
> EEEEEmygads....what is the world coming to !

C++ is a preprocessor to C - hence the C-compiler does not get messed with (much). I prefer Objective C - could we have someone do a talk on that sometime (Committee, please ??)

Unix 4.3 BOF chaired by Miriam Amos and Kevin Dunlap

Whats a BOF - well, BOFFF is the sound a bird makes when it hits the ground involuntarily, losing all its feathers...hence BOF stands for Birds Of a Feather (or a "Special Interest Group").

This was a question and answer session re. 4.3 where poor Miri and Kevin tried vainly to answer everything at once. Some very interesting points arose however...

> They have this rdist program, which handles distributing sources and maintaing software across a network. This means that it will distribute updated source files immediately and try to recompile and re-install the software on different machines.
>
> Berkeley is working on its own distributed filesystem. It will not take an existing one like Newcastle or Sun's NFS to integrate into 4.4.
>
> 4.3 should be binary compatible with 4.2 (except getsockopt syscall).
>
> I/O on files (fread & fwrite) have been made 10 times faster.
>
> dbx will now also debug pascal programs. All known dbx bugs have been fixed.
>
> And lots more....yours for only $500 !!!

This BOF was a sort of name server BOF as well, because Berkeley are also giving us a name server in 4.3. A lot of discussion arose regarding its stability. Someone asked "So what if someone bombs mcvax ??". At this point I really burst out laughing because I could just see poor Jim McKie, sitting on the smouldering wreckage of his ex-VAX, clothed by nothing but the charred remains of an EUUG T-shirt.

Evening Entertainment

The conference dinner was in the Tivoli. The talloc() (table allocation) routine panicked because we arrived and asked for a contiguous block of nine places. So in the end we all scattered. I was most delighted to find myself sitting between Jean Wood and Chris Holmes (who works for Bigelow and Holmes (the font people)). If Jean has got strong hands, well Chris has got very steady hands: she is a font artist, this means she actually draws the letters in big (and they then get reduced and digitized/mutilated).

In her own words: 'One morning, I had just gotten up and had a cup of coffee. There was a bee sitting on the breakfast table and it had a little drop of dew or water on its back. I flicked off the drop with one finger - and the bee never noticed. THATS when I knew, I had steady hands...'. Another amazing profession. The dinner was good too (despite Achims comments re. portion size again...he really is insatiable), the company was enchanting and it was over far too soon. Theo de Ridder held a very funny speech (as usual), Dave Tilbrook made rude comments (as usual) and later the crowds disbanded to watch the fireworks.

We (Achim, Jean, Bjarne (of C++ fame), Daniel, Miriam and Kevin) went sauntering around the Tivoli. Well, we wound up in this beer place, where we had a good laugh, telling each other about "my worst ever system crash". I had lots of good ones to tell. When the Tivoli closed at Midnight, we all went off to the Stro/get, Copenhagens pedestrians delight. Good fun until Jean tried to extricate her care from the hands of a vigilant car park attendant, who wanted to be in bed ages ago. Thankfully Bjarne was around to handle the Danish negotiations.

Day 3 (Friday)

The final day. Very many people were late...

Screen based History for the Shell by Mike Burrows

Mike is one of the lads from Cambridge. He showed us his shell the MSH (pronounced Mush). This is a nice idea, because he does away with the idea of having to recall the command name by event number (i.e. !21). Instead you type the first few chars of the command name and the shell substitutes the rest. You trigger substitution by use of a single key, like up-arrow for example.

For example, suppose you did several commands with a c at the beginning, like a cat, a cc and a cal. Then you type a 'c' and msh would bung in 'cal'. Type ^ and msh will put in 'cc'; type ^ again and msh would substitute 'cat'. It then goes around the same iterations for the filenames.

This sounds so nice that I will mail him and ask for a version. He claims that people really like it. It seems like the best thing out of Cambridge since the Ring.

Copenhagen

The next few talks were really about standardisation and internationalisation. I could not really bear the thought of yet another AT&T extravaganza, so I decided to skip off and have a look around Ko/benhavn.

It's a lovely little town and I got along very well with the locals. Its got such a nice atmosphere that I could surely live there. There are plenty of antique bookshops (Groan, said the VISA card) and I managed to pick up several mint first editions of some old A.E. van Vogt and Delaney. That made me very happy indeed.

Then I returned to the Bella Centre, to meet Heidi from Unix Europe. I had this unresolved licensing problem and she offered to help me out. She did too. She must do miracles for AT&T's cash flow: she tells you very honestly and nicely that your problems are over if you just pay $12,000 - and thats a special offer ! - and you just HAVE to believe her. I do too. At least Ma Bell does not sell herself cheap...

After all these lengthy and hair-raising discussions I returned to the conference proper.

The X/OPEN Standard by Jacques Febvre

He discussed this very same book, which is being published. In the following discussion the old AT&T vs. 4.2 controversy resurfaced. A show of hands showed, that the proportion of one or the other was about half. Then why take AT&T as standard ? The answer is obviously that Berkeley does not own half the dollars as well. I guess the only chance would be a 4.2 Unix standard issued by the Coca Cola Corp. Then at least we would have enough money to back us. Does that mean we could have Classic Unix ?

Farewell

Generally the feeling was positive, except regarding the the DIS Congress Service.

So the conference motto for this one was:

> Your hotel has been DISorganised !

Rumours

It is rumoured that Dave Tilbrook bought something "very nice" for Helen Gibbons in the sex shop, next to the Hotel Triton.

Memorable Quotes

"Hello Robert, by an innings, by an innings, for goodness sake."
- Dave Tilbrook

"I think this system is like LEGO...do you have LEGO in Denmark ???"
- Tony Highwood

"I will shoot the next person, who asks a question"
- Helen Gibbons

"UNIX is like AIDS: its not very interesting unless you've got it or you're a doctor"
- Name and Address withheld by request.

"I do not like people corrupting themselves, be it Pascal or any other drug."
- Bjarne Stroustrup

"Is there such a thing as TOUGH FISH ?"
- overheard at the Cafeteria Nord

;login:

The USENIX Association Newsletter

Volume 11, Number 1 **January/February 1986**

CONTENTS

Load Balancing With Maitre d'

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

December 9, 1985

ABSTRACT

As the number of machines in a computer installation increases, the likelihood that they are all being equally used is very small. We have implemented a load-balancing system to increase the overall utilization and throughput of a network of computers. With this system, a busy machine will locate an underutilized one and attempt to process certain types of CPU intensive jobs there. We present here a complete functional description of the system and an analysis of its performance.

1. Introduction

In any multi-machine computing installation, there is a need to evenly distribute the workload over the participating machines. Otherwise, some machines will remain idle while others become overloaded. The Berkeley UNIX environment, with its networking capabilities, provides for the inter-connection of powerful processors. The busier machines may move tasks to the less busy machines, offering a more even distribution of workload across the entire system, and a decrease in overall command response time.

This paper describes one load-balancing system called *Maitre d'*[1] that is currently in use in the EECS Department of the University of California at Berkeley. For a given class of relatively expensive jobs, *Maitre d'* will attempt to locate a lightly loaded machine and process the job at that machine. In this way, imbalances in processor demand across machines can be smoothed through an automatic redistribution of the load.

This paper is broken into several sections: background, functional description, operational considerations, implementation notes and performance analysis.

2. Background

Maitre d' was originally proposed by several students at Berkeley to relieve peak usage demands on the machines used by the Computer Science Division of EECS, particularly those dedicated to instruction. These machines had always been VAX 11/750s and 11/780s, characterized by extremely uneven workloads. Research and administrative machines alike suffered from high daytime loads, while being relatively idle at night. Instructional machines would go unused for long periods of time, but would become so loaded in the days prior to an assignment being due that they would become almost unusable.

In December 1984 the University received a gift of six VAX 11/750's from Digital Equipment Corporation[3] as part of a grant earmarked for undergraduate research and instruction. These machines were not ready for assignment to formal classes when the school semester began, but they were accessible through a 10 megabit Ethernet [1,6] connection with the rest of campus. Consequently, the new VAXes were designated as remote process servers for overloaded instructional computers being rented by the Department, with *Maitre d'* acting as the agent.

[1] *Maitre d'* is French for host or server.

2.1. Functional Description

Maitre d' operates around a modified state-broadcast algorithm. Every potential client maintains a list of known server machines. Associated with each server is a binary value representing that server's availability, as determined and advertised *by* the server. When a user invokes an application modified to run under *Maitre d'*, a decision is made as to whether the job should be performed remotely or on the user's machine by comparing the UNIX five minute load average against a minimum load threshold.[2] If it is determined that a remote machine should be used (local load average > sendoff threshold), the list of known servers is consulted, and the least recently used available server in the list is chosen to perform the job. That prospective server is contacted, informed of its selection and then told to perform the service.

The process of selection and contact can be made entirely transparent to the user. This is especially necessary in an instructional environment, where most users are naive and unable to handle exceptional conditions (such as failure). They care only that their job gets done, and not where.

Maitre d' can be used to off-load almost any type of non-interactive job. The initial version of *Maitre d'* exported Pascal and C compiles. It has since been expanded to include typesetting (*troff*), circuit simulation (*spice*), a true Pascal compiler (*pc*), and others (see details in Appendix A). New applications are being added as need and opportunity arise.

3. How It Works

This section describes the operational details of *Maitre d'*. Readers uninterested in the mechanics of the system should skip to section four.

3.1. Establishing Connections

Before considering the total operation of *Maitre d'*, it is important to review some of the basics of interprocess communication. This section describes the establishment of connections and the relationship between the client and server machines. It assumes no familiarity with UNIX Interprocess Communications (*IPC*). We present a brief review of *IPC* under UNIX in the following paragraph. The reader unfamiliar with UNIX *IPC* may wish to consult either [5] or [7] for a more thorough description.

Separate processes may communicate with one another using any of several alternative methods. We describe here only the user-level protocol for a connected, bi-directional stream communication capable of crossing machine boundaries. The terms *client* and *server* are used to describe the parties during the connection phase. The server is generally referred to as a *daemon*, or a process which runs indefinitely, usually blocked while waiting for an event. The server daemon listens at some well-known address[3] waiting for requests for service. The client calls the server via some high-speed communications medium (in this case, the Ethernet) and requests a connection. Implicit in this client's call is the requester's originating address. The server accepts by completing the connection with the client. Once the connection has been established, general practice is to have the server create a duplicate server process which interacts only with the initiating client, leaving the original server free to continue listening for further requests. Once the connection is established, both the client and server may read from and write to their common connection as though it were a standard UNIX file. This makes data transfer between the two processes extremely simple.

[2] The UNIX five minute load average is defined as the average number of jobs in the run queue exponentially smoothed over the past five minutes. It is a limited metric, but very cheap to obtain. For a more complete evaluation of load metrics, see [2].

[3] An address is the combination of the machine's internet address and a local port number.

3.2. System Components

Load balancing under *Maitre d'* comprises four distinct components:

(1) *maitrd*[4]
All machines which are to be able to off-load jobs to other processors run a *maitrd* daemon. This process maintains the list of all server machines, including status information as to whether or not they are currently willing to accept jobs. For the remainder of this paper, the terms *maitrd* and *client* may be used interchangeably.

(2) *garcon*
This is the server daemon running on each machine which is to be a compute server. It has two functions: maintaining status connections with client machines, and accepting jobs from application programs. *Garcon* and *server* may be used interchangeably.

(3) *application programs*
The *maitrd* and *garcon* components provide only a control environment through which the execution of programs may actually be distributed among many processors. The software that provides the interface between the user's requested task and *Maitre d'* is referred to as the *application.*

(4) *miscellaneous*
This includes the black-box library routines used to interface with *maitrd* (described later), and a dynamic control program called *mdc* used to tune parameters while the system is running.

When a *maitrd* process is started, it tries to create control channels with the *garcon* daemons running at a pre-designated set of servers. This set is given in a startup configuration file (Appendix A) kept at the *maitrd*'s machine. Through these control channels, the *maitrd* process can determine which machines are up and accepting jobs, which are up and not accepting, and which are down.

The counterpart to a *maitrd* process is *garcon.* This daemon listens at its well-known control port for requests. When a connection from a *maitrd* is accepted, the *garcon* daemon lets the *maitrd* know whether the *garcon* host is willing to accept jobs. A server declares itself ready if its UNIX five-minute load average is less than some threshold *and* there are fewer than a given number of active users logged in to the server machine. Both of these thresholds can be set from a configuration file. After the connection is made, *garcon* checks its own status every 30 seconds, informing each of its connected clients (i.e., multicasting) whenever availability changes.

The typical configuration for *Maitre d'* is to have a set of machines in which each runs both the *maitrd* and *garcon* daemons. Each machine in the cooperative would look for help from others whenever it became too busy, and in return would be willing to take on jobs from its peers when it would otherwise be idle. One alternative to this is to introduce dedicated servers into the system (machines running only *garcon*), which accept jobs, but never send them out. A second alternative involves running a *clearinghouse maitrd* and is discussed below.

For each client/server pair, there is an open stream connection maintained for the duration of the relationship. This imposes a limit on the number of servers that may be kept track of by a single maitrd process.[5] Although not particularly cumbersome for a small number of machines, the total number of status connections for N clients and M servers grows as NM, and is order N^2 when all N machines are accommodating both *garcon* and *maitrd.* To slow this growth, *Maitre 'd* has the capability for *clearinghouse* clients. Instead of running a *maitrd* on every machine that is to offload, applications can request an available machine from a remote *maitrd.* This is most effective in clusters of diskless workstations. It is sufficient to have a single *maitrd* running on the file server handling requests from all of the file server's clients. Any reliability gained from having redundant *maitrd's* would be pointless, as the workstations aren't very useful when the file server is down.

[4] *Maitre d'* (with a capital *M* and spelled correctly) is the name of the system. One of the component programs is called *maitrd* (with a lower case *m* and modified spelling).

[5] In 4.2BSD, this limit was set by the operating system at 20. The newer 4.3 allows 64.

Certainty of state is the main reason why open connections are used for the control channels. A status update from a *garcon* is guaranteed to arrive at each connected *maitrd.* If an update is undeliverable, *garcon* assumes the intended *maitrd* no longer exists and removes it from its multicast list. Similarly, if a *garcon* disappears, UNIX *IPC* enables each connected *maitrd* daemon to recognize this *immediately* and mark the associated server machine as unavailable. The *maitrd* daemon attempts to re-establish connections to downed servers every five minutes.

3.3. Selecting A Machine

The *maitrd/garcon* connection performs no real work. Its only purpose is to give the local *maitrd* a pool of processors from which it may choose a server. In addition to maintaining connections with remote servers, *maitrd* also listens in on a second, local socket for requests from an application program, which is any program that has been modified to run under *Maitre d'.* These applications first connect to the local *maitrd* process, asking for an available machine. If the local load is less than the sendoff threshold, or no remote servers are presently available, the application is told to perform the job locally. Otherwise, *maitrd* does a round-robin traversal of its list until it finds a machine where the *garcon* has advertised a willingness to accept jobs. It passes back to the application program the internet address of this server and terminates the local connection with the application.

3.4. Program Execution

In addition to listening on its control port, *garcon* also listens on a data or service port. It is this address that *maitrd* gives to the application, and it is the responsibility of the application to create the connection. Once the connection is created, the *garcon* process creates a copy of itself. This copy communicates with the local application and executes the requested task on the remote machine, leaving the original *garcon* free to handle further requests from other applications. All communication between the application and the server is done through this data port. Commands and data are passed from the application process to the remote machine; results and an exit status are passed back from the server to the application process.

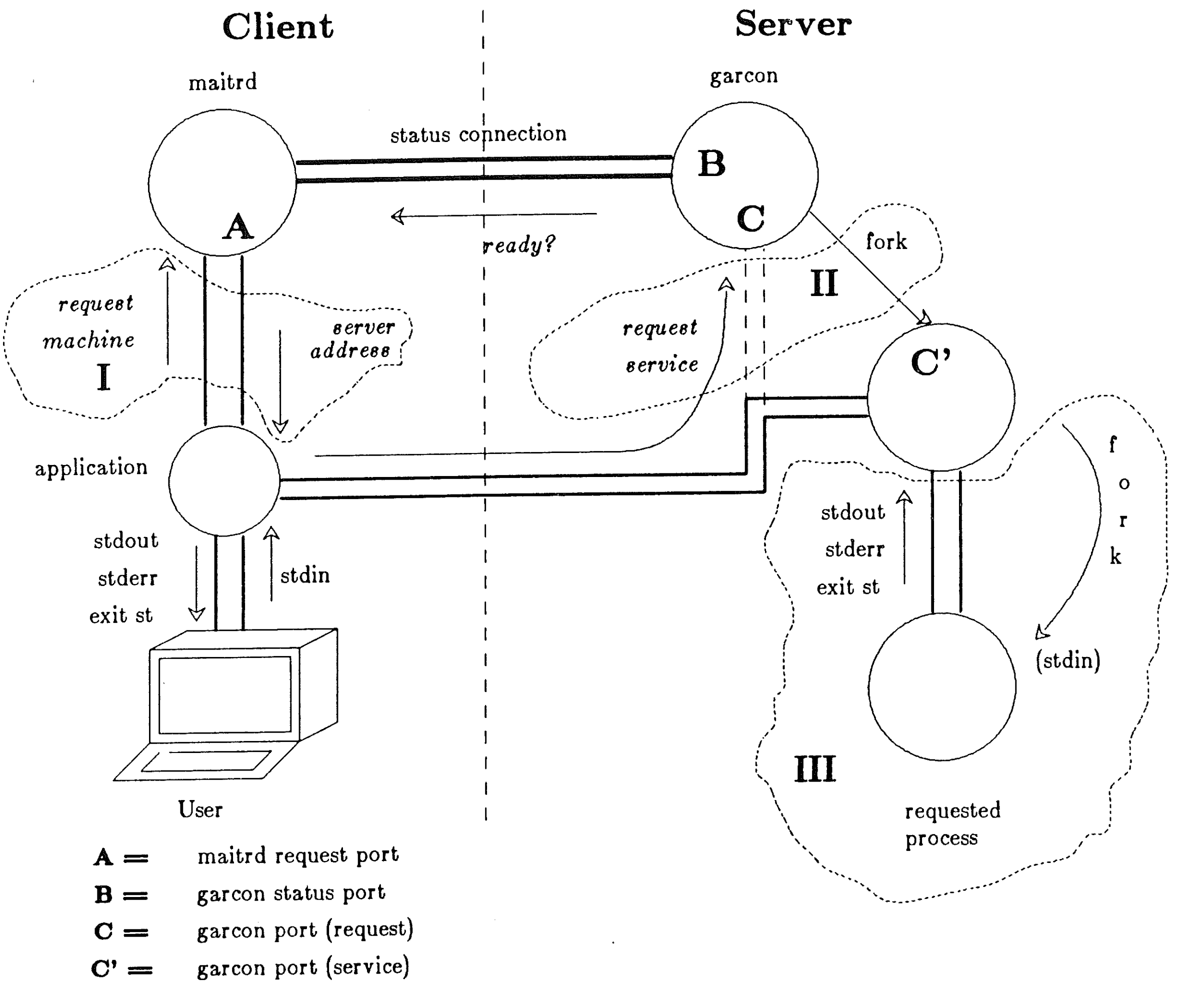
Figure I shows a typical *Maitre d'* interaction occurring in three stages. Permanent communication streams are shown in thick black; temporary ones in thin. In stage I the user invokes some application (compiler, formatter, etc.) which connects to the local *maitrd* and requests an available server. This *maitrd* has been maintaining status connections with many *garcons* (only one shown in the picture). The client daemon passes a server address out of its *maitrd* port (A) to the application program and terminates its connection with the application. Stage II shows the application connecting to the *garcon* port (C) that was returned by the local *maitrd* and requesting a service. If the application and request are valid (see **Security**), the server creates a copy of itself (called *forking* in UNIX). Note that the service port is passed down to the newly created dedicated child process (C'). The dashed lines leading into the *garcon* port indicate that the parent stops attending to the application on the other end of the channel once the child *garcon* has taken over. Stage III has the dedicated *garcon* creating the process to perform the requested task and acting as a data buffer between the application program and that task. When the requested process finishes, its exit status is returned to the application at the originating machine, and the child *garcon* terminates.

3.5. I/O Handling

The C-shell provides every process with three default *file descriptors* or data channels: standard input *(stdin)*, standard output *(stdout)* and standard error *(stderr). Stdout* and *stderr* are both output channels; *stdin* is the only input channel. Many UNIX programs are capable of taking their input from *stdin*, and placing their output on *stdout.* Convention has error messages going to *stderr.* All processes return an 8 bit status value upon termination.

Stdin to a *garcon* task is passed directly from the originating machine. But, on the return path of the service channel, *Maitre d'* provides only one data stream. The extra bandwidth needed to handle *stdout, stderr* and the exit status is obtained by returning all data from the server in finite

Figure I



packets, and prepending each packet with a four byte header. The first byte indicates the type of data being returned: *stdout*, *stderr* or exit status. If the header indicates an exit status, the second byte indicates how the process exited, and the third byte is the exit status. Otherwise, the final three bytes in the header give the size of the packet to be sent. The dedicated *garcon* process, acting as a buffer between the application and the remote task, handles the data encoding.

3.5.1. The Black Box

Although the sequence of connection, selection, and output decoding required by every application program appears complicated, there is a function called **RemoteRun** which does it all:

```
RemoteRun(inFD, outFD, cmdp);
int inFD;
int outFD;
struct pipepiece *cmdp;
```

where

```
struct pipepiece {
                    char *pp_name;
                    char **pp_args;
}
```

and **inFD** and **outFD** are the input and output file descriptors that should be used for input to and output from the remote task. There are no provisions for redirecting *stderr.* Note that **cmdp** is a *pointer* to **struct pipepiece**. A list of these structs indicates a sequence of piped commands, allowing multiple tasks to be piped together on the remote end.

4. Operational Considerations

4.1. Error Handling

Maitre d' itself does not have any provision for handling errors other than reporting them to the application. Some common error situations are:

- lost connection with remote host
- remote *garcon* prematurely exited
- remote machine could not execute process

Whenever possible, an application should recover from the error and accommodate the user's task as quietly and politely as possible. This may be done either by finding another host, or by processing the job on the user's machine. All of our applications attempt to run the job locally if there is a remote failure.

Because of redirection facilities and pipes in UNIX, a difficult situation arises if the remote server is the recipient of a local process's output, and one of the errors listed above occurs. The process cannot be restarted in any way. For example:

```
tbl file.me | matfront nroff -me > file.out
```

where *matfront* is a generic front end that attempts to process its arguments on a remote machine using *stdin* as input. Once data from *tbl* is passed to *matfront,* it cannot be retrieved if the remote *nroff* terminates due to some error related to *Maitre d'.* Even if *matfront* kept a copy of *tbl*'s output (which would be prohibitively expensive), output already generated by the remote *nroff* would have gone to *file.out.* Any attempt to restart the job might cause duplicate output to appear in *file.out.* Because *Maitre d'* operates at the user level, above the C-shell and UNIX kernel, no simple solution exists for this problem. Consequently, if an error occurs after a remote job has started executing, and the job is receiving its input from a pipe, the user's task terminates with an error message. Processes not using redirection do not have this problem, and can be restarted.

4.2. Security

When a machine services users on other machines, various security problems arise. Some of the concerns are:

Unauthorized Use
Administrative barriers must be respected.

Unauthorized Access
Use of spare cycles should not allow access to restricted files.

Unauthorized Execution
Not all machines should have to provide all services to all clients.

Maitre d' solves these security problems with client verification, task verification, non-privileged users, and logfiles.

When a connection is requested to either of *garcon's* two ports, the originating host is checked against a list of authorized hostnames contained in the startup configuration file. The request is ignored if the host is not authorized and the illegal access attempt is recorded in a log file.

If a request for service arrives, *garcon* guards against unscrupulous applications by checking to see that it has actually advertised itself as *available.* If not, the request is ignored. The request is then checked against a list of *reasonable* services that *garcon* has been told about in the configuration file. If it is not in the list, *garcon* informs the application that it doesn't know about the service. It is then up to the application to either choose another server or process the job locally.

Associated with each process under UNIX is a user name governing that process' access rights. When a task runs under *garcon,* the privileges are first set to those of some named user given in the configuration file. In Berkeley's implementation, all service tasks run as *nobody,* which is an actual entry in the password file originally created for system administration functions. *Nobody* has no password, home directory or shell and can only read public files. If process accounting is being run on the server machines, it would be worthwhile to create a dummy account used only by *garcon.* In this way, the standard accounting software could determine the percentage of resources which are being used on behalf of remote requests.

Once a server has begun a remote job, and if its load has risen above the acceptance threshold, it is possible to have this job's priority lowered or re-*niced* during the period that the server is not accepting new jobs. Active jobs from other machines will then not impinge upon jobs coming from a server machine's own users. The priority is raised again once the server's load falls below the threshold.

5. Implementation Notes

All programs that are to operate under the *Maitre d'* system require a front end (using **RemoteRun**) on the client machine to decode the returned data. Those applications that use *stdin, stdout,* and *stderr* for I/O do not need a front end on the remote machine, and may simply use *matfront* as an interface. Since *Maitre d'* supports only these three channels of data transfer, a few programs did not easily integrate into the system.

5.1. The Compilers

5.1.1. C

Creating the application to handle C compiles was relatively trivial due to the structure of the compiler, which is broken down into four distinct parts: pre-processing, compilation, assembly and loading. Pre-processing and loading are always performed on the local machine. The compilation and assembly, which comprise almost 70% of the total cycles, are done on the remote machine.

5.1.2. Pascal

A good example of a package that did not lend itself well to running under *Maitre d'* was the Berkeley Pascal interpreter (*pi*). There were problems on both the client and server end.

Server

Berkeley Pascal (*pi*) does not use *stdin* for anything, but instead requires that its input come from a file (or several files). The executable image produced by *pi* goes directly to a file and cannot be directed elsewhere (such as *stdout*). Things are further complicated by the fact that *pi* uses *stderr* for only one error message. All other error messages go to *stdout.* This causes problems for *garcon,* which expects remote tasks to communicate back to the application via *stdout* and *stderr.*

Client

Because of the way *#include*[6] files are handled in *pi,* it is semantically legal to concatenate all of the files and compile them as a single stream. Originally, we ran a very simple Pascal preprocessor over all of the user's source, scanning for *#include* directives and including them as we found them, sending over all of the files as one large program. Unfortunately, all of the

[6] When the interpreter encounters a line of the form "*#include filename*" in the source file, it reads the named file into its input stream as though it were coded in-line by the user[4].

debugging and diagnostic errors produced by Berkeley Pascal have line numbers relative to the beginning of *each* source file, so this was not a satisfactory solution. Students were being told that they had a syntax error on line 1237, when their largest file contained only 250 lines.

These problems were solved by building a simple file transfer protocol on top of the three data channels already provided for by *Maitre d'*. This extra level required another layer of pre-processors on both the client and the server machines. A user runs *Rpi* on the local machine which reads the user's program, looking for *#include* directives. Instead of starting up *pi* on the server machine, *Rpi* instead requests that *Spi* (server *pi*) be executed. Data going from *Rpi* to *Spi* includes a header giving the size of the file, the length of the file name, the file name, and finally the file. This is done for every file that is needed in the compilation. *Spi* reconstructs the original source code in a temporary directory on the server machine. When all files have been received, *Spi* executes *pi* on those files in the temporary directory and the compilation is performed. *Spi* takes care of transferring *pi*'s *stdout* to *stderr*. If the compilation completes successfully, *Spi* reads the executable image left behind by *pi* and sends it to *stdout*. This is picked up by *garcon* and sent to *Rpi*, which knows that anything coming to *stdout* from the remote machine belongs in an executable file on the local machine. The relationships between the processes, machines, and files are shown in figure II.

Maitre d' provides only a primitive connection mechanism through which a task on a remote machine may communicate with its calling process on the local machine. For tasks flexible enough to operate using only three data channels, the mechanism is sufficient. The point of this discussion on the Pascal application is to show that those tasks requiring more complicated file access *can* be accommodated by building on top of the interface provided. It may seem cumbersome and expensive to transfer all of a user's source files from client to server for each application's invocation. But, until a reliable remote file system becomes widely available, this type of explicit data transfer is necessary.[7]

6. Evaluation & Performance

6.1. Design Criteria

Alonso[2] describes six points for choosing a good load-balancing strategy: stability, implementability, cost, autonomy, transparency, and tunability. *Maitre d'* meets all of these criteria.

(1) *Stability.* A server machine could become inundated with requests from clients if it had recently announced its availability. As instantaneous state information is very hard to come by, use of an algorithm that avoids processor flooding is very important. *Maitre d'* relies on a round-robin selection mechanism of available hosts to minimize the possibility that any one client might overload a server. Although it is possible for several clients to all simultaneously request the same server, this type of selection synchronization is highly unlikely.

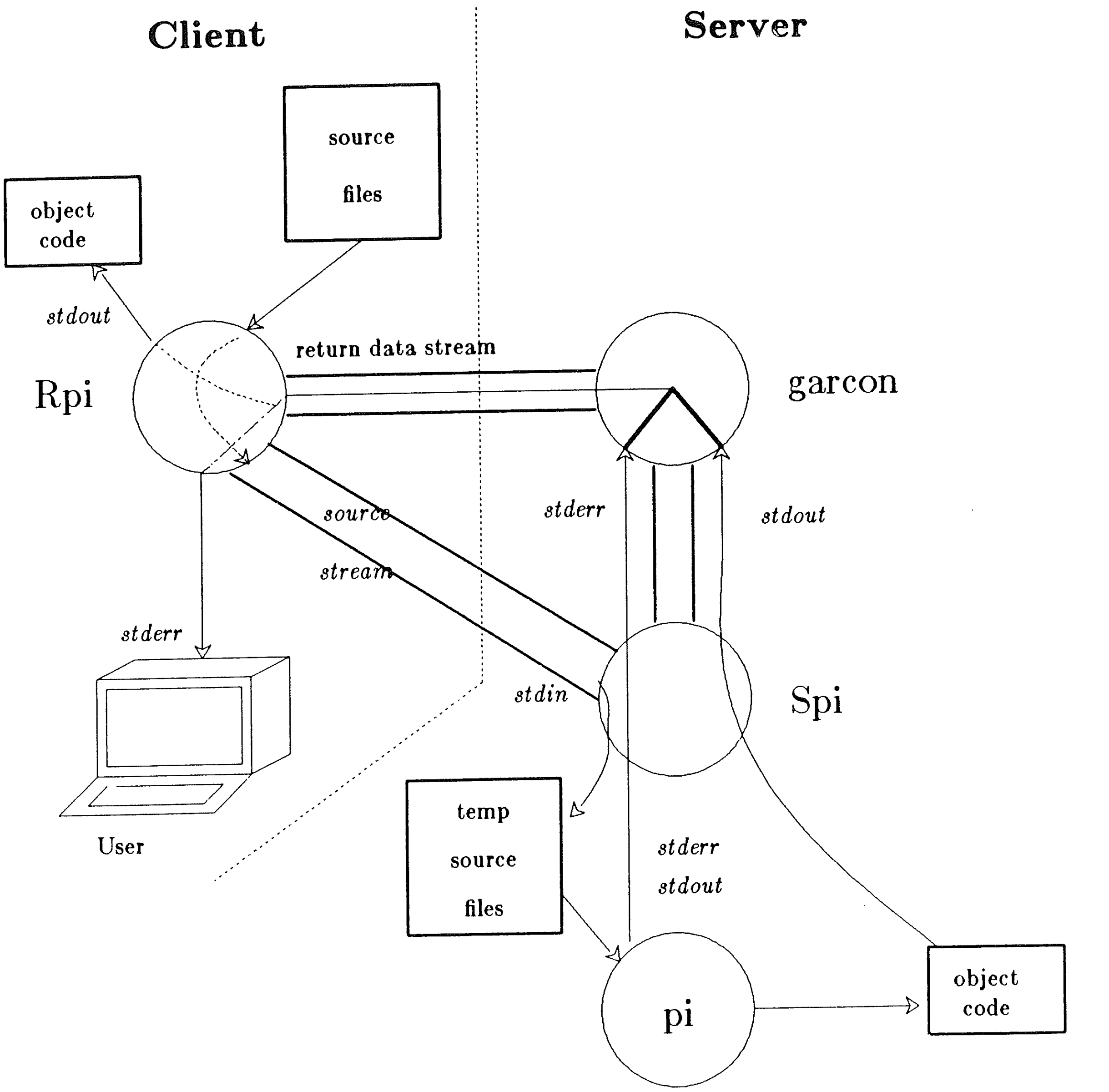
(2) *Implementability.* *Maitre d'* runs at the user level and requires no modifications to the UNIX kernel. It is running on VAX 750s, 780s, 785s, MicroVAX IIs, and a Sequent parallel processor. The basic package is about 4000 lines of well-commented C code (approximately 50K object) and is portable to any machine running 4.2 or 4.3 BSD UNIX.[8]

(3) *Cost.* The overhead in running *Maitre d'* is minimal. There are three considerations for cost: client overhead, network traffic, and server overhead. In the worst case, when no servers are available, the user's process usually requires less than .5 seconds of real time to determine that the job must be performed locally. Since the class of jobs running under *Maitre d'* are typically long-lived, this time is unnoticeable (but consistent). Traces show that it is rare for jobs to require more than a few hundred kilobytes of data to be transferred between client and server machines. On a 3 or 10 megabit Ethernet, the impact of the extra traffic is minimal. Since servers only broadcast *changes* in state information and spend most of their time blocked waiting on requests, the cost of running the server is also low. Using acceptance load thresholds of 5

[7] Even with a remote file system, the data would still need to be moved between machines. The transfer would just be completely transparent.

[8] There were some problems initially with the Sequent, but it turned out to be a bug in their 4.2 release and not in *Maitre d'*.

Figure II



and 2 on our VAX 785's and 750's respectively,[9] each server was averaging about 40 state changes a day.

(4) *Autonomy.* The decision to accept or reject jobs is *completely* up to the server, so no machine can be forced to take on work from outside clients. In addition, the types of jobs a server will accept, and the client machines from which those jobs may arrive is definable by the system administrator in a configuration file. The server machine may also impose a preemption policy on remote jobs, always giving priority to its own users.

[9] Experience has shown these loads to be comfortable operating points. Below these values, the machines tend toward idleness. Above them, response time becomes intolerable.

(5) *Transparency.* The fact that a process is being executed on a remote machine *can* be made completely transparent to the user. Users need never know that their tasks are being performed elsewhere.

(6) *Tunability.* System parameters can be set and reset through a configuration file and a dynamic control program (*mdc*). Thus, *Maitre d'* can be run in various environments without the need for recompilation.

6.2. Performance

Several factors contribute to the success of *Maitre d'*. First, the decision to export only high-cost, CPU intensive jobs allows a machine, regardless of how busy it might be, to provide swift response time for those jobs. As the less expensive, non-ported jobs are no longer competing with the more costly ones for resources, they too enjoy an improvement in response time. Table I shows comparative statistics for April 1984 and April 1985 taken on a VAX 11/780 (ucbcory). In 1984, the machine ran without *Maitre d'*. In April 1985, the only jobs being offloaded were Pascal and C compiles. About 3000 compiles per week were being exported to two VAX 750's operating as dedicated remote servers. Given are the UNIX five minute load averages; the times to start up the editor on a trivial file; and the times (in seconds) to compile and execute locally the following short CPU intensive program:

```
main()
{
        int i, j, m;
        for (i=0 ; i<1000 ; i++)
                for (j=0 ; j<1000 ; j++)
                        m = m+1;
}
```

The demands on the machine from instructional coursework were identical for the two months being compared. The increase in performance can be seen across the board in all the statistics, with an average improvement factor of over 2. The increase in perceived machine performance is even more dramatic considering that the VAX 780 was running with 16 megabytes of memory in 1984, but only half that during the 1985 sampling period. There were no other configuration changes. The decrease in variance for all the figures demonstrates that a balanced system has the added feature of offering predictable response times.

One metric used to gauge the relative utilization of machines over a long period of time is the *mean* load average. This is simply the average value of a machine's load average when sampled at regular intervals. From this value we also found:

(a) that those machines whose mean load average before *Maitre d'* was less than *garcon's* acceptance threshold value tended to have their mean increase after load sharing was in place, and

(b) that those machines whose mean load average was above *maitrd*'s sendoff threshold saw a decrease in their mean load average, as well as a marked decrease in the variance of the load average.

This simply means that those machines most in need of assistance will benefit the most from load sharing, and machines which were idle will find themselves more busy. This is exactly what should be happening and is not surprising.

Table I
VAX 11/780 Performance Comparisons

ucbcory	April 84 (w/o Maitre d')	April 85 (w/ Maitre d')	Improvement Factor
# samples	2140	2134	-
mean users	20.01	19.31	-
median users	20	19.74	-
variance users	97	100	-
mean load	6.12	2.52	2.42
median load	4.6	1.8	-
variance load	23.95	6.07	-
mean editor (secs)	1.46	.82	1.78
median editor	1	1	-
variance editor	1.87	.362	-
mean compile (secs)	11.90	7.04	1.69
median compile	8	6	-
variance compile	94.6	20.42	-
mean execution (secs)	63	26.7	2.35
median execution	30	14	-
variance execution	5564	1142	-

7. Conclusions

We have implemented an effective load-balancing system and demonstrated its utility. Further applications can be easily introduced to operate within the package's environment. Performance metrics indicate that this system is highly successful in creating a more responsive and pleasant working environment for users.

Acknowledgments

This work was made possible in part by a donation of equipment from Digital Equipment Corporation and the support of the State of California MICRO Program[3]. The original version of *Maitre d'* was initially proposed and authored by undergraduate students Chris Williams and Chris Guthrie[8]. Professors Richard Fateman and Lou Katz should be credited for their patience in seeing this project through to the end. Finally, special thanks to Doug Cooper for his tireless advice on this paper's organization and presentation.

Sites interested in obtaining the *Maitre d'* load balancing software should contact the author in care of the Computer Systems Support Group (CSSG) at UC Berkeley.

References

(1) G. Almes and E. Lazowska, *The Behavior of Ethernet-Like Computer Communications Networks.*

(2) R. Alonso, *The Design Of Load Balancing Systems For Local Area Network Based Distributions*, U.C. Berkeley Publication, Fall 1983.

(3) R. Fateman, *DEC MICRO Research Proposal*, May 16, 1984, Department of Electrical Enginering and Computer Science, University of California, Berkeley.

(4) W. Joy et al., *Berkeley Pascal Users Manual Version 3.0*, Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley.

(5) S. J. Leffler, R. S. Fabry & W. N. Joy, 1983, *A 4.2BSD Interprocess Communication Primer*, Computer Systems Research Group, Department of Electrical Engineering and Computer Science, University of California, Berkeley.

(6) R. M. Metcalfe and D. R. Boggs, *Ethernet: Distributed Packet For Local Computer Networks*, Comm ACM 19, 7, July 1976, 395-404.

(7) S. Sechrest, *Tutorial Examples of Interprocess Communication In Berkeley UNIX 4.2 BSD*, Computer Science Research Group, Department of Electrical Engineering and Computer Science, University of California, Berkeley.

(8) C. Williams and C. Guthrie, *A Proposal To Study Remote Processing of CPU Bound Jobs On Idle Computers*, Computer Systems Support Group, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1984.

Appendix A

Sample Configuration File

```
# UCBSIM
#
# Maitrd Load Balancing Configuration File
#
#               Instructions available to outside world
#                       I<command name          command path  proc uid>
#               Possible Client
#                       C<ucbxyzzy>
#               People Servers for me
#                       S<ucbxyzzy>             ... no entries means all
#               Entry both a client and a server
#                       B<ucbzyzzy>
#               Garcon configuration option
#                       G<option>       <value>
#               Clearing House Machine
#                       m<ucbxyzzy>
#
# These machines are allowed in:
Cucbear
Cucbeast
Cucbdali
Cucbarpa
# These machines should be willing to take jobs from me:
Sucbfranny
Sucbzooey
Sucbsim
Cucbbuddy
# This guy takes and receives:
Bucbernie
#
# Request from          (localhost == ourselves)
mlocalhost
#
##########################################################
#               Commands We Can Run
#
#
# Anything here can be run by people on the outside
# First column = name by request
# Second column = path of associated program
# Third column = user name for task
#
#       Icsh    /bin/csh root   would be a bad entry
#
#
Iccom           /lib/ccom               nobody
Inroff          /usr/bin/nroff          nobody
Icat            /bin/cat                nobody
Idate           /bin/date               nobody
Iecho           /bin/echo               nobody
Iwhoami         /usr/ucb/whoami         nobody
```

```
lhostname       /bin/hostname           nobody
lSpi            /usr/local/Spi          nobody          # pascal front end
lScc            /usr/local/Scc          nobody          # C front end
ltroff          /usr/bin/troff          nobody
ltroff_p        /usr/local/troff_p      nobody          # ditroff
lrwho           /usr/ucb/rwho           nobody
lspice          /cad/bin/spice          nobody          # circuit simulation
#
#
##########################################################
#
#                       Server runtime options....
#
# Load threshold above which jobs are not accepted
Gload           3.0
# Number of non-idle users above which jobs are not accepted
Gusers          20
# Maximum number of clients allowed in
Gclients 15
# When the load exceeds threshold, raise priority (renice) of
# active jobs to argument.  In unix, this is analogous to preemption.
Gnice           4
```

A Perspective on the USENET

Erik E. Fair

Technical Consultant
USENIX Association
P. O. Box 7
El Cerrito, CA 94530

Introduction

This section will be a quick description of the USENET. If you are already familiar with the basic structure of it, you can skip to the next section.

The USENET is a large, distributed computer conferencing network. The primary difference between the USENET and traditional computer conferencing (or bulletin board) systems is that the USENET is a network of computer systems, and the information is automatically transmitted to the user's computer system from other peer systems, rather than having the user call up a central conferencing system at some remote location in order to participate. The network consists of some 2000 medium to large computer systems, most of which run the UNIX operating system. These systems primarily exchange messages of English text, although foreign language text, program source code, and even executable binaries have been sent.

The software that implements this network is called *netnews.* The traffic consists of individual messages, called *articles,* which are organized into topics, called *newsgroups.* The majority of newsgroups are distributed network-wide (i.e. worldwide), but newsgroups can have a restricted distribution region (e.g. "ca.general" goes only to sites in the state of California).

In terms of the UNIX file system, articles are stored as individual files, newsgroups are directories (organized hierarchically, of course), and articles may appear in more than one newsgroup at a time without extra cost in transmission or disk space, courtesy of file links. Dots in the newsgroup name are translated to slashes to make a directory name (e.g. net.news.group → net/news/group).

When a user *posts* an article to a selected newsgroup (or newsgroups; posting to multiple newsgroups at once is called *cross-posting*), the netnews software causes the article to be transmitted to the site's immediate network neighbors, who, in turn, transmit it to their neighbors, ad infinitum. Thus the effect of broadcasting is achieved by this "store and forward" network. Although the netnews software can initiate article transmission over almost any medium, articles are transferred between USENET sites primarily over 1200 baud dial-up lines with the UNIX to UNIX Copy (UUCP) communication software.

When articles arrive on a system, they are put into the news spool directory in the appropriate newsgroup directories, under sequentially numbered files (e.g. /usr/spool/news/net/general/105). When users on that system invoke any of the user interface programs, the program will present to the user those articles on the system that he or she has not already read. After reading each article, the user can respond to the item either by electronic mail directly to the author, or by posting an article to the network for the entire network community to read. Articles remain on a system for some period of time (default is two weeks), after which they are removed by the *expire* program, to free up disk space for more incoming articles.

The USENET has been built up over the last five years by individual system administrators making agreements with each other to exchange netnews articles. There has never been any central authority or control over the network as a whole, so when network-wide decisions need to be made, the USENET operates by consensus with a dash of anarchy thrown in.

Successes

If you already have a UNIX system, the network is cheap to join: the cost is a modem and the phone bills to call the nearest USENET site. The netnews software is in the public domain, and is freely available from any site that already has it. It runs on all the versions of UNIX that we know about, and it is maintained by a cadre of very dedicated volunteers.

Because of the low cost of joining, the USENET has grown explosively during its five year history. The earliest network map I have is dated April, 1981, and it shows 35 sites in the United States and Canada. Presently the best guess is that the USENET is somewhere between 2000 and 2500 sites in size, covering North America, Europe, Australia and the Far East.

The varieties of human interest have also found expression in the nearly 300 newsgroups covering a very diverse set of topics from the excruciatingly technical (e.g. net.unix-wizards and net.lang.c) to the mundane (e.g. net.cooks and net.movies).

In addition to the worldwide and regional newsgroups, some corporations have found the netnews software useful for organizing and distributing their internal corporate communications in internally distributed newsgroups (e.g. AT&T, Tektronix).

The bottom line is that there is a wealth of information that is easily and cheaply available to anyone with a UNIX system and the time to set things up.

So What's Wrong?

Because of the unchecked growth of the USENET, coupled with a high user turnover rate, most of the users of the network did not live through the events during which the network's conventions and standards of etiquette were established. New users have also shown a remarkable unwillingness to educate themselves, and the majority of system administrators, on whom the network community depends to educate the users, have not made sufficient efforts at doing so.

The network community also depends upon system administrators to run the latest release of the software so that bugs can be quickly squashed, and improvements can be used as soon as implemented. Unfortunately, system administrators often can't or won't do an update of the software each time it is released, so the software maintainers must keep backward compatibility in mind when they make changes.

It is widely perceived that a large fraction of the articles flowing through the network are junk. Unfortunately, the definition of "junk" varies sufficiently from individual to individual, making network-wide decisions about content a very charged political issue. The anarchic consensus process that is used to make decisions at the network-wide level compounds this problem considerably.

The network also has its share of self-righteous and/or malicious users who cause problems for the network community.

Being a site on the USENET is an embarrassment of riches, in that there are too many articles flowing through the network for any one person to plow through all of them. Average traffic for a site that gets all the newsgroups is approximately one megabyte per day, and while "newsgroups" are a reasonable system for general separation of articles into topics of interest, there is no organization or structure of the articles below that level. To compound things, there are no filtering mechanisms provided by the user interfaces, other than refusing individual articles (which takes too long because there are so many articles) or entire newsgroups (which is too coarse a level of cutoff).

Lastly, most of the effort in improving the software has been directed at making transmission, processing, and storage more efficient, with comparatively little development of either existing or new user interfaces. While the former is indeed a worthy goal (particularly if we wish the software to continue to handle the increasing volume of traffic gracefully), most of the network's problems result from the behavior of individual users, and we need to change the user interfaces in order to influence that behavior toward fewer articles of higher quality.

Thinking About Solutions

Since neither users nor system administrators are behaving as we would like, we should change the software to do as much of the education as possible, and make administration of netnews as "turnkey" as possible, since we can expect that both users and system administrators will (on average) put in minimum effort, and will expect maximum return on their investment of time.

In order to do this, there are some questions to think about:

1. How can we best organize articles in a newsgroup for presentation to a user?
2. What sort of article filtering tools should be available to the overwhelmed user?
3. Since the majority of articles are responses, rather than original articles, what is the ideal behavior of a user responding to an article, and how can we encourage that behavior in the software?
4. What netnews administration tasks can be partially or entirely automated so that administration of netnews takes a minimum of effort?

Article Presentation

Ideally, one should be able to read all the articles in a newsgroup in the order in which they were posted. However, the current presentation programs present the articles in the order in which the arrived on the local system, which is frequently not the right order. Also, there is usually more than one discussion thread in a newsgroup, so that straight chronological order isn't right, either.

To create structure in the way that articles are presented, we can begin by grouping articles whose 'Subject:' header lines match. This gives us the thread of a discussion.

Unfortunately, articles can and frequently do arrive on a system out of order. To get the ordering right within a discussion we need to sort each discussion by the date of submission of each article in the discussion. This information is in the 'Date:' header line of every article, expressed in Greenwich Mean Time, so that the time stamp is absolute. This operation will give us a discussion in the chronological order in which it occurred.

However, much like conversation at a cocktail party, discussion topics tend to wander away from the original subject, but on the USENET the 'Subject:' header line is usually not changed by a user composing a response to an article. The positive aspect of this is that it is easy to find the same discussion you've been following for some weeks. Unfortunately, some poor innocent might get tricked into reading a debate about something very different from what the subject line indicated, which would be a waste of time, if it were not serendipitous.

Fortunately, there is even more information in the header that we can use to order articles into a discussion more accurately than with 'Subject:' and 'Date:.' Each article has a network-wide unique message identifier string assigned to it, usually consisting of a sequence number (or some number generated from the date and time), and the name of the host that the article originated from. When a response to an article is composed, the netnews software puts the message-id into another header line called 'References:,' so that the response has a pointer back to the article to which it is a response.

Presently, the only use that any of the user interfaces make of this field is for finding the "parent" article of the current article (i.e. the article to which the current article is a response). However, if the article processing program (*rnews*) built the doubly linked list implied by this single back pointer, and constructed a database of discussions, the following things are possible:

1. Accurate ordering and presentation of the discussions that take place on the network.
2. When the content of a message is no longer reflected by the 'Subject:' line, the users can feel free to change it without worrying that the user interface will lose the thread of discussion, because the references will still cause the response with the different subject to be grouped with the discussion.
3. Since the users will be free to change the subject lines as the subject changes, we can easily differentiate between the various sub-branches of the tree of a discussion (e.g. one branch might go off discussing "foo" from "foobar," while another discusses "bar" from "foobar").

4. With the notion of discussions firmly entrenched into the user interfaces, it will not be necessary to include the text of the article to which one is responding, because that article can be easily fetched for display.

Article Filtering

Unfortunately, even with wonderful presentation methods, there's still too much material to read. As the volume of messages in the newsgroups increases, there are two steps that a user can take to be more efficient:

1. The user can arrange to read netnews at a higher baud rate (instead of 1200 baud, how about 9600 or 19200?). This will allow making article selections faster, and hopefully more articles per unit time can be handled than at the lower speed. Ideally, an instantaneous display system would eliminate all waiting for the computer in the article selection process.
2. The user can prioritize the his or her list of newsgroups and remove some newsgroups from the bottom of the list, until the volume presented is manageable again. Unfortunately, unsubscribing to a newsgroup that one is interested in tends to leave the feeling that one might miss something really terrific.

However, these traditional mechanisms for limiting time spent reading netnews are no longer sufficient, because they're not specific enough, or too specific. What we need now is a set of automatic filtering mechanisms for articles.

Consider the following information which is contained in (or can be derived from) a netnews article header that might be useful to filter by:

author	(also known as the 'bozo' filter)
site	(they're all bozos on at that site)
date	(e.g. refuse articles that are 'n' days old)
transit-time	(e.g. refuse articles that took more than 'n' days to get here)
length	(e.g. refuse anything too small or too big)
newsgroups	(e.g. in a multiple group posting, refuse if 'net.flame' is one of the other groups)

It should be noted that in addition to refusing articles by these criteria, we can also seek them (e.g. show me all the articles by <x@y.UUCP> in net.unix-wizards).

Finally, one more: moderators are a filtering mechanism, in that they select appropriate articles to broadcast to the network. In our electronic publishing medium, they are editors.

While the whole concept of moderation has generated controversy over perceived censorship on the USENET, the ARPA Internet community has found it most useful for keeping mailing lists from degenerating into prolonged "more heat than light" debates, and that community trusts their moderators implicitly. I have observed that a good moderator can do wonders to keep a discussion on track and productive, and I think that this is one case where the USENET community really has a lot to learn from its forebears on the ARPA Internet.

Responses and User Behavior

Since the vast majority of the articles on the network are responses, rather than original postings, if we want to cut down on traffic while increasing the information content of what is posted, we should consider carefully the behavior of a user when he sees something that he wants to respond to. It is this behavior that we should attempt to affect by making changes in the user interface software.

In particular, this is the time that the user should be checked and double checked as far as the software can discern. This is the time to ask, "Are you sure that's what you want to do?" to discourage spur of the moment postings.

One of the most chronic problems of both the USENET and of electronic mailing lists are hundreds of responses to a trivial query. A user will post a message asking a question, to which everyone knows the answer and *everyone* immediately posts a response as they see the query. The network is deluged with different copies of the same answer to this trivial query, and no one is happy plowing through the resulting traffic. This occurs because both reading electronic mail and reading netnews are essentially sequential processes, with each message being handled discretely.

However, given that our hypothetical user interface is aware of "discussions" as described in the previous section, and given that there are more articles on the local system which refer to the query (one presumes that they would be answers), the user interface could hold the follow-up article until the user has read all the articles in the discussion. After the entire discussion has been read, the user interface could ask whether the submitted response is still appropriate to post network-wide, or mail to the author. Alternately, the user interface could inform the user that there are more articles to read on that particular subject, and suggest that they all be read before responding.

This could even be implemented simply by providing a mechanism in the user interface to "mark" articles that the user might want to respond to, and recapitulating these articles when all the articles in the newsgroup have been read.

While this technique is limited by the propagation speed of the network (it does no good if the responses aren't there yet), not everyone is hanging on the edge of their modem, waiting eagerly for the very next article to arrive, and that being the case, either of these two measures should cause the incidence of this problem to drop significantly, resulting in a decrease in total network traffic. Also, the propagation time of a given article should go down over the long run as the speed of transmission of articles over the network increases because of modem and network technology improvements.

The other serious problem is too many response articles posted network wide, when all parties concerned would have been better served by the respondent sending electronic mail to the author of the article that inspired the response. Right now, all the user interfaces ask what to do with a response *before* it has been composed by the user. This interrupts the flow from decision to respond, to composition of response, and does not allow consideration of the content of the response in the decision of how to send, because the response has not been composed yet.

The question of "How to send?" is best asked after the response has been composed, so that the user can go straight from the decision to respond, to composition of the response, and *then* to the decision as to whether the response should be electronic mail to the author of the article or posted to the entire network.

Asking after composition allows the user to consider the content of his response in the decision of how to send it, and also allows the composition period to be a "cool down" period, if the decision to respond was made passionately. It would also allow the user interface implementors to reduce the number of commands for responding.

Site Administration and Network Administration

Speaking as the lazy administrator of two USENET sites, I think that netnews administration takes more effort than it should. However, there are some things that can be done to help this situation:

1. Control messages that require administrator intervention could be handled as part of the user interface, when the administrator reads the *control* newsgroup. Then control messages from the wrong people or wrong sites could be filtered with the same mechanisms that normal articles are. It would be even better if they could be aggregated so that when the administrator got to the end of the *control* newsgroup, all of the actions could be executed at once (and hopefully those actions that canceled each other out could be culled before execution).

   Alternately, it might be reasonable to let all control messages be executed after a grace period for administrator intervention (i.e. if the administrator does nothing, let the control messages be executed).

2. More of the myriad files that netnews needs could be maintained automatically from a central network point (or from several points). The files that come to mind are: *aliases, moderators,* and

distributions. In particular, software to automatically generate mail paths for the *moderators* file would be helpful.

3. A *rename* control message for newsgroups so that newsgroup name space management would be a little easier. This control message should do all the work that is done by hand now (change the active file, add to the aliases file, etc.).

In short, some special handling in the user interfaces for netnews administrators would make their job easier.

A Word About *rn* and *notesfiles*

At this point I'll bet that users from both the *rn* and *notesfiles* camps are bursting to tell me that their interfaces do a lot of what I have been writing about. I disagree.

Both *rn* and *notesfiles* are still trying to follow discussions by matching subjects. While this is a good start, that method has limitations, and there is (as I have described) a vastly better method for doing so available. On the other hand, only *rn* has a good article filtering mechanism.

I think that *rn* would be a terrific base to develop the things described herein, and *notesfiles* has the subject menu done right (sequence number, number of articles, subject; such that only one line per discussion appears in the menu). Neither of them is quite what we need to cope with the great flood of information that we're currently wading through.

Some Blue Sky

Here's an incomplete wish list for whizzy new features to add to the netnews software. Fortunately for us, the header of the netnews article is easily extensible.

1. **Article types.** Suppose I have a graphics terminal, and I want to send a pretty picture. Given I could reduce the picture to some intermediate code (NAPLPS, gremlin, etc.), it would all work better if the article could be tagged such that the body of the article is passed through a program (or a pipeline) before being displayed on a terminal. Possible types or tags include: *nroff, naplps, gremlin, face.*

2. **Voting and survey support.** Imagine being able to format a survey as a list of questions and possible responses for a program that would ask the questions, collect the specific responses from the user, and mail them off to the surveyor (as opposed to posting the results network-wide, as frequently happens now). This program would be invoked by the user interface for each user that read the article, with the body of the article as *stdin*. This can be accomplished with the article type described above (the formal input could be tagged *survey* with the survey program invoked to provide the special processing).

   This might even be able to aid in network-wide decision-making with proper surveys of network opinion.

3. **Article accolades.** In many offices, different people read different publications, and if any one of them finds a particularly interesting article, they usually photocopy it for the entire office. Imagine if you will, a control message or command to mark articles that you thought were good or socially redeeming in some way, so that other people at the same site could dip into a newsgroup that they rarely read, and only read those articles marked with some number of accolade marks or with accolade marks from certain local users.

   While it might be interesting to have this information distributed network-wide, I expect that the extra traffic required would be prohibitive, so I don't recommend it.

   (Thanks to Chuq Von Rospach for the name "accolades.")

4. **Education in software.** Imagine a beginner's interface to netnews, much like the editor *edit* is to *ex,* or as the *learn* program was intended to teach the basic concepts of UNIX, that would put forth the basic concepts (what's an article, a newsgroup, a response; how distributions are used) and network etiquette.

Better help systems in the standard user interfaces might also accomplish this just as effectively.

5. **Posting to a region outside your own.** It is presently not possible to post a message to nj.general if you're at a site in California, for example. At least, not without the collusion of a friend in New Jersey.
6. **Authentication with encryption.** The network currently has no security from forgeries. It is nearly impossible for someone to be sure that article 'x' was in fact written by the person claimed in the header. If RSA style digital signatures were used, you could be sure beyond reasonable doubt. This might find best use in authenticating control messages for article cancellation, and other network-wide management.

Summary

It should be fairly obvious at this point that I believe in "better networking through software technology." This is primarily because I know how to program computers vastly better than I can program people, and there is really only one set of software that needs to change to effect the desired changes in the network, versus the thousands (or tens of thousands) of people that participate in the USENET. If the programming were properly done, it could produce a consistency of education in the conventions and etiquette of the network that would benefit the USENET community a great deal.

I think that Larry Wall, author of the *rn* user interface, has done a marvelous job at designing the first interface with real power to deal with the great volumes of articles that the USENET has been circulating in recent years, and that program is the right base for the "ideal" user interface that I'm looking for.

However, the volume of traffic keeps going up, and unless we do something more to limit the number of articles (particularly if we can distinguish and limit the number of "junk" articles), the traffic will become more than our current methods of transmission can handle or afford.

Bibliography

Erik E. Fair, "Information Overload and What We Can Do About It," *USENET message-id <10381@ucbvax.ARPA>,* September 14, 1985.

Douglas E. Comer, Larry L. Peterson, Purdue University, "DRAGONMAIL: A Prototype Conversation-Based Mail System," *Salt Lake City USENIX Conference Proceedings,* June 1984, p. 42.

Evan L. Ivie, Brigham Young University, "The Readers Workbench - A System for Computer Assisted Reading," *Salt Lake City USENIX Conference Proceedings,* June 1984, p. 270.

Starr Roxanne Hiltz, Murray Turoff, "Structuring Computer-Mediated Communication Systems to Avoid Information Overload," *Communications of the ACM,* July 1985, Vol 28, #7, p. 680.

Douglas E. Comer, Purdue University, Larry L. Peterson, University of Arizona, "Conversation-Based Mail," DRAFT TR, August 26, 1985.

;login:

The USENIX Association Newsletter

Volume 11, Number 2 **March/April 1986**

CONTENTS

;login:

The USENIX Association Newsletter

Volume 11, Number 3 **May/June 1986**

CONTENTS

The closing date for submissions for the next issue of *;login:* is July 3, 1986

USENIX THE PROFESSIONAL AND TECHNICAL UNIX® ASSOCIATION

1986 Membership Survey Results

Jim Ferguson

According to the results of the 1986 Membership Survey the *mythically* average or typical USENIX member:

- is employed by a large corporation
- is a manager or programmer with management responsibilities
- has a master's degree
- is in his mid-thirties
- earns about $40,000 a year
- has been using UNIX® for 5 years
- uses UNIX® for:
  document preparation
  electronic mail
  programming and debugging
  software design and development
- has been a USENIX member for 3 years
- is also a member of ACM and IEEE
- has attended at least one USENIX Conference
- thinks USENIX Conferences and *;login:* should remain the same, covering both highly technical topics and general interest topics
- thinks USENIX and /usr/group should hold concurrent (same city, same time) winter meetings
- reads:
  ;login:
  Byte
  Communications of the ACM
  IEEE Computer Magazine
  UNIX® Review
  UNIX® / World

That sure shoots down the stereotype image many people have of USENIX members!!!

The returns on this survey far exceeded our wildest hopes. Of the more than 1,700 current dues-paid members, slightly over 500 are new members in 1986, and very few of them participated in the survey.

Survey forms were mailed to only those members who were billed renewal dues for 1986. About 1,000 of the 1,200 who paid their renewal dues for 1986 also answered the survey – an exceptionally high return of 83%!!!

We thank all of you who answered the questions and returned the survey.

As with most surveys when the results are tallied, the author wishes he had asked some of the questions just a little differently. Thus, a few notes and comments are needed to better understand the results.

70-80% of the 332 respondents working for organizations with over 10,000 employees or students are from universities and government agencies. Most of the members do not work for corporations with over 10,000 employees.

The few members who report using UNIX® for over 12 years appear to be employees of Bell Laboratories and probably helped develop the system, rather than just exaggerating their experience.

At least 150 members – who indicated they have *NOT* attended any USENIX Conferences or UniForum Shows – favor having joint events thereby distorting the comparative percentages in Question 12. Persons who attended both the 1984 joint meeting in Washington and the 1985 concurrent meetings in Dallas, appear to favor the concurrent meetings format by a wide margin.

The ideas, comments, and suggestions in response to Questions 11, 14, 15, and 16 need further study and consolidation. They will be the subject of a future article in *;login:*.

The number of responses to the $90,000+ income range is too high as several jokesters checked that category even though their experience, education, and positions obviously place them at much lower levels.

There are two surprises. First is the decline both in actual numbers and percentage of members who work for educational institutions. The other surprise is the high percentage of members who are in executive, management, and supervisory positions – over 64%.

The following pages are copies of the survey showing the relevant percentage comparisons as well as the raw data tallies for each question.

1986 USENIX Membership Survey Percentages

Based upon 1,016 survey forms returned by May 7, 1986. Includes multiple answers where appropriate.

1. Employer or organization?

49%	Business or Corporation
28%	Educational Institution
4%	Government Agency
3%	Nonprofit Organization
10%	Research Institution
5%	Self-employed
1%	Other

1,111 responses

2. Size of employer or organization?

33%	Over 10,000 employees or students
10%	Over 5,000 employees or students
15%	Over 1,000 employees or student
7%	Over 500 employees or students
12%	Over 100 employees or students
5%	Over 50 employees or students
9%	Over 10 employees or students
9%	Under 10 employees or students

1,011 responses

3. Your position level or title?

33%	Executive or Manager
29%	Project Leader or Senior Programmer
20%	Programmer or Technical Staff Member
1%	Sales or Service Staff Member
2%	Dean or Department Chairman
7%	Professor - Associate - Assistant
2%	Instructor or Teacher
1%	Graduate Student or Student
5%	Consultant or Self-employed

1,062 responses

4. Education level?

18%	Doctorate
7%	Work on Doctorate
27%	Master's Degree
12%	Work on Master's
26%	Bachelor's Degree
9%	College - No Degree
1%	High School Graduate

1,031 responses

5. Number of years USENIX member?

1–22%	2–23%	3–24%	4–13%	5–8%
6–3%	7–2%	8–2%	9–1%	10–2%

948 responses

6. Member of what other organizations?

49%	ACM	15%	Other
36%	IEEE	26%	No answer
28%	/usr/group		

1,016 responses

7. Number of years using UNIX®?

1–3%	2–9%	3–12%	4–10%	5–15%
6–10%	7–7%	8–6%	9–4%	10–10%
11–2%	12–1%	14–.8%	15–.2%	

No answer–10% 1,016 responses

8. At its conferences and in *;login:* should USENIX:

14%	Broaden the scope of coverage to more general interest topics (e.g., e-mail)
10%	Narrow the scope of coverage to more highly technical topics (e.g., kernel)
76%	Remain the same, covering some of both

941 responses

9. Did you attend:

. . . the *concurrent* USENIX Winter Conference and /usr/group UniForum in Dallas in January 1985?

34% Yes 66% No

. . . the *joint* /usr/group and USENIX UniForum Conference in Washington, DC, in January 1984?

35% Yes 65% No

. . . the *separate* USENIX Summer Conferences in Portland in June 1985 and/or Salt Lake City in June 1984?

27%	Yes, Portland	25%	Yes, Salt Lake City
73%	No, Portland	75%	No, Salt Lake City

1,016 responses

10. Were you satisfied with the format and arrangements in:

. . . Dallas?	81% Yes	19% No
. . . Washington, DC?	76% Yes	24% No
. . . Portland?	95% Yes	5% No
. . . Salt Lake City?	92% Yes	8% No

11. What improvements do you recommend, or what was unsatisfactory?

(Responses will be in a future issue of *;login:*.)

12. Do you think USENIX and /usr/group should *again* try to hold the USENIX Winter Conference and the /usr/group UniForum either jointly or concurrently in the same city?

12% No - different cities, different times
32% Yes - concurrent - same city, same time
31% Yes - joint events
25% No answer
1,066 responses

13. Earlier USENIX Conferences you attended?

24%	Toronto - 7/83	24%	San Diego - 1/83
18%	Boston - 7/82	15%	Santa Monica - 1/82
6%	Austin - 6/81	12%	San Francisco - 1/81
4%	Newark - 6/80	7%	Boulder - 1/80
55%	No answer		1,016 responses

14. What areas or topics would you like to see covered at USENIX Conferences?

(Responses will be in a future issue of *;login:*.)

15. What areas or topics would you like to see written about in *;login:?*

(Responses will be in a future issue of *;login:*.)

16. What other benefits and services would you like to see USENIX offer its members?

(Responses will be in a future issue of *;login:*.)

17. How are you using UNIX®?

13% Accounting or budgeting
21% Administration or management
31% Data base management
62% Document preparation
65% Electronic mail
12% Hardware design and development
4% Hardware marketing
71% Programming and debugging
45% Research
68% Software design and development
9% Software marketing
37% System administration
29% Teaching or student
5% Other 1,016 responses

18. Age range?

.2%	15-19	14%	40-44
4%	20-24	6%	45-49
21%	25-29	3%	50-54
26%	30-34	2%	55-59
23%	35-39	.8%	60+

959 responses

19. Income range?

1%	$00-10,000	15%	$50-60,000
4%	$10-20,000	7%	$60-70,000
12%	$20-30,000	2%	$70-80,000
25%	$30-40,000	2%	$80-90,000
26%	$40-50,000	6%	$90,000 +

808 responses

20. What computer magazines and technical publications do you read?

79% *;login:*
1% Access
6% Business Computer Systems
51% Byte
56% Communications of the ACM
26% CommUNIXations
7% Computer & Software News
5% Computer Business News
20% Computer Design
9% Computer Graphics World
27% Computerworld
1% Data Management Magazine
29% Datamation
26% Digital Review
17% Dr. Dobbs Journal
9% Electronic Week
9% Electronic News
24% Hardcopy
38% IEEE Computer Magazine
9% Infosystems
17% Infoworld
2% Interface Age
8% MIS Week
26% Mini Micro Systems
13% PC World
12% PC Tech Journal
4% Personal Computing
2% Popular Computing
1% Programmers Journal
6% Software News
14% Software Practice & Experience
7% Systems Software
15% The C Journal
57% UNIX® Review
52% UNIX® / World
20% Other
4% No answer
1,016 responses

1986 USENIX Membership Survey Response Tallies

Based upon 1,016 survey forms returned by May 7, 1986. Includes multiple answers where appropriate.

1. Employer or organization?

538	Business or Corporation
309	Educational Institution
47	Government Agency
38	Nonprofit Organization
113	Research Institution
56	Self-employed
10	Other

1,111 responses

2. Size of employer or organization?

332	Over 10,000 employees or students
103	Over 5,000 employees or students
154	Over 1,000 employees or student
70	Over 500 employees or students
121	Over 100 employees or students
51	Over 50 employees or students
91	Over 10 employees or students
89	Under 10 employees or students

1,011 responses

3. Your position level or title?

345	Executive or Manager
306	Project Leader or Senior Programmer
213	Programmer or Technical Staff Member
12	Sales or Service Staff Member
23	Dean or Department Chairman
76	Professor - Associate - Assistant
17	Instructor or Teacher
15	Graduate Student or Student
55	Consultant or Self-employed

1,062 responses

4. Education level?

185	Doctorate
75	Work on Doctorate
280	Master's Degree
123	Work on Master's
263	Bachelor's Degree
94	College - No Degree
11	High School Graduate

1,031 responses

5. Number of years USENIX member?

1–204 2–219 3–234 4–122 5–81

6–34 7–17 8–14 9–5 10–18

948 responses

6. Member of what other organizations?

501	ACM	157	Other
370	IEEE	263	No answer
287	/usr/group		

1,016 responses

7. Number of years using UNIX®?

1–27 2–85 3–124 4–104 5–149

6–105 7–66 8–62 9–41 10–102

11–22 12–13 14–9 15–3

104 No answer 1,016 responses

8. At its conferences and in *;login:* should USENIX:

127	Broaden the scope of coverage to more general interest topics (e.g., e-mail)
97	Narrow the scope of coverage to more highly technical topics (e.g., kernel)
717	Remain the same, covering some of both

941 responses

9. Did you attend:

. . . the *concurrent* USENIX Winter Conference and /usr/group UniForum in Dallas in January 1985?

346 Yes 670 No

. . . the *joint* /usr/group and USENIX UniForum Conference in Washington, DC, in January 1984?

358 Yes 658 No

. . . the *separate* USENIX Summer Conferences in Portland in June 1985 and/or Salt Lake City in June 1984?

278 Yes, Portland	257 Yes, Salt Lake City
738 No, Portland	759 No, Salt Lake City

1,016 responses

10. Were you satisfied with the format and arrangements in:

. . . Dallas?	275 Yes	66 No
. . . Washington, DC?	266 Yes	82 No
. . . Portland?	264 Yes	14 No
. . . Salt Lake City?	237 Yes	20 No

11. What improvements do you recommend, or what was unsatisfactory?

(Responses will be in a future issue of *;login:*.)

12. Do you think USENIX and /usr/group should *again* try to hold the USENIX Winter Conference and the /usr/group UniForum either jointly or concurrently in the same city?

128	No – different cities, different times
344	Yes – concurrent – same city, same time
326	Yes – joint events
268	No answer

1,066 responses

13. Earlier USENIX Conferences you attended?

244	Toronto – 7/83	240	San Diego – 1/83
185	Boston – 7/82	153	Santa Monica – 1/82
65	Austin – 6/81	117	San Francisco – 1/81
38	Newark – 6/80	72	Boulder – 1/80
555	No answer		

1,016 responses

14. What areas or topics would you like to see covered at USENIX Conferences?

(Responses will be in a future issue of *;login:*.)

15. What areas or topics would you like to see written about in *;login:?*

(Responses will be in a future issue of *;login:*.)

16. What other benefits and services would you like to see USENIX offer its members?

(Responses will be in a future issue of *;login:*.)

17. How are you using UNIX®?

133	Accounting or budgeting
216	Administration or management
317	Data base management
626	Document preparation
658	Electronic mail
120	Hardware design and development
36	Hardware marketing
718	Programming and debugging
461	Research
678	Software design and development
91	Software marketing
372	System administration
298	Teaching or student
46	Other

1,016 responses

18. Age range?

2	15-19	134	40-44
42	20-24	58	45-49
201	25-29	30	50-54
249	30-34	15	55-59
221	35-39	7	60+

959 responses

19. Income range?

12	$00-10,000	125	$50-60,000
34	$10-20,000	57	$60-70,000
95	$20-30,000	20	$70-80,000
200	$30-40,000	12	$80-90,000
209	$40-50,000	44	$90,000 +

808 responses

20. What computer magazines and technical publications do you read?

804	*;login:*
12	Access
58	Business Computer Systems
517	Byte
566	Communications of the ACM
266	CommUNIXations
74	Computer & Software News
53	Computer Business News
205	Computer Design
93	Computer Graphics World
274	Computerworld
12	Data Management Magazine
298	Datamation
261	Digital Review
172	Dr. Dobbs Journal
94	Electronic Week
96	Electronic News
248	Hardcopy
381	IEEE Computer Magazine
88	Infosystems
176	Infoworld
17	Interface Age
83	MIS Week
266	Mini Micro Systems
130	PC World
126	PC Tech Journal
41	Personal Computing
21	Popular Computing
15	Programmers Journal
64	Software News
144	Software Practice & Experience
73	Systems Software
157	The C Journal
579	UNIX® Review
526	UNIX® / World
205	Other
39	No answer

1,016 responses

;login:

The USENIX Association Newsletter

Volume 11, Number 4 **July/August 1986**

CONTENTS

The closing date for submissions for the next issue of *;login:* is August 29, 1986

USENIX THE PROFESSIONAL AND TECHNICAL UNIX® ASSOCIATION

Reflections On A UNIX® Scheduler

Thomas A. Bohannon
Harris Corporation
Business Information Systems Operations
Systems Development
1704 Chantilly Dr. NE, Suite C
Atlanta, Georgia 30324
{gatech,akgua}!galbp!tab

ABSTRACT

Modifications made to an existing UNIX scheduler have resulted in a simple, straightforward scheduler that gives significant performance gains with a minimal amount of kernel changes. As currently implemented on UNIX Version 7 and UNIX System III ports, the scheduler has increased interactive response from 31%-87% under varied load conditions. Far from perfect, the changes appear to be a step in the right direction. Every attempt was made to enhance the simplicity and elegance familiar in the UNIX programming environment. The three main areas of discussion include the traditional UNIX scheduler, the changes made, and the results of the changes. The traditional scheduler is discussed briefly along with the change in the nature of most applications that necessitated the scheduler change. Second, the concepts behind the new scheduler along with the actual kernel changes are discussed. Thirdly, the results of the changes as they affect particular processes as well as the overall system performance are discussed.

1. Primary Goals

Reworking the existing scheduler had several primary goals. The system should provide good interactive response and a way to keep any one process from monopolizing the system resources. Any given process should not be able to consciously alter the scheduling mechanism. Perhaps most importantly, however, any changes should be kept to a bare minimum and as simple and straightforward as possible to remain well within the realm of the spirit of the UNIX environment.

The system should give good interactive response, which generally means that a user should not be able to outpace the character echo of a process functioning in either raw or cooked mode. Before full screen editors such as *vi*, nearly all utilities executed in cooked mode relying on the terminal drivers to echo typed characters. A process awaiting terminal input would block itself until a full line of input was typed. Only then would a task switch occur giving the process an opportunity to receive the incoming data. However, in a process executing in raw mode a task switch must occur for every incoming character allowing the process to receive and echo the character if needed. Generally this means that interactive processes need only a small amount of CPU time at any instant, but they need it as soon as possible.

A process should not be able to monopolize the system resources. One way this condition can occur in the standard scheduler is in a process that has a high ratio of system time to user time. The longer a process is executing kernel code, the greater the chances that it encounters a critical region and temporarily disables interrupts. More importantly, as long as a process is executing in kernel code and not blocked on a resource, a task switch can not occur to allow a preempted process to continue executing until it exits system mode.

A scheduler that allows a user process to alter its own behavior within the system immediately hinders program portability to other versions of UNIX. Resource allocation should be controlled solely by the operating system. To maintain transparency to user processes, no system or function calls should be added. A process should not have any means to alter the scheduling mechanism except the *nice(2)* system call that is found on all UNIX systems.

Any changes to the kernel should be minimal and kept as simple as possible. The UNIX scheduler is generally known to be a target of kernel hackers and tends to be the testing field of various new academically inspired algorithms. Scheduler changes can quickly become unbelievably complex. When this condition occurs the simplicity and elegance of UNIX quickly breaks down. The Berkeley 4.[23] scheduler is a good example of an idea gone awry.[1]

2. Traditional Scheduler

The traditional UNIX scheduler gives every process a fixed quantum of one second and uses a basic round-robin scheduling mechanism. A compute to real-time ratio for each process is recalculated every second. This ratio is the main value used in calculating a process's priority. The longer a process runs the larger its ratio becomes causing it to have a lower priority. On the other hand, the longer a process waits for the opportunity to run, the smaller the ratio becomes causing it to have a more favorable priority. When a process awakens after waiting for some system event to occur it is assigned a good initial priority that usually causes it to preempt the current running process. [2]

As the nature of the processes that run in the UNIX environment changes, the requirements of the scheduler change. Originally, most interactive processes executed in "cooked" mode meaning that the serial device drivers were responsible for all canonical processing. Incoming characters were placed on a queue and were not given to the process until a newline was received. As more and more interactive applications are written for the UNIX environment the system resource demands for "raw" mode processing increases. Raw mode simply means that as each character is received it is given to the process to perform the canonical processing. A good example of cooked and raw processes are the UNIX editors **ed** and **vi. ed** is a line editor which executes in cooked mode whereas **vi** is a full screen editor which executes in raw mode.

3. New Scheduler Implementation

3.1. Data Structure Changes

Very few changes were made to the existing structures within the kernel and only one new structure was added. The only existing structure modified was the process table. Three additional members were added to the process structure:

```
short p_quantum;  /* cumulative quantum value  */
char  p_person;   /* personality type          */
char  p_ratio;    /* ratio of stime to utime   */
```

The *p_quantum* member contains the current scheduling quantum of the process measured in clock ticks. The *p_person* member contains a value associated with a particular personality. This value is really a subscript into another table that will be discussed later. The *p_ratio* member contains the current value of the process's total system time divided by total user time, measured in clock ticks.

A new structure was added that defines various bounds according to the personality of a process. Currently there are four possible personalities (cpu, disk, tty, pipe) each with its own characteristic scheduling behavior.

```
struct vsched {
      short m_cpu;             /* maximum cpu clicks allowed        */
      short m_cpuinc;          /* p_cpu increment                   */
      short m_quantum;         /* p_quantum increment value         */
      short m_divisor;         /* divisor for p_cpu decay           */
      short m_cum_quan;        /* max. cumulative quantum size      */
} vsched[4];
```

The member *m_cpu* gives the maximum value that the *p_cpu* value can accumulate. *m_cpuinc* is the amount by which *p_cpu* is incremented every clock tick that the process is caught with control of the CPU. *m_quantum* is the quantum size added to *p_quantum* each time a process is given control of the CPU. The member *m_divisor* is used in the decay of the *p_cpu* value while a process is not in control of the CPU. The member *m_cum_quan* is the highest value that a process's quantum is allowed to attain.

3.2. Process Personality Types

Processes are assigned one of four personalities *(cpu, tty, disk,* or *pipe).* All processes are initially classified as CPU intensive until they prove themselves otherwise. As a process's behavior changes, its personality value will vary. The purpose is to allow the scheduler to favor one type of process over another type. Flexibility is given to the system administrator to categorize the four possible types of processes from most favorable to least favorable.

A hook was placed in *rdwr()* to set the personality byte to *tty* if the process is accessing a character device, to *disk* if the process is accessing a block device, and to *pipe* if the process is accessing a pipe or fifo. The *rdwr()* routine was chosen as the site for the hook placement for simplicity: the desired effect could be achieved by changing only one routine.

At the end of a process's quantum the high bit of the personality byte is examined. If set then the personality has been reconfirmed during the quantum just completed. As a result, the process is allowed to keep its current personality. If on the other hand the high bit is not set, then its current personality has not been reconfirmed during the last quantum and is therefore reassigned to *cpu*. This prevents, for example, a CPU intensive process from initially performing a *printf()* to gain a *tty* type personality. As long as the process is performing terminal output it will have a *tty* personality, but as soon as it becomes *cpu* intensive its personality type will change accordingly. The resources that a process receives are based on the personality type.

The CPU usage variable for each process is not allowed to grow higher than the allowable limit for its personality. Each time the process's CPU usage variable is incremented it increases by the amount found in the increment value of the given personality. The quantum size is added to the process's quantum (not to exceed the maximum cumulative limit) each time the process gains control of the CPU. This allows a process to save the remaining portion of its current quantum for future use if it blocks itself or is preempted by another process. The cumulative limit is set to prevent a process from temporarily hogging the system resources with an enormous quantum. The CPU usage variable decays while a process is not running so its priority will gradually become better. The rate at which the CPU usage decays is a function of the divisor variable which can differ according to the personality, thus allowing varying decay rates for each personality type. If one wishes to penalize a pipe intensive process more than a CPU intensive process he might assign a larger CPU usage divisor to the pipe type that will cause the CPU usage value to decay at a much slower rate than the CPU intensive process. The net result is that the priority of the pipe intensive process remains low (higher value) for a longer period of time.

3.3. Quantum Management

A quantum is a slice of processor time given to a process during which it executes. Since the UNIX scheduler is basically a round-robin algorithm, large quanta tend to make it act as a first come first serve (FCFS) scheduler. A process will execute until it completes or blocks itself. When small quanta are chosen the system resources are more evenly distributed among the runnable processes. Small quanta are favorable to interactive environments allowing quicker character by character processing for programs executing in raw mode giving users better interactive response. If quanta too small are chosen, then more time could be spent in system mode performing task switches than actually running user processes.[3] The bulk of the quantum management takes place within the clock interrupt handler. The only other routine dealing with quantum management is *swtch()*. Just before a process is set to run, its quantum size is increased according to its personality (not to exceed the given limit).

The following code was added to the clock interrupt handler to be executed at every clock tick.

```
if ( pp != &proc[0] ) {
    struct      proc  *ap;

    if ( pp->p_cpu < vsched[pp->p_person&VS].m_cpu )
        pp->p_cpu += vsched[pp->p_person&VS].m_cpuinc;

    pp->p_ratio = (u.u_stime / ( (u.u_utime)?u.u_utime:1 ))&0x7f;

    if ( --pp->p_quantum <= 0 ) {
        if ( pp->p_quantum < 0 )
            pp->p_quantum++;
        pp->p_person = (pp->p_person&VSCHG) ?
            pp->p_person&VS : VSCPU;
        runrun++;
        for ( ap = &proc[1]; ap <= &proc[Nproc]; ap++ )
            if ( ap->p_stat )
                setpri(ap);
        curpri = pp->p_pri;
    }
}
```

At each clock tick, if the current running process is not the swapper then perform quantum management tasks. If a process has not exceeded the maximum amount of CPU usage allowable, then increment its usage value by the increment variable for the process's personality. Priority calculations are based on the CPU usage value. However, there is a point where the scheduler no longer cares the exact amount of CPU usage but only that the process is using a lot. Priority calculations are also based on the ratio of system time to user time and is calculated every clock tick for the current running process.

Once a process's quantum expires, its personality is verified, the priority of all processes is recalculated, and a task switch happens. During each quantum a process must reassert its personality or it will default back to CPU.

Code still remains to calculate priorities every second for each process. Generally all lightning bolt processing is left undisturbed.

3.4. Setting Processes' Priority

It seems that Version 7 UNIX contains a function called *setpri()* that calculates the priority of a given process whereas UNIX System III does not. The basic idea of the *setpri()* function remains unchanged with the exception that the CPU usage value is decayed **after** it is used in the priority calculation.

```
setpri(pp)
struct      proc  *pp;
{
    pp->p_pri = pp->p_cpu + PUSER + pp->p_nice - NZERO + pp->p_ratio;
    pp->p_pri &= 0x7f;

    if ( pp->p_cpu && pp != u.u_procp ) {
        pp->p_cpu -= (pp->p_cpu/vsched[pp->p_person&VS].m_divisor+1);
        pp->p_cpu = (pp->p_cpu < 0)? 0 : pp->p_cpu;
    }

    if ( pp->p_pri < curpri )
        runrun++;

    return ( pp->p_pri );
}
```

3.4.1. Priority Calculation

The priority of a process is the sum of its total CPU usage, a constant (PUSER 50), the nice value, and the ratio of system to user time.

The purpose of NZERO, which is 20, is to serve as the baseline value for nice. A normal process has a nice value of 20 with a net result in the priority calculations of 0. A super-user process may nice itself down by setting nice to zero which decreases the priority value by 20 resulting in a more favorable priority. On the other hand, a considerate process may nice itself by setting nice to 40 increasing the priority value by 20 resulting in a less favorable priority.

The CPU usage value is a multiple of how many times the clock interrupt occurs and catches the process with control of the CPU. The PUSER constant is a baseline value. The best priority that a process can ever hope to have as a result of calling *setpri()* is PUSER, unless, of course, the process has been niced in which case the priority could be PUSER-NZERO (30). Note that a process can have a priority lower than PUSER or PUSER-NZERO only as a result of going to sleep waiting on an event such as disk I/O.

The idea behind the ratio of system to user time is to punish a process as it spends more time in system mode than in user mode. While a process is executing in system mode, no task switch can occur unless the process blocks itself or returns to user mode. A process awakening from a blocked state will usually have a better priority than the current running process. For example, if an interrupt occurs signaling some I/O complete, enabling a process with a priority less than the current running process, a task switch cannot occur until the current process exits from system mode.

3.4.2. CPU Usage Decay

As long as a process is executing, its CPU usage field grows thus causing it to have a progressively worse priority. On the other hand, when a process is not executing its CPU usage field decays causing its priority to gradually become better. The rate at which the value decays is a key element in calculating the process's priority. Each personality has associated with it a divisor value by which the CPU usage value is divided. The result is used to decay the process's CPU usage value. The general effect is that the CPU usage decays rapidly at first and eventually levels off to become zero.

The idea is that as a process executes, its priority becomes worse until its quantum expires and a task switch occurs. As the process awaits its next quantum, its priority becomes more desirable. When the next task switch occurs, the process with the best priority of all runnable processes is chosen.

3.4.3. Preempting Process

Whenever a process's priority is calculated within *setpri()* it is checked against the priority of the current running process. If the priority of the current running process is found to be worse than that of the process just calculated, the task switch flag *runrun* is set causing *swtch()* to be called at the first available opportunity.

3.5. Scheduler Tailoring

Obviously the values of the vsched structures play a crucial role in the behavior of the scheduler. Experimentation was done by changing the initialization values of the structures and relinking the kernel. This quickly became a slow and monotonous process. Therefore a pseudo-device was created to allow on-the-fly change of the vsched structure.

The scheduler device (*/dev/sched*) is a pseudo-device that allows behavior change of the scheduler without relinking and rebooting the kernel. A *read(2)* of the device returns the contents of the vsched array of structures and a *write(2)* to the device replaces the contents of the structures, thus allowing instant change of the scheduler. The device was initially created only as a means for testing and tuning of the scheduler. However, the scheduler pseudo-device appears to have several positive side effects that justify its remaining an integral part of the new scheduler. It might be very beneficial, for example, to have a daemon modify the scheduling mechanism in the evening hours to favor pipe or compute intensive processes and to discourage interactive processes. On a broader scope, the ease of varying the scheduler's behavior allows individual installations to fine tune the kernel for a particular environment.

One must realize that a lack of discretion by the system administrator concerning the pseudo-device can have negative consequences. However, the same holds true for other devices and particularly for the root filesystem device! With a little common sense, */dev/sched* can be a great asset in obtaining the maximum horse-power available from the system.

4. Actual Experiences

The effects of the new scheduler have been far greater than expected. In attempting to increase interactive response under loaded conditions, some degradation of CPU intensive processes was expected. However, they improved by 7-8% and interactive processes improved from 31% under light load conditions to 87% under heavier load conditions! Before the scheduler modifications, interactive response degraded rapidly as the machine load increased. With the new scheduler, interactive processes show virtually no degradation as machine load increases.

4.1. Benchmarks

The benchmark consisted of a shell script that initiated dictionary **sort**'s (25,000 words) background and performed a **cat**(1) of the *termcap* file foreground. The **sort**'s and the **cat** both have a moderate amount of disk activity with the **sort**'s tending to be CPU intensive and the **cat**'s tending to be tty intensive. In lieu of **cat**'ing the *termcap* file, further tests were performed using a process with repetitive writes to the terminal and no disk access. It, too, showed performance gains in line with the previous tests.

4.2. Everyday Life

All too often, benchmarks are developed to prove some point and rarely measure general performance. With the new scheduler, the benchmarks discussed are not the only tests that show improvement. Normally, on our processor response in **vi**(1) is intolerable with just one or two **make**(1) commands running background. With the scheduler modifications, **vi** responds extremely well with four and five **make** commands running background and shows virtually no degradation!

A real example is a terminal emulation program that continuously polls both a serial port and the keyboard for incoming characters. The select type function spends a fair amount of time in system mode and unlike other I/O calls can never block itself. When this process was running, a login sequence from another port that would normally take 12-14 seconds took over 2 minutes! With the new scheduler the login sequence takes 15 seconds with one (and even more) of the emulation processes running. Furthermore, the data transfer effectiveness of the emulation processes remains high. Although the mentioned program is poorly designed, no one process should be able to monopolize the system resources.

4.3. A Comfortable Configuration

The following table illustrates the configuration of the system on which the mentioned experiences and tests were performed. A clock interrupt occurs every 25 milliseconds (40Hz).

CPU: Intel 80186 10MHz Memory: 1.5MB				
Variable	Personality Type			
	CPU	TTY	DISK	PIPE
m_cpu	70	70	70	70
m_cpuinc	5	5	5	5
m_quantum	4	2	3	5
m_divisor	4	2	3	4
m_cum_quan	4	4	6	15

4.4. UNIX Versions & Hardware

The scheduler modifications were initially made to a port of the UNIX System III kernel to the Intel 80186 microprocessor. Later, the same scheduler was applied to a port of the UNIX Version 7 kernel to the same processor and showed performance gains equivalent to those found with System III.

Although the modified kernel has only been tested on microcomputers, there appears to be no reason that similar performance gains could not be achieved on hardware of considerable more horsepower. Every attempt was made to ensure that the scheduler not be overly complex or specific to any one type of hardware architecture.

5. Further Work

The hook placed in *rdwr()* to set the personality byte has several shortcomings. At this level within the I/O request many assumptions are being made that can conceivably be incorrect. For instance, just because a process is accessing a character device does not necessarily mean that the device is a terminal device. A process could be accessing a magnetic tape device via the raw device driver. Also at this level of the I/O request it is not known when accessing a block device whether the requested block is currently in the buffer cache. Therefore the process may or may not block itself awaiting completion of the I/O request. Perhaps a more exact distinction could be made between processes really performing terminal I/O as opposed to accessing a raw device and between I/O requests that really cause disk activity to occur as opposed to requests that are fulfilled directly from the buffer cache.

Modifications were made to the **ps**(1) command to display the three additional fields within the process table (*p_quantum, p_person, p_ratio*). This is somewhat helpful in observing process behavior. Because of the amount of processing time **ps** needs in order to locate and display the process table information, it is not a good tool for monitoring the effects of various *vsched* variables on process behavior. Perhaps a more useful observation tool would be another pseudo-device which, when read, would return proc table information for each process for a given number of consecutive clock ticks after the read request is initiated.

All of the scheduler modifications only address the problem of processor utilization and do not address the issue of process swapping. More investigation as to the impact of the modifications on the swapping procedures needs to take place.

6. Conclusions

All of the primary goals were achieved with the performance improvements being far greater than expected. The ability to change the scheduler's behavior via a pseudo-device was originally intended for initial tuning purposes only. However, after implementation, benefits other than those originally intended were seen, and as a result the drivers remain.

Attempting to solve the multitude of problems associated with scheduling is an overwhelming task that cannot be taken lightly. It is so easy to make changes that at first seem logical, but turn out to have unforeseen negative effects. Throughout the entire effort, every attempt was made to keep kernel changes to a minimum with the maximum amount of positive results. More importantly the author hopes not to have contributed to the corruption of simple elegant UNIX.

References

[1] J. H. Straathof, A. K. Thareja, A. K. Agrawala, "UNIX Scheduling for Large Systems," 1986 Winter USENIX Technical Conference Proceedings.

[2] K. Thompson, "UNIX Implementation," The Bell System Technical Journal, July-August 1978, Vol. 57, No. 6, Part 2.

[3] D. C. Tsichritzis and P. A. Bernstein, *Operating Systems, Computer Science and Applied Mathematics,* Academic Press, 1974

THIS PAGE UNINTENIONALLY LEFT BLANK

ED.

C/o Dept of Mines and Energy,
PO Box 2901,
DARWIN, NT 5794.

The Editor, AUUGN,
School of Electronic Engineering
and Computer Science,
University of New South Wales,
PO Box 1,
KENSINGTON NSW 2033

30 January, 1986

Dear Editor,

Saying, saying, sed

Ken Frame's article on 'awk' has encouraged me to contribute a couple of programs that are based on another member of the family of Unix filters, viz. 'sed'.

It often happens, if you have a data file in tabular format, that you wish to subsitute a string for the 'n'th to 'm'th characters on each line of the file. However, the interactive Unix editors contain no command which would allow you to globally replace, say, the 56th to the 59th characters on each line, unless you reckon you can accurately enter the following:

```
:g/./s/\(.......................................................\)..../repl/
```

where 'repl' stands for the replacement string that you wish to substitute.

I keep in my bin directory two files, called 'dots1' and 'dots2', which consist of lists of 'sed' commands that can help to do this job. The first five lines of 'dots1' are given below:

```
############################################
s/./repl/
s/\(.\)./1repl/
s/\(..\)./1repl/
s/\(...\)./1repl/
s/\(....\)./1repl/
[etc]
############################################
```

The first five lines of 'dots2' are:

```
############################################
s/\.\//\.\//
s/\.\//\.\.\//
s/\.\//\.\.\.\//
s/\.\//\.\.\.\.\//
s/\.\//\.\.\.\.\.\//
[etc]
############################################
```

I also keep in the same directory a C-shell command-file called 'repl':

```
##############################################################################
# repl - A program to substitute, on each line of a file, a specified
#        replacement string for a string defined in terms of its number
#        of characters and the column-number of its first character
##############################################################################
echo -n "Enter the replacement string   "                                  #1
set REPL = `echo $<`                                                       #2
echo " "                                                                   #3
echo -n "Enter the number of characters that the string is to replace   "  #4
set CNUM = `echo $<`                                                       #5
echo " "                                                                   #6
echo -n "Enter the filename   "                                            #7
set FILE = `echo $<`                                                       #8
expand $FILE >tmp$FILE                                                     #9
echo " "                                                                   #10
echo "Now examine $FILE to find the column number of the first character"  #11
echo " "                                                                   #10
echo "that is to be replaced"; sleep 3                                     #13
vi tmp$FILE                                                                #14
echo " "                                                                   #15
echo "Enter the column number of the first"                                #16
echo " "                                                                   #17
echo -n "character that is to be replaced    "                             #18
set CNUMF = `echo $<`                                                      #19
sed -n {$CNUMF}p $HOME/bin/dotsl >sedfilel                                 #20
sed -n {$CNUM}p $HOME/bin/dots2 >sedfile2                                  #21
sed -f sedfile2 sedfilel >sedfile3                                         #22
sed s/repl/"$REPL"/ sedfile3 >sedfile4                                     #23
sed -f sedfile4 tmp$FILE >$FILE                                            #24
unexpand -a $FILE >tmp$FILE                                                #25
sed 's/→[ ] [ ]*/→/g' tmp$FILE >$FILE                                      #26
rm sedfile*                                                                #27
##############################################################################
```

Lines 1-8 of 'repl' prompt the user for the replacement string, the number of characters to be replaced, and the filename of the data-file, and then set the variables REPL, CNUM, and FILE. Line 9 expands tabs to spaces in the specified file. Lines 10-19 allow the user to view the expanded file in 'vi', to ascertain and then to enter the column-number of the first character that is to be replaced. The variable CNUMF is then set. Lines 20 and 21 select the requisite command-lines from $HOME/bin/dotsl and $HOME/bin/dots2, according to the numbers which constitute the values of the variables CNUMF and CNUM respectively, and redirect the output to the temporary files 'sedfilel' and 'sedfile2'. Line 22 uses 'sedfile2' to replace the single '.' after the tagged regular expression in 'sedfilel' with the required number of dots, and redirects the output to 'sedfile3'. Line 23 substitutes the value of the variable REPL for the string 'repl' in 'sedfile3' and redirects the output to 'sedfile4', which is used to perform the desired substitution on the data-file. Lines 25-26, in which the right-arrows represent tabs, restore the tabs and eliminate any redundant spaces. Line 27 cleans up.

The command lines in 'dots1' 'dots2' make use of a 'tagged regular expression' enclosed by '\(' and '\)', and referred to in the replacement part of the substitution command by '\1'. The dots that comprise this 'regular expression' stand for the first CNUMF minus 1 characters on each line of the data-file that are to remain unchanged.

Lines 20-26 contain some interesting features of 'sed' and the C-shell. The '-n' option suppresses all lines in the output, except that specified by the addresses represented by $CNUMF and $CNUM; the curly brackets around these variable values are needed to isolate the variable name from the 'sed' command 'p'. The double quotes around $REPL are needed in case the replacement string contains spaces. Note that the quotes that are needed around the 'sed' command in line 26 to prevent the meta-characters from being seen by the shell are not used in line 23; this is because there are no meta-characters that we want to protect from the shell; in fact, we wish the shell to see "$REPL". Had there been meta-characters in the pattern 'repl' that we wished to protect from the shell, while still requiring the shell to see "$REPL", we could have used two pairs of single quotes:

```
sed 's/repl/'"$REPL"'/' sedfile3 >sedfile4                   #23
```

Yours sincerely,

David Dundas

David Dundas

Australian Unix† Systems User Group
1986 Winter Meeting
Australian National University, Canberra

September 1-2, 1986.

The 1986 Winter Meeting of AUUG will be held at the Australian National University, Canberra, on Monday and Tuesday, 1 & 2, September, 1986.

Local arrangements are under the direction of Peter Wishart at the Department of Computer Science, at the A.N.U.

The keynote speaker will be Mike Banahan from The Instruction Set in the U.K. He is on the ANSI C committee as well as being involved with the System V port of SUN NFS and the X/OPEN standard.

Call for Papers

Papers on subjects related to the Unix† system, or Unix†-like systems have been called for. Papers relating to new developmentss, new applications, or new insights or research into the Unix† system are particularly requested, but tutorial and survey papers are also very welcome.

The program committee consists of Chris Campbell (chairman), Ross Nealon and Piers Lauder. Should you wish to present a paper, abstracts should be sent to the Chairman or one of the committee members as soon as possible.

Abstracts and full papers may be sent by ordinary mail to:

Chris Campbell,
c/- Olivetti Australia Pty Ltd,
140 William Street,
Sydney 2011.
[(02) 358-2655 x 466, or 'chris@olisyd']

or by ACSnetwork mail to Piers Lauder
[(02) 692-2824, or 'piers@basser.oz']

Ross Nealon is at the University of Wollongong
[(042) 27-0802, or 'ross@uowcsa']

Hardware Display

There will be a display of appropriate hardware and software in conjunction with the conference, and vendors/manufacturers are invited to demonstrate their products. For more information on participating in this display, contact:

Stephen Rothwell
Department of Computer Science,
Australian National University,
GPO Box 4,
Canberra, ACT 2601.
[(062) 49-3890, or ‘sfr@anucsd.oz’]

Registration

The registration fee will be $50, with a $10 discount for AUUG members.

Each speaker will receive a free ticket to the conference dinner, and will be entitled to a complete remission of fees if a final version of the paper in publishable format (2-4000 words) is received by August 5.

Accommodation

Accommodation will be available at Burgmann College on the University campus at the cost of $23 per night. The ANU is situated close to the centre of Canberra. Numerous motels of varying standards are within walking distance of the conference venues.

Contacts

GENERAL	E-Mail Inquiries	ACSnet: auug@anucsd.oz ARPA: auug%anucsd.oz@seismo.css.gov UUCP: ...!seismo!munnari!anucsd.oz!auug	
	Peter Wishart	pjw@anucsd.oz	(062) 49 3850
REGISTRATION	Tony Sloane	tony@anucsd.oz	(062) 49 3890
ACCOMODATION	Jeremy Webber	jeremy@facl.anu.oz	(062) 49 2750
DISPLAY	Stephen Rothwell	sfr@anucsd.oz	(062) 49 3890

or via snail mail ...

AUUG Winter Meeting 1986
Department of Computer Science
Australian National University
GPO Box 4, Canberra, ACT 2601

UNIX is a trade mark of AT&T Bell Laboratories

Australian UNIX†-Systems Users' Group
1986 Winter Meeting, September 1-2
Australian National University
Registration Form

Name: ....

Address: ....

....

....

....

Phone: .... Network Address: ....

Organisation: ....

Fees

Registration		$50	
AUUG Member's Discount Subtract		$10	
Early Bird Registration			
(before 18th August) Subtract		$10	....
Dinner	....	at $30 each	....
Total (payable on arrival)			______________

Please reserve accommodation for me at Burgmann College for the nights of____ at $23 per night. (Payable in advance, or one night deposit in advance, balance on arrival.)

Signed:...

† UNIX is a trade mark of AT&T Bell Laboratories

Australian UNIX†-Systems Users' Group
1986 Winter Meeting, September 1, 2
Australian National University

Hardware Display Questionnaire.

1 **Company Name:** ....
Contact: .... **Phone No.:** ....

2 **Space Required.** ($\$20/m^2$) .... m^2

3 **Power Requirements.**

Number of outlets.				....	
Total heat output.				....	BTU's or kJ's.
Type of connector.	....	....	....	....	
Current capacity.	....	....	....	....	Amperes.
Current used.	....	....	....	....	(Startup)
	....	....	....	....	(Running)

4 **What Software will you display?**

....

....

5 **What Hardware will you display?**

....

....

6 We will provide 1 table, 2 chairs and power to the above specifications if possible.

Signed: ....

Please return to:
Stephen Rothwell,
Department of Computer Science,
Australian National University,

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

Australian UNIX* systems User Group
(AUUG)

Membership Application

I, ______________________________ do hereby apply for ordinary($50)/student($30)** membership of the Australian UNIX systems User Group and do agree to abide by the rules of the association especially with respect to non-disclosure of confidential and restricted licensed information. I understand that the membership fee entitles me to receive the Australian UNIX systems User Group Newsletter and I enclose payment of $_______ herewith.

Signed ______________________________ Date ____________________

==

Name ____

Mailing address for AUUG information ______________________________

____

____

Telephone number (including area code) ____________________________

UNIX Network address ____

I agree to my name and address being made available to software/hardware vendors YES □ NO □

==

Student Member Certification

I certify that ______________________________ is a full-time student at

____

Expected date of graduation _______________________________________

Faculty signature ______________________________ Date ____________________

==

Office use only 10/85

* UNIX is a trademark of AT&T Bell Laboratories

** Delete one

Australian UNIX* systems User Group Newsletter (AUUGN)

Subscription Application

I wish to subscribe to the Australian UNIX systems User Group Newsletter and enclose payment of $__________ herewith for the items indicated below.

Signed ____ Date ______________________

==

□	One years subscription (6 issues) available on microfiche or paper	$30.00
□	Back issues of Volume 1 (6 issues) available only on microfiche	$24.00
□	Back issues of Volume 2 (6 issues) available only on microfiche	$24.00
□	Back issues of Volume 3 (6 issues) available only on microfiche	$24.00
□	Back issues of Volume 4 (6 issues) available on microfiche, some paper copies	$24.00
□	Back issues of Volume 5 (6 issues) available on microfiche, some paper copies	$24.00
□	Subscribers outside Australia must pay more per volume to cover surface mail costs	$10.00
□	Subscribers outside Australia must pay more per volume to cover air mail costs	$30.00

==

Name ____

Mailing address ____

____

____

Telephone number (including area code) ______________________________

UNIX Network address ____

I agree to my name and address being made available to software/hardware vendors YES □ NO □

==

Office use only 10/85

* UNIX is a trademark of AT&T Bell Laboratories